IBM SPSS Modeler 18.2.1 Guida per script Python e automazione

Nota
Prima di utilizzare queste informazioni e il relativo prodotto, leggere le informazioni disponibili in “Informazioni particolari” a pagina 387.

Informazioni sul prodotto

Questa edizione si applica alla versione 18, release 2, livello di modifica 0 di IBM SPSS Modeler e a tutte le successive release e modifiche a meno che non sia diversamente indicato nelle nuove edizioni.

Indice

Capitolo 1. Script e linguaggio di script

Panoramica sugli script

Gli script di IBM® SPSS Modeler sono un potente strumento per automatizzare i processi nell'interfaccia utente. Tramite gli script è possibile eseguire gli stessi tipi di azioni eseguite con il mouse o la tastiera, nonché automatizzare le attività ripetitive o la cui esecuzione manuale richiederebbe un tempo molto maggiore.

È possibile utilizzare gli script per:

- Imporre un ordine specifico per l'esecuzione dei nodi in un flusso.
- Impostare le proprietà di un nodo ed eseguire le derivazioni utilizzando un sottoinsieme di CLEM (Control Language for Expression Manipulation).
- Specificare una sequenza automatica di operazioni che in genere richiedono l'intervento dell'utente, per esempio la creazione e la verifica di un modello.
- Impostare processi di grande complessità per i quali sono necessari interventi sostanziali da parte dell'utente, per esempio le procedure di convalida incrociata che richiedono più processi di creazione e verifica dei modelli.
- Impostare i processi di manipolazione dei flussi, ad esempio recuperare un flusso di addestramento per un modello, eseguirlo e creare il flusso di verifica del modello corrispondente in modo automatico.

In questo capitolo sono fornite descrizioni approfondite ed esempi di script a livello di flusso, script autonomi e script all'interno di Supernodi nell'interfaccia IBM SPSS Modeler. Per ulteriori informazioni sul linguaggio di script, la sintassi e i comandi, consultare i capitoli che seguono.

Nota:

Non è possibile importare ed eseguire script creati in IBM SPSS Statistics in IBM SPSS Modeler.

Tipi di script

IBM SPSS Modeler utilizza tre tipi di script:

- Gli **script del flusso** sono archiviati come proprietà di stream e quindi salvati e caricati con un flusso specifico. Per esempio, è possibile scrivere uno script del flusso che automatizza il processo di addestramento e applicazione di un nugget del modello. È anche possibile specificare che, ogni volta che viene eseguito un determinato stream, venga eseguito lo script anziché il contenuto dell'area del flusso.
- Gli **script autonomi** non sono associati ad alcun flusso particolare e vengono salvati in file di testo esterni. È possibile utilizzare uno script autonomo, per esempio, per manipolare insieme più flussi.
- Gli **script del Supernodo** vengono archiviati come proprietà del flusso Supernodo. Gli script del Supernodo sono disponibili solo nei Supernodi terminali. È possibile utilizzare uno script del Supernodo per controllare la sequenza di esecuzione del contenuto del Supernodo. Per i Supernodi non terminali (origine o di elaborazione), è possibile definire le proprietà del Supernodo o direttamente i nodi che esso contiene nello script del flusso.

Script del flusso

È possibile utilizzare gli script per personalizzare le operazioni all'interno di un flusso specifico e salvarli insieme al flusso. Gli script del flusso possono essere utilizzati per specificare un particolare ordine di esecuzione per i nodi terminali all'interno di un flusso. La finestra di dialogo di script del flusso consente di modificare lo script salvato insieme al flusso corrente.

Per accedere alla scheda dello script dello stream nella finestra di dialogo Proprietà stream:

1. Dal menu **Strumenti**, scegliere:
   Proprietà flusso > **Esecuzione**
2. Fare clic sulla scheda **Esecuzione** per utilizzare gli script per il flusso corrente.

Utilizzare le icone della barra degli strumenti nella parte superiore della finestra di dialogo dello script del flusso per le operazioni riportate di seguito:

- Importare nella finestra il contenuto di uno script autonomo preesistente.
- Salvare lo script come file di testo.
- Stampare uno script.
- Accodare lo script di default.
- Modificare uno script (annullare l'operazione, tagliare, copiare, incollare ed altre funzioni di modifica comuni).
- Eseguire l'intero script corrente.
- Eseguire righe selezionate di uno script.
- Arrestare uno script durante l'esecuzione. Questa icona è abilitata solo durante l'esecuzione di uno script.
- Controllare la sintassi dello script e, in caso di errori, visualizzarli nel riquadro inferiore della finestra di dialogo per esaminarli.

Nota: A partire dalla versione 16.0, SPSS Modeler utilizza il linguaggio di script Python. Tutte le versioni precedenti alla 16.0 utilizzavano un linguaggio di script univoco di SPSS Modeler, ora indicato come script Legacy. In base al tipo di script utilizzato, nella scheda **Esecuzione**, selezionare la modalità di esecuzione **Predefinita (script facoltativo)**, quindi selezionare **Python** o **Legacy**.

È possibile specificare se uno script debba essere eseguito o meno quando viene eseguito il flusso. Per eseguire lo script ogni volta che viene eseguito il flusso, rispettando l'ordine di esecuzione dello script, selezionare **Esegui questo script**. L'automazione a livello di flusso garantita in questo modo consente di accelerare la creazione del modello. Tuttavia, l'impostazione di default ignora questo script durante l'esecuzione del flusso. Anche se si seleziona l'opzione **Ignora questo script**, è sempre possibile eseguire lo script direttamente da questa finestra di dialogo.

L'editor di script include le seguenti funzioni che rendono più semplice la creazione di script:

- Evidenziazione della sintassi: parole chiave, valori letterali (come stringhe e numeri) e commenti sono evidenziati.
- Numerazione delle righe.
- Corrispondenza del blocco: quando il cursore viene posizionato all'inizio di un blocco di programma, viene evidenziato anche il blocco finale corrispondente.
- Suggerimenti per il completamento automatico.

Gli stili di testo e colori utilizzati dal programma di evidenziazione della sintassi possono essere personalizzati utilizzando le preferenze di visualizzazione di IBM SPSS Modeler. Per accedere alle preferenze di visualizzazione, selezionare **Strumenti** > **Opzioni** > **Opzioni utente** e selezionare la scheda **Sintassi**.

È possibile accedere ad un elenco di completamenti della sintassi suggeriti selezionando **Suggerimento automatico** dal menu di contesto oppure premendo Ctrl + Spazio. Utilizzare i tasti cursore per spostarsi verso l'alto e verso il basso all'interno dell'elenco, quindi premere Invio per inserire il testo selezionato. Per uscire dalla modalità di suggerimento automatico senza modificare il testo esistente, premere Esc.

La scheda **Debug** visualizza i messaggi di debug e può essere utilizzata per valutare lo stato dello script una volta eseguito lo script. La scheda **Debug** è composta da un'area di testo di sola lettura e da un

campo di testo di input a riga singola. L'area di testo visualizza il testo inviato dagli script all'output standard o all'errore standard, ad esempio mediante il testo del messaggio di errore. Il campo del testo di input accetta l'input da parte dell'utente. Tale input viene valutato all'interno del contesto dello script eseguito più recentemente all'interno della finestra di dialogo (detto *contesto di script*). L'area di testo contiene i comandi e l'output risultante, in modo che gli utenti possano visualizzare una traccia dei comandi. Il campo di input del testo contiene sempre il prompt dei comandi (`-->` per gli script Legacy).

Nelle seguenti circostanze viene creato un nuovo contesto di script:

- Uno script viene eseguito utilizzando **Esegui questo script** oppure **Esegui righe selezionate**.
- Il linguaggio di script viene modificato.

Se viene creato un nuovo contesto di script, l'area di testo viene svuotata.

Nota: L'esecuzione di un flusso al di fuori del riquadro dello script non modifica il contesto dello script del riquadro dello script. I valori delle variabili creati come parte dell'esecuzione non sono visibili all'interno della finestra di dialogo dello script.

Esempio di script del flusso: Addestramento di una rete neurale

È possibile utilizzare un flusso per addestrare una rete neurale durante l'esecuzione. La verifica del modello prevede in genere l'esecuzione del nodo di creazione modelli per aggiungere il modello al flusso, l'esecuzione delle connessioni appropriate e l'esecuzione del nodo Analisi.

Con uno script di IBM SPSS Modeler, è possibile automatizzare il processo di verifica del nugget del modello creato. Per esempio, il seguente script del flusso per il flusso di esempio `druglearn.str` (disponibile nella cartella `/Demos/streams/` dell'installazione di IBM SPSS Modeler) può essere eseguito dalla finestra di dialogo Proprietà flusso (**Strumenti > Proprietà flusso > Script**):

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

L'elenco riportato di seguito descrive ogni riga in questo esempio di script.

- La prima riga definisce una variabile che punta al flusso corrente.
- Nella riga 2, lo script rileva il nodo builder Rete neurale.
- Nella riga 3, lo script crea un elenco in cui è possibile archiviare i risultati dell'esecuzione.
- Nella riga 4, viene creato il nugget del modello Rete Neurale. Tale elemento viene archiviato nell'elenco definito alla riga 3.
- Nella riga 5, per il nugget del modello viene creato un nodo Applicazione del modello che viene posizionato nell'area di disegno del flusso.
- Nella riga 6, viene creato un nodo di analisi denominato `Drug`.
- Nella riga 7, lo script trova il nodo Tipo.
- Nella riga 8, lo script collega il nodo Applicazione del modello creato alla riga 5 tra il nodo Tipo ed il nodo Analisi.
- Infine, viene eseguito il nodo Analisi per produrre il report di analisi.

È possibile utilizzare uno script per creare ed eseguire un flusso nuovo, partendo da un'area vuota. Per ulteriori informazioni sul linguaggio di script in generale, vedere Panoramica sul linguaggio di script.

Limiti di dimensione del codice Jython

Jython compila ciascuno script nel bytecode Java, che viene quindi eseguito dalla JVM (Java Virtual Machine). Tuttavia, Java impone un limite sulla dimensione di un singolo bytecode. Quindi quando Jython tenta di caricare il bytecode, può determinare la chiusura anomala della JVM. IBM SPSS Modeler non può impedire che ciò accada.

Assicurarsi di scrivere gli script Jython utilizzando le pratiche di codifica corrette (quali la riduzione del codice duplicato utilizzando variabili o funzioni per calcolare valori intermedi comuni). Se necessario, è possibile suddividere il codice su diversi file di origine o definirlo utilizzando moduli che saranno quindi compilati in file bytecode separato.

Script autonomi

Nella finestra di dialogo Script autonomo è possibile creare o modificare uno script salvato come file di testo. Nella finestra viene visualizzato il nome del file e sono disponibili funzionalità per il caricamento, il salvataggio, l'importazione e l'esecuzione degli script.

Per accedere alla finestra di dialogo dello script autonomo:

Dal menu principale, scegliere:

Strumenti > **Script autonomi**

Per gli script autonomi e del flusso sono disponibili la stessa barra degli strumenti e le stesse opzioni di controllo della sintassi degli script. Per ulteriori informazioni, consultare l'argomento “Script del flusso” a pagina 1.

Esempio di script autonomo: salvataggio e caricamento di un modello

Gli script autonomi sono utili per la manipolazione dei flussi. Si supponga di avere due flussi, uno che crea un modello e un altro che utilizza grafici per analizzare l'insieme di regole generato dal primo flusso mediante i campi di dati esistenti. Uno script autonomo per questa situazione potrebbe essere simile al seguente:

```
taskrunner = modeler.script.session().getTaskRunner()

# Modify this to the correct Modeler installation Demos folder.
# Note use of forward slash and trailing slash.
installation = "C:/Program Files/IBM/SPSS/Modeler/19/Demos/"

# First load the model builder stream from file and build a model
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Save the model to file
taskrunner.saveModelToFile(results[0], "rule.gm")

# Now load the plot stream, read the model from file and insert it into the stream
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivenode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
```

```
modelapplier.setPositionBetween(derivenode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivenode, plotnode)
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

Nota: Per ulteriori informazioni sul linguaggio di script in generale, vedere "Panoramica sul linguaggio di script" a pagina 15.

Esempio di script autonomo: generazione di un modello di selezione funzioni

Iniziando con un'area vuota, questo esempio crea un flusso che genera un Modello di selezione funzioni, applica il modello e crea una tabella che elenca i 15 campi più importanti relativi all'obiettivo specificato.

```
stream = modeler.script.session().createProcessorStream("featureselection", True)

statisticsimportnode = stream.createAt("statisticsimport", "Statistics File", 150, 97)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/customer_dbase.sav")

typenode = stream.createAt("type", "Type", 258, 97)
typenode.setKeyedPropertyValue("direction", "response_01", "Target")

featureselectionnode = stream.createAt("featureselection", "Feature Selection", 366, 97)
featureselectionnode.setPropertyValue("top_n", 15)
featureselectionnode.setPropertyValue("max_missing_values", 80.0)
featureselectionnode.setPropertyValue("selection_mode", "TopN")
featureselectionnode.setPropertyValue("important_label", "Check Me Out!")
featureselectionnode.setPropertyValue("criteria", "Likelihood")

stream.link(statisticsimportnode, typenode)
stream.link(typenode, featureselectionnode)
models = []
featureselectionnode.run(models)

# Assumes the stream automatically places model apply nodes in the stream
applynode = stream.findByType("applyfeatureselection", None)
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96, applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

Questo script crea un nodo origine nel quale leggere i dati, utilizza un nodo Tipo per impostare il ruolo (direzione) del campo `response_01` su `Obiettivo`, quindi crea ed esegue un nodo Selezione funzioni. Inoltre, lo script connette i nodi e le posizioni nell'area del flusso per generare un layout leggibile. Il nugget del modello così ottenuto viene quindi connesso a un nodo Tabella, che elenca i 15 campi più importanti come determinato dalle proprietà `selection_mode` e `top_n`. Per ulteriori informazioni, consultare l'argomento "proprietà featureselectionnode" a pagina 213.

Script dei Supernodi

È possibile creare e salvare script all'interno di qualsiasi Supernodo terminale utilizzando il linguaggio di script di IBM SPSS Modeler. Questi script sono disponibili solo per i Supernodi terminali e vengono spesso utilizzati durante la creazione di modelli di stream o per imporre un ordine di esecuzione speciale per il contenuto del Supernodo. Gli script del Supernodo consentono anche l'esecuzione di più di uno script all'interno di un flusso.

Per esempio, si supponga che sia stato necessario specificare l'ordine di esecuzione di un flusso complesso e che il Supernodo contenga più nodi tra cui un nodo Calcola globali, che deve essere eseguito prima di creare un nuovo campo utilizzato in un nodo Plot. In tal caso, è possibile creare uno script del

Supernodo che esegue prima il nodo Calcola globali. I valori calcolati da questo nodo, quali la media o la deviazione standard, possono quindi essere utilizzati quando viene eseguito il nodo Plot.

All'interno di uno script del Supernodo è possibile specificare le proprietà del nodo analogamente agli altri script. In alternativa, è possibile modificare e definire le proprietà di qualsiasi Supernodo o dei suoi nodi incapsulati direttamente da uno script del flusso. Per ulteriori informazioni, consultare l'argomento Capitolo 21, “Proprietà dei Supernodi”, a pagina 371. Questo metodo funziona per i Supernodi origine e di elaborazione e per i Supernodi terminali.

Nota: poiché solo i Supernodi terminali possono eseguire i propri script, la scheda Script della finestra di dialogo Supernodo è disponibile solo per i Supernodi terminali.

Per aprire la finestra di dialogo Script Supernodo dall'area principale:

Selezionare un Supernodo terminale nell'area dello script e, dal menu Supernodo, scegliere:

Script Supernodo...

Per aprire la finestra di dialogo Script Supernodo dall'area del Supernodo in modalità Zoom avanti:

Fare clic con il tasto destro del mouse sull'area del Supernodo e, dal menu di scelta rapida, selezionare:

Script Supernodo...

Esempio di script di un Supernodo

Lo script del Supernodo riportato di seguito dichiara l'ordine in cui devono essere eseguiti i nodi terminali all'interno del Supernodo. Questo ordine assicura che il nodo Calcola globali venga eseguito per primo, in modo che i valori calcolati da questo nodo possano successivamente essere utilizzati quando viene eseguito un altro nodo.

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

Blocco e sblocco dei supernodi

L'esempio riportato di seguito mostra come è possibile bloccare e sbloccare un supernodo:

```
stream = modeler.script.stream()
superNode=stream.findByID('id854RNTSD5MB')
# unlock one super node
print 'unlock the super node with password abcd'
if superNode.unlock('abcd'):
    print 'unlocked.'
else:
    print 'invalid password.'
# lock one super node
print 'lock the super node with password abcd'
superNode.lock('abcd')
```

Esecuzioni di cicli ed esecuzione condizionale nei flussi

Dalla Versione version 16.0 in poi, SPSS Modeler consente di creare alcuni script di base all'interno di un flusso selezionando i valori all'interno di varie finestre di dialogo invece di dover scrivere istruzioni direttamente nel linguaggio di script. I due tipi principali di script che è possibile creare in questo modo sono cicli semplici e un modo per eseguire i nodi se una condizione è stata soddisfatta.

È possibile combinare le regole sia dell'esecuzione di cicli che dell'esecuzione condizionale all'interno di un flusso. Ad esempio, si supponga di avere i dati relativi alle vendite di automobili dai produttori di tutto il mondo. È possibile impostare un ciclo per elaborare i dati in un flusso, identificando i dettagli per paese di produzione ed creare output di dati in grafici diversi che mostrano i dettagli come ad esempio il volume di vendite per modello, i livelli di emissione sia per produttore che per dimensione del motore e così via. Se si fosse interessati ad analizzare solo le informazioni Europee, si potrebbero anche aggiungere condizioni nell'esecuzione del ciclo che forniscano grafici creati per produttori situati in America e Asia.

Nota: Poichè sia l'esecuzione di cicli che l'esecuzione condizionale sono basate su script in background, questi vengono appplicati solo ad un flusso totale quando viene eseguito.

- **Esecuzione di cicli** È possibile utilizzare l'esecuzione di cicli per automatizzare attività ripetitive. Ad esempio, questo potrebbe significare l'aggiunta di un dato numero di nodi a un flusso e la modifica di un parametro del nodo ogni volta. In alternativa, è possibile controllare l'esecuzione di un flusso o ramo ancora una volta per un dato numero di volte, come nei seguenti esempi:
  - Eseguire il flusso un dato numero di volte e modificare l'origine ogni volta.
  - Eseguire il flusso un dato numero di volte modificando il valore di una variabile ogni volta.
  - Eseguire il flusso un dato numero di volte immettendo un campo aggiuntivo ad ogni esecuzione.
  - Costruire un modello un dato numero di volte e modificare le impostazioni del modello ogni volta.
- **Esecuzione Condizionale** È possibile utilizzarla per controllare come i nodi terminali vengono eseguiti, in base alle condizioni che si predefiniscono, gli esempi possono includere i seguenti:
  - In base a se un dato valore è vero o falso, controlla se un nodo verrà eseguito.
  - Definisce se un'esecuzione di cicli di nodi verrà eseguita in parallelo o sequenziale.

Sia l'esecuzione di cicli che l'esecuzione condizionale vengono configurate sulla scheda Esecuzione all'interno della finestra di dialogo Proprietà del flusso. I nodi che vengono utilizzati nei requisiti condizionali o di cicli vengono mostrati con un simbolo aggiuntivo a loro allegato sull'area di disegno del flusso per indicare che stanno prendendo parte nell'esecuzione di cicli e nell'esecuzione condizionale.

È possibile accedere alla scheda Esecuzione in uno dei 3 modi:

- Utilizzando i menu nella parte superiore della finestra di dialogo principale:
  1. Dal menu Strumenti, scegliere:

     Proprietà flusso > **Esecuzione**
  2. Fare clic sulla scheda Esecuzione per utilizzare gli script per il flusso corrente.
- Dall'interno di un flusso:
  1. Fare clic col tasto destro su un nodo e scegliere **Esecuzione Cicli/Condizionale**.
  2. Selezionare l'opzione pertinente del sottomenu.
- Dalla barra degli strumenti del grafico nella parte superiore della finestra di dialogo principale, fare clic sull'icona proprietà del flusso.

Se questa è la prima volta che si configurano i dettagli o dell'esecuzione di cicli o dell'esecuzione condizionale, nella scheda Esecuzione selezionare la modalità di esecuzione **Esecuzione Cicli/Condizionale** e poi selezionare o la sottoscheda **Condizionale** o quella **Cicli**.

Esecuzione di cicli nei flussi

Con l'esecuzione di cicli è possibile automatizzare le attività ripetitive nei flussi; alcuni esempio potrebbero essere i seguenti:

- Eseguire il flusso un dato numero di volte e modificare l'origine ogni volta.
- Eseguire il flusso un dato numero di volte modificando il valore di una variabile ogni volta.
- Eseguire il flusso un dato numero di volte immettendo un campo aggiuntivo ad ogni esecuzione.
- Costruire un modello un dato numero di volte e modificare le impostazioni del modello ogni volta.

Le condizioni da soddisfare vengono impostate nella sottoscheda **Esecuzione di cicli** della scheda Esecuzione del flusso. Per visualizzare la sottoscheda, selezionare la modalità di esecuzione **Esecuzione di cicli/Esecuzione Condizionale** .

Ogni requisito dell'esecuzione di cicli che viene definita avrà effetto quando il flusso viene eseguito, se è stata impostata la modalità di esecuzione **Esecuzione di cicli/Esecuzione Condizionale**. Se lo si desidera,è possibile generare il codice dello script per i requisiti di esecuzione dei cicli ed incollarlo nell'editor dello script facendo clic su **Incolla...** nell'angolo in basso a destra della sottoscheda Esecuzione di Cicli; la scheda principale Esecuzione visualizza le modifiche da mostrare nella modalità di esecuzione **Default (script facoltativo)** con lo script nella parte in alto della scheda. Questo significa che è possibile definire una struttura di esecuzione dei cicli utilizzando le varie opzioni delle finestre di dialogo di esecuzione dei cicli prima di generare uno script che è possibile personalizzare ulteriormente nell'editor dello script. Si noti che quando si fa clic su **Incolla...** ogni requisito di esecuzione condizionale che è stato definito, verrà visualizzato nello script generato.

Importante: Le variabili nei cicli impostate in un flusso SPSS Modeler potrebbero essere sovrascritte se si esegue il flusso in un lavoro IBM SPSS Collaboration and Deployment Services. Ciò si verifica perché la voce dell'editor lavori IBM SPSS Collaboration and Deployment Services sostituisce la voce SPSS Modeler. Ad esempio, se si imposta una variabile nel ciclo nel flusso per creare un diverso nome del file di output per ciascun ciclo, i file vengono denominati correttamente in SPSS Modeler ma vengono sovrascritti dalla voce prefissata immessa nella scheda Risultato di IBM SPSS Collaboration and Deployment Services Deployment Manager.

Impostazione di un ciclo

1. Creare una chiave di iterazione per definire una struttura principale dell'esecuzione dei cicli che devono essere eseguiti nel flusso. Consultare Creare una chiave di iterazione per ulteriori informazioni.
2. Quando necessario, definire una o più variabili di iterazione. Consultare Creare una variabile di iterazione per ulteriori informazioni.
3. Le iterazioni, e qualsiasi varibile creata, vengono mostrate nel corpo principale della sottoscheda. Per default, le iterazioni vengono eseguite nell'ordine in cui appaiono; per spostare una iterazione su o giù nell'elenco, selezionarla con un clic e quindi utilizzare le frecce su o giù nella colonna a destra della sottoscheda, per modificarne l'ordine.

Creazione di una chiave di iterazione per l'esecuzione di cicli nei flussi

Si utilizza una chiave di iterazione per definire la struttura principale dell'esecuzione dei cicli che devono essere eseguiti nel flusso. Ad esempio, se si sta analizzando la vendita delle automobili, si potrebbe creare un parametro di flusso *Paese di produzione* e utilizzarlo come chiave di iterazione; quando il flusso viene eseguito, questa chiave è impostata su ogni valore dei diversi paesi nei propri dati durante ogni iterazione. Utilizzare la finestra di dialogo Definisci chiave di iterazione per impostare la chiave.

Per aprire la finestra di dialogo, selezionare o il pulsante **Chiave di iterazione...** nell'angolo in basso a sinistra della sottoscheda di Esecuzione dei cicli o fare clic sul pulsante destro del mouse su qualsiasi nodo nel flusso e selezionare o **Esecuzione cicli/Esecuzione condizionale** > **Definisci Chiave di iterazione (Campi)** o **Esecuzione cicli/Esecuzione condizionale** > **Definisci Chiave di iterazione (Valori)**. Se si apre la finestra di dialogo del flusso, alcuni dei campi possono essere completati automaticamente per l'utente, come ad esempio il nome del nodo.

Per impostare una chiave iterazione, completare i seguenti campi:

Agisce sui. È possibile selezionare una delle seguenti opzioni:

- **Parametro di flusso - Campi**. Utilizzare questa opzione per creare un ciclo che imposti il valore di un parametro di flusso esistente in ogni campo specificato mano mano.
- **Parametro di flusso - Valori**. Utilizzare questa opzione per creare un ciclo che imposti il valore di un parametro di flusso esistente in ogni valore specificato mano mano.

- **Proprietà del nodo - Campi**. Utilizzare questa opzione per creare un ciclo che imposti il valore della proprietà del nodo in ogni campo specificato mano mano.
- **Proprietà del nodo - Valori**. Utilizzare questa opzione per creare un ciclo che imposti il valore della proprietà del nodo in ogni valore specificato mano mano.

Cosa impostare. Scegliere l'elemento che avrà il valore impostato ogni volta che il ciclo viene eseguito. È possibile selezionare una delle seguenti opzioni:

- **Parametro**. Disponibile solo se si seleziona **Parametro di flusso– Campi** o **Parametro di flusso – Valori**. Selezionare il parametro richiesto dall'elenco disponibile.
- **Nodo**. Disponibile solo se si seleziona **Proprietà del nodo– Campi** o **Proprietà del nodo – Valori**. Selezionare il nodo per cui si desidera impostare un ciclo. Fare clic sul pulsante sfoglia per aprire la finestra di dialogo Seleziona Nodo e scegliere il nodo che si desidera; se vi sono troppi nodi elencati, è possibile filtrare la visualizzazione per mostrare solo certi tipi di nodi selezionando una delle seguenti categorie: Nodi Origine, Processo, Grafico, Modello, Output, Esporta o Modelli applicati.
- **Proprietà**. Disponibile solo se si seleziona **Proprietà del nodo– Campi** o **Proprietà del nodo – Valori**. Selezionare la proprietà del nodo dall'elenco disponibile.

Campi da utilizzare. Disponibile solo se si seleziona **Parametro di flusso– Campi** o **Proprietà del nodo – Campi**. Scegliere il campo, o i campi, all'interno di un nodo da utilizzare per fornire i valori di iterazione. È possibile selezionare una delle seguenti opzioni:

- **Nodo**. Disponibile solo se si seleziona **Parametro di flusso – Campi**. Selezionare il nodo che contiene i dettagli per i quali si desidera impostare un ciclo. Fare clic sul pulsante sfoglia per aprire la finestra di dialogo Seleziona Nodo e scegliere il nodo che si desidera; se vi sono troppi nodi elencati, è possibile filtrare la visualizzazione per mostrare solo certi tipi di nodi selezionando una delle seguenti categorie: Nodi Origine, Processo, Grafico, Modello, Output, Esporta o Modelli applicati.
- **Elenco campi**. Fare clic sul pulsante elenco nella colonna destra per visualizzare la finestra di dialogo Seleziona Campi, all'interno della quale è possibile selezionare i campi nel nodo per fornire i dati di iterazione. Consultare “Selezione campi per le iterazioni” a pagina 10 per ulteriori informazioni.

Valori da utilizzare. Disponibile solo se si seleziona **Parametro di flusso – Valori** o **Proprietà del nodo – Valori**. Scegliere il valore, o i valori, all'interno del campo selezionato da utilizzare come valori di iterazione. È possibile selezionare una delle seguenti opzioni:

- **Nodo**. Disponibile solo se si seleziona **Parametro di flusso – Valori**. Selezionare il nodo che contiene i dettagli per i quali si desidera impostare un ciclo. Fare clic sul pulsante sfoglia per aprire la finestra di dialogo Seleziona Nodo e scegliere il nodo che si desidera; se vi sono troppi nodi elencati, è possibile filtrare la visualizzazione per mostrare solo certi tipi di nodi selezionando una delle seguenti categorie: Nodi Origine, Processo, Grafico, Modello, Output, Esporta o Modelli applicati.
- **Elenco campi**. Selezionare il campo nel nodo per fornire i dati di iterazione.
- **Elenco valori**. Fare clic sul pulsante elenco nella colonna destra per visualizzare la finestra di dialogo Seleziona Valori, all'interno della quale è possibile selezionare i valori nel campo per fornire i dati di iterazione.

Creazione di una variabile di iterazione per l'esecuzione di cicli nei flussi

È possibile utilizzare le variabili di iterazione per modificare i valori dei parametri del flusso o le proprietà dei nodi selezionati all'interno di un flusso ogni volta che viene eseguito un ciclo. Ad esempio, se il ciclo del flusso sta analizzando i dati di vendita delle automobili e sta utilizzando *Paese di produzione* come chiave di iterazione, si potrebbe avere un grafico di output che mostra le vendite per modello e un altro grafico di output che mostra le informazioni sulle emissioni di gas di scarico. In questi casi, è possibile creare variabili di iterazione che creano nuovi titoli per i grafici risultanti, per esempio *Emissioni veicoli svedesi* e *Vendite automobili giapponesi per modello*. Utilizzare la finestra di dialogo Definisci variabile di iterazione per impostare una qualsiasi variabile che si desidera richiedere.

Per aprire la finestra di dialogo, selezionare il pulsante **Aggiungi variabile...** nell'angolo in basso alla sinistra della scheda secondaria Esecuzione di cicli, o fare clic con il tasto destro del mouse su qualsiasi nodo nel flusso e selezionare:**Esecuzione di cicli/Esecuzione condizionale** > **Definisci variabile di iterazione**.

Per impostare una variabile di iterazione, completare i seguenti campi:

Modifica. Selezionare il tipo di attributo che che si desidera modificare. È possibile scegliere o tra **Parametro di flusso** o **Proprietà del nodo**.

- Se si seleziona **Parametro di flusso**, scegliere il parametro richiesto e quindi, utilizzando una delle seguenti opzioni, se disponibili nel proprio flusso, definire quale valore di quel parametro deve essere impostato con ogni iterazione del ciclo:
  - **Variabile globale**. Selezionare la variabile globale che il parametro di flusso deve impostare.
  - **Cella tabella di output**. Per impostare un parametro di flusso come valore nella cella della tabella di output, selezionare la tabella dall'elenco e immettere la **Riga** e la **Colonna** da utilizzare.
  - **Immettere manualmente**. Selezionare questa opzione se si desidera immettere manualmente un valore per questo parametro da prendere in ogni iterazione. Quando si torna alla sottoscheda esecuzione di cicli viene creata una nuova colonna in cui si inserisce il testo richiesto.
- Se si seleziona **Proprietà del nodo**, scegliere il nodo richiesto e una delle relative proprietà e quindi impostare il valore che si desidera utilizzare per tale proprietà. Impostare il nuovo valore della proprietà utilizzando una delle seguenti opzioni:
  - **Singolo**. Il valore della proprietà utilizzerà il valore della chiave di iterazione. Consultare "Creazione di una chiave di iterazione per l'esecuzione di cicli nei flussi" a pagina 8 per ulteriori informazioni.
  - **Come prefisso per Ramo**. Utilizza il valore della chiave di iterazione come prefisso di quello che è stato immesso nel campo **Ramo**.
  - **Come suffisso per Ramo**. Utilizza il valore della chiave di iterazione come suffisso di quello che è stato immesso nel campo **Ramo**.

  Se si seleziona l'opzione prefisso o suffisso viene richiesto di aggiungere il testo aggiuntivo al campo **Ramo**. Per esempio, se il valore della chiara di iterazione è *Paese di produzione* e si seleziona **Come prefisso per Ramo**, è possibile immettere ì - *vendite per modello* in questo campo.

Selezione campi per le iterazioni

Quando si creano le iterazioni è possibile selezionare uno o più campi utilizzando la finestra di dialogo Seleziona Campi.

Ordina per Per visualizzare i campi disponibili in un determinato ordine, sono disponibili le seguenti opzioni:

- **Naturale** Visualizza l'ordine dei campi secondo la modalità di passaggio a valle nel flusso di dati nel nodo corrente.
- **Nome** Utilizza l'ordine alfabetico per ordinare i campi per la visualizzazione.
- **Tipo** Visualizza i campi ordinati in base al relativo livello di misurazione. Questa opzione è utile quando si selezionano campi con un determinato livello di misurazione.

Selezionare i campi dall'elenco uno per volta, oppure utilizzare i metodi Maiusc-clic e Ctrl-clic per selezionare più campi contemporaneamente. È anche possibile utilizzare i pulsanti nella parte inferiore dell'elenco per selezionare gruppi di campi in base al livello di misurazione, oppure per selezionare o deselezionare tutti i campi nella tabella.

Si noti che il campi disponibile per essere selezionati sono filtrati in modo da mostrare solo i campi che sono appropriati per i parametri del flusso o le proprietà del nodo che si sta utilizzando. Per esempio, se si sta utilizzando un parametro di flusso che ha Stringa come tipo di archiviazione, vengono mostrati solo i campi che hanno come tipo di archiviazione Stringa.

Esecuzione condizionale nei flussi

Con l'esecuzione condizionale è possibile controllare il modo in cui i nodi terminali vengono eseguiti, in base ai contenuti del flusso corrispondenti alle condizioni che si desidera definire; esempi possono includere:

- In base a se un dato valore è vero o falso, controlla se un nodo verrà eseguito.
- Definisce se un'esecuzione di cicli di nodi verrà eseguita in parallelo o sequenziale.

Si impostano le condizioni che devono essere soddisfatte nella sottoscheda **Condizionale** della scheda Esecuzione del flusso. Per visualizzare la sottoscheda, selezionare la modalità di esecuzione **Esecuzione di cicli/Esecuzione Condizionale** .

Qualsiasi requisito dell'esecuzione condizionale che si definisce avrà effetto quando si eseguirà il flusso, se è stata impostata la modalità di esecuzione **Esecuzione di Cicli/Condizionale**. Facoltativamente, è possibile generare il codice dello script per i propri requisiti di esecuzione condizionale e incollarlo nell'editor dello script facendo clic su **Incolla...** nell'angolo destro in basso della sottoscheda Condizionale; la scheda principale Esecuzione visualizza le modifiche da mostrare nella modalità di esecuzione **Default (script facoltativo)** con lo script nella parte in alto della scheda. Questo significa che è possibile definire le condizioni utilizzando le varie finestre di dialogo delle opzioni per l'esecuzione dei cicli prima di generare uno script che è possibile personalizzare ulteriormente nell'editor dello script. Si noti che quando si fa clic su **Incolla...** qualunque requisito che è stato definito per l'esecuzione dei cicli sarà visualizzato nello script generato.

Per impostare una condizione:

1. Nella colonna a destra della scheda secondaria Condizionale, fare clic sul pulsante Aggiungi nuova condizione  per aprire la finestra di dialogo Aggiungi istruzione di esecuzione condizionale. In questa finestra di dialogo viene specificata la condizione che deve essere soddisfatta per far si che il nodo venga eseguito.
2. Nella finestra di dialogo Aggiungi istruzione di esecuzione condizionale, specificare quanto segue:
   a. **Nodo**. Selezionare il nodo per cui si desidera impostare un'esecuzione condizionale. Fare clic sul pulsante sfoglia per aprire la finestra di dialogo Seleziona Nodo e scegliere il nodo che si desidera; se vi sono troppi nodi elencati, è possibile filtrare la visualizzazione per mostrare i nodi da una delle seguenti categorie: Nodo Esporta, Grafico, Modello o Output.
   b. **Condizione basata su**. Specificare la condizione che deve essere soddisfatta per il nodo da eseguire. È possibile sceglie tra quattro opzioni: **Parametri flusso**, **Variabile globale**, **Cella tabella di output** oppure **Sempre vero**. I dettagli immessi nella metà inferiore della finestra di dialogo vengono controllati dalle condizioni scelte.
      - **Parametri di flusso**. Selezionare il parametro dall'elenco disponibile e quindi scegliere **Operatore** per quel parametro; per esempio, l'operatore potrebbe essere Maggiore di, Uguale, Minore, Tra e così via. Quindi immettere il **Valore**, o i valori minimo e massimo, in base all'operatore.
      - **Variabile globale**. Selezionare la variabile dall'elenco disponibile; per esempio, potrebbe essere: Media, Somma, Valore minimo, Valore massimo oppure Deviazione standard. Quindi selezionare il campo **Operatore** ed i valori richiesti.
      - **Cella tabella di output**. Selezionare il nodo tabella dall'elenco disponibile e quindi scegliere **Riga** e **Colonna** nella tabella. Quindi selezionare il campo **Operatore** ed i valori richiesti.
      - **Sempre vero**. Selezionare questa opzione se il nodo deve essere sempre eseguito. Se si seleziona questa opzione, non ci sono ulteriori parametri da selezionare.
3. Ripetere i passi 1 e 2 il numero di volte necessario all'impostazione di tutte le condizioni richieste. Il nodo selezionato e la condizione da rispettare prima che il nodo venga eseguito, sono mostrati neo corpo principale della sottoscheda rispettivamente nelle colonne **Nodo di esecuzione** e **Se questa condizione è verae**.

4. Per default, i nodi e le condizioni vengono eseguite nell'ordine di visualizzazione; per spostare un nodo o una condizione su o giù nell'elenco, selezionarlo con un clic e quindi utilizzare le frecce su o giù nella colonna a destra della sottoscheda per modificare l'ordine.

Inoltre, è possibile impostare le seguenti opzioni nella parte inferiore della sottoscheda Condizionale:

- **Valuta tutti in ordine**. Selezionare questa opzione per valutare ogni condizione nell'ordine in cui sono visualizzate nella sottoscheda. I nodi per i quali le condizioni vengono verificate essere "Vero" saranno tutti eseguiti una volta che tutte le condizioni sono state valutate.
- **Eseguire uno alla volta**. Disponibile solo se è stato selezionato **Valuta tutti in ordine**. Selezionando questa opzione significa che, se una condizione viene valutata come "Vera", il nodo associato con quella condizione viene eseguito prima che la condizione successiva venga valutata.
- **Valuta fino al primo risultato**. Selezionando questa opzione significa che sarà eseguito solo il primo nodo che ritorna una valutazione "Vero" dalle condizioni specificate.

Esecuzione ed interruzione degli script

Sono disponibili diversi sistemi per l'esecuzione degli script. Per esempio, nello script del flusso o nella finestra di dialogo dello script autonomo, il pulsante "Esegui questo script" esegue lo script completo:

Figura 1. Pulsante Esegui questo script

Il pulsante "Esegui solo righe selezionate" esegue una sola riga o un blocco di righe adiacenti selezionate nello script:

Figura 2. Pulsante Esegui solo righe selezionate

Per eseguire gli script è possibile utilizzare i metodi seguenti:

- Fare clic sul pulsante "Esegui questo script" o "Esegui solo righe selezionate" all'interno dello script di un flusso o nella finestra di dialogo dello script autonomo.
- Eseguire un flusso nel quale il metodo di esecuzione predefinito impostato è **Esegui questo script**.
- Utilizzare il flag `-execute` all'avvio in modalità interattiva. Per ulteriori informazioni, consultare l'argomento “Utilizzo degli argomenti della riga di comando” a pagina 63.

Nota: uno script del Supernodo verrà eseguito insieme al Supernodo se nella finestra di dialogo Script Supernodo è stata selezionata l'opzione **Esegui questo script**.

Interruzione dell'esecuzione degli script

Nella finestra di dialogo dello script di un flusso, il pulsante rosso Interrompi viene attivato durante l'esecuzione dello script. Questo pulsante consente di interrompere l'esecuzione dello script e di qualsiasi stream corrente.

Trova e sostituisci

La finestra di dialogo Trova/Sostituisci è disponibile ogni volta che è possibile modificare il testo di script o di espressioni, compreso l'editor di script, il generatore di espressioni CLEM e quando si definisce un modello nel nodo Report. Quando si modifica un testo in una di queste aree, premere `Ctrl+F` per accedere alla finestra di dialogo, assicurandosi che il cursore sia posizionato in un'area di testo. In un

nodo Riempimento, per esempio, è possibile accedere alla finestra di dialogo da qualsiasi area di testo della scheda Impostazioni oppure dal campo testo nel Generatore di espressioni.

1. Con il cursore posizionato in un'area di testo, premere Ctrl+F per accedere alla finestra di dialogo Trova/Sostituisci.
2. Immettere il testo da cercare oppure sceglierne uno dall'elenco a discesa degli elementi cercati di recente.
3. Se necessario, immettere il testo sostitutivo.
4. Fare clic su **Trova successivo** per avviare la ricerca.
5. Fare clic su **Sostituisci** per sostituire la selezione corrente oppure scegliere **Sostituisci tutto** per aggiornare tutte le istanze o quelle selezionate.
6. Al termine di ogni operazione, la finestra di dialogo si chiude. Premere F3 da qualsiasi area di testo per ripetere l'ultima operazione di ricerca oppure premere Ctrl+F per accedere nuovamente alla finestra di dialogo.

Opzioni di ricerca

Caratteri maiuscoli/minuscoli. Specifica se l'operazione di ricerca fa distinzione tra caratteri maiuscoli/minuscoli, per esempio se *miavar* corrisponde a *miaVar*. Il testo sostitutivo viene sempre inserito esattamente come viene digitato, indipendentemente da questa impostazione.

Solo parole intere. Specifica se l'operazione di ricerca cerca le occorrenze che sono parole intere. Se questa opzione è selezionata, la ricerca di *palla* non consentirà di trovare per esempio *pallavolo* o *Palladio*.

Espressioni regolari. Specifica se è utilizzata la sintassi delle espressioni regolari (vedere la sezione seguente). Quando questa opzione è selezionata, l'opzione **Solo parole intere** è disattivata e il relativo valore viene ignorato.

Solo testo selezionato. Controlla l'ambito della ricerca quando si utilizza l'opzione **Sostituisci tutto**.

Sintassi delle espressioni regolari

Le espressioni regolari consentono di cercare caratteri speciali, quali tabulazioni o caratteri di nuova riga, classi o intervalli di caratteri quali *a - d*, cifre e caratteri diversi da cifre, nonché limiti, per esempio l'inizio o la fine di una riga. Sono supportati i seguenti tipi di espressioni.

Tabella 1. Corrispondenze di caratteri

Caratteri	Corrispondenze
x	Il carattere x
\\	Il carattere barra rovesciata
\0n	Il carattere con valore ottale 0n (0 <= n <= 7)
\0nn	Il carattere con valore ottale 0nn (0 <= n <= 7)
\0mnn	Il carattere con valore ottale 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	Il carattere con valore esadecimale 0xhh
\uhhhh	Il carattere con valore esadecimale 0xhhhh
\t	Il carattere di tabulazione ('\u0009')
\n	Il carattere di nuova riga (avanzamento riga) ('\u000A')
\r	Il carattere di ritorno a capo ('\u000D')
\f	Il carattere di avanzamento carta ('\u000C')
\a	Il carattere di avviso (campanello) ('\u0007')
\e	Il carattere di escape ('\u001B')

Tabella 1. Corrispondenze di caratteri (Continua)

Caratteri	Corrispondenze
\cx	Il carattere di controllo corrispondente a x

Tabella 2. Corrispondenze di classi di caratteri

Classi di caratteri	Corrispondenze
[abc]	a, b o c (classe semplice)
[^abc]	Qualsiasi carattere, eccetto a, b o c (sottrazione)
[a-zA-Z]	a-z oppure A-Z, incluse (intervallo)
[a-d[m-p]]	a-d oppure m-p (unione). In alternativa è possibile specificare [a-dm-p].
[a-z&&[def]]	a-z + d, e oppure f (intersezione)
[a-z&&[^bc]]	a-z, eccetto b e c (sottrazione). In alternativa è possibile specificare [ad-z].
[a-z&&[^m-p]]	a-z, eccetto m-p (sottrazione). In alternativa è possibile specificare [a-lq-z].

Tabella 3. Classi di caratteri predefinite

Classi di caratteri predefinite	Corrispondenze
.	Qualsiasi carattere (può corrispondere o meno a terminazioni di riga)
\d	Qualsiasi cifra: [0-9]
\D	Un carattere diverso da una cifra: [^0-9]
\s	Un spazio vuoto: [\n\x0B \t \f\r]
\S	Uno spazio non vuoto: [^\s]
\w	Un carattere alfanumerico: [a-zA-Z_0-9]
\W	Un carattere diverso da alfanumerico: [^\w]

Tabella 4. Corrispondenze di limiti

Corrispondenze di limiti	Corrispondenze
^	L'inizio di una riga
$	La fine di una riga
\b	Un limite di parola
\B	Un limite diverso da un limite di parola
\A	L'inizio dell'input
\Z	La fine dell'input ma per la terminazione finale, se disponibile
\z	La fine dell'input

Capitolo 2. Linguaggio di script

Panoramica sul linguaggio di script

La funzione di script per IBM SPSS Modeler consente di creare script che eseguono operazioni sull'interfaccia utente di SPSS Modeler, modificano gli oggetti di output ed eseguono sintassi dei comandi. È possibile eseguire gli script direttamente dall'interno di SPSS Modeler.

Gli script in IBM SPSS Modeler sono scritti nel linguaggio di script Python. L'implementazione di Python basata su Java utilizzata da IBM SPSS Modeler è denominata Jython. Il linguaggio di script dispone delle seguenti funzioni:

- Un formato per i riferimenti a nodi, flussi, progetti, output e altri oggetti IBM SPSS Modeler.
- Un insieme di istruzioni o comandi di script che può essere utilizzato per la manipolazione di questi oggetti.
- Un linguaggio di espressioni script per l'impostazione dei valori di variabili, parametri e altri oggetti.
- Supporto per commenti, continuazioni e blocchi di testo letterale.

Le sezioni riportate di seguito descrivono il linguaggio di script Python, l'implementazione Jython di Python e la sintassi di base per iniziare ad utilizzare gli script all'interno di IBM SPSS Modeler. Le sezioni che seguono contengono informazioni su proprietà e comandi specifici.

Python e Jython

Jython è un'implementazione del linguaggio di script Python, scritto nel linguaggio Java ed integrato con la piattaforma Java. Python è un potente linguaggio di script orientato agli oggetti. Jython è utile perché fornisce le funzioni di produttività di un solido linguaggio di script e, a differenza di Python, viene eseguito in un ambiente che supporta una JVM (Java virtual machine). Ciò significa che le librerie Java sulla JVM sono disponibili per l'utilizzo durante la scrittura di programmi. Con Jython, è possibile sfruttare questa differenza ed utilizzare la sintassi e la maggior parte delle funzioni del linguaggio Python

Come linguaggio di script, Python (e la relativa implementazione Jython) è semplice da apprendere ed efficace da codificare e dispone della struttura minima richiesta per creare un programma in esecuzione. Il codice può essere immesso in m,odo interattivo, vale a dire una riga alla volta. Python è un linguaggio di script interpretato; non è disponibile alcun passo di precompilazione, come in Java. I programmi Python sono semplicemente file di testo interpretati al momento dell'input (dopo l'analisi degli errori di sintassi). Le espressioni semplici, come i valori definiti, e le azioni più complesse, come le definizioni di funzioni, sono immediatamente eseguite e disponibili per l'utilizzo. Le modifiche apportate al codice possono essere verificate rapidamente. Tuttavia, l'interpretazione degli script presenta alcuni svantaggi. Ad esempio, l'utilizzo di una variabile non definita non è un errore del compilatore, per cui viene rilevato solo se (e quando) viene eseguita l'istruzione in cui viene utilizzata la variabile. In questo caso, il programma può essere modificato ed eseguito per il debug dell'errore.

Python considera tutti gli elementi, inclusi tutti i dati e tutto il codice, come un oggetto. Pertanto, è possibile modificare tali oggetti con righe di codice. Alcuni tipi di selezione, come numeri e stringhe, vengono considerati come valori e non come oggetti; questa modalità è supportata in Python. È supportato un valore null. Tale valore null ha il nome riservato `None`.

Per un'introduzione più approfondita agli script Python e Jython e per alcuni script di esempio, consultare http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html e http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html.

Script Python

Questa guida al linguaggio di script Python rappresenta un'introduzione ai componenti che hanno maggiori probabilità di essere utilizzati durante la creazione di script in IBM SPSS Modeler, inclusi concetti ed elementi di base della programmazione. Ciò fornirà una serie di informazioni sufficienti per iniziare a sviluppare i propri script Python da utilizzare all'interno di IBM SPSS Modeler.

Operazioni

L'assegnazione viene eseguita utilizzando il simbolo di uguaglianza (=). Ad esempio, per assegnare il valore "3" ad una variabile denominata "x", viene utilizzata la seguente istruzione:

```
x = 3
```

Il simbolo di uguaglianza viene utilizzato anche per assegnare dati di tipo stringa ad una variabile. Ad esempio, per assegnare il valore "a string value" alla variabile "y", viene utilizzata la seguente istruzione:

```
y = "a string value"
```

La tabella riportata di seguito elenca alcune delle operazioni numeriche e di confronto utilizzate frequentemente e le relative descrizioni.

Tabella 5. Operazioni numeriche e di confronto comuni

Operazione	Descrizione
`x < y`	x è minore di y?
`x > y`	x è maggiore di y?
`x <= y`	x è minore o uguale a y?
`x >= y`	x è maggiore o uguale a y?
`x == y`	x è uguale a y?
`x != y`	x non è uguale a y?
`x <> y`	x non è uguale a y?
`x + y`	Aggiungi y a x
`x - y`	Sottrai y da x
`x * y`	Moltiplica x per y
`x / y`	Dividi x per y
`x ** y`	Eleva x alla potenza y

Elenchi

Gli elenchi sono sequenze di elementi. Un elenco può contenere qualsiasi numero di elementi e gli elementi dell'elenco possono essere oggetti di qualsiasi tipo. È possibile pensare agli elenchi anche come ad array. Il numero di elementi in un elenco può aumentare o diminuire man mano che gli elementi vengono aggiunti, rimossi o sostituiti.

Esempi

`[]`	Qualsiasi elenco vuoto.
`[1]`	Un elenco con un elemento singolo, un intero.
`["Mike", 10, "Don", 20]`	Un elenco con quattro elementi, due elementi stringa e due elementi interi.
`[[],[7],[8,9]]`	Un elenco di elenchi. Ciascun elenco secondario è un elenco vuoto oppure un elenco di elementi interi.

```
x = 7; y = 2; z = 3;
[1, x, y, x + y]
```

Un elenco di interi. Questo esempio illustra l'utilizzo di variabili ed espressioni.

È possibile assegnare un elenco ad una variabile; ad esempio:

```
mylist1 = ["one", "two", "three"]
```

È quindi possibile accedere ad elementi specifici dell'elenco, ad esempio:

```
mylist[0]
```

Il risultato sarà l'output riportato di seguito:

```
one
```

Il numero nelle parentesi quadre ([]) è conosciuto come *indice* e fa riferimento ad un particolare elemento dell'elenco. Gli elementi di un elenco sono indicizzati a partire da 0.

È anche possibile selezionare un intervallo di elementi di un elenco; questa operazione è definita *sezionamento*. Ad esempio, `x[1:3]` seleziona il secondo e terzo elemento di `x`. L'indice finale è un'unità dopo la selezione.

Stringhe

Una *stringa* è una sequenza immutabile di caratteri considerata come un valore. Le stringhe supportano tutti gli operatori e le funzioni di sequenza che risultano in una nuova stringa. Ad esempio, `"abcdef"[1:4]` ha come risultato l'output `"bcd"`.

In Python, i caratteri sono rappresentati da stringhe di lunghezza uno.

I letterali di stringa sono definiti mediante l'utilizzo di tripli o singoli apici. Le stringhe definite utilizzando apici singoli non possono essere suddivise su più righe, al contrario delle stringhe definite utilizzando tripli apici. Una stringa può essere racchiusa tra apici singoli (') o doppi ("). Un carattere di quotatura può contenere l'altro carattere di quotatura senza carattere di escape o il carattere di quotatura con carattere di escape, preceduto dal carattere barra retroversa (\).

Esempi

```
"This is a string"
'This is also a string'
"It's a string"
'This book is called "Python Scripting and Automation Guide".'
"This is an escape quote (\") in a quoted string"
```

Più stringhe separate da spazi vengono automaticamente concatenate dal parser Python. In questo modo è più semplice immettere stringhe estese e combinare tipi di apici in una singola stringa, come ad esempio:

```
"This string uses ' and " 'that string uses ".'
```

Si ottiene il seguente risultato:

```
This string uses ' and that string uses ".
```

Le stringhe supportano diversi metodi utili. Alcuni di tali metodi sono indicati nella tabella riportata di seguito.

Tabella 6. Metodi stringa

Metodo	Utilizzo
`s.capitalize()`	Lettera maiuscola iniziale `s`

Tabella 6. Metodi stringa (Continua)

Metodo	Utilizzo
`s.count(ss {,start {,end}})`	Conta le ricorrenze di `ss` in `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Verifica se `s` inizia con `str` Verifica se `s` termina con `str`
`s.expandtabs({size})`	Sostituisce le tabulazioni con gli spazi; il valore `size` predefinito è 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Individua il primo indice di `str` in `s`; se non trovato, il risultato è -1. `rfind` esegue la ricerca da destra a sinistra.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Trova il primo indice di `str` in `s`; se non trovato, genera `ValueError`. `rindex` esegue la ricerca da destra a sinistra.
`s.isalnum`	Verifica se la stringa è alfanumerica
`s.isalpha`	Verifica se la stringa è alfabetica
`s.isnum`	Verifica se la stringa è numerica
`s.isupper`	Verifica se la stringa contiene tutte lettere maiuscole.
`s.islower`	Verifica se la stringa contiene tutte lettere minuscole
`s.isspace`	Verifica se la stringa contiene tutti spazi vuoti.
`s.istitle`	Verifica se la stringa è una sequenza di stringhe alfanumeriche con lettera maiuscola iniziale

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Converte in lettere minuscole
	Converte in lettere maiuscole
	Converte nel caso opposto
	Converte in caratteri del titolo

`s.join(seq)`	Unisce le stringhe in `seq` con `s` come separatore
`s.splitlines({keep})`	Suddivide `s` in righe, se keep è `true`, conserva la nuove righe
`s.split({sep {, max}})`	Suddivide `s` in "parole" utilizzando `sep` (il valore predefinito `sep` è uno spazio) per un massimo di `max` volte

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Giustifica la stringa a sinistra in un campo di larghezza `width`
	Giustifica la stringa a destra in un campo di larghezza `width`
	Giustifica la stringa al centro in un campo di larghezza `width`
	Per il riempimento, viene utilizzato 0.

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Rimuove lo spazio vuoto iniziale
	Rimuove le spazio vuoto finale
	Rimuove lo spazio vuoto iniziale e finale

`s.translate(str {,delc})`	Converte `s` utilizzando la tabella, dopo aver rimosso tutti i caratteri in `delc`. `str` deve essere una stringa con lunghezza `== 256`.
`s.replace(old, new {, max})`	Sostituisce tutte o le `max` ricorrenze della stringa `old` con la stringa `new`

Contrassegni

I contrassegni sono commenti introdotti dal carattere cancelletto (#). Tutto il testo che segue il carattere cancelletto sulla stessa riga viene considerato come parte del contrassegno e viene ignorato. Un contrassegno può iniziare in qualsiasi colonna. L'esempio riportato di seguito illustra l'utilizzo dei contrassegni:

```
#The HelloWorld application is one of the most simple
print 'Hello World' # print the Hello World line
```

Sintassi delle istruzioni

La sintassi delle istruzioni in Python è molto semplice. In generale, ciascuna riga di origine è una singola istruzione. Ad eccezione delle istruzioni `expression` e `assignment`, ciascuna istruzione è introdotta da una parola chiave, come `if` o `for`. È possibile inserire righe vuote o di contrassegno un qualsiasi punto tra le istruzioni nel codice. Se una riga contiene più di una istruzione, ciascuna istruzione deve essere separata mediante un punto e virgola (`;`).

Le istruzioni molto lunghe possono continuare su più righe. In questo caso, l'istruzione che deve continuare alla riga successiva deve terminare con una barra retroversa (\), ad esempio:

```
x = "A looooooooooooooooooong string" + \
     "another looooooooooooooooooong string"
```

Quando una struttura è racchiusa tra parentesi tonde (`()`), quadre (`[]`) o graffe (`{}`), l'istruzione può continuare su una riga successiva dopo qualsiasi virgola, senza che sia necessario inserire una barra retroversa; ad esempio:

```
x = (1, 2, 3, "hello",
      "goodbye", 4, 5, 6)
```

Identificativi

Gli identificativi vengono utilizzati per assegnare nomi a variabili, funzioni, classi e parole chiave. Gli identificativi possono avere qualsiasi lunghezza, ma devono iniziare con un carattere alfabetico maiuscolo o minuscolo o con il carattere di sottolineatura (_). I nomi che iniziano con il carattere di sottolineatura sono generalmente riservati a nomi interni o privati. Dopo il primo carattere, l'identificativo può contenere qualsiasi numero e combinazione di caratteri alfabetici, numeri da 0 a 9 ed il carattere di sottolineatura.

In Jython, alcune parole riservate non possono essere utilizzate per assegnare nomi a variabili, funzioni o classi. Tali parole chiave sono suddivise nelle seguenti categorie:

- **Parole che introducono istruzioni:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` e `while`
- **Parole che introducono parametri:** `as`, `import` e `in`
- **Operatori:** `and`, `in`, `is`, `lambda`, `not` e `or`

Generalmente, un utilizzo non appropriato delle parole chiave determina un errore `SyntaxError`.

Blocchi di codice

I blocchi di codice sono gruppi di istruzioni utilizzati in punti in cui sono previste istruzioni singole. I blocchi di codice possono seguire tutte le istruzioni riportate di seguito: `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` e `class`. Tali istruzioni introducono il blocco di codice con il carattere due punti (`:`), ad esempio:

```
if x == 1:
    y = 2
    z = 3
elif:
    y = 4
    z = 5
```

Per delimitare i blocchi di codice, viene utilizzato il rientro (invece delle parentesi graffe utilizzate in Java). Tutte le righe in un blocco devono essere rientrate alla stessa posizione. Questo perché una modifica del rientro indica la fine di un blocco di codice. Generalmente, vengono utilizzati rientri di quattro spazi per ciascun livello. Per il rientro delle righe, si consiglia di utilizzare gli spazi invece delle tabulazioni. Gli spazi e le tabulazioni non devono essere utilizzati contemporaneamente. Le righe nel blocco più esterno di un modulo devono iniziare alla colonna uno; in caso contrario, si verifica un errore `SyntaxError`.

Le istruzioni che costituiscono un blocco di codice (e seguono i due punti) possono essere anche su una riga singola, separate da punto e virgola, ad esempio:

```
if x == 1: y = 2; z = 3;
```

Passaggio di argomenti ad uno script

Il passaggio di argomenti ad uno script è utile perché significa che è possibile utilizzare più volte uno script senza modifiche. Gli argomenti passati sulla riga comandi vengono passati come valori nell'elenco `sys.argv`. È possibile ottenere il numero di valori passati utilizzando il comando `len(sys.argv)`. Ad esempio:

```
import sys
print "test1"
print sys.argv[0]
print sys.argv[1]
print len(sys.argv)
```

In questo esempio, il comando `import` importa l'intera classe `sys`, in modo che sia possibile utilizzare i metodi esistenti per tale classe, come, ad esempio, `argv`.

Lo script in questo esempio può essere richiamato utilizzando la riga riportata di seguito:

```
/u/mjloos/test1 mike don
```

Il risultato è il seguente output:

```
/u/mjloos/test1 mike don
test1
mike
don
3
```

Esempi

La parola chiave `print` stampa gli argomenti immediatamente successivi. Se l'istruzione è seguita da una virgola, nell'output non viene inclusa una nuova riga. Ad esempio:

```
print "This demonstrates the use of a",
print " comma at the end of a print statement."
```

Il risultato sarà l'output riportato di seguito:

```
This demonstrates the use of a comma at the end of a print statement.
```

L'istruzione `for` viene utilizzata per l'iterazione attraverso un blocco di codice. Ad esempio:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    print lv
    continue
```

In questo esempio, tre stringhe vengono assegnate all'elenco `mylist1`. Gli elementi dell'elenco vengono quindi stampati, con un elemento di ciascuna riga. Il risultato sarà l'output riportato di seguito:

```
one
two
three
```

In questo esempio, l'iteratore `lv` prende il valore di ciascun elemento nell'elenco `mylist1` man mano che il loop for implementa il blocco di codice per ciascun elemento. Un iteratore può essere qualsiasi identificativo valido di qualsiasi lunghezza.

L'istruzione `if` è un'istruzione condizionale. Valuta la condizione e restituisce true o false, in base al risultato della valutazione. Ad esempio:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    if lv == "two"
        print "The value of lv is ", lv
    else
        print "The value of lv is not two, but ", lv
    continue
```

In questo esempio, viene valutato il valore dell'iteratore `lv`. Se il valore di `lv` è `two`, viene restituita una stringa differente da quella restituita se il valore di `lv` è diverso da `two`. Si ottiene il seguente risultato:

```
The value of lv is not two, but one
The value of lv is two
The value of lv is not two, but three
```

Metodi matematici

Dal modulo `math`, è possibile accedere a utili metodi matematici. Alcuni di tali metodi sono indicati nella tabella riportata di seguito. Se non diversamente specificato, tutti i valori vengono restituiti come float.

Tabella 7. Metodi matematici

Metodo	Utilizzo
`math.ceil(x)`	Restituisce il limite superiore di `x` come float, vale a dire il più piccolo intero maggiore o uguale a `x`
`math.copysign(x, y)`	Restituisce `x` con il segno di `y`. `copysign(1, -0.0)` restituisce `-1`
`math.fabs(x)`	Restituisce il valore assoluto di `x`
`math.factorial(x)`	Restituisce `x` fattoriale. Se `x` è negativo o non è un intero, viene generato un errore `ValueError`.
`math.floor(x)`	Restituisce il limite inferiore di `x` come float, vale a dire l'intero più grande minore o uguale a `x`
`math.frexp(x)`	Restituisce la mantissa (`m`) e l'esponente (`e`) di `x` come coppia (`m, e`). `m` è un float ed `e` è un intero, in modo che `x == m * 2**e` esattamente. Se `x` è zero, restituisce `(0.0, 0)`, in caso contrario `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Restituisce una somma float accurata dei valori in `iterable`
`math.isinf(x)`	Verifica se il float `x` è infinito positivo o negativo
`math.isnan(x)`	Verifica se il float `x` è `NaN` (not a number - non un numero)
`math.ldexp(x, i)`	Restituisce `x * (2**i)`. Questa è la funzione inversa della funzione `frexp`.
`math.modf(x)`	Restituisce le parti intera e frazionaria di `x`. Entrambi i risultati hanno il segno di `x` e sono float.
`math.trunc(x)`	Restituisce il valore `Real x`, troncato ad un `Integral`.
`math.exp(x)`	Restituisce `e**x`
`math.log(x[, base])`	Restituisce il logaritmo di `x` al valore fornito `base`. Se `base` non è specificato, viene restituito il logaritmo naturale di `x`.

Tabella 7. Metodi matematici (Continua)

Metodo	Utilizzo
`math.log1p(x)`	Restituisce il logaritmo naturale di `1+x` (base e)
`math.log10(x)`	Restituisce il logaritmo in base 10 di `x`
`math.pow(x, y)`	Restituisce `x` elevato alla potenza `y`. `pow(1.0, x)` e `pow(x, 0.0)` restituisce sempre 1, anche quando `x` è zero o NaN.
`math.sqrt(x)`	Restituisce la radice quadrata di `x`

Oltre alle funzioni matematiche, sono presenti anche alcuni utili metodi trigonometrici. Tali metodi sono illustrati nella tabella riportata di seguito.

Tabella 8. Metodi trigonometrici

Metodo	Utilizzo
`math.acos(x)`	Restituisce l'arcocoseno di `x` in radianti
`math.asin(x)`	Restituisce l'arcoseno di `x` in radianti
`math.atan(x)`	Restituisce l'arcotangente di `x` in radianti
`math.atan2(y, x)`	Restituisce `atan(y / x)` in radianti.
`math.cos(x)`	Restituisce il coseno di `x` in radianti.
`math.hypot(x, y)`	Restituisce la norma Euclidea `sqrt(x*x + y*y)`. Questa è la lunghezza del vettore dall'origine al punto `(x, y)`.
`math.sin(x)`	Restituisce il seno di `x` in radianti
`math.tan(x)`	Restituisce la tangente di `x` in radianti
`math.degrees(x)`	Converte l'angolo `x` da radianti a gradi
`math.radians(x)`	Converte l'angolo `x` da gradi a radianti
`math.acosh(x)`	Restituisce il coseno iperbolico inverso di `x`
`math.asinh(x)`	Restituisce il seno iperbolico inverso di `x`
`math.atanh(x)`	Restituisce la tangente iperbolica inversa di `x`
`math.cosh(x)`	Restituisce il coseno iperbolico di `x`
`math.sinh(x)`	Restituisce il coseno iperbolico di `x`
`math.tanh(x)`	Restituisce la tangente iperbolica di `x`

Sono disponibili anche due costanti matematiche. Il valore di `math.pi` è la costante matematica pi. Il valore di `math.e` è la costante matematica e.

Utilizzo di caratteri Non-ASCII

Per utilizzare caratteri non-ASCII, Python richiede una codifica esplicita e una decodifica di stringhe in Unicode. In IBM SPSS Modeler, gli script Python si assume che siano codificati in UTF-8, che è la codifica standard Unicode che supporta caratteri non-ASCII. Il seguente script verrà compilato poiché il compilatore Python è stato impostato a UTF-8 da SPSS Modeler.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "テストノード", 96, 64)
```

Tuttavia, il nodo risultante avrà un'etichetta errata.

Figura 3. Etichetta del nodo contenente caratteri non-ASCII, visualizzata in modo errato

L'etichetta è errata perché la stessa stringa letterale è stata convertita in una stringa ASCII da Python.

Python consente alle stringhe letterali Unicode di essere specificate aggiungendo un carattere `u` come prefisso prima della stringa letterale:

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", u"テストノード", 96, 64)
```

Ciò creerà una stringa Unicode e l'etichetta verrà visualizzata correttamente.

Figura 4. Etichetta del nodo contenente caratteri non-ASCII, visualizzata correttamente

L'utilizzo di Python e Unicode è un argomento vasto che va oltre l'ambito di questo documento. Sono disponibili molti libri e risorse online che comprendono questo argomento in grande dettaglio.

Programmazione orientata agli oggetti

La programmazione orientata agli oggetti è basata sul concetto di creazione di un modello del problema di destinazione nei propri programmi. La programmazione orientata agli oggetti riduce gli errori di programmazione e favorisce il riutilizzo del codice. Python è un linguaggio orientato agli oggetti. Gli oggetti definiti in Python hanno le seguenti caratteristiche:

- **Identità.** Ciascun oggetto deve essere distinto e verificabile. A questo scopo, sono disponibili i test `is` e `is not`.
- **Stato.** Ciascun oggetto deve essere in grado di memorizzare lo stato. A questo scopo, sono disponibili gli attributi, come i campi e le variabili dell'istanza.
- **Comportamento.** Ciascun oggetto deve essere in grado di modificare il proprio stato. A questo scopo, sono disponibili alcuni metodi.

Python include le seguenti funzioni per il supporto della programmazione orientata agli oggetti:

- **Creazione di oggetti basati su classi.** Le classi sono modelli per la creazione degli oggetti. Gli oggetti sono strutture di dati con un comportamento associato.
- **Ereditarietà con polimorfismo.** Python supporta l'ereditarietà singola e multipla. Tutti i metodi dell'istanza Python sono polimorfici e possono essere sovrascritti dalle classi secondarie.
- **Incapsulamento con dati nascosti.** Python consente di nascondere gli attributi. Quando nascosti, è possibile accedere agli attributi dall'esterno della classe solo attraverso i metodi della classe. Le classi implementano i metodi per modificare i dati.

Definizione di una classe

All'interno di una classe Python, è possibile definire variabili e metodi. A differenza di Java, in Python è possibile definire qualsiasi numero di classi pubbliche per file di origine (o *modulo*). Quindi, un modulo in Python può essere considerato simile ad un package in Java.

In Python, le classi sono definite utilizzando l'istruzione `class`. Di seguito è riportato il formato dell'istruzione `class`:

```
class name (superclasses): statement
```

o

```
class name (superclasses):
    assignment
    .
    .
    function
    .
    .
```

Quando viene definita una classe, è possibile fornire zero o più istruzioni *assignment*. Tali istruzioni creano attributi della classe condivisi da tutte le istanze della classe. È anche possibile fornire zero o più definizioni *function*. Tali definizioni di funzione creano metodi. L'elenco delle superclassi è facoltativo.

Il nome della classe deve essere univoco nello stesso ambito, vale a dire all'interno di un modulo, una funzione o una classe. È possibile definire più variabili per fare riferimento alla stessa classe.

Creazione di un'istanza della classe

Le classi vengono utilizzate per conservare gli attributi della classe (o condivisi) oppure per creare istanze della classe. Per creare un'istanza di una classe, è possibile richiamare la classe come se fosse una funzione. Ad esempio, considerare la classe riportata di seguito:

```
class MyClass:
    pass
```

In questo caso, viene utilizzata l'istruzione `pass` perché è necessaria un'istruzione per completare la classe, ma non è richiesta alcuna azione in modo programmatico.

L'istruzione riportata di seguito crea un'istanza della classe `MyClass`:

```
x = MyClass()
```

Aggiunta di attributi ad un'istanza della classe

A differenza di Java, in Python i client possono aggiungere attributi ad un'istanza di una classe. Viene modificata solo quella particolare istanza. Ad esempio, per aggiungere attributi ad un'istanza `x`, impostare nuovi valori su tale istanza:

```
x.attr1 = 1
x.attr2 = 2
    .
    .
x.attrN = n
```

Definizione dei metodi e degli attributi della classe

Qualsiasi variabile collegata in una classe è un *attributo della classe*. Qualsiasi funzione definita all'interno di una classe è un *metodo*. I metodi ricevono un'istanza della classe, denominata per convenzione `self`, come primo argomento. Ad esempio, per definire alcuni metodi ed attributi della classe, è possibile utilizzare il codice riportato di seguito:

```
class MyClass
    attr1 = 10        #class attributes
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #reference the class attribute

    def method2(self):
        print MyClass.attr2   #reference the class attribute

    def method3(self, text):
        self.text = text        #instance attribute
        print text, self.text   #print my argument and my attribute

    method4 = method3   #make an alias for method3
```

All'interno di una classe, è necessario qualificare tutti i riferimenti agli attributi della classe con il nome della classe; ad esempio, `MyClass.attr1`. Tutti i riferimenti agli attributi dell'istanza devono essere qualificati con la variabile `self`; ad esempio, `self.text`. All'esterno della classe, è necessario qualificare tutti i riferimenti agli attributi della classe con il nome della classe (ad esempio, `MyClass.attr1`) oppure con un'istanza della classe (ad esempio, `x.attr1`, dove `x` è un'istanza della classe). All'esterno della classe, tutti i riferimenti alle variabili dell'istanza devono essere qualificati con un'istanza della classe; ad esempio, `x.text`.

Variabili nascoste

È possibile nascondere i dati creando variabili *Private*. Alle variabili Private può accedere solo la classe stessa. Se vengono dichiarati nomi nel formato `__xxx` o `__xxx_yyy`, vale a dire con due caratteri di sottolineatura iniziali, il programma di analisi Python aggiunge automaticamente il nome della classe al nome dichiarato, creando variabili nascoste; ad esempio:

```
class MyClass:
    __attr = 10   #private class attribute

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #private attribute
```

A differenza di Java, in Python tutti i riferimenti alle variabili dell'istanza devono essere qualificati con `self`; non è previsto l'utilizzo implicito di `this`.

Ereditarietà

La possibilità di ereditare dalle classi è fondamentale per la programmazione orientata ad oggetti. Python supporta l'ereditarietà singola e multipla. Il termine *ereditarietà singola* indica che può esistere una sola superclasse. Il termine *ereditarietà multipla* indica che può essere più di una superclasse.

L'ereditarietà viene implementata inserendo altre classi come sottoclassi. Qualsiasi numero di classi Python possono essere superclassi. Nell'implementazione Jython di Python, è possibile ereditare direttamente o indirettamente solo da una classe Java. Non è necessario fornire una superclasse.

Qualsiasi attributo o metodo in una superclasse è anche in qualsiasi sottoclasse e può essere utilizzato dalla classe stessa o da qualsiasi client, purché l'attributo o il metodo non sia nascosto. Qualsiasi istanza di una sottoclasse può essere utilizzata in qualsiasi punto in cui può essere utilizzata un'istanza di una superclasse; questo è un esempio di *polimorfismo*. Tali funzioni abilitano il riutilizzo e la semplicità di estensione.

Esempio

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

class Class4(Class3, Class2): pass     #multiple inheritance
```

Capitolo 3. Script in IBM SPSS Modeler

Tipi di script

In IBM SPSS Modeler sono disponibili tre tipi di script:

- Gli *script del flusso* sono utilizzati per controllare l'esecuzione di un singolo flusso e sono archiviati all'interno del flusso.
- Gli *script del Supernodo* vengono utilizzati per controllare il funzionamento dei supernodi.
- Gli *script autonomi o di sessione* possono essere utilizzati per coordinare l'esecuzione attraverso un numero di flussi differenti.

Sono disponibili diversi metodi che possono essere utilizzati negli script in IBM SPSS Modeler con cui è possibile accedere ad una vasta gamma di funzionalità di SPSS Modeler. Tali metodi sono utilizzati anche in Capitolo 4, "API di script", a pagina 37 per creare funzioni più avanzate.

Flussi, flussi SuperNodo e diagrammi

Il più delle volte il termine *flusso* significa la stessa cosa, indipendentemente se si tratta di un flusso caricato da un file o utilizzato all'interno di un SuperNodo. Generalmente indica un insieme di nodi che sono connessi insieme e possono essere eseguiti. Negli script, comunque, non tutte le operazioni sono supportate i tutti i posti, vale a dire che un autore di uno script dovrebbe essere consapevole di quale variante di flusso si sta utilizzando.

Flussi

Un flusso è il tipo di documento principale di IBM SPSS Modeler. Può essere salvato, caricato, modificato edeseguito. I flussi possono anche avere parametri, valori globali, uno script ed altre informazioni ad essi associati.

Flussi SuperNodo

Un *flusso supernodo* è un tipo di flusso utilizzato all'interno di un supernodo. Come un flusso normale, contiene i nodi che sono collegati tra di loro. I flussi di supernodo hanno una serie di differenze rispetto ad un normale flusso:

- I parametri ed ogni script sono associati con il supernodo proprietario del flusso del supernodo, piuttosto che con il flusso del supernodo stesso.
- I flussi di supernodo hanno dei nodi connettori aggiuntivi di input ed output, a seconda del tipo di supernodo. Questi nodi connettori sono utilizzati per passare le informazioni in entrata ed in uscita al flusso del supernodo e vengono automaticamente creati quando viene creato il supernodo stesso.

Diagrammi

Il termine *diagramma* comprende le funzioni che sono supportate sia dai flussi normali che dai flussi SuperNodo, come aggiungere e rimuovere nodi e modificare la connessioni tra i nodi.

Esecuzione di un flusso

L'esempio riportato di seguito esegue tutti i nodi eseguibili nel flusso e rappresenta il tipo più semplice di script del flusso:

```
modeler.script.stream().runAll(None)
```

L'esempio riportato di seguito, inoltre, esegue tutti i nodi eseguibili nel flusso:

```
stream = modeler.script.stream()
stream.runAll(None)
```

In questo esempio, il flusso è memorizzato in una variabile denominata `stream`. L'archiviazione del flusso in una variabile è utile perché uno script viene generalmente utilizzato per modificare il flusso o i nodi all'interno di un flusso. La creazione di una variabile che memorizza il flusso ha come risultato uno script più breve.

Contesto di script

Il modulo `modeler.script` fornisce il contesto in cui viene eseguito uno script. Il modulo viene importato automaticamente in uno script SPSS Modeler al runtime. Il modulo definisce quattro funzioni che forniscono ad uno script l'accesso al proprio ambiente di esecuzione:

- La funzione `session()` restituisce la sessione per lo script. La sessione definisce informazioni come la locale ed il backend the SPSS Modeler (un processo locale o un SPSS Modeler Server di rete) utilizzati per l'esecuzione dei flussi.
- La funzione `stream()` può essere utilizzata con script di supernodi e flussi. Questa funzione restituisce il flusso proprietario dello script del flusso o dello script del supernodo eseguito.
- La funzione `diagram()` può essere utilizzata con gli script del supernodo. Questa funzione restituisce il diagramma all'interno del supernodo. Per gli altri tipi di script, questa funzione restituisce lo stesso risultato della funzione `stream()`.
- La funzione `supernode()` può essere utilizzata con gli script del supernodo. Questa funzione restituisce il supernodo proprietario dello script che viene eseguito.

Le quattro funzioni ed i relativi output sono riepilogati nella tabella riportata di seguito.

Tabella 9. Riepilogo delle funzioni di `modeler.script`

Tipo di script	`session()`	`stream()`	`diagram()`	`supernode()`
Autonomo	Restituisce una sessione	Restituisce il flusso gestito corrente nel momento in cui è stato richiamato lo script (ad esempio, il flusso passato mediante l'opzione `-stream` della modalità batch) oppure `None`.	Come `stream()`	Non applicabile
Flusso	Restituisce una sessione	Restituisce un flusso	Come `stream()`	Non applicabile
Supernodo	Restituisce una sessione	Restituisce un flusso	Restituisce un flusso Supernodo	Restituisce un supernodo

Il modulo `modeler.script`, inoltre, definisce un modo per terminare lo script con un codice di uscita. La funzione `exit(exit-code)` arresta l'esecuzione dello script e restituisce il codice di uscita intero fornito.

Uno dei metodi definiti per un flusso è `runAll(List)`. Questo metodo esegue tutti i nodi eseguibili. Qualsiasi modello o output generato mediante l'esecuzione dei nodi viene aggiunto all'elenco fornito.

È comune per un'esecuzione di flusso generare output come modelli, grafici ed altro output. Per catturare tale output, uno script può fornire una variabile inizializzata in un un elenco, ad esempio:

```
stream = modeler.script.stream()
results = []
stream.runAll(results)
```

Una volta completata l'esecuzione, è possibile accedere a tutti gli oggetti generati dall'esecuzione dall'elenco `results`.

Riferimento a nodi esistenti

Spesso un flusso già dispone di alcuni parametri che è necessario modificare prima che il flusso venga eseguito. La modifica di tali parametri implica le attività riportate di seguito:

1. Individuazione dei nodi nel relativo flusso.
2. Modifica delle impostazioni del nodo o del flusso (o di entrambi).

Ricerca di nodi

I flussi forniscono diversi modi per ricercare un nodo esistente. Tali metodi sono riepilogati nella tabella riportata di seguito.

Tabella 10. Metodi per la ricerca di un nodo esistente

Metodo	Tipo di restituzione	Descrizione
`s.findAll(type, label)`	Raccolta	Restituisce un elenco di tutti i nodi con il tipo e l'etichetta specificati. Il tipo o l'etichetta possono essere `None`; in questo caso, viene utilizzato l'altro parametro.
`s.findAll(filter, recursive)`	Raccolta	Restituisce una raccolta di tutti i nodi accettati dal filtro specificato. Se l'indicatore recursive è `True`, viene eseguita la ricerca anche nei supernodi all'interno del flusso specificato.
`s.findByID(id)`	Nodo	Restituisce il nodo con l'ID fornito oppure `None` se non esiste alcun nodo di questo tipo. La ricerca è limitata al flusso corrente.
`s.findByType(type, label)`	Nodo	Restituisce il nodo con il tipo e/o l'etichetta forniti. Il tipo o il nome può essere `None`; in questo caso viene utilizzato l'altro parametro. Se si verifica una corrispondenza per più nodi, viene scelto e restituito un nodo arbitrario. Se non si verifica alcuna corrispondenza, il valore di restituzione è `None`.
`s.findDownstream(fromNodes)`	Raccolta	Ricerca dall'elenco di nodi fornito e restituisce l'insieme di nodi downstream dei nodi forniti. L'elenco restituito include i nodi forniti originariamente.
`s.findUpstream(fromNodes)`	Raccolta	Ricerca dall'elenco di nodi fornito e restituisce l'insieme dei nodi upstream dei nodi forniti. L'elenco restituito include i nodi forniti originariamente.

Ad esempio, se un flusso contiene un singolo nodo Filter per cui lo script richiede l'accesso, è possibile trovare il nodo Filter utilizzando lo script riportato di seguito:

```
stream = modeler.script.stream()
node = stream.findByType("filter", None)
...
```

In alternativa, per trovare il nodo è possibile utilizzare l'ID del nodo (visualizzato nella scheda Annotazioni della finestra di dialogo del nodo), se noto; ad esempio:

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # the filter node ID
...
```

Impostazione delle proprietà

I nodi, i flussi, i modelli e gli output dispongono di proprietà a cui è possibile accedere e che, nella maggior parte dei casi, è possibile impostare. Generalmente, le proprietà sono utilizzate per modificare il funzionamento o l'aspetto dell'oggetto. I metodi disponibili per l'accesso e l'impostazione delle proprietà dell'oggetto sono riepilogati nella tabella riportata di seguito.

Tabella 11. Metodi per l'accesso e l'impostazione delle proprietà dell'oggetto

Metodo	Tipo di restituzione	Descrizione
`p.getPropertyValue(propertyName)`	Oggetto	Restituisce il valore della proprietà indicata o `None` se non esiste alcuna proprietà di questo tipo.
`p.setPropertyValue(propertyName, value)`	Non applicabile	Imposta il valore della proprietà indicata.
`p.setPropertyValues(properties)`	Non applicabile	Imposta i valori delle proprietà indicate. Ciascuna voce nella mappa delle proprietà è composta da una chiave che rappresenta il nome della proprietà e dal valore che deve essere assegnato a tale proprietà.
`p.getKeyedPropertyValue( propertyName, keyName)`	Oggetto	Restituisce il valore della proprietà denominata e la chiave associata o `None` se non esistono una proprietà o una chiave di questo tipo.
`p.setKeyedPropertyValue( propertyName, keyName, value)`	Non applicabile	Imposta il valore delle proprietà indicata e della chiave.

Ad esempio, se si desidera impostare il valore di un nodo Variable File all'inizio di un flusso, è possibile utilizzare il seguente script:

```
stream = modeler.script.stream()
node = stream.findByType("variablefile", None)
node.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
...
```

In alternativa, è possibile che si desideri filtrare un campo da un nodo Filter. In questo caso, il valore è anche associato con chiave al nome del campo; ad esempio:

```
stream = modeler.script.stream()
# Locate the filter node ...
node = stream.findByType("filter", None)
# ... and filter out the "Na" field
node.setKeyedPropertyValue("include", "Na", False)
```

Creazione di nodi e modifica dei flussi

In alcune situazioni, è possibile che si desideri aggiungere nuovi nodi ai flussi esistenti. L'aggiunta di nodi ai flussi esistenti generalmente implica le seguenti attività:

1. Creazione dei nodi.
2. Collegamento dei nodi al flusso esistente.

Creazione di nodi

I flussi forniscono diversi modi per creare i nodi. Tali metodi sono riepilogati nella tabella riportata di seguito.

Tabella 12. Metodi per la creazione di nodi

Metodo	Tipo di restituzione	Descrizione
`s.create(nodeType, name)`	Nodo	Crea un nodo del tipo specificato e lo aggiunge al flusso specificato.
`s.createAt(nodeType, name, x, y)`	Nodo	Crea un nodo del tipo specificato e lo aggiunge al flusso specificato nel percorso specificato. Se x < 0 oppure y < 0, il percorso non viene impostato.
`s.createModelApplier(modelOutput, name)`	Nodo	Crea un nodo applicatore del modello derivato dall'oggetto di output del modello fornito.

Ad esempio, per creare un nuovo nodo Type in un flusso, è possibile utilizzare lo script riportato di seguito:

```
stream = modeler.script.stream()
# Create a new type node
node = stream.create("type", "My Type")
```

Collegamento e scollegamento di nodi

Quando un nuovo nodo viene creato all'interno di un flusso, deve essere connesso in una sequenza di nodi prima di poter essere utilizzato. I flussi forniscono diversi metodi per collegare e scollegare i nodi. Tali metodi sono riepilogati nella tabella riportata di seguito.

Tabella 13. Metodi per il collegamento e lo scollegamento dei nodi

Metodo	Tipo di restituzione	Descrizione
`s.link(source, target)`	Non applicabile	Crea un nuovo collegamento tra i nodi di origine e di destinazione.
`s.link(source, targets)`	Non applicabile	Crea nuovi collegamenti tra il nodo di origine e ciascun nodo di destinazione nell'elenco fornito.
`s.linkBetween(inserted, source, target)`	Non applicabile	Connette un nodo tra due altre istanze del nodo (i nodi di origine e destinazione) ed imposta la posizione del nodo inserito tra di essi. Qualsiasi collegamento diretto tra i nodi di origine e destinazione viene rimosso prima.
`s.linkPath(path)`	Non applicabile	Crea un nuovo percorso tra le istanze del nodo. Il primo nodo viene collegato al secondo, il secondo viene collegato al terzo e così via.

Tabella 13. Metodi per il collegamento e lo scollegamento dei nodi (Continua)

Metodo	Tipo di restituzione	Descrizione
`s.unlink(source, target)`	Non applicabile	Rimuove qualsiasi collegamento diretto tra i nodi di origine e di destinazione.
`s.unlink(source, targets)`	Non applicabile	Rimuove i collegamenti diretti tra il nodo di origine e ciascun oggetto nell'elenco delle destinazioni.
`s.unlinkPath(path)`	Non applicabile	Rimuove qualsiasi percorso esistente tra le istanze del nodo.
`s.disconnect(node)`	Non applicabile	Rimuove qualsiasi collegamento tra il nodo fornito e qualsiasi altro nodo nel flusso specificato.
`s.isValidLink(source, target)`	*booleano*	Restituisce `True` se è valido creare un collegamento tra l'origine specificata ed i nodi di destinazione. Questo metodo verifica che entrambi gli oggetti appartengano al flusso specificato, che il nodo di origine possa fornire un collegamento e che il nodo di destinazione possa ricevere un collegamento, e che la creazione di un collegamento di questo tipo non causi circolarità nel flusso.

Lo script di esempio riportato di seguito esegue queste cinque attività:

1. Crea un nodo di input Variable File, un nodo Filter ed un nodo di output Table.
2. Connette i nodi tra loro.
3. Imposta il nome del file sul nodo di input Variable File.
4. Filtra il campo "Drug" dall'output risultante.
5. Esegue il nodo Table.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "My File Input ", 96, 64)
filternode = stream.createAt("filter", "Filter", 192, 64)
tablenode = stream.createAt("table", "Table", 288, 64)
stream.link(filenode, filternode)
stream.link(filternode, tablenode)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
filternode.setKeyedPropertyValue("include", "Drug", False)
results = []
tablenode.run(results)
```

Importazione, sostituzione ed eliminazione di nodi

Oltre alla creazione ed alla connessione dei nodi, è spesso necessario sostituire ed eliminare nodi dal flusso. I metodi disponibili per l'importazione, la sostituzione e l'eliminazione dei nodi sono riepilogati nella seguente tabella.

Tabella 14. Metodi per l'importazione, la sostituzione e l'eliminazione dei nodi

Metodo	Tipo di restituzione	Descrizione
`s.replace(originalNode, replacementNode, discardOriginal)`	Non applicabile	Sostituisce il nodo specificato dal flusso specificato. Il nodo originale ed il nodo sostitutivo devono essere di proprietà del flusso specificato.

Tabella 14. Metodi per l'importazione, la sostituzione e l'eliminazione dei nodi (Continua)

Metodo	Tipo di restituzione	Descrizione
`s.insert(source, nodes, newIDs)`	Elenco	Inserisce copie dei nodi nell'elenco fornito. Si suppone che tutti i nodi nell'elenco fornito siano contenuti nel flusso specificato. L'indicatore `newIDs` indica se è necessario generare nuovi ID per ciascun nodo o se è necessario copiare ed utilizzare l'ID esistente. Si suppone che tutti i nodi in un flusso dispongano di un ID univoco, per cui questo indicatore deve essere impostato su `True` se il flusso di origine è uguale al flusso specificato. Il metodo restituisce l'elenco dei nodi appena inseriti, in cui l'ordine dei nodi è indefinito (l'ordinamento non è necessariamente uguale all'ordine dei nodi nell'elenco di input).
`s.delete(node)`	Non applicabile	Elimina il nodo specificato dal flusso specificato. Il nodo deve essere di proprietà del flusso specificato.
`s.deleteAll(nodes)`	Non applicabile	Elimina tutti i nodi specificati dal flusso specificato. Tutti i nodi nella raccolta devono appartenere al flusso specificato.
`s.clear()`	Non applicabile	Elimina tutti i nodi dal flusso specificato.

Attraversamento dei nodi in un flusso

Un requisito comune è quello di identificare i nodi upstream o downstream di un particolare nodo. Il flusso fornisce diversi metodi che è possibile utilizzare per identificare tali nodi. Tali metodi sono riepilogati nella tabella riportata di seguito.

Tabella 15. Metodi per identificare i nodi upstream e downstream

Metodo	Tipo di restituzione	Descrizione
`s.iterator()`	Iteratore	Restituisce un iteratore sugli oggetti del nodo contenuti nel flusso specificato. Se il flusso viene modificato tra le chiamate della funzione `next()`, il funzionamento dell'iteratore non è definito.
`s.predecessorAt(node, index)`	Nodo	Restituisce il predecessore immediato specificato del nodo fornito oppure `None` se l'indice non è compreso nei limiti.
`s.predecessorCount(node)`	*int*	Restituisce il numero di predecessori immediati del nodo fornito.
`s.predecessors(node)`	Elenco	Restituisce i predecessori immediati del nodo fornito.
`s.successorAt(node, index)`	Nodo	Restituisce il successore immediato specificato del nodo fornito oppure `None` se l'indice non è compreso nei limiti.

Tabella 15. Metodi per identificare i nodi upstream e downstream (Continua)

Metodo	Tipo di restituzione	Descrizione
`s.successorCount(node)`	*int*	Restituisce il numero di successori immediati del nodo fornito.
`s.successors(node)`	Elenco	Restituisce i successori immediati del nodo fornito.

Cancellazione o rimozione di elementi

Gli script legacy supportano diversi utilizzi del comando `clear`, ad esempio:

- `clear outputs` Per eliminare tutti gli elementi dell'output dalla tavolozza dei manager.
- `clear generated palette` Per eliminare tutti i nugget del modello dalla tavolozza dei modelli.
- `clear stream` Per rimuovere il contenuto di un flusso.

Gli script Python supportano un insieme simile di funzioni; il comando `removeAll()` viene utilizzato per cancellare i manager Flussi, Output e Modelli. Ad esempio:

- Per cancellare il manager Flussi:

```
session = modeler.script.session()
session.getStreamManager.removeAll()
```

- Per cancellare il manager Output:

```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```

- Per cancellare il manager Modelli:

```
session = modeler.script.session()
session.getModelOutputManager().removeAll()
```

Acquisizione delle informazioni relative ai nodi

I nodi possono essere suddivisi in diverse categorie, come, ad esempio, nodi di importazione ed esportazione dei dati, nodi di creazione dei modelli ed altri tipi di nodi. Ciascun nodo fornisce una serie di metodi che è possibile utilizzare per individuare informazioni relative al nodo.

I metodi che è possibile utilizzare per ottenere l'ID, il nome e l'etichetta di un nodo sono riepilogati nella tabella riportata di seguito.

Tabella 16. Metodi per ottenere l'ID, il nome e l'etichetta di un nodo

Metodo	Tipo di restituzione	Descrizione
`n.getLabel()`	*stringa*	Restituisce l'etichetta di visualizzazione del nodo specificato. L'etichetta è il valore della proprietà `custom_name` solo se tale proprietà è una stringa non vuota e la proprietà `use_custom_name` non è impostata; in caso contrario, l'etichetta è il valore di `getName()`.

Tabella 16. Metodi per ottenere l'ID, il nome e l'etichetta di un nodo (Continua)

Metodo	Tipo di restituzione	Descrizione
`n.setLabel(label)`	Non applicabile	Imposta l'etichetta di visualizzazione del nodo specificato. Se la nuova etichetta è una stringa non vuota viene assegnata alla proprietà `custom_name`, e `False` viene assegnato alla proprietà `use_custom_name` in modo che l'etichetta specificata ha la precedenza; altrimenti una stringa vuota viene assegnata alla proprietà `custom_name` e `True` viene assegnato alla proprietà `use_custom_name`.
`n.getName()`	*stringa*	Restituisce il nome del nodo specificato.
`n.getID()`	*stringa*	Restituisce l'ID del nodo specificato. Viene creato un nuovo ID ogni volta che viene creato un nuovo nodo. L'ID viene reso permanente con il nodo quando viene salvato come parte di un flusso, in modo che quando il flusso viene aperto, gli ID del nodo vengono conservati. Tuttavia, se un nodo salvato viene inserito in un flusso, il nodo inserito viene considerato come un nuovo oggetto e ad esso verrà assegnato un nuovo ID.

I metodi che è possibile utilizzare per ottenere altre informazioni relative ad un nodo sono riepilogati nella seguente tabella.

Tabella 17. Metodi per ottenere informazioni relative ad un nodo

Metodo	Tipo di restituzione	Descrizione
`n.getTypeName()`	*stringa*	Restituisce il nome di script di questo nodo. È lo stesso nome che può essere utilizzato per creare una nuova istanza di questo nodo.
`n.isInitial()`	*Booleano*	Restituisce `True` se si tratta di un nodo *iniziale*, vale a dire un nodo che si verifica all'inizio di un flusso.
`n.isInline()`	*Booleano*	Restituisce `True` se questo è un nodo *in linea*, vale a dire un nodo presente a metà del flusso.
`n.isTerminal()`	*Booleano*	Restituisce `True` se questo è un nodo *terminale*, vale a dire un nodo presente alla fine di un flusso.
`n.getXPosition()`	*int*	Restituisce l'offset della posizione x del nodo nel flusso.
`n.getYPosition()`	*int*	Restituisce l'offset della posizione y del nodo nel flusso.
`n.setXYPosition(x, y)`	Non applicabile	Imposta la posizione del nodo nel flusso.

Tabella 17. Metodi per ottenere informazioni relative ad un nodo (Continua)

Metodo	Tipo di restituzione	Descrizione
`n.setPositionBetween(source, target)`	Non applicabile	Imposta la posizione del nodo nel flusso, in modo che sia posizionato tra i nodi forniti.
`n.isCacheEnabled()`	*Booleano*	Restituisce `True` se la cache è abilitata; in caso contrario, restituisce `False`.
`n.setCacheEnabled(val)`	Non applicabile	Abilita o disabilita la cache per questo oggetto. Se la cache è piena e la memorizzazione nella cache viene disabilitata, la cache viene svuotata.
`n.isCacheFull()`	*Booleano*	Restituisce `True` se la cache è piena; in caso contrario, restituisce `False`.
`n.flushCache()`	Non applicabile	Svuota la cache di questo nodo. Non ha alcun effetto se la cache non è abilitata o non è piena.

Capitolo 4. API di script

Introduzione all'API di script

L'API di script fornisce l'accesso ad una vasta gamma di funzionalità di SPSS Modeler. Tutti i metodi descritti fanno parte dell'API ed è possibile eseguirvi l'accesso in modo implicito all'interno dello script senza ulteriori importazioni. Tuttavia, se si desidera fare riferimento alle classi API, è necessario importare l'API in modo esplicito con la seguente istruzione:

```
import modeler.api
```

Tale istruzione di importazione è richiesta da molti degli esempi di API di script.

Una guida completa relativa alle classi, ai metodi ed ai parametri disponibili nell'API di script è disponibile nel documento *IBM SPSS Modeler Python Scripting API Reference Guide*.

Esempio 1: ricerca di nodi utilizzando un filtro personalizzato

La sezione "Ricerca di nodi" a pagina 29 conteneva un esempio di ricerca di un nodo in un flusso in cui veniva utilizzato il nome del tipo del nodo come criterio di ricerca. In alcune situazioni, è richiesta una ricerca più generica ed è possibile utilizzare la classe `NodeFilter` ed il metodo `findAll()` del flusso. Questo tipo di ricerca implica le due fasi riportate di seguito:

1. Creazione di una nuova classe che estende `NodeFilter` ed implementa una versione personalizzata del metodo `accept()`.
2. Richiamo del metodo `findAll()` del flusso con un'istanza di questa nuova classe. In questo modo vengono restituiti tutti i nodi che soddisfano i criteri definiti nel metodo `accept()`.

L'esempio riportato di seguito illustra come ricercare i nodi in un flusso con la cache del nodo abilitata. L'elenco dei nodi restituito può essere utilizzato per svuotare o disabilitare le cache di tali nodi.

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
 """A node filter for nodes with caching enabled"""
 def accept(this, node):
  return node.isCacheEnabled()

cachingnodes = modeler.script.stream().findAll(CacheFilter(), False)
```

Esempio 2: consente agli utenti di ottenere informazioni sulla directory o sul file in base ai propri privilegi

Per evitare che il PSAPI venga aperto agli utenti, è possibile utilizzare un metodo chiamato `session.getServerFileSystem()` chiamando la funzione PSAPI per creare un oggetto file system.

Il seguente esempio mostra come consentire ad un utente di ottenere informazioni sulla directory o sul file in base ai privilegi dell'utente che si connette al IBM SPSS Modeler Server.

```
import modeler.api
stream = modeler.script.stream()
sourceNode = stream.findByID('')
session = modeler.script.session()
fileSystem = session.getServerFileSystem()
parameter = stream.getParameterValue('VPATH')
serverDirectory = fileSystem.getServerFile(parameter)
files = fileSystem.getFiles(serverDirectory)
for f in files:
```

```
    if f.isDirectory():
        print 'Directory:'
    else:
        print 'File:'
        sourceNode.setPropertyValue('full_filename',f.getPath())
        break
    print f.getName(),f.getPath()
stream.execute()
```

Metadati: informazioni sui dati

Poichè i nodi sono collegati insieme in un flusso, sono disponibili tutte le informazioni relative alle colonne o ai campi che sono disponibili per ogni singolo nodo. Per esempio, nella interfaccia del Modeler, questo permette di selezionare quali campi utilizzare per ordinare o aggregare. Queste informazioni sono chiamate modello dati.

Gli script possono accedere al modello dati esaminando i campi entrano o escono da un nodo. Per alcuni nodi, i modelli dei dati in input ed in output coincidono, per esempio un nodo Odina semplicemente riordina i record ma non cambierà il modello dati. Alcuni, come il nodo Ricava, possono aggiungere nuovi campi. Altri, come il nodo Filtro possono rinominare o rimuovere i campio.

Nell'esempio seguente, lo script prende il flusso standard IBM SPSS Modeler `druglearn.str` e, per ogni campo, costruisce un modello con uno dei campi in input eliminato. Questo viene realizzato così:

1. Accesso al modello dati output dal nodo Tipo.
2. Esecuzione di cicli per ogni campo nel modello dati output.
3. Modificare il nodo Filtro per ogni campo di input.
4. Cambiare il nome del modello che si sta costruendo.
5. Eseguire il nodo di costruzione del modello.

Nota: Prima di eseguire lo script nel flusso `druglean.str`, ricordarsi di impostare il linguaggio dello script su Python (il flusso è stato creato in una versione precedente di IBM SPSS Modeler e quindi il linguaggio di script del flusso è impostato su Legacy).

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Always use a custom model name
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # If this is the target field then ignore it
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Re-enable the field that was most recently removed
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Remove the field
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Set the name of the new model then run the build
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

L'oggetto DataModel fornisce diversi metodi per accedere alle informazioni dei campi o delle colonne all'interno di un certo modello dati. Tali metodi sono riepilogati nella tabella riportata di seguito.

Tabella 18. Metodi dell'oggetto DataModel per accedere alle informazioni relative ai campi o colonne

Metodo	Tipo di restituzione	Descrizione
`d.getColumnCount()`	*int*	Restituisce il numero di colonne nel modello dati.
`d.columnIterator()`	Iteratore	Restituisce un iteratore che restituisce ogni colonna nell'ordine di inserimento "naturale". L?iteratore restituisce istanze di Colonna.
`d.nameIterator()`	Iteratore	Restituisce un iteratore che restituisce il nome di ogni colonna nell'ordine di inserimento "naturale".
`d.contains(name)`	*Booleano*	Restituisce `True` se esiste una colonna con il nome fornito in questo DataModel, `False` altrimenti.
`d.getColumn(name)`	Colonna	Restituisce la colonna con il nome specificato.
`d.getColumnGroup(name)`	ColumnGroup	Restituisce il gruppo di colonne specificato oppure `None` se il gruppo colonna non esiste.
`d.getColumnGroupCount()`	*int*	Restituisce il numero di gruppi colonna in questo modello dati.
`d.columnGroupIterator()`	Iteratore	Restituisce un iteratore che restituisce ogni gruppo colonna.
`d.toArray()`	Column[]	Restituisce il modello dati come un array di colonne. Le colonne vengono ordinate secondo il loro ordine "naturale" di inserimento.

Ogni campo (oggetto colonna) include un numero di metodi per accedere alle informazioni relative alla colonna. La tabella seguente mostra una selezione di questi.

Tabella 19. Metodi dell'oggetto Colonna per accedere alle informazioni relative ad una colonna

Metodo	Tipo di restituzione	Descrizione
`c.getColumnName()`	*stringa*	Restituisce il nome della colonna.
`c.getColumnLabel()`	*stringa*	Restituisce l'etichetta della colonna oppure una stringa vuota se non c'è nessuna etichetta associata con la colonna.
`c.getMeasureType()`	MeasureType	Restituisce il tipo misura per la colonna.
`c.getStorageType()`	StorageType	Restituisce il tipo archiviazione per la colonna.
`c.isMeasureDiscrete()`	*Booleano*	Restituisce `True` se la colonna ha un valore discreto. Colonne che sono un insieme o un indicatore sono considerate discrete.
`c.isModelOutputColumn()`	*Booleano*	Restituisce `True` se la colonna è una colonna di modello di output.

Tabella 19. Metodi dell'oggetto Colonna per accedere alle informazioni relative ad una colonna (Continua)

Metodo	Tipo di restituzione	Descrizione
`c.isStorageDatetime()`	*Booleano*	Restituisce `True` se l'archiviazione della colonna è un valore di tipo ora, data o timestamp.
`c.isStorageNumeric()`	*Booleano*	Restituisce `True` se l'archiviazione della colonna è un intero o un numero reale.
`c.isValidValue(value)`	*Booleano*	Restituisce `True` se il valore specificato è valido per questa archiviazione e `valid` quando i valori validi della colonna sono noti.
`c.getModelingRole()`	ModelingRole	Restituisce il ruolo di modellazione per la colonna.
`c.getSetValues()`	Object[]	Restituisce un array di valori validi per la colonna o `None` se i valori non sono noti o la colonna non è un insieme.
`c.getValueLabel(value)`	*stringa*	Restituisce l'etichetta per il valore nella colonna, oppure una stringa vuota se non c'è alcuna etichetta associata con il valore.
`c.getFalseFlag()`	Oggetto	Restituisce il valore di indicatore "false" per la colonna o `None` se il valore non è noto oppure se la colonna non è un indicatore.
`c.getTrueFlag()`	Oggetto	Restituisce il valore di indicatore "true" per la colonna o `None` se il valore non è noto oppure se la colonna non è un indicatore.
`c.getLowerBound()`	Oggetto	Restituisce il valore del limite inferiore per i valori nella colonna o `None` se il valore non è noto oppure se la colonna non è continua.
`c.getUpperBound()`	Oggetto	Restituisce il valore del limite superiore per i valori nella colonna o `None` se il valore non è noto oppure se la colonna non è continua.

Si noti che la maggior parte dei metodi che accedono alle informazioni relative ad una colonna hanno metodi equivalenti definiti sull'oggetto DataModel stesso. Per esempio le istruzioni seguenti sono equivalenti:

```
dataModel.getColumn("someName").getModelingRole()
dataModel.getModelingRole("someName")
```

Accesso agli oggetti generati

Generalmente, l'esecuzione di un flusso implica la creazione di ulteriori oggetti di output. Tali oggetti aggiuntivi possono essere un nuovo modello oppure una parte di output che fornisce informazioni da utilizzare nelle esecuzioni successive.

Nell'esempio riportato di seguito, il flusso `druglearn.str` viene utilizzato nuovamente come punto di partenza per il flusso. In questo esempio, tutti i nodi nel flusso vengono eseguiti ed i risultati vengono

archiviati in un elenco. Lo script, quindi, esegue un loop all'interno dei risultati e tutti gli output del modello risultanti dall'esecuzione vengono salvati come un file di modello IBM SPSS Modeler (.gm) ed il modello viene esportato in PMML.

```
import modeler.api

stream = modeler.script.stream()

# Set this to an existing folder on your system.
# Include a trailing directory separator
modelFolder = "C:/temp/models/"

# Execute the stream
models = []
stream.runAll(models)

# Save any models that were created
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # If the stream execution built other outputs then ignore them
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

    label = model.getLabel()
    algorithm = model.getModelDetail().getAlgorithmName()

    # save each model...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...and export each model PMML...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

La classe task runner fornisce un modo per l'esecuzione di varie attività comuni. I metodi disponibili in tale classe sono riepilogati nella tabella riportata di seguito.

Tabella 20. Metodi della classe task runner per l'esecuzione di attività comuni

Metodo	Tipo di restituzione	Descrizione
`t.createStream(name, autoConnect, autoManage)`	Flusso	Crea e restituisce un nuovo flusso. Notare che il codice che deve creare i flussi privatamente senza renderli visibili all'utente deve impostare l'indicatore `autoManage` su `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	Non applicabile	Esporta la descrizione del flusso in un file utilizzando il formato di file specificato.
`t.exportModelToFile(modelOutput, filename, fileFormat)`	Non applicabile	Esporta il modello in un file utilizzando il formato di file specificato.
`t.exportStreamToFile(stream, filename, fileFormat)`	Non applicabile	Esporta il flusso in un file utilizzando il formato file specificato.
`t.insertNodeFromFile(filename, diagram)`	Nodo	Legge e restituisce un nodo dal file specificato, inserendolo nel diagramma fornito. Notare che può essere utilizzato per leggere gli oggetti nodo e supernodo.
`t.openDocumentFromFile(filename, autoManage)`	DocumentOutput	Legge e restituisce un documento dal file specificato.

Tabella 20. Metodi della classe task runner per l'esecuzione di attività comuni (Continua)

Metodo	Tipo di restituzione	Descrizione
`t.openModelFromFile(filename, autoManage)`	ModelOutput	Legge e restituisce un modello dal file specificato.
`t.openStreamFromFile(filename, autoManage)`	Flusso	Legge e restituisce un flusso dal file specificato.
`t.saveDocumentToFile( documentOutput, filename)`	Non applicabile	Salva il documento nel percorso di file specificato.
`t.saveModelToFile(modelOutput, filename)`	Non applicabile	Salva il modello nel percorso di file specificato.
`t.saveStreamToFile(stream, filename)`	Non applicabile	Salva il flusso nel percorso di file specificato.

Gestione degli errori

Il linguaggio Python fornisce la gestione degli errori mediante il blocco di codice `try...except`. Tale blocco può essere utilizzato all'interno degli script per racchiudere eccezioni e gestire problemi che, in caso contrario, potrebbero causare la fine dello script.

Nello script di esempio riportato di seguito, viene eseguito un tentativo di richiamo di un modello da un IBM SPSS Collaboration and Deployment Services Repository. Questa operazione può causare la generazione di un'eccezione, ad esempio, le credenziali di accesso al repository potrebbero non essere state impostate correttamente oppure il percorso del repository non è corretto. Nello script, ciò può causare un'eccezione `ModelerException` (tutte le eccezioni generate da IBM SPSS Modeler sono derivate da `modeler.api.ModelerException`).

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
```

Nota: Alcune operazioni di script potrebbero causare la generazione di eccezioni Java standard; tali eccezioni non derivano da `ModelerException`. Per rilevare tali eccezioni, è possibile utilizzare un ulteriore blocco per rilevare tutte le eccezioni Java, ad esempio:

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

Parametri stream, sessione e Supernodo

I parametri rappresentano un utile modo per il passaggio dei valori al runtime, invece della codifica diretta in uno script. I parametri ed i relativi valori vengono definiti nello stesso modo dei flussi, vale a dire come voci nella tabella dei parametri di un flusso o supernodo oppure come parametri della riga comandi. Le classi Stream e SuperNode implementano una serie di funzioni definite dall'oggetto ParameterProvider, come illustrato nella tabella riportata di seguito. La sessione fornisce una chiamata `getParameters()` che restituisce un oggetto che definisce tali funzioni.

Tabella 21. Funzioni definite dall'oggetto ParameterProvider

Metodo	Tipo di restituzione	Descrizione
`p.parameterIterator()`	Iteratore	Restituisce un iteratore di nomi di parametri per questo oggetto.
`p.getParameterDefinition( parameterName)`	ParameterDefinition	Restituisce la definizione del parametro per il parametro con il nome specificato oppure `None` se in questo provider non esiste alcun parametro di questo tipo. Il risultato può essere un'istantanea della definizione nel momento in cui il metodo è stato richiamato e non deve riflettere tutte le modifiche successive apportate al parametro mediante questo provider.
`p.getParameterLabel(parameterName)`	*stringa*	Restituisce l'etichetta del parametro indicato oppure `None` se non esiste alcun parametro di questo tipo.
`p.setParameterLabel(parameterName, label)`	Non applicabile	Imposta l'etichetta del parametro indicato.
`p.getParameterStorage( parameterName)`	ParameterStorage	Restituisce l'archiviazione del parametro indicato oppure `None` se non esiste alcun parametro di questo tipo.
`p.setParameterStorage( parameterName, storage)`	Non applicabile	Imposta l'archiviazione del parametro indicato.
`p.getParameterType(parameterName)`	Tipo Parametro	Restituisce il tipo del parametro indicato oppure `None` se non esiste alcun parametro di questo tipo.
`p.setParameterType(parameterName, type)`	Non applicabile	Imposta il tipo del parametro indicato.
`p.getParameterValue(parameterName)`	Oggetto	Restituisce il valore del parametro indicato oppure `None` se non esiste alcun parametro di questo tipo.
`p.setParameterValue(parameterName, value)`	Non applicabile	Imposta il valore del parametro indicato.

Nell'esempio riportato di seguito, lo script aggrega alcuni dati Telco per individuare la regione con i dati di reddito medio più basso. Con questa regione viene quindi impostato un parametro stream. Tale parametro stream viene quindi utilizzato in un nodo Select per escludere tale regione dai dati, prima che sulla parte rimanente venga creato un modello churn.

L'esempio è artificiale perché lo script genera il nodo Select da solo e, pertanto, potrebbe aver generato il valore corretto direttamente nell'espressione del nodo Select. Tuttavia, i flussi sono generalmente pregenerati, per cui l'impostazione dei parametri in questo modo rappresenta un esempio utile.

La prima parte dello script di esempio crea il parametro stream che conterrà la regione con il reddito medio più basso. Lo script crea anche i nodi nel ramo di aggregazione e nel ramo di creazione del modello e li collega tra loro.

```
import modeler.api

stream = modeler.script.stream()

# Initialize a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)
```

Lo script di esempio crea il seguente flusso.

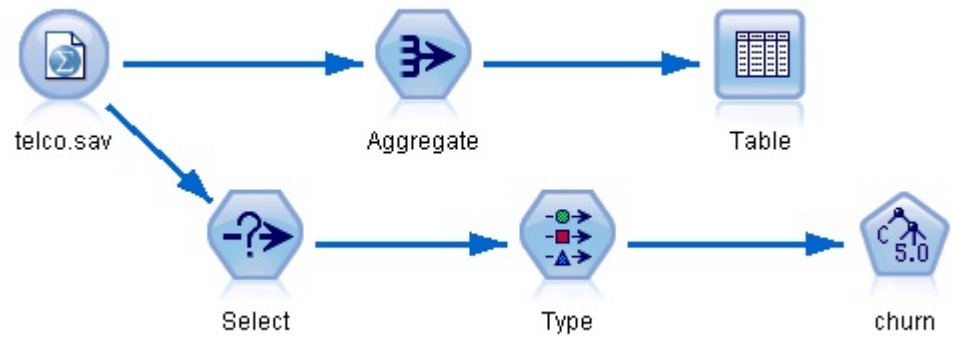

Figura 5. Flusso che risulta dallo script di esempio

La seguente parte dello script di esempio esegue il nodo Table alla fine del ramo di aggregazione.

```
# First execute the table node
results = []
tablenode.run(results)
```

La seguente parte dello script di esempio accede all'output della tabella generato dall'esecuzione del nodo Table. Lo script, quindi, esegue un'iterazione attraverso le righe nella tabella, ricercando la regione con il reddito medio più basso.

```
# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1
```

La seguente parte dello script utilizza la regione con il reddito medio più basso per impostare il parametro del flusso "LowestRegion" precedentemente creato. Lo script, quindi, esegue il builder del modello con la regione specificata esclusa dai dati di addestramento.

```
# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

Lo script di esempio completo è riportato di seguito.

```
import modeler.api

stream = modeler.script.stream()

# Create a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)
```

```
stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)


# First execute the table node
results = []
tablenode.run(results)

# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

Valori globali

I valori globali vengono utilizzati per calcolare statistiche di riepilogo per campi specificati. È possibile accedere a tali valori di riepilogo in qualsiasi punto all'interno del flusso. I valori globali sono simili ai parametri del flusso perché ad essi si accede in base al nome attraverso il flusso. Sono differenti dai parametri del flusso perché i valori associati vengono aggiornati automaticamente quando viene eseguito un nodo Calcola globali, invece di essere assegnati mediante script o dalla riga comandi. È possibile accedere ai valori globali per un flusso richiamando il metodo `getGlobalValues()` del flusso.

L'oggetto GlobalValues definisce le funzioni indicate nella tabella riportata di seguito.

Tabella 22. Funzioni definite dall'oggetto GlobalValues

Metodo	Tipo di restituzione	Descrizione
`g.fieldNameIterator()`	Iteratore	Restituisce un iteratore per ciascun nome campo con almeno un valore globale.
`g.getValue(type, fieldName)`	Oggetto	Restituisce il valore globale per il nome del campo ed il tipo specificati oppure `None` se non è possibile individuare alcun valore. Generalmente, il valore restituito previsto è un numero, sebbene funzionalità future potrebbero restituire tipi di valori differenti.

Tabella 22. Funzioni definite dall'oggetto GlobalValues (Continua)

Metodo	Tipo di restituzione	Descrizione
`g.getValues(fieldName)`	Mappa	Restituisce una mappa che contiene le voci note per il nome del campo specificato oppure `None` se non esistono voci per il campo.

`GlobalValues.Type` definisce il tipo di statistiche di riepilogo disponibili. Sono disponibili le statistiche di riepilogo riportate di seguito:

- `MAX`: il valore massimo del campo.
- `MEAN`: il valore medio del campo.
- `MIN`: il valore minimo del campo.
- `STDDEV`: la deviazione standard del campo.
- `SUM`: la somma dei valori nel campo.

Ad esempio, lo script riportato di seguito accede al valore medio del campo "income", calcolato da un nodo Calcola globali:

```
import modeler.api

globals = modeler.script.stream().getGlobalValues()
mean_income = globals.getValue(modeler.api.GlobalValues.Type.MEAN, "income")
```

Utilizzo di più flussi: script autonomi

Per utilizzare più flussi, è necessario utilizzare uno script autonomo. Lo script autonomo può essere modificato ed eseguito all'interno dell'interfaccia utente di IBM SPSS Modeler oppure passato come parametro della riga comandi in modalità batch.

Lo script autonomo riportato di seguito apre due flussi. Uno di tali flussi crea un modello, mentre l'altro flusso traccia la distribuzione dei valori previsti.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/18.2.1/DEMOS/streams/"

session = modeler.script.session()
tasks = session.getTaskRunner()

# Open the model build stream, locate the C5.0 node and run it
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Create a model applier node, insert it between the derive and histogram nodes
# then run the histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

# Finally, tidy up the streams
buildstream.close()
plotstream.close()
```

Il seguente esempio mostra in che modo è possibile iterare nei flussi aperti (tutti i flussi aperti nella scheda Flussi). Questa attività è supportata solo negli script autonomi.

```
for stream in modeler.script.streams():
    print stream.getName()
```

Capitolo 5. Suggerimenti per gli script

Questa sezione contiene una panoramica dei suggerimenti e delle tecniche di utilizzo degli script, incluse tecniche relative a modifica dell'esecuzione del flusso, utilizzo di una password codificata in uno script e accesso a oggetti in IBM SPSS Collaboration and Deployment Services Repository.

Modifica dell'esecuzione del flusso

Durante l'esecuzione di un flusso, i relativi nodi terminali vengono eseguiti in base all'ordine ottimale per la situazione di default. In alcuni casi, è tuttavia preferibile utilizzare un ordine di esecuzione diverso. Per modificare l'ordine di esecuzione di un flusso, completare i passi riportati di seguito dalla scheda Esecuzione della finestra di dialogo delle proprietà del flusso:

1. Iniziare con uno script vuoto.
2. Fare clic sul pulsante **Accoda lo script di default** nella barra degli strumenti per aggiungere lo script del flusso di default.
3. Modificare l'ordine delle istruzioni contenute nello script del flusso di default in base all'ordine in cui si desidera che vengano eseguite le istruzioni.

Esecuzione di cicli attraverso i nodi

È possibile eseguire un ciclo `for` per eseguire un cicli su tutti i nodi in un flusso. Ad esempio, i due esempi di script riportati di seguito eseguono un ciclo su tutti i nodi e cambiano da minuscolo a maiuscolo i nomi dei campi nei nodi Filtro.

Tali script possono essere utilizzati in qualsiasi flusso che dispone di un nodo Filtro, anche se non viene effettivamente filtrato alcun campo. È sufficiente aggiungere un nodo Filtro che passi tutti i campi per far sì che tutti i nomi dei campi diventino maiuscoli.

```
# Alternative 1: using the data model nameIterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() returns the field names
        for field in node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Alternative 2: using the data model iterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() returns the field objects so we need
        # to call getColumnName() to get the name
        for field in node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

Lo script esegue un ciclo su tutti i nodi nel flusso corrente e verifica se ciascun nodo è un nodo Filtro. In questo caso, lo script esegue un ciclo su ciascun campo nel nodo ed utilizza la funzione `field.upper()` o `field.getColumnName().upper()` per modificare il nome in maiuscolo.

Accesso a oggetti nel IBM SPSS Collaboration and Deployment Services Repository

Se si dispone di una licenza per IBM SPSS Collaboration and Deployment Services Repository, è possibile archiviare e richiamare gli oggetti dal repository utilizzando i comandi script. Utilizzare il repository per gestire il ciclo di vita dei modelli di data mining e gli oggetti predittivi correlati nel contesto di applicazioni enterprise, strumenti e soluzioni.

Connessione a IBM SPSS Collaboration and Deployment Services Repository

Per accedere al repository, è necessario innanzitutto impostare una connessione valida al repository, mediante il menu **Strumenti** dell'interfaccia utente SPSS Modeler o mediante la riga comandi. Per ulteriori informazioni, vedere "Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository" a pagina 67.

Come ottenere accesso al repository

È possibile accedere al repository dalla sessione, ad esempio:

```
repo = modeler.script.session().getRepository()
```

Richiamo degli oggetti dal repository

All'interno di uno script, utilizzare le funzioni `retrieve*` per accedere a diversi oggetti, inclusi flussi, modelli, output e nodi. Nella seguente tabella è riportato un riepilogo delle funzioni di richiamo.

Tabella 23. Funzioni script di richiamo

Tipo di oggetto	Funzione repository
Flusso	repo.retrieveStream(percorso stringa, versione stringa, etichetta stringa, autoManage booleano)
Modello	repo.retrieveModel(percorso stringa, versione stringa, etichetta stringa, autoManage booleano)
Output	repo.retrieveDocument(percorso stringa, versione stringa, etichetta stringa, autoManage booleano)
Nodo	repo.retrieveProcessor(percorso stringa, versione stringa, etichetta stringa, diagramma ProcessorDiagram)

Ad esempio, è possibile richiamare un flusso dal repository con la seguente funzione:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", None, "production", True)
```

Questo esempio richiama il flusso `risk_score.str` dalla cartella specificata. L'etichetta `production` identifica la versione del flusso da richiamare e l'ultimo parametro specifica che SPSS Modeler deve gestire il flusso (ad esempio, in questo modo il flusso viene visualizzato nella scheda **Flussi** se l'interfaccia utente SPSS Modeler è visibile). In alternativa, per utilizzare una versione specifica senza etichetta:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", "0:2015-10-12 14:15:41.281", None, True)
```

Nota: Se i parametri della versione e dell'etichetta sono entrambi `None`, viene restituita la versione più recente.

Archiviazione degli oggetti nel repository

Per utilizzare gli script per archiviare gli oggetti nel repository, utilizzare le funzioni `store*`. Nella seguente tabella è riportato un riepilogo delle funzioni di archiviazione.

Tabella 24. Funzioni script di archiviazione

Tipo di oggetto	Funzione repository
Flusso	repo.storeStream(flusso ProcessorStream, percorso stringa, etichetta stringa)
Modello	repo.storeModel(ModelOutput modelOutput, percorso stringa, etichetta stringa)
Output	repo.storeDocument(DocumentOutput documentOutput, percorso stringa, etichetta stringa)
Nodo	repo.storeProcessor(nodo Processor, percorso stringa, etichetta stringa)

Ad esempio, è possibile archiviare una nuova versione del flusso `risk_score.str` con la seguente funzione:

```
versionId = repo.storeStream(stream, "/projects/retention/risk_score.str", "test")
```

Questo esempio archivia una nuova versione del flusso, associa ad esso l'etichetta `"test"` e restituisce il contrassegno della versione per la versione appena creata.

Nota: Se non si desidera associare un'etichetta alla nuova versione, passare `None` per l'etichetta.

Gestione delle cartelle del repository

Utilizzando le cartelle all'interno del repository, è possibile organizzare gli oggetti in gruppi logici e visualizzare più facilmente gli oggetti correlati. Creare le cartelle utilizzando la funzione `createFolder()`, come nell'esempio riportato di seguito:

```
newpath = repo.createFolder("/projects", "cross-sell")
```

Questo esempio crea una nuova cartella denominata `"cross-sell"` nella cartella `"/projects"`. La funzione restituisce il percorso completo della nuova cartella.

Per ridenominare una cartella, utilizzare la funzione `renameFolder()`:

```
repo.renameFolder("/projects/cross-sell", "cross-sell-Q1")
```

Il primo parametro è il percorso completo della cartella da ridenominare ed il secondo è il nuovo nome da assegnare a tale cartella.

Per eliminare una cartella vuota, utilizzare la funzione `deleteFolder()`:

```
repo.deleteFolder("/projects/cross-sell")
```

Blocco e sbocco di oggetti

Da uno script, è possibile bloccare un oggetto per impedire qualsiasi aggiornamento delle sue versioni esistenti o la creazione di nuove versioni. È inoltre possibile sbloccare un oggetto che è stato bloccato.

La sintassi per bloccare e sbloccare un oggetto è:

```
repo.lockFile(REPOSITORY_PATH)
repo.lockFile(URI)

repo.unlockFile(REPOSITORY_PATH)
repo.unlockFile(URI)
```

Come con l'archiviazione e il recupero di oggetti, `REPOSITORY_PATH` fornisce la posizione dell'oggetto nel repository. Il percorso deve essere racchiuso tra virgolette ed è necessario utilizzare le barre (/) come delimitatori. Il percorso non fa distinzione tra caratteri maiuscoli/minuscoli.

```
repo.lockFile("/myfolder/Stream1.str")
repo.unlockFile("/myfolder/Stream1.str")
```

In alternativa, è possibile utilizzare un URI (Uniform Resource Identifier) anziché un percorso di repository per indicare la posizione dell'oggetto. L'URI deve includere il prefisso `spsscr:` e deve essere racchiuso tra virgolette. Come delimitatori di percorso sono consentite solo le barre (/) e gli spazi devono essere codificati, ovvero utilizzare `%20` anziché lo spazio nel percorso. L'URI non fa distinzione tra caratteri maiuscoli/minuscoli. Di seguito sono riportati alcuni esempi:

```
repo.lockFile("spsscr:///myfolder/Stream1.str")
repo.unlockFile("spsscr:///myfolder/Stream1.str")
```

Tenere presente che il blocco dell'oggetto si applica a tutte le versioni di un oggetto. Non è possibile, infatti, bloccare o sbloccare versioni singole.

Creazione di una password codificata

In alcuni casi, è possibile includere una password in uno script, per esempio se si desidera accedere a un'origine dati protetta da password. Le password codificate possono essere utilizzate in:

- Proprietà dei nodi per nodi origine Database e nodi output
- Argomenti della riga di comando per l'accesso al server
- Proprietà di connessione al database archiviate in un file *.par* (file del parametro generato dalla scheda Pubblica di un nodo di esportazione)

Tramite l'interfaccia utente, è disponibile uno strumento per generare password codificate in base all'algoritmo Blowfish (per ulteriori informazioni, vedere http://www.schneier.com/blowfish.html). Dopo la codifica, è possibile copiare e archiviare la password in file script e argomenti della riga di comando. La proprietà del nodo `epassword` utilizzata per i nodi `databasenode` e `databaseexportnode` consente di memorizzare la password codificata.

1. Per generare una password codificata, dal menu Strumenti scegliere:

   Codifica password...
2. Specificare una password nella casella di testo Password.
3. Fare clic su **Codifica** per generare una codifica casuale per la password.
4. Fare clic sul pulsante Copia per copiare la password codificata negli Appunti.
5. Incollare la password nello script o nel parametro desiderato.

Controllo degli script

Per controllare in modo rapido la sintassi di tutti i tipi di script, fare clic sul pulsante di verifica di colore rosso disponibile nella barra degli strumenti della finestra di dialogo Script autonomo.

Figura 6. Icone della barra degli strumenti Script del flusso

Durante il controllo degli script verranno segnalati gli eventuali errori del codice e forniti suggerimenti per la risoluzione. Per visualizzare la riga contenente gli errori, fare clic sul feedback visualizzato nella metà inferiore della finestra di dialogo. L'errore verrà evidenziato in rosso.

Script dalla riga di comando

Gli script consentono di eseguire operazioni che in genere vengono eseguite nell'interfaccia utente. È sufficiente specificare ed eseguire uno script autonomo dalla riga di comando quando si avvia IBM SPSS Modeler. Ad esempio:

```
client -script scores.txt -execute
```

Il flag `-script` carica lo script specificato, mentre il flag `-execute` esegue tutti i comandi contenuti nel file di script.

Compatibilità con le versioni precedenti

Questa versione di IBM SPSS Modeler supporta il funzionamento degli script creati nelle versioni precedenti. Tuttavia, ora è possibile inserire automaticamente i nugget del modello nel flusso (questa è l'impostazione di default) nonché sostituire o integrare un nugget esistente dello stesso tipo all'interno del flusso. Il fatto che questo accada o meno dipende dalle impostazioni delle opzioni **Aggiungi modello al flusso** e **Sostituisci modello precedente** (**Strumenti > Opzioni > Opzioni utente > Notifiche**). Per esempio, può essere necessario modificare uno script di una versione precedente in cui la sostituzione di un nugget avviene eliminando il nugget esistenti e inserendone uno nuovo.

Gli script creati in questa versione potrebbero non funzionare in versioni precedenti.

Se uno script creato in una versione precedente utilizza un comando che è stato nel frattempo sostituito o che non è più supportato, il formato sarà supportato, ma verrà visualizzato un messaggio di avviso. Per esempio, la vecchia parola chiave `generated` è stata sostituita da `model` e `clear generated` è stato sostituito da `clear generated palette`. Gli script che utilizzano i vecchi formati verranno eseguiti, tuttavia verrà visualizzato un messaggio di avviso.

Accesso ai risultati dell'esecuzione del flusso

Molti nodi IBM SPSS Modeler producono oggetti di output come modelli, grafici e dati in formato tabellare. Molti di tali output contengono valori utili che possono essere utilizzati dagli script per guidare la successiva esecuzione. Tali valori vengono raggruppati in contenitori del contenuto (indicati semplicemente come contenitori) a cui è possibile accedere utilizzando tag o ID che identificano ciascun contenitore. La modalità di accesso a tali valori dipende dal formato o "modello di contenuto" utilizzato da tale contenitore.

Ad esempio, molti output del modello predittivo utilizzano una variante di XML denominata PMML per rappresentare le informazioni relative al modello, come, ad esempio, i campi utilizzati da una struttura ad albero delle decisioni in ciascuna suddivisione oppure il modo in cui i neuroni sono connessi in una rete neurale e con quale intensità. Gli output del modello che utilizzano PMML forniscono un modello di contenuto XML che può essere utilizzato per accedere a tali informazioni. Ad esempio:

```
stream = modeler.script.stream()
# Assume the stream contains a single C5.0 model builder node
# and that the datasource, predictors and targets have already been
# set up
modelbuilder = stream.findByType("c50", None)
results = []
modelbuilder.run(results)
modeloutput = results[0]

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")

# The PMML content model is a generic XML-based content model that
# uses XPath syntax. Use that to find the names of the data fields.
# The call returns a list of strings match the XPath values
dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
```

IBM SPSS Modeler supporta i seguenti modelli di contenuto negli script:

- Il**modello di contenuto tabella** fornisce l'accesso a dati in formato tabellare semplici rappresentati come righe e colonne.

- Il**modello di contenuto XML** fornisce l'accesso al contenuto archiviato in formato XML.
- Il**modello di contenuto JSON** fornisce l'accesso al contenuto archiviato in formato JSON.
- Il**modello di contenuto delle statistiche di colonne** fornisce l'accesso a statistiche di riepilogo relative ad un campo specifico.
- Il**modello di contenuto delle statistiche di colonne pair-wise** fornisce l'accesso a statistiche di riepilogo tra due campi o valori tra due campi separati.

I seguenti nodi non contengono questi modelli di contenuto:

- Serie temporali
- Discriminante
- SLRM
- TCM
- Tutti i nodi Python
- Tutti i nodi Spark
- Tutti i nodi di modellazione del database
- Modello estensione
- STP

Modello di contenuto tabella

Il modello di contenuto tabella fornisce un modello semplice per l'accesso ai dati di righe e colonne semplici. I valori in una particolare colonna devono avere tutti lo stesso tipo di archiviazione (ad esempio, stringhe o interi).

API

Tabella 25. API

Restituzione	Metodo	Descrizione
`int`	`getRowCount()`	Restituisce il numero di righe in questa tabella.
`int`	`getColumnCount()`	Restituisce il numero di colonne in questa tabella.
`Stringa`	`getColumnName(int columnIndex)`	Restituisce il nome della colonna all'indice della colonna specificato. L'indice della colonna parte da 0.
`StorageType`	`getStorageType(int columnIndex)`	Restituisce il tipo di archiviazione della colonna all'indice specificato. L'indice della colonna parte da 0.
`Oggetto`	`getValueAt(int rowIndex, int columnIndex)`	Restituisce il valore all'indice della riga e della colonna specificato. Gli indici di riga e colonna partono da 0.
`void`	`reset()`	Svuota la memoria interna associata a questo modello di contenuto.

Nodi ed output

Questa tabella elenca i nodi che creano output che includono questo tipo di modello di contenuto.

Tabella 26. Nodi ed output

Nome nodo	Nome output	ID contenitore
`table`	`table`	`"table"`

Script di esempio

```
stream = modeler.script.stream()
from modeler.api import StorageType

# Set up the variable file import node
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Next create the aggregate node and connect it to the variable file node
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configure the aggregate node
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Then create the table output node and connect it to the aggregate node
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Execute the table node and capture the resulting table output object
results = []
tablenode.run(results)
tableoutput = results[0]

# Access the table output's content model
tablecontent = tableoutput.getContentModel("table")

# For each column, print column name, type and the first row
# of values from the table content
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

L'output nella scheda Debug dello script sarà simile a quello riportato di seguito:

```
Age_Min Integer 15
Age_Max Integer 74
Na_Mean Real 0.730851098901
Na_SDev Real 0.116669731242
Drug String drugY
Record_Count Integer 91
```

Modello di contenuto XML

Il modello di contenuto XML fornisce l'accesso al contenuto basato su XML.

Il modello di contenuto XML supporta la possibilità di accedere ai componenti in base alle espressioni XPath. Le espressioni XPath sono stringhe che definiscono gli elementi o gli attributi richiesti dal chiamante. Il modello di contenuto XML nasconde i dettagli relativi alla costruzione di diversi oggetti ed alla compilazione di espressioni generalmente richiesti dal supporto XPath. In questo modo, è più semplice eseguire chiamate dagli script Python.

Il modello di contenuto XML include una funzione che restituisce il documento XML come stringa. In questo modo, gli utenti di script Python possono utilizzare la propria libreria Python preferita per analizzare il codice XML.

API

Tabella 27. API

Restituzione	Metodo	Descrizione
`Stringa`	`getXMLAsString()`	Restituisce il codice XML come stringa.
`number`	`getNumericValue(String xpath)`	Restituisce il risultato della valutazione del percorso con un tipo di restituzione numerico (ad esempio, conteggia il numero di elementi che corrispondono all'espressione del percorso).
`booleano`	`getBooleanValue(String xpath)`	Restituisce il risultato booleano della valutazione dell'espressione del percorso specificata.
`Stringa`	`getStringValue(String xpath, String attribute)`	Restituisce il valore dell'attributo o il valore del nodo XML che corrisponde al percorso specificato.
`Elenco di stringhe`	`getStringValues(String xpath, String attribute)`	Restituisce un elenco di tutti i valori di attributo o valori del nodo XML che corrispondono al percorso specificato.
`Elenco di elenchi di stringhe`	`getValuesList(String xpath, <List of strings> attributes, boolean includeValue)`	Restituisce un elenco di tutti i valori di attributo che corrispondono al percorso specificato insieme al valore del nodo XML se richiesto.
`Tabella hash (key:string, value:list of string)`	`getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)`	Restituisce una tabella hash che utilizza l'attributo chiave o il valore del nodo XML come chiave e l'elenco dei valori dell'attributo specificato come valori della tabella.
`booleano`	`isNamespaceAware()`	Indica se i parser XML devono riconoscere gli spazi dei nomi. Il valore di default è `False`.
`void`	`setNamespaceAware(boolean value)`	Imposta se i parser XML devono riconoscere gli spazi dei nomi. Inoltre, richiama `reset()` per garantire che le modifiche siano utilizzate dalle chiamate successive.
`void`	`reset()`	Svuota la memoria interna associata a questo modello di contenuto (ad esempio, un oggetto DOM memorizzato nella cache).

Nodi ed output

Questa tabella elenca i nodi che creano output che includono questo tipo di modello di contenuto.

Tabella 28. Nodi ed output

Nome nodo	Nome output	ID contenitore
Maggior parte dei builder di modello	Maggior parte dei modelli generati	"PMML"
"autodataprep"	n/d	"PMML"

Script di esempio

Il codice di script Python per accedere al contenuto potrebbe essere simile a quello riportato di seguito:

```
results = []
modelbuilder.run(results)
modeloutput = results[0]
cm = modeloutput.getContentModel("PMML")

dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
predictedNames = cm.getStringValues("//MiningSchema/MiningField[@usageType='predicted']", "name")
```

Modello di contenuto JSON

Il modello di contenuto JSON viene utilizzato per fornire il supporto per il contenuto in formato JSON. Fornisce un'API di base per consentire ai chiamanti di estrarre i valori, basandosi sull'ipotesi che questi ultimi conoscano i valori a cui accedere.

API

Tabella 29. API

Restituzione	Metodo	Descrizione
Stringa	getJSONAsString()	Restituisce il contenuto JSON come stringa.
Oggetto	getObjectAt(<List of cbjecta> path, JSONArtifact artifact) throws Exception	Restituisce l'oggetto nel percorso specificato. La risorsa root fornita può essere null; in questo caso, viene utilizzata la root del contenuto. Il valore restituito può essere una stringa a valore letterale, un intero, un numero reale o un valore booleano oppure una risorsa (un oggetto JSON o un array JSON).
Hash table (key:object, value:object>	getChildValuesAt(<List of object> path, JSONArtifact artifact) throws Exception	Restituisce i valori figlio del percorso specificato se il percorso indica un oggetto JSON o altrimenti null. Le chiavi nella tabella sono stringhe mentre il valore associato può essere una stringa a valore letterale, un intero, un numero reale o un valore booleano oppure una risorsa JSON (un oggetto JSON o un array JSON).

Tabella 29. API (Continua)

Restituzione	Metodo	Descrizione
`Elenco di oggetti`	`getChildrenAt(<List of object> path path, JSONArtifact artifact) throws Exception`	Restituisce l'elenco di oggetti nel percorso specificato se il percorso indica un array JSON o altrimenti null. I valori restituiti possono essere una stringa a valore letterale, un intero, un numero reale o un valore booleano oppure una risorsa JSON (un oggetto JSON o un array JSON).
`void`	`reset()`	Svuota la memoria interna associata a questo modello di contenuto (ad esempio, un oggetto DOM memorizzato nella cache).

Script di esempio

Se è presente un nodo builder di output che crea un output basato sul formato JSON, è possibile utilizzare il codice riportato di seguito per accedere alle informazioni relative ad un insieme di libri:

```
results = []
outputbuilder.run(results)
output = results[0]
cm = output.getContentModel("jsonContent")

bookTitle = cm.getObjectAt(["books", "ISIN123456", "title"], None)

# Alternatively, get the book object and use it as the root
# for subsequent entries
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Get all child values for aspecific book
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Get the third book entry. Assumes the top-level "books" value
# contains a JSON array which can be indexed
bookInfo = cm.getObjectAt(["books", 2], None)

# Get a list of all child entries
allBooks = cm.getChildrenAt(["books"], None)
```

Modello di contenuto delle statistiche di colonne e modello di contenuto delle statistiche pairwise

Il modello di contenuto delle statistiche di colonne fornisce l'accesso alle statistiche che possono essere calcolate per ciascun campo (statistiche univariate). Il modello di contenuto delle statistiche pairwise fornisce l'accesso alle statistiche che possono essere calcolate tra coppie di campi o valori in un campo.

Le misure delle statistiche possibili sono:

- `Count`
- `UniqueCount`
- `ValidCount`
- `Mean`
- `Sum`
- `Min`

- Max
- Range
- Variance
- StandardDeviation
- StandardErrorOfMean
- Skewness
- SkewnessStandardError
- Kurtosis
- KurtosisStandardError
- Median
- Mode
- Pearson
- Covarianza
- TTest
- FTest

Alcuni valori sono appropriati solo da statistiche di colonne singole mentre altri sono appropriati solo per le statistiche pairwise.

Di seguito sono riportati i nodi che producono tali valori:

- Il **nodo Statistiche** produce le statistiche di colonna e può produrre statistiche pairwise quando sono specificati i campi di correlazione
- Il **nodo Esplora** produce statistiche di colonne ed è in grado di produrre statistiche pairwise quando viene specificato un campo di sovrapposizione.
- Il **nodo Medie** produce statistiche pairwise quando viene eseguito il confronto tra coppie di campi o vengono confrontati i valori di un campo con i riepiloghi di un altro campo.

I modelli di contenuto e le statistiche disponibili dipendono dalle capacità del nodo e dalle impostazioni all'interno del nodo.

API ColumnStatsContentModel

Tabella 30. API ColumnStatsContentModel.

Restituzione	Metodo	Descrizione
`List<StatisticType>`	`getAvailableStatistics()`	Restituisce le statistiche disponibili in questo modello. Non tutti i campi dispongono necessariamente di valori per tutte le statistiche.
`List<String>`	`getAvailableColumns()`	Restituisce i nomi delle colonne per cui sono state calcolate le statistiche.
`Number`	`getStatistic(String column, StatisticType statistic)`	Restituisce i valori statistici associati alla colonna.
`void`	`reset()`	Svuota la memoria interna associata a questo modello di contenuto.

API PairwiseStatsContentModel

Tabella 31. API PairwiseStatsContentModel.

Restituzione	Metodo	Descrizione
`List<StatisticType>`	`getAvailableStatistics()`	Restituisce le statistiche disponibili in questo modello. Non tutti i campi dispongono necessariamente di valori per tutte le statistiche.
`List<String>`	`getAvailablePrimaryColumns()`	Restituisce i nomi delle colonne primarie per cui sono state calcolate le statistiche.
`List<Object>`	`getAvailablePrimaryValues()`	Restituisce i valori della colonna primaria per cui sono state calcolate le statistiche.
`List<String>`	`getAvailableSecondaryColumns()`	Restituisce i nomi delle colonne secondarie per cui sono state calcolate le statistiche.
`Number`	`getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)`	Restituisce i valori statistici associati alle colonne.
`Number`	`getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)`	Restituisce i valori statistici associati al valore della colonna primaria ed alla colonna secondaria.
`void`	`reset()`	Svuota la memoria interna associata a questo modello di contenuto.

Nodi ed output

Questa tabella elenca i nodi che creano output che includono questo tipo di modello di contenuto.

Tabella 32. Nodi ed output.

Nome nodo	Nome output	ID contenitore	Note
`"means"` (nodo Medie)	`"means"`	`"columnStatistics"`	
`"means"` (nodo Medie)	`"means"`	`"pairwiseStatistics"`	
`"dataaudit"` (nodo Esplora)	`"means"`	`"columnStatistics"`	
`"statistics"` (nodo Statistiche)	`"statistics"`	`"columnStatistics"`	Generato solo quando vengono esaminati campi specifici.
`"statistics"` (nodo Statistiche)	`"statistics"`	`"pairwiseStatistics"`	Generato solo quando i campi sono correlati.

Script di esempio

```
from modeler.api import StatisticType
stream = modeler.script.stream()

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
```

```
# Now create the statistics node. This can produce both
# column statistics and pairwise statistics
statisticsnode = stream.createAt("statistics", "Stats", 192, 96)
statisticsnode.setPropertyValue("examine", ["Age", "Na", "K"])
statisticsnode.setPropertyValue("correlate", ["Age", "Na", "K"])
stream.link(varfile, statisticsnode)

results = []
statisticsnode.run(results)
statsoutput = results[0]
statscm = statsoutput.getContentModel("columnStatistics")
if (statscm != None):
 cols = statscm.getAvailableColumns()
 stats = statscm.getAvailableStatistics()
 print "Column stats:", cols[0], str(stats[0]), " = ", statscm.getStatistic(cols[0], stats[0])

statscm = statsoutput.getContentModel("pairwiseStatistics")
if (statscm != None):
 pcols = statscm.getAvailablePrimaryColumns()
 scols = statscm.getAvailableSecondaryColumns()
 stats = statscm.getAvailableStatistics()
 corr = statscm.getStatistic(pcols[0], scols[0], StatisticType.Pearson)
 print "Pairwise stats:", pcols[0], scols[0], " Pearson = ", corr
```

Capitolo 6. Argomenti della riga di comando

Modalità di richiamo del software

È possibile utilizzare la riga di comando del sistema operativo per avviare IBM SPSS Modeler:

1. Sul computer in cui è installato IBM SPSS Modeler, aprire una finestra DOS (prompt dei comandi).
2. Per avviare l'interfaccia di IBM SPSS Modeler in modalità interattiva, digitare il comando `modelerclient` seguito dagli argomenti richiesti, ad esempio:

```
modelerclient -stream report.str -execute
```

Gli argomenti disponibili (flag) consentono di connettersi a un server, caricare stream, eseguire script o specificare altri parametri.

Utilizzo degli argomenti della riga di comando

È possibile aggiungere alcuni argomenti della riga di comando (denominati anche *flag*) al comando `modelerclient` iniziale per modificare il modo in cui IBM SPSS Modeler viene richiamato.

Diversi tipi di argomenti della riga di comando non sono disponibili e vengono descritti in seguito in questa sezione.

Tabella 33. Tipi di argomenti della riga di comando.

Tipo di argomento	**Dove descritto**
Argomenti di sistema	Per ulteriori informazioni, consultare l'argomento "Argomenti di sistema" a pagina 64.
Argomenti dei parametri	Per ulteriori informazioni, consultare l'argomento "Argomenti dei parametri" a pagina 65.
Argomenti per la connessione del server	Per ulteriori informazioni, consultare l'argomento "Argomenti per la connessione del server" a pagina 66.
Argomenti di connessione a IBM SPSS Collaboration and Deployment Services Repository	Per ulteriori informazioni, consultare l'argomento "Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository" a pagina 67.
Argomenti per la connessione a IBM SPSS Analytic Server	Per ulteriori informazioni, consultare l'argomento "Argomenti per la connessione a IBM SPSS Analytic Server" a pagina 68.

Per esempio, è possibile utilizzare gli argomenti della riga di comando `-server`, `-stream` ed `-execute` per connettersi a un server e caricare ed eseguire un flusso, come indicato di seguito:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Si noti che in caso di esecuzione dalla riga di comando con Clementine Client installato localmente, gli argomenti di connessione al server non sono necessari.

È possibile racchiudere tra virgolette i valori di parametri che contengono spazi, ad esempio:

```
modelerclient -stream mystream.str -Pusername="Joe User" -execute
```

Questa soluzione consente anche di eseguire stati e script di IBM SPSS Modeler, utilizzando rispettivamente i flag `-state` e `-script`.

Nota: Se in un comando viene utilizzato un parametro strutturato, è necessario inserire le virgolette una barra rovesciata. In questo modo, le virgolette non vengono rimosse durante l'interpretazione della stringa.

Debug degli argomenti della riga di comando

Per eseguire il debug di una riga di comando, utilizzare il comando `modelerclient` per avviare IBM SPSS Modeler con gli argomenti desiderati. Ciò consente di verificare che i comandi vengano eseguiti come previsto. È possibile confermare i valori di qualsiasi parametro passato dalla riga di comando nella finestra di dialogo Parametri di sessione (menu Strumenti, Imposta parametri di sessione).

Argomenti di sistema

Nella tabella seguente sono illustrati gli argomenti di sistema disponibili per il richiamo dell'interfaccia utente dalla riga di comando.

Tabella 34. Argomenti di sistema

Argomento	Comportamento/descrizione
`@` <Filecomando>	Il carattere '@' seguito da un nome di file specifica un elenco di comandi. Quando `modelerclient` incontra un argomento che inizia con @, utilizza i comandi del file esattamente come se fossero stati specificati nella riga di comando. Per ulteriori informazioni, consultare l'argomento "Combinazione di più argomenti" a pagina 68.
`-directory` <dir>	Consente di impostare la directory di default. Nella modalità locale tale directory viene utilizzata sia per i dati che per l'output. Esempio: `-directory c:/` o `-directory c:\\`
`-server_directory` <dir>	Consente di impostare la directory del server di default per i dati. Tale directory, specificata mediante il flag `-directory`, viene utilizzata per l'output.
`-execute`	Dopo l'avvio, consente di eseguire eventuali flussi, stati o script caricati all'avvio. Se oltre a un flusso o stato viene caricato anche uno script, verrà eseguito solamente lo script.
`-stream` <stream>	Consente di caricare all'avvio il flusso specificato. È possibile specificare più flussi, ma l'ultimo flusso specificato verrà impostato come flusso corrente.
`-script` <script>	Consente di caricare all'avvio lo script autonomo specificato. Come illustrato di seguito, tale script può essere specificato insieme a un flusso o a uno stato, ma è possibile caricare un solo script all'avvio.
`-model` <modello>	Consente di caricare all'avvio il modello generato specificato (formato di file `.gm`).
`-state` <stato>	Consente di caricare all'avvio lo stato salvato specificato.
`-project` <progetto>	Consente di caricare il progetto specificato. È possibile caricare un solo progetto all'avvio.
`-output` <output>	Consente di caricare l'oggetto di output salvato (file di formato `COU`) all'avvio.
`-help`	Consente di visualizzare un elenco di argomenti della riga di comando. Quando si specifica questa opzione, tutti gli altri argomenti vengono ignorati e viene visualizzata la finestra della Guida in linea.
`-P` <nome>=<valore>	Utilizzato per impostare un parametro di avvio. Può essere utilizzato anche per impostare le proprietà dei nodi (parametri di slot).

Nota: È possibile impostare le directory di default anche nell'interfaccia utente. Per accedere alle opzioni, scegliere **Imposta directory** o **Imposta directory server** dal menu File.

Caricamento di più file

Dalla riga di comando è possibile caricare più flussi, stati e output all'avvio ripetendo l'argomento rilevante per ogni oggetto caricato. Per esempio, per caricare ed eseguire due flussi denominati `report.str` e `train.str`, è necessario utilizzare il seguente comando:

```
modelerclient -stream report.str -stream train.str -execute
```

Caricamento di oggetti da IBM SPSS Collaboration and Deployment Services Repository

Poiché è possibile caricare determinati oggetti da un file o da IBM SPSS Collaboration and Deployment Services Repository (se concesso in licenza), il prefisso `spsscr:` che precede il nome file e, facoltativamente, `file:` (per oggetti su disco) indica IBM SPSS Modeler dove cercare l'oggetto. Il prefisso viene utilizzato con i seguenti flag:

- `-stream`
- `-script`
- `-output`
- `-model`
- `-project`

Il prefisso utilizzato per creare un URI che specifica l'ubicazione dell'oggetto, ad esempio `-stream "spsscr:///folder_1/scoring_stream.str"`. La presenza del prefisso `spsscr:` richiede che nello stesso comando sia stata specificata una connessione valida a IBM SPSS Collaboration and Deployment Services Repository. Pertanto il comando completo si presenterà come segue:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///cartella_1/punteggio_stream.str".
```

Si noti che dalla riga di comando è *necessario* utilizzare un URI. Non è infatti supportato il semplice `REPOSITORY_PATH` che viene utilizzato solo all'interno di script. Per ulteriori informazioni sugli URI per gli oggetti nel IBM SPSS Collaboration and Deployment Services Repository, consultare l'argomento “Accesso a oggetti nel IBM SPSS Collaboration and Deployment Services Repository” a pagina 50.

Argomenti dei parametri

I parametri possono essere utilizzati come flag durante l'esecuzione della riga di comando di IBM SPSS Modeler. Negli argomenti della riga di comando il flag `-P` consente di specificare un parametro nel formato `-P` <*nome*>=<*valore*>.

I parametri possono essere dei seguenti tipi:

- **Parametri semplici** o parametri utilizzati direttamente nelle espressioni CLEM.
- **Parametri di slot**, detti anche **proprietà dei nodi**. Questi parametri vengono utilizzati per modificare le impostazioni dei nodi nel flusso. Per ulteriori informazioni, consultare l'argomento “Panoramica delle proprietà del nodo” a pagina 71.
- **Parametri della riga di comando** che consentono di modificare il richiamo di IBM SPSS Modeler.

Per esempio, è possibile specificare nomi utente e password per le sorgenti dei dati sotto forma di flag della riga di comando, come nel seguente esempio:

```
modelerclient -stream response.str -P:databasenode.datasource="{\"ORA 10gR2\",user1,mypsw,false}"
```

Il formato è lo stesso del parametro `datasource` della proprietà del nodo `databasenode`. Per ulteriori informazioni, consultare: “proprietà databasenode” a pagina 84.

L'ultimo parametro dovrebbe essere impostato su `true` se si sta passando una password codificata. Considerare anche che non si dovrebbe utilizzare nessuno spazio iniziale davanti al nome utente database e password (a meno che, naturalmente, il nome utente o password effettivamente non contengono uno spazio iniziale).

Nota: Se il nodo è denominato, è necessario racchiudere il nome del nodo tra virgolette e utilizzare la barra rovesciata come carattere di escape per le virgolette. Ad esempio, se il nodo dell'origine dati nell'esempio precedente è denominato `Source_ABC`, la voce è simile a quella riportata di seguito:

```
modelerclient -stream response.str -P:databasenode.\"Source_ABC\".datasource="{\"ORA 10gR2\",
 user1,mypsw,true}"
```

È inoltre necessario utilizzare una barra rovesciata prima delle virgolette che identificano un parametro strutturato, come riportato nel seguente esempio di origine dati TM1:

```
clemb -server -hostname 9.115.21.169 -port 28053 -username administrator
    -execute -stream C:\Share\TM1_Script.str -P:tm1import.pm_host="http://9.115.21.163:9510/pmhub/pm"
    -P:tm1import.tm1_connection={\"SData\",\"\",\"admin\",\"apple\"}
    -P:tm1import.selected_view={\"SalesPriorCube\",\"salesmargin%\"}
```

Argomenti per la connessione del server

Il flag `-server` indica che è necessario che IBM SPSS Modeler si connetta a un server pubblico e i flag `-hostname`, `-use_ssl`, `-port`, `-username`, `-password` e `-domain` vengono utilizzati per fornire a IBM SPSS Modeler i parametri necessari per la connessione al server pubblico. Se non viene specificato un argomento -server, viene utilizzato il server di default o locale.

Esempi

Per connettersi a un server pubblico:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Per connettersi a un cluster di server:

```
modelerclient -server -cluster "QA Machines" \
-spsscr_hostname pes_host -spsscr_port 8080 \
-spsscr_username asmith -spsscr_epassword xyz
```

Si noti che la connessione a un cluster di server richiede il plug-in Coordinator of Processes attraverso IBM SPSS Collaboration and Deployment Services, quindi l'argomento `-cluster` deve essere utilizzato in combinazione con le opzioni di connessione al repository (`spsscr_*`). Per ulteriori informazioni, consultare l'argomento "Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository" a pagina 67.

Tabella 35. Argomenti per la connessione del server

Argomento	Comportamento/descrizione
`-server`	Esegue IBM SPSS Modeler in modalità server, effettuando la connessione a un server pubblico tramite i flag `-hostname`, `-port`, `-username`, `-password` e `-domain`.
`-hostname <nome>`	Nome host del server. Disponibile solo nella modalità server.
`-use_ssl`	Specifica che la connessione deve utilizzare il protocollo SSL (Secure Socket Layer). Flag facoltativo; l'impostazione predefinita *non* prevede l'uso di SSL.
`-port <numero>`	Numero di porta del server specificato. Disponibile solo nella modalità server.

Tabella 35. Argomenti per la connessione del server (Continua)

Argomento	Comportamento/descrizione
`-cluster` <nome>	Specifica una connessione a un cluster di server invece che a un server denominato. Argomento alternativo a `hostname`, `port` e `use_ssl`. Il nome è il nome del cluster o un URI univoco che identifica il cluster in IBM SPSS Collaboration and Deployment Services Repository. Il cluster di server viene gestito da Coordinator of Processes attraverso IBM SPSS Collaboration and Deployment Services. Per ulteriori informazioni, consultare l'argomento "Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository".
`-username` <nome>	Nome utente utilizzato per l'accesso al server. Disponibile solo nella modalità server.
`-password` <password>	Password utilizzata per l'accesso al server. Disponibile solo nella modalità server. **Nota:** Se l'argomento `-password` non viene specificato, verrà richiesta l'immissione di una password.
`-epassword` <stringapasswordcodificata>	Password codificata utilizzata per l'accesso al server. Disponibile solo nella modalità server. **Nota:** per creare una password codificata, utilizzare il menu Strumenti dell'applicazione IBM SPSS Modeler.
`-domain` <nome>	Dominio utilizzato per l'accesso al server. Disponibile solo nella modalità server.
`-P` <nome>=<valore>	Utilizzato per impostare un parametro di avvio. Può essere utilizzato anche per impostare le proprietà dei nodi (parametri di slot).

Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository

Se si desidera archiviare o recuperare oggetti da IBM SPSS Collaboration and Deployment Services tramite la riga di comando, è necessario specificare una connessione valida a IBM SPSS Collaboration and Deployment Services Repository. Ad esempio:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///cartella_1/punteggio_stream.str".
```

Nella tabella riportata di seguito sono elencati gli argomenti da utilizzare per impostare la connessione.

Tabella 36. Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository

Argomento	Comportamento/descrizione
`-spsscr_hostname` <nome host o indirizzo IP>	Nome host o indirizzo IP del server su cui è installato IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_port` <numero>	Numero di porta su cui IBM SPSS Collaboration and Deployment Services Repository accetta connessioni, generalmente è la porta 8080 di default.
`-spsscr_use_ssl`	Specifica che la connessione deve utilizzare il protocollo SSL (Secure Socket Layer). Flag facoltativo; l'impostazione predefinita *non* prevede l'uso di SSL.
`-spsscr_username` <nome>	Nome utente utilizzato per l'accesso a IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <password>	Password utilizzata per l'accesso a IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` <password codificata>	Password codificata utilizzata per l'accesso a IBM SPSS Collaboration and Deployment Services Repository.

Tabella 36. Argomenti per la connessione a IBM SPSS Collaboration and Deployment Services Repository (Continua)

Argomento	Comportamento/descrizione
`-spsscr_providername <name>`	Il provider di autenticazione utilizzato per l'accesso a IBM SPSS Collaboration and Deployment Services Repository (Active Directory o LDAP). Non è richiesto se si utilizza il provider nativo (repository locale).

Argomenti per la connessione a IBM SPSS Analytic Server

Se si desidera archiviare o recuperare oggetti da IBM SPSS Analytic Server attraverso la riga di comando, è necessario specificare una connessione valida a IBM SPSS Analytic Server.

Nota: L'ubicazione predefinita di Analytic Server viene ottenuta da SPSS Modeler Server. Gli utenti possono anche definire le proprie connessioni Analytic Server tramite **Strumenti** > **Connessioni Analytic Server**.

Nella tabella riportata di seguito sono elencati gli argomenti da utilizzare per impostare la connessione.

Tabella 37. Argomenti per la connessione a IBM SPSS Analytic Server

Argomento	Comportamento/descrizione
`-analytic_server_username`	Il nome utente con cui eseguire l'accesso a IBM SPSS Analytic Server.
`-analytic_server_password`	La password con cui eseguire l'accesso a IBM SPSS Analytic Server.
`-analytic_server_epassword`	La password codificata con cui eseguire l'accesso a IBM SPSS Analytic Server.
`-analytic_server_credential`	Le credenziali utilizzate per l'accesso a IBM SPSS Analytic Server.

Combinazione di più argomenti

È possibile combinare più argomenti in un unico file dei comandi, che potrà essere specificato all'avvio utilizzando il simbolo @ seguito dal nome del file. In questo modo è possibile abbreviare il richiamo dalla riga di comando e superare eventuali limitazioni di lunghezza dei comandi previste dal sistema operativo. Per esempio, il seguente comando di avvio utilizza gli argomenti specificati nel file indicato da `<NomeFilecomando>`.

```
modelerclient @<NomeFilecomando>
```

Se è necessario specificare degli spazi, racchiudere il nome del file e il percorso tra virgolette, per esempio:

```
modelerclient @ "C:\Program Files\IBM\SPSS\Modeler\nn\scripts\my_command_file.txt"
```

Il file dei comandi può contenere tutti gli argomenti che in precedenza venivano specificati singolarmente all'avvio, con un argomento per riga, Ad esempio:

```
-stream report.str
-Porder.full_filename=APR_orders.dat
-Preport.filename=APR_report.txt
-execute
```

Quando si scrivono o si richiamano file dei comandi è importante attenersi alle seguenti indicazioni:

- Specificare un solo comando per riga.
- Non incorporare un argomento `@fileComando` in un file dei comandi.

Capitolo 7. Guida alle proprietà

Panoramica sui riferimenti alle proprietà

È possibile specificare numerose proprietà differenti per nodi, flussi, progetti e supernodi. Alcune proprietà sono comuni a tutti i nodi, per esempio name, annotation e ToolTip, altre invece sono specifiche di alcuni tipi di nodi. Altre proprietà fanno riferimento a operazioni di alto livello dei flussi, quali la memorizzazione nella cache o il funzionamento dei Supernodi. È possibile accedere alle proprietà tramite l'interfaccia utente standard (per esempio, tramite la finestra di dialogo per la modifica delle opzioni di un nodo) e utilizzarle in molti modi.

- È possibile modificare le proprietà tramite gli script, come illustrato in questa sezione. Per ulteriori informazioni, vedere "Sintassi per le proprietà".
- È possibile utilizzare le proprietà dei nodi nei parametri dei Supernodi.
- Le proprietà dei nodi possono inoltre essere specificate come parte di un'opzione della riga di comando (mediante il flag `-P`) all'avvio di IBM SPSS Modeler.

Per gli script di IBM SPSS Modeler, le proprietà dei nodi e dei flussi vengono spesso denominate **parametri di slot**. In questa guida, verranno invece definite come proprietà dei nodi o dei flussi.

Per ulteriori informazioni sul linguaggio di script, vedere Linguaggio di script.

Sintassi per le proprietà

È possibile impostare le proprietà utilizzando la sintassi riportata di seguito

```
OBJECT.setPropertyValue(PROPERTY, VALUE)
```

oppure:

```
OBJECT.setKeyedPropertyValue(PROPERTY, KEY, VALUE)
```

È possibile richiamare il valore delle proprietà utilizzando la seguente sintassi:

```
VARIABLE = OBJECT.getPropertyValue(PROPERTY)
```

oppure:

```
VARIABLE = OBJECT.getKeyedPropertyValue(PROPERTY, KEY)
```

dove `OBJECT` è un nodo o un output, `PROPERTY` è il nome della proprietà del nodo a cui fa riferimento l'espressione e `KEY` è il valore della chiave per le proprietà basate su chiavi. Ad esempio, la seguente sintassi viene utilizzata per trovare il nodo filtro e quindi impostare il valore predefinito per includere tutti i campi e filtrare il campo `Age` dai dati downstream:

```
filternode = modeler.script.stream().findByType("filter", None)
filternode.setPropertyValue("default_include", True)
filternode.setKeyedPropertyValue("include", "Age", False)
```

Tutti i nodi utilizzati in IBM SPSS Modeler possono essere individuati utilizzando la funzione `findByType(TYPE, LABEL)` del flusso. È necessario specificare almeno uno tra `TYPE` o `LABEL`.

Proprietà strutturate

Le proprietà strutturate vengono utilizzate negli script per semplificare l'analisi ed essenzialmente per due motivi:

- Per strutturare i nomi delle proprietà dei nodi complessi, quali i nodi tipo, filtro o bilanciamento.
- Per rendere disponibile un formato per la specifica di più proprietà contemporaneamente.

Strutturazione per interfacce complesse

Gli script per i nodi con tabelle e altre interfacce complesse quali i nodi tipo, filtro e bilanciamento devono avere una struttura particolare per poter eseguire l'analisi correttamente. A queste proprietà deve essere assegnato un nome più complesso di quello di un singolo identificativo; questo nome è denominato la chiave. Per esempio, all'interno di un nodo Filtro ogni campo disponibile (nel lato a monte) è attivato o disattivato. Per fare riferimento a queste informazioni, il nodo Filtro archivia un'informazione per campo (se ogni campo è true o false). Questa proprietà può avere (o alla quale può essere assegnato) il valore `True` o `False`. Si supponga che un nodo Filtro denominato `mionodo` includa un campo (nel lato upstream) denominato `Età`. Per disattivarlo, impostare la proprietà `include`, con la chiave `Età`, sul valore `False`, come indicato di seguito:

```
mynode.setKeyedPropertyValue("include", "Age", False)
```

Strutturazione per l'impostazione di proprietà multiple

Per molti nodi è possibile assegnare più di una proprietà di nodo o di flusso contemporaneamente. Questo tipo di operazione è definita **comando multiset** o **blocco di impostazioni**.

A volte le proprietà strutturate possono essere molto complesse. Di seguito è riportato un esempio:

```
sortnode.setPropertyValue("keys", [["K", "Descending"], ["Age", "Ascending"], ["Na", "Descending"]])
```

Un altro vantaggio delle proprietà strutturate consiste nella possibilità di impostare numerose proprietà in un nodo prima che questo diventi stabile. Per default, un comando multiset definisce tutte le proprietà del blocco prima di eseguire qualsiasi operazione basata sull'impostazione di una singola proprietà. Per esempio, se si definisce un nodo Testo fisso e si utilizzano due passaggi per impostare le proprietà dei campi, verranno generati degli errori perché il nodo non è coerente fino a quando entrambe le impostazioni non sono valide. La definizione delle proprietà come un multiset elude il problema, perché entrambe le proprietà vengono impostate prima dell'aggiornamento del modello di dati.

Abbreviazioni

Per le proprietà dei nodi, nella sintassi vengono utilizzate abbreviazioni standard il cui apprendimento può risultare utile per la creazione degli script.

Tabella 38. Abbreviazioni standard utilizzate nella sintassi

Abbreviazione	Significato
abs	In valore assoluto
len	Lunghezza
min	Minimo
max	Massimo
correl	Correlazione
covar	Covarianza
num	Numero o numerico
pct	Percento o percentuale
transp	Trasparenza
xval	Convalida incrociata
var	Varianza o variabile (nei nodi origine)

Esempi di proprietà dei nodi e dei flussi

IBM SPSS Modeler consente di utilizzare le proprietà di nodi e stream in vari modi. Nella maggior parte dei casi vengono utilizzate come parte di uno script, sia di uno **script autonomo** che consente di automatizzare più flussi o più operazioni, sia di uno **script del flusso** che consente di automatizzare i

processi all'interno di un singolo flusso. È possibile specificare i parametri dei nodi anche utilizzando le proprietà dei nodi all'interno del Supernodo. A livello di base, le proprietà possono inoltre essere utilizzate come opzioni della riga di comando per l'avvio di IBM SPSS Modeler. L'utilizzo dell'argomento `-p` nella chiamata alla riga di comando consente di modificare un'impostazione del flusso mediante una proprietà del flusso.

Tabella 39. Esempi di proprietà dei nodi e dei flussi

Proprietà	Significato
`s.max_size`	Fa riferimento alla proprietà `max_size` del nodo denominato `s`.
`s:samplenode.max_size`	Fa riferimento alla proprietà `max_size` del nodo denominato `s`, che deve essere un nodo Campione.
`:samplenode.max_size`	Fa riferimento alla proprietà `max_size` del nodo Campione del flusso corrente (deve essere presente un solo nodo Campione).
`s:samplenode.max_size`	Fa riferimento alla proprietà `max_size` del nodo denominato `s`, che deve essere un nodo Campione.
`t.direction.Età`	Fa riferimento al ruolo del campo *Età* nel nodo Tipo `t`.
`:.max_size`	*** NON VALIDO *** È necessario specificare il nome o il tipo del nodo.

L'esempio `s:sample.max_size` indica che non è necessario scrivere per esteso i tipi di nodo.

L'esempio `t.direction.Age` indica che anche i nomi di alcuni slot possono essere strutturati, se gli attributi di un nodo sono più complessi delle singole configurazioni con valori singoli. Questi slot sono definiti proprietà **strutturate** o **complesse**.

Panoramica delle proprietà del nodo

Per ogni tipo di nodo è disponibile un insieme specifico di proprietà valide. Questo tipo può essere generale, numero, indicatore o stringa, in cui le impostazioni relative alla proprietà vengono forzate sul tipo corretto. Se questo non è possibile, viene generato un errore. In alternativa, è possibile che il riferimento alla proprietà specifichi l'intervallo di valori validi, per esempio `Discard`, `PairAndDiscard` e `IncludeAsText`, nel qual caso verrà generato un errore se si utilizza un qualsiasi altro valore. Le proprietà flag dovrebbero essere lette o impostate utilizzando i valori `vero` e `falso`. (Per l'impostazione di questi valori è inoltre possibile utilizzare variazioni quali `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` o `0` vengono riconosciute anche durante l'impostazione dei valori ma, in alcuni casi, potrebbero causare errori durante la lettura dei valori delle proprietà. Tutti gli altri valori vengono considerati come vero. L'utilizzo coerente di `vero` e `falso` consente di evitare confusioni. Nelle tabelle di riferimento di questa guida, le proprietà strutturate vengono indicate come tali nella colonna **Descrizione proprietà** e vengono inoltre forniti i relativi formati di utilizzo.

Proprietà comuni dei nodi

Esistono numerose proprietà comuni a tutti i nodi di IBM SPSS Modeler, inclusi i Supernodi.

Tabella 40. Proprietà comuni dei nodi

Nome proprietà	Tipo di dati	Descrizione proprietà
`use_custom_name`	*indicatore*	
`name`	*stringa*	La proprietà di sola lettura che legge il nome (automatico o personalizzato) per un nodo nell'area.

Tabella 40. Proprietà comuni dei nodi (Continua)

Nome proprietà	Tipo di dati	Descrizione proprietà
custom_name	*stringa*	Specifica un nome personalizzato per il nodo.
tooltip	*stringa*	
annotation	*stringa*	
keywords	*stringa*	Slot strutturato che specifica un elenco di parole chiave associate all'oggetto, per esempio, ["Keyword1" "Keyword2"]
cache_enabled	*indicatore*	
node_type	source_supernode process_supernode terminal_supernode tutti i nomi di nodi, come specificato per gli script	Proprietà di sola lettura utilizzata per fare riferimento a un nodo in base al tipo. Per esempio, invece di fare riferimento a un nodo utilizzando solo il nome, quale real_income, è anche possibile specificarne il tipo, per esempio userinputnode o filternode.

Le proprietà specifiche dei Supernodi vengono illustrate separatamente, analogamente a tutti gli altri nodi. Per ulteriori informazioni, consultare l'argomento Capitolo 21, “Proprietà dei Supernodi”, a pagina 371.

Capitolo 8. Proprietà dei flussi

Gli script consentono di controllare numerose proprietà dei flussi. Per indicare le proprietà del flusso, è necessario impostare il metodo di esecuzione per utilizzare gli script:

```
stream = modeler.script.stream()
stream.setPropertyValue("execute_method", "Script")
```

Esempio

La proprietà del nodo viene utilizzata per fare riferimento ai nodi nel flusso corrente. Un esempio è costituito dallo script del flusso seguente:

```
stream = modeler.script.stream()
annotation = stream.getPropertyValue("annotation")

annotation = annotation + "\n\nThis stream is called \"" + stream.getLabel() + "\" and
 contains the following nodes:\n"

for node in stream.iterator():
    annotation = annotation + "\n" + node.getTypeName() + " node called \"" + node.getLabel()
 + "\""

stream.setPropertyValue("annotation", annotation)
```

Nell'esempio riportato sopra, la proprietà del nodo viene utilizzata per creare un elenco di tutti i nodi dello stream e per scrivere tale elenco nelle annotazioni dello stream. L'annotazione generata si presenta nel modo seguente:

```
Il flusso è denominato "druglearn" e contiene i nodi seguenti:

type node called "Define Types"
derive node called "Na_to_K"
variablefile node called "DRUG1n"
neuralnetwork node called "Drug"
c50 node called "Drug"
filter node called "Discard Fields"
```

Nella tabella seguente vengono illustrate le proprietà dei flussi.

Tabella 41. Proprietà dei flussi

Nome proprietà	Tipo di dati	Descrizione proprietà
execute_method	Normal Script	

Tabella 41. Proprietà dei flussi (Continua)

Nome proprietà	Tipo di dati	Descrizione proprietà
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	
date_baseline	*number*	
date_2digit_baseline	*number*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	*indicatore*	
import_datetime_as_string	*indicatore*	
decimal_places	*number*	
decimal_symbol	Default Period Comma	
angles_in_radians	*indicatore*	
use_max_set_size	*indicatore*	
max_set_size	*number*	
ruleset_evaluation	Voting FirstHit	

Tabella 41. Proprietà dei flussi (Continua)

Nome proprietà	Tipo di dati	Descrizione proprietà
refresh_source_nodes	*indicatore*	Consente di aggiornare automaticamente i nodi origine all'esecuzione del flusso.
script	*stringa*	
annotation	*stringa*	
name	*stringa*	**Nota:** questa proprietà è di sola lettura. Se si desidera modificare il nome di un flusso, salvarlo con un nome diverso.
parametri		Utilizzare questa proprietà per aggiornare i parametri dei flussi dall'interno di uno script autonomo.
nodes		Vedere le informazioni dettagliate riportate di seguito.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*booleano*	
stream_rewriting_maximise_sql	*booleano*	
stream_rewriting_optimise_clem_esecuzione	*booleano*	
stream_rewriting_optimise_syntax_esecuzione	*booleano*	
enable_parallelism	*booleano*	
sql_generation	*booleano*	
database_caching	*booleano*	
sql_logging	*booleano*	
sql_generation_logging	*booleano*	
sql_log_native	*booleano*	
sql_log_prettyprint	*booleano*	
record_count_suppress_input	*booleano*	
record_count_feedback_interval	*numero intero*	
use_stream_auto_create_node_impostazioni	*booleano*	Se true, vengono utilizzate le impostazioni specifiche del flusso; in caso contrario, vengono utilizzate le preferenze utente.
create_model_applier_for_new_modelli	*booleano*	Se true, quando un builder del modello crea un nuovo modello e non dispone di collegamenti di aggiornamento attivi, viene aggiunto un nuovo applicatore del modello. **Nota:** Se si utilizza IBM SPSS Modeler Batch versione 15, è necessario aggiungere in modo esplicito l'applicatore del modello all'interno del proprio script.

Tabella 41. Proprietà dei flussi (Continua)

Nome proprietà	Tipo di dati	Descrizione proprietà
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Definisce il tipo di collegamento creato quando un nodo applicatore del modello viene aggiunto automaticamente.
create_source_node_from_builders	*booleano*	Se true, quando un builder di origine crea un nuovo output di origine e non dispone di collegamenti di aggiornamento attivi, viene aggiunto un nuovo nodo di origine.
create_source_node_update_links	createEnabled createDisabled doNotCreate	Definisce il tipo di collegamento creato quando un nodo di origine viene aggiunto automaticamente.
has_coordinate_system	*booleano*	Se true, viene applicato un sistema di coordinate al flusso intero.
coordinate_system	*stringa*	Il nome del sistema di coordinate proiettate selezionato.
deployment_area	ModelRefresh Calcolo del punteggio Nessuno	Scegliere come si desidera eseguire la distribuzione del flusso. Se questo valore è impostato su None, non vengono utilizzate altre voci di distribuzione.
scoring_terminal_node_id	*stringa*	Scegliere il ramo di calcolo del punteggio nel flusso. Può essere un qualsiasi nodo terminale nel flusso.
scoring_node_id	*stringa*	Scegliere il nugget nel ramo di calcolo del punteggio.
model_build_node_id	*stringa*	Scegliere il nodo di modeling nel flusso.

Capitolo 9. Proprietà dei nodi origine

Proprietà comuni dei nodi origine

Di seguito vengono elencate le proprietà comuni a tutti i nodi origine, corredate da informazioni sui nodi specifici negli argomenti che seguono.

Esempio 1

```
varfilenode = modeler.script.stream().create("variablefile", "Var. File")
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
varfilenode.setKeyedPropertyValue("check", "Age", "None")
varfilenode.setKeyedPropertyValue("values", "Age", [1, 100])
varfilenode.setKeyedPropertyValue("type", "Age", "Range")
varfilenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Esempio 2

In questo script si suppone che il file di dati specificato contenga un campo denominato `Region` che rappresenta una stringa a più righe.

```
from modeler.api import StorageType
from modeler.api import MeasureType

# Create a Variable File node that reads the data set containing
# the "Region" field
varfilenode = modeler.script.stream().create("variablefile", "My Geo Data")
varfilenode.setPropertyValue("full_filename", "C:/mydata/mygeodata.csv")
varfilenode.setPropertyValue("treat_square_brackets_as_lists", True)

# Override the storage type to be a list...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region", StorageType.LIST)
# ...and specify the type if values in the list and the list depth
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region", StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Now change the measurement to indentify the field as a geospatial value...
varfilenode.setKeyedPropertyValue("measure_type", "Region", MeasureType.GEOSPATIAL)
# ...and finally specify the necessary information about the specific
# type of geospatial object
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
varfilenode.setKeyedPropertyValue("coordinate_system", "Region", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabella 42. Proprietà comuni dei nodi origine.

Nome proprietà	Tipo di dati	Descrizione proprietà

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Proprietà basata su chiavi per i ruoli del campo.
		
		Formato di utilizzo:
		`NODE.direction.FIELDNAME`
		Nota: i valori `In` e `Out` sono obsoleti. Nelle versioni future potrebbero non essere più supportati.

Tabella 42. Proprietà comuni dei nodi origine (Continua).

Nome proprietà	Tipo di dati	Descrizione proprietà
type	Range Flag Set Typeless Discrete Insieme ordinato Default	Tipo di campo. L'impostazione di questa proprietà su *Default* cancella qualsiasi impostazione di proprietà values e se value_mode è impostata su *Specifica*, verrà reimpostata su *Leggi*. Se value_mode è già impostata su *Passa* o *Leggi*, non verrà influenzata dall'impostazione type. Formato di utilizzo: NODE.type.FIELDNAME
storage	Unknown String Integer Real Time Date Timestamp	Proprietà basata su chiavi in sola lettura per il tipo di archiviazione del campo. Formato di utilizzo: NODE.storage.FIELDNAME
check	None Annulla Coerce Discard Warn Abort	Proprietà basata su chiavi per il controllo del tipo di campo e dell'intervallo. Formato di utilizzo: NODE.check.FIELDNAME
values	[*valore valore*]	Per un campo continuo (intervallo), il primo valore corrisponde al minimo e l'ultimo valore al massimo. Per i campi nominali (insieme), specificare tutti i valori. Nel caso dei campi flag, il primo valore rappresenta *falso* e l'ultimo valore rappresenta *vero*. L'impostazione automatica di questa proprietà consente di impostare la proprietà value_mode su *Specify*. L'archiviazione è determinata in base al primo valore nell'elenco, ad esempio, se il primo valore è una *stringa*, l'archiviazione viene impostata su String. Formato di utilizzo: NODE.values.FIELDNAME
value_mode	Read Pass Leggi+ Current Specify	Determina la modalità di impostazione dei valori per un campo nel passaggio di dati successivo. Formato di utilizzo: NODE.value_mode.FIELDNAME Si noti che non è possibile impostare questa proprietà direttamente su *Specify*. Per utilizzare valori specifici, impostare la proprietà values.
default_value_mode	Read Pass	Specifica il metodo di default per l'impostazione dei valori di tutti i campi. Formato di utilizzo: NODE.default_value_mode È possibile sovrascrivere questa impostazione per campi specifici utilizzando la proprietà value_mode.

Tabella 42. Proprietà comuni dei nodi origine (Continua).

Nome proprietà	Tipo di dati	Descrizione proprietà
`extend_values`	*indicatore*	Viene applicato quando `value_mode` è impostata su *Read*. Per aggiungere valori appena letti a eventuali valori esistenti per il campo, impostare su *T*. Per scartare i valori esistenti e sostituirli con i valori appena letti, impostare su *F*. Formato di utilizzo: `NODE.extend_values.FIELDNAME`
`value_labels`	*stringa*	Utilizzata per specificare l'etichetta di un valore. Tenere presente che i valori devono essere specificati prima.
`enable_missing`	*indicatore*	Se impostato su *V*, attiva la registrazione dei valori mancanti per il campo. Formato di utilizzo: `NODE.enable_missing.FIELDNAME`
`missing_values`	[*valore valore ...*]	Specifica i valori dei dati che indicano dati mancanti. Formato di utilizzo: `NODE.missing_values.FIELDNAME`
`range_missing`	*indicatore*	Quando questa proprietà è impostata su *T*, specifica se viene definito un intervallo di valori mancanti (vuoti) per un campo. Formato di utilizzo: `NODE.range_missing.FIELDNAME`
`missing_lower`	*stringa*	Se `range_missing` è impostata su vero, specifica il limite inferiore dell'intervallo di valori mancanti. Formato di utilizzo: `NODE.missing_lower.FIELDNAME`
`missing_upper`	*stringa*	Se `range_missing` è impostata su vero, specifica il limite superiore dell'intervallo di valori mancanti. Formato di utilizzo: `NODE.missing_upper.FIELDNAME`
`null_missing`	*indicatore*	Se questa proprietà è impostata su *T*, i valori null (valori non definiti, visualizzati come `$null$` nel software) vengono considerati valori mancanti. Formato di utilizzo: `NODE.null_missing.FIELDNAME`
`whitespace_missing`	*indicatore*	Quando questa proprietà è impostata su *T*, i valori contenenti solo uno spazio vuoto (spazi, tabulazioni e nuove righe) vengono considerati valori mancanti. Formato di utilizzo: `NODE.whitespace_missing.FIELDNAME`
`descrizione`	*stringa*	Utilizzata per specificare una descrizione o etichetta di campo.

Tabella 42. Proprietà comuni dei nodi origine (Continua).

Nome proprietà	Tipo di dati	Descrizione proprietà
default_include	*indicatore*	Proprietà basata su chiavi utilizzata per specificare se il comportamento di default determina il passaggio o il filtro di campi: NODE.default_include Esempio: set mynode:filternode.default_include = false
include	*indicatore*	Proprietà basata su chiavi utilizzata per determinare se i singoli campi vengono inclusi o filtrati: NODE.include.FIELDNAME.
new_name	*stringa*	
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Questa proprietà basata su chiavi è simile a type in quanto può essere utilizzata per definire la misurazione associata al campo. La differenza consiste nel fatto che, negli script Python, alla funzione setter può essere passato anche uno dei valori MeasureType mentre getter viene restituito sempre sui valori MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Per i campi di raccolta (elenchi con profondità uguale a 0), questa proprietà basata su chiavi definisce il tipo di misurazione associato ai valori sottostanti.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Per i campi geospaziali, questa proprietà basata su chiavi definisce il tipo di oggetto geospaziale rappresentato da questo campo. Questo valore deve essere coerente con la profondità di elenco dei valori.
has_coordinate_system	*booleano*	Per i campi geospaziali, questa proprietà definisce se questo campo dispone di un sistema di coordinate
coordinate_system	*stringa*	Per i campi geospazialli, questa proprietà basata su chiavi definisce il sistema di coordinate per questo campo.

Tabella 42. Proprietà comuni dei nodi origine (Continua).

Nome proprietà	Tipo di dati	Descrizione proprietà
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Questa proprietà basata su chiavi è simile a custom_storage in quanto può essere utilizzata per definire l'archiviazione di sostituzione per il campo. La differenza consiste nel fatto che, negli script Python, alla funzione setter può essere passato anche uno dei valori StorageType mentre getter viene restituito sempre sui valori StorageType.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Per i campi di elenco, questa proprietà basata su chiavi specifica il tipo di archiviazione dei valori sottostanti.
custom_list_depth	*numero intero*	Per i campi di elenco, questa proprietà basata su chiavi specifica la profondità del campo
max_list_length	*numero intero*	Disponibile solo per i dati con livello di misurazione *Geospaziale* o *Raccolta*. Impostare la lunghezza massima dell'elenco specificando il numero di elementi che l'elenco può contenere.
max_string_length	*numero intero*	Disponibile solo per i dati *typeless* ed utilizzato quando viene generato il codice SQL per creare una tabella. Immettere il valore della stringa di dimensioni maggiori nei propri dati; in questo modo, nella tabella viene generata una colonna di grandezza sufficiente per contenere la stringa.

Proprietà asimport

L'origine di Analytic Server consente di eseguire un flusso su HDFS (Hadoop Distributed File System).

Esempio

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false","","","",""])
```

Tabella 43. Proprietà asimport.

Proprietà asimport	Tipo di dati	Descrizione proprietà
data_source	*stringa*	Il nome dell'origine dati.
use_default_as	*booleano*	Se impostata su True, utilizza la connessione Analytic Server predefinita configurata nel file options.cfg del server. Se impostata su False, utilizza la connessione di questo nodo.

Tabella 43. Proprietà asimport (Continua).

Proprietà asimport	Tipo di dati	Descrizione proprietà
connection	["string","string","string", "string","string","string","string", "string","string","string", "string" ,"string"]	Una proprietà elenco che contiene i dettagli della connessione di Analytic Server. Il formato è: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] dove: is_secure_connect: indica se viene utilizzata la connessione protetta e può essere true o false. use-kerberos-auth: indica se viene utilizzata l'autenticazione kerberos e può essere true o false. enable-kerberos-debug: indica se viene utilizzata la modalità di debug dell'autenticazione kerberos; può essere true o false.

Proprietà del nodo cognosimport

Il nodo origine IBM Cognos importa i dati dai database Cognos Analytics.

Esempio

```
node = stream.create("cognosimport", "My node")
node.setPropertyValue("cognos_connection", ["http://mycogsrv1:9300/p2pd/servlet/dispatch",
 True, "", "", ""])
node.setPropertyValue("cognos_package_name", "/Public Folders/GOSALES")
node.setPropertyValue("cognos_items", ["[GreatOutdoors].[BRANCH].[BRANCH_CODE]", "[GreatOutdoors]
.[BRANCH].[COUNTRY_CODE]"])
```

Tabella 44. Proprietà del nodo cognosimport.

Proprietà del nodo cognosimport	Tipo di dati	Descrizione proprietà
mode	Data Report	Specifica se importare i dati (impostazione predefinita) o i report Cognos.

Tabella 44. Proprietà del nodo cognosimport (Continua).

Proprietà del nodo cognosimport	**Tipo di dati**	**Descrizione proprietà**
cognos_connection	*["string",flag,"string", "string" ,"string"]*	Una proprietà elenco contenente i dettagli di connessione per il server Cognos. Il formato è: ["Cognos_server_URL", login_mode, "namespace", "username", "password"] dove: Cognos_server_URL è l'URL del server Cognos contenente la sorgente. login_mode indica se viene utilizzato un accesso anonimo e può essere true o false; se impostato su true, i seguenti campi devono essere impostati su "". namespace specifica il provider di protezione per l'autenticazione utilizzato per accedere al server. username e password sono i dati utilizzati per accedere al server Cognos. Invece di login_mode, sono disponibili le seguenti modalità: • anonymousMode. Ad esempio: ['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"] • credentialMode. Ad esempio: ['Cognos_server_url', 'credentialMode', "namespace", "username", "password"] • storedCredentialMode. Ad esempio: ['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"] Dove stored_credential_name rappresenta il nome delle credenziali Cognos nel repository.
nome_package_cognos	*stringa*	Il percorso e il nome del package Cognos da cui importare gli oggetti di dati, per esempio: /Public Folders/GOSALES **Nota:** sono valide solo le barre (/).
cognos_items	*["campo","campo", ... ,"campo"]*	Il nome di uno o più oggetti di dati da importare. Il formato di *campo* è [*spaziodeinomi*].[*oggetto_query*].[*elemento_query*]
cognos_filters	*campo*	Il nome di uno o più filtri da applicare prima di importare i dati.
cognos_data_parameters	*elenco*	Valori per i parametri di prompt per i dati. Le coppie nome-valore sono racchiuse tra parentesi quadre, più coppie sono separate tra loro da virgole e l'intera stringa è racchiusa tra parentesi quadre. Formato: [["*param1*", "*valore*"],...,["*paramN*", "*valore*"]]

Tabella 44. Proprietà del nodo cognosimport (Continua).

Proprietà del nodo cognosimport	**Tipo di dati**	**Descrizione proprietà**
cognos_report_directory	*campo*	Il percorso Cognos di una cartella o un package da cui si importano i report, per esempio: /Public Folders/GOSALES **Nota:** sono valide solo le barre (/).
cognos_report_name	*campo*	Il percorso ed il nome nella posizione del report da importare.
cognos_report_parameters	*elenco*	Valori per i parametri di report. Le coppie nome-valore sono racchiuse tra parentesi quadre, più coppie sono separate tra loro da virgole e l'intera stringa è racchiusa tra parentesi quadre. Formato: [["*param1*", "*valore*"],...,["*paramN*", "*valore*"]]

proprietà databasenode

Il nodo Database può essere utilizzato per importare i dati da una varietà di altri package che utilizzano ODBC (Open Database Connectivity), incluso Microsoft SQL Server, Db2, Oracle e altri.

Esempio

```
import modeler.api
stream = modeler.script.stream()
node = stream.create("database", "My node")
node.setPropertyValue("mode", "Table")
node.setPropertyValue("query", "SELECT * FROM drug1n")
node.setPropertyValue("datasource", "Drug1n_db")
node.setPropertyValue("username", "spss")
node.setPropertyValue("password", "spss")
node.setPropertyValue("tablename", ".Drug1n")
```

Tabella 45. proprietà databasenode

proprietà databasenode	**Tipo di dati**	**Descrizione proprietà**
mode	Table Query	Specificare *Table* per connettersi a una tabella di database tramite i controlli della finestra di dialogo, oppure *Query* per eseguire una query del database selezionato tramite SQL.
datasource	*stringa*	Nome database (vedere anche la nota riportata sotto).
username	*stringa*	Dettagli connessione database (vedere anche la nota riportata sotto).
password	*stringa*	

Tabella 45. proprietà databasenode (Continua)

proprietà databasenode	Tipo di dati	Descrizione proprietà
credential	*stringa*	Nome della credenziale memorizzata in IBM SPSS Collaboration and Deployment Services. Può essere utilizzato in sostituzione delle proprietà username e password. Il nome utente e la password della credenziale devono corrispondere al nume utente e alla password richiesti per accedere al database.
use_credential		Impostare su True o False.
epassword	*stringa*	Specifica una password codificata come alternativa all'hardcoding di una password in uno script. Per ulteriori informazioni, consultare l'argomento "Creazione di una password codificata" a pagina 52. Proprietà di sola lettura durante l'esecuzione.
tablename	*stringa*	Nome della tabella a cui si desidera accedere.
strip_spaces	None Left Right Both	Opzioni per scartare gli spazi iniziali e finali nelle stringhe.
use_quotes	AsNeeded Always Never	Specifica se si desidera che i nomi delle tabelle e delle colonne vengano racchiusi tra virgolette quando le query vengono inviate al database (per esempio nel caso in cui tali nomi contengano spazi o punteggiatura).
query	*stringa*	Specifica il codice SQL per la query che si desidera effettuare.

Nota: Se il nome del database (nella proprietà datasource) contiene uno o più spazi, punti o caratteri di sottolineatura, è possibile utilizzare il formato "barra rovesciata virgolette" per considerarlo come stringa. Ad esempio: "{\"db2v9.7.6_linux\"}" o: "{\"TDATA 131\"}". Inoltre, racchiudere sempre i valori stringa datasource tra virgolette e parentesi graffe, come nell'esempio riportato di seguito: "{\"SQL Server\",spssuser,abcd1234,false}".

Nota: Se il nome del database (nella proprietà datasource) contiene degli spazi, invece delle singole proprietà per datasource, username e password, è possibile utilizzare una singola proprietà datasource nel seguente formato:

Tabella 46. Proprietà databasenode specifiche dell'origine dati

proprietà databasenode	Tipo di dati	Descrizione proprietà
datasource	*stringa*	Formato: [database_name,username,password[,true | false]] L'ultimo parametro è destinato all'uso con le password crittografate. Se è impostato su true, prima dell'uso la password verrà decrittografata.

Utilizzare questo formato anche se si sta modificando la sorgente dati; tuttavia, se si desidera soltanto modificare il nome utente o la password, è possibile utilizzare le proprietà `username` o `password`.

Proprietà datacollectionimportnode

Il nodo Data Collection Importazione dati importa dati di indagine basati sul Modello dati di Data Collection utilizzato dai prodotti di ricerca di mercato. Per utilizzare questo nodo, è necessario che sia installata Data Collection Data Library.

Esempio

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("casedata_name", "mrQvDsc")
node.setPropertyValue("casedata_source_type", "File")
node.setPropertyValue("casedata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

Tabella 47. proprietà datacollectionimportnode

Proprietà `datacollectionimportnode`	Tipo di dati	Descrizione proprietà

`metadata_name`	*stringa*	Nome dell'MDSC. Il valore speciale `DimensionsMDD` indica che deve essere utilizzato il documento metadati di Data Collection standard. Gli altri valori possibili sono:
		`mrADODsc`
		`mrI2dDsc`
		`mrLogDsc`
		`mrQdiDrsDsc`
		`mrQvDsc`
		`mrSampleReportingMDSC`
		`mrSavDsc`
		`mrSCDsc`
		`mrScriptMDSC`
		
		Il valore speciale `none` indica che non è presente alcun MDSC.

`metadata_file`	*stringa*	Nome del file nel quale sono archiviati i metadati.

Tabella 47. proprietà datacollectionimportnode (Continua)

Proprietà **datacollectionimportnode**	Tipo di dati	Descrizione proprietà
casedata_name	*stringa*	Nome del CDSC. I valori possibili sono: mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc Il valore speciale none indica che non è presente alcun CDSC.
casedata_source_type	Unknown File Folder UDL DSN	Indica il tipo di sorgente del CDSC.
casedata_file	*stringa*	Se casedata_source_type è *File*, specifica il file contenente i dati del caso.
casedata_folder	*stringa*	Se casedata_source_type è *Folder*, specifica la cartella contenente i dati del caso.
casedata_udl_string	*stringa*	Se casedata_source_type è *UDL*, specifica la stringa di connessione OLD-DB della sorgente dati contenente i dati del caso.
casedata_dsn_string	*stringa*	Se casedata_source_type è *DSN*, specifica la stringa di connessione ODBC della sorgente dati.
casedata_project	*stringa*	Durante la lettura dei dati del caso da un database di Data Collection, è possibile immettere il nome del progetto. Per tutti gli altri tipi di dati del caso, questa impostazione deve essere lasciata vuota.
version_import_mode	All Latest Specify	Definisce quante versioni devono essere gestite.
specific_version	*stringa*	Se version_import_mode è *Specify*, definisce la versione dei dati del caso da importare.
use_language	*stringa*	Definisce se debbano essere utilizzate delle etichette di una lingua specifica.
language	*stringa*	Se use_language è impostata su vero, specifica il codice della lingua da utilizzare nell'importazione. Il codice della lingua deve essere uno di quelli disponibili nei dati del caso.
use_context	*stringa*	Definisce se debba essere importato un contesto specifico. I contesti vengono utilizzati per variare la descrizione associata alle risposte.

Tabella 47. proprietà datacollectionimportnode (Continua)

Proprietà **datacollectionimportnode**	Tipo di dati	Descrizione proprietà
context	*stringa*	Se use_context è impostato su vero, definisce il contesto da importare. Il contesto deve essere uno di quelli disponibili nei dati del caso.
use_label_type	*stringa*	Definisce se debba essere importato un tipo specifico di etichetta.
label_type	*stringa*	Se use_label_type è impostato su vero, definisce il tipo di etichetta da importare. Il tipo di etichetta deve essere uno di quelli disponibili nei dati del caso.
user_id	*stringa*	Per i database che richiedono un login esplicito, è possibile fornire un ID utente e una password per accedere alla sorgente dati.
password	*stringa*	
import_system_variables	Common None All	Specifica quali variabili di sistema vengono importate.
import_codes_variables	*indicatore*	
import_sourcefile_variables	*indicatore*	
import_multi_response	MultipleFlags Single	

Proprietà dataviewimport

Il nodo Vista dati importa i dati della Vista dati in IBM SPSS Modeler.

Esempio

```
stream = modeler.script.stream()

dvnode = stream.createAt("dataviewimport", "Data View", 96, 96)
dvnode.setPropertyValue("analytic_data_source",
["","/folder/adv", "LATEST"])
dvnode.setPropertyValue("table_name", ["","com.ibm.spss.Table"])
dvnode.setPropertyValue("data_access_plan",
["","DataAccessPlan"])
dvnode.setPropertyValue("optional_attributes",
[["","NewDerivedAttribute"]])
dvnode.setPropertyValue("include_xml", True)
dvnode.setPropertyValue("include_xml_field", "xml_data")
```

Tabella 48. Proprietà dataviewimport

Proprietà dataviewimport	Tipo di dati	Descrizione proprietà
analytic_data_source	*stringa*	L'oggetto Vista dati analitici archiviato in IBM SPSS Collaboration and Deployment Services. Il nome del percorso e l'etichetta della versione per la versione da utilizzare. ["Object ID","Full path", "Version"]
table_name	*stringa*	La tabella della vista dati utilizzata nella Vista dati analitici. Il nome della tabella essere qualificato con il package.È possibile ottenere il package esportando il BOM dal client IBM SPSS Collaboration and Deployment Services Deployment Manager e ricercando nel file default.bom nell'archivio zip esportato. Il nome del package deve essere sempre uguale, a meno che il BOM non sia stato importato da IBM Operational Decision Management (iLOG). ["Object ID", "Name"]
data_access_plan	*stringa*	Il piano di accesso dati utilizzato per fornire i dati per la Vista dati analitici. ["Object ID","Name"]
optional_attributes	*stringa*	Un elenco degli attributi derivati da includere. [["ID1","Name1"], ["ID2", "Name2"]]
include_xml	*booleano*	True se deve essere incluso un campo con dati di istanza XOM. A meno che non vengano utilizzati nodi IBM Analytical Decision Management iLOG, l'impostazione consigliata è false. L'attivazione di questa proprietà può aggiungere un'elevata quantità di elaborazione supplementare.
include_xml_field	*stringa*	Il nome del campo da aggiungere quando include_xml è impostato su true.

Proprietà excelimportnode

Il nodo Importazione da Excel importa i dati da Microsoft Excel nel formato file .xlsx. Non è richiesta alcuna sorgente dati ODBC.

Esempi

```
#Per utilizzare un intervallo denominato:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("use_named_range", True)
node.setPropertyValue("named_range", "DRUG")
node.setPropertyValue("read_field_names", True)

#Per utilizzare un intervallo esplicito:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
```

```
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("worksheet_mode", "Name")
node.setPropertyValue("worksheet_name", "Drug")
node.setPropertyValue("explicit_range_start", "A1")
node.setPropertyValue("explicit_range_end", "F300")
```

Tabella 49. proprietà excelimportnode.

proprietà excelimportnode	Tipo di dati	Descrizione proprietà
excel_file_type	Excel2007	
full_filename	*stringa*	Il nome del file completo compreso il percorso.
use_named_range	*Booleano*	Indica se utilizzare o meno un intervallo denominato. Se vero, la proprietà named_range viene utilizzata per specificare l'intervallo da leggere, mentre le altre impostazioni relative al foglio di lavoro e all'intervallo dati vengono ignorate.
named_range	*stringa*	
worksheet_mode	Index Name	Specifica se il foglio di lavoro è definito in base all'indice o al nome.
worksheet_index	*numero intero*	Indice dei fogli di lavoro da leggere che inizia con 0 per il primo foglio di lavoro, 1 per il secondo e così via.
worksheet_name	*stringa*	Nome del foglio di lavoro da leggere.
data_range_mode	FirstNonBlank ExplicitRange	Specifica come viene determinato l'intervallo.
blank_rows	StopReading ReturnBlankRows	Se data_range_mode è *FirstNonBlank*, specifica come vanno gestite le righe vuote.
explicit_range_start	*stringa*	Se data_range_mode è *ExplicitRange*, specifica il punto di partenza dell'intervallo da leggere.
explicit_range_end	*stringa*	
read_field_names	*Booleano*	Specifica se la prima riga dell'intervallo specificato deve essere utilizzata come nome di campo (colonna).

Proprietà extensionimportnode

Con il nodo Importazione estensione, è possibile eseguire script R o Python for Spark per esportare i dati.

Esempio Python for Spark

```
##### Script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_importer", "extension_importer")
node.setPropertyValue("syntax_type", "Python")

python_script = """
```

```
import spss.pyspark
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

_schema = StructType([StructField('id', LongType(), nullable=False), \
StructField('age', LongType(), nullable=True), \
StructField('Sex', StringType(), nullable=True), \
StructField('BP', StringType(), nullable=True), \
StructField('Cholesterol', StringType(), nullable=True), \
StructField('K', DoubleType(), nullable=True), \
StructField('Na', DoubleType(), nullable=True), \
StructField('Drug', StringType(), nullable=True)])

if cxt.isComputeDataModelOnly():
 cxt.setSparkOutputSchema(_schema)
else:
 df = cxt.getSparkInputData()
 if df is None:
  drugList=[(1,23,'F','HIGH','HIGH',0.792535,0.031258,'drugY'), \
(2,47,'M','LOW','HIGH',0.739309,0.056468,'drugC'),\
              (3,47,'M','LOW','HIGH',0.697269,0.068944,'drugC'),\
              (4,28,'F','NORMAL','HIGH',0.563682,0.072289,'drugX'),\
              (5,61,'F','LOW','HIGH',0.559294,0.030998,'drugY'),\
              (6,22,'F','NORMAL','HIGH',0.676901,0.078647,'drugX'),\
              (7,49,'F','NORMAL','HIGH',0.789637,0.048518,'drugY'),\
              (8,41,'M','LOW','HIGH',0.766635,0.069461,'drugC'),\
              (9,60,'M','NORMAL','HIGH',0.777205,0.05123,'drugY'),\
              (10,43,'M','LOW','NORMAL',0.526102,0.027164,'drugY')]
  sqlcxt = cxt.getSparkSQLContext()
  rdd = cxt.getSparkContext().parallelize(drugList)
  print 'pyspark read data count = '+str(rdd.count())
  df = sqlcxt.createDataFrame(rdd, _schema)

 cxt.setSparkOutputData(df)
"""

node.setPropertyValue("python_syntax", python_script)
```

Esempio R

```
#### Script example for R
node.setPropertyValue("syntax_type", "R")

R_script = """# 'JSON Import' Node v1.0 for IBM SPSS Modeler
# 'RJSONIO' package created by Duncan Temple Lang - http://cran.r-project.org/web/packages/RJSONIO
# 'plyr' package created by Hadley Wickham http://cran.r-project.org/web/packages/plyr
# Node developer: Danil Savine - IBM Extreme Blue 2014
# Description: This node allows you to import into SPSS a table data from  a JSON.
# Install function for packages
packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x,character.only=TRUE)){
    install.packages(pkgs=x,repos="http://cran.r-project.org")
    require(x,character.only=TRUE)
  }
}
#  packages
packages(RJSONIO)
packages(plyr)
### This function is used to generate automatically the dataModel
getMetaData <- function (data) {
  if (dim(data)[1]<=0) {

    print("Warning : modelerData has no line, all fieldStorage fields set to strings")
    getStorage <- function(x){return("string")}
```

```
  } else {

    getStorage <- function(x) {
      res <- NULL
      #if x is a factor, typeof will return an integer so we treat the case on the side
      if(is.factor(x)) {
        res <- "string"
      } else {
        res <- switch(typeof(unlist(x)),
                      integer = "integer",
                      double = "real",
                      character = "string",
                      "string")
      }
      return (res)
    }
  }

  col = vector("list", dim(data)[2])
  for (i in 1:dim(data)[2]) {
    col[[i]] <- c(fieldName=names(data[i]),
                  fieldLabel="",
                  fieldStorage=getStorage(data[i]),
                  fieldMeasure="",
                  fieldFormat="",
                  fieldRole="")
  }
  mdm<-do.call(cbind,col)
  mdm<-data.frame(mdm)
  return(mdm)
}
# From JSON to a list
txt <- readLines('C:/test.json')
formatedtxt <- paste(txt, collapse = '')
json.list <- fromJSON(formatedtxt)
  # Apply path to json.list
  if(strsplit(x='true', split='
  ' ,fixed=TRUE)[[1]][1]) {
    path.list <- unlist(strsplit(x='id_array', split=','))
    i = 1
    while(i<length(path.list)+1){
      if(is.null(getElement(json.list, path.list[i]))){
         json.list <- json.list[[1]]
      }else{
      json.list <- getElement(json.list, path.list[i])
      i <- i+1
      }
    }
  }
# From list to dataframe via unlisted json
i <-1
filled <- data.frame()
while(i < length(json.list)+ 1){
  unlisted.json <- unlist(json.list[[i]])
  to.fill <- data.frame(t(as.data.frame(unlisted.json, row.names = names(unlisted.json))), stringsAsFactors=FALSE)
  filled <- rbind.fill(filled,to.fill)
  i <-  1 + i
}
# Export to SPSS Modeler Data
modelerData <- filled
print(modelerData)
modelerDataModel <- getMetaData(modelerData)
print(modelerDataModel)

"""

node.setPropertyValue("r_syntax", R_script)
```

Tabella 50. Proprietà extensionimportnode

Proprietà extensionimportnode	Tipo di dati	Descrizione proprietà
syntax_type	*R* *Python*	Specifica quale script viene eseguito, R o Python (R è il valore predefinito).
r_syntax	*stringa*	La sintassi di script R da eseguire.
python_syntax	*stringa*	La sintassi di script Python da eseguire.

Proprietà fixedfilenode

Il nodo Testo fisso importa dati da file di testo a campi fissi, ovvero file i cui campi non vengono delimitati ma iniziano nella stessa posizione e hanno una lunghezza fissa. Nel formato a campi fissi vengono in genere archiviati dati di versioni precedenti o generati dalla macchina.

Esempio

```
node = stream.create("fixedfile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("record_len", 32)
node.setPropertyValue("skip_header", 1)
node.setPropertyValue("fields", [["Age", 1, 3], ["Sex", 5, 7], ["BP", 9, 10], ["Cholesterol",
 12, 22], ["Na", 24, 25], ["K", 27, 27], ["Drug", 29, 32]])
node.setPropertyValue("decimal_symbol", "Period")
node.setPropertyValue("lines_to_scan", 30)
```

Tabella 51. proprietà fixedfilenode

proprietà fixedfilenode	Tipo di dati	Descrizione proprietà
record_len	*number*	Specifica il numero di caratteri in ogni record.
line_oriented	*indicatore*	Ignora il carattere di nuova riga alla fine di ogni record.
decimal_symbol	Default Comma Period	Tipo di separatore decimale utilizzato nella sorgente dati.
skip_header	*number*	Specifica il numero di righe che si desidera ignorare all'inizio del primo record. Utile per ignorare le intestazioni di colonna.
auto_recognize_datetime	*indicatore*	Specifica se data e ora vengono identificate automaticamente nei dati di origine.
lines_to_scan	*number*	
fields	*elenco*	Proprietà strutturata.
full_filename	*stringa*	Nome completo del file da leggere, inclusa la directory.
strip_spaces	None Left Right Both	Scarta gli spazi iniziali e finali nelle stringhe durante l'importazione.

Tabella 51. proprietà fixedfilenode (Continua)

proprietà **fixedfilenode**	Tipo di dati	Descrizione proprietà
invalid_char_mode	Discard Replace	Rimuove i caratteri non validi (null, 0 o qualsiasi carattere non esistente nella codifica corrente) dall'input dei dati o sostituisce i caratteri non validi con il simbolo a un carattere specificato.
invalid_char_replacement	*stringa*	
use_custom_values	*indicatore*	
custom_storage	Unknown String Integer Real Time Date Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "GG-MM-AA" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	Questa proprietà è applicabile solo se è stata specificata un'archiviazione personalizzata.
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	

Tabella 51. proprietà fixedfilenode (Continua)

proprietà fixedfilenode	Tipo di dati	Descrizione proprietà
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Questa proprietà è applicabile solo se è stata specificata un'archiviazione personalizzata.
custom_decimal_symbol	*campo*	Valida solo se è stata specificata un'archiviazione personalizzata.
encoding	StreamDefault SystemDefault "UTF-8"	Specifica il metodo di codifica del testo.

Proprietà del nodo gsdata_import

Utilizzare il nodo origine geospaziale per inserire i dati spaziali o della mappa nella propria sessione di data mining.

Tabella 52. proprietà del nodo gsdata_import

Proprietà del nodo gsdata_import	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Immettere il percorso del file .shp che si desidera caricare.
map_service_URL	*stringa*	Immettere l'URL del servizio mappa a cui connettersi.
map_name	*stringa*	Solo se si utilizza map_service_URL; questo contiene la struttura della cartella di livello superiore del servizio mappa.

Proprietà jsonimportnode

Il nodo origine JSON importa i dati da un file JSON. Consultare Nodo origine JSON per ulteriori informazioni.

Tabella 53. Proprietà jsonimportnode

Proprietà jsonimportnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Il nome del file completo compreso il percorso.
string_format	*records* *values*	Specificare il formato della stringa JSON. Il valore predefinito è records.

Proprietà sasimportnode

Il nodo File SAS importa dati SAS in IBM SPSS Modeler.

Esempio

```
node = stream.create("sasimport", "My node")
node.setPropertyValue("format", "Windows")
node.setPropertyValue("full_filename", "C:/data/retail.sas7bdat")
node.setPropertyValue("member_name", "Test")
node.setPropertyValue("read_formats", False)
node.setPropertyValue("full_format_filename", "Test")
node.setPropertyValue("import_names", True)
```

Tabella 54. proprietà sasimportnode.

proprietà sasimportnode	Tipo di dati	Descrizione proprietà
format	Windows UNIX Transport SAS7 SAS8 SAS9	Formato del file di importazione.
full_filename	*stringa*	Il nome del file completo che è stato specificato e il relativo percorso.
member_name	*stringa*	Specifica il membro da importare dal file di trasporto SAS specificato.
read_formats	*indicatore*	Legge i formati dei dati (quali etichette di variabile) dal file del formato specificato.
full_format_filename	*stringa*	
import_names	NamesAndLabels LabelsasNames	Specifica il metodo per la mappatura di nomi ed etichette di variabili durante l'importazione.

Proprietà simgennode

Il nodo Genera simulazione fornisce un modo semplice per generare dati simulati — partendo da zero utilizzando distribuzioni statistiche specificate dall'utente oppure automaticamente utilizzando le distribuzioni ottenute dall'esecuzione di un nodo Adattamento simulazione su dati cronologici esistenti. Ciò è utile quando si decide di valutare il risultato di un modello predittivo in presenza di incertezza negli input del modello.

Tabella 55. Proprietà simgennode.

Proprietà simgennode	Tipo di dati	Descrizione proprietà
fields	Proprietà strutturata	Vedere l'esempio
correlations	Proprietà strutturata	Vedere l'esempio
keep_min_max_setting	*booleano*	
refit_correlations	*booleano*	
max_cases	*numero intero*	Il valore minimo è 1000, il valore massimo è 2.147.483.647
create_iteration_field	*booleano*	
iteration_field_name	*stringa*	
replicate_results	*booleano*	
random_seed	*numero intero*	
parameter_xml	*stringa*	Restituisce il codice Xml del parametro come stringa

Esempio di fields

Questo è un parametro di slot strutturato con la seguente sintassi:

```
simgennode.setPropertyValue("fields", [
    [field1, storage, locked, [distribution1], min, max],
    [field2, storage, locked, [distribution2], min, max],
    [field3, storage, locked, [distribution3], min, max]
])
```

distribution è una dichiarazione del nome della distribuzione seguito da un elenco contenente le coppie di nomi di attributo e valori. Ciascuna distribuzione è definita nel seguente modo:

```
[distributionname, [[par1], [par2], [par3]]]
```

```
simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [["Age", "integer", False, ["Uniform",[["min","1"],["max","2"]]], "", ""]])
```

Ad esempio, per creare un nodo che genera un solo campo con una distribuzione Binomial è possibile utilizzare il seguente script:

```
simgen_node1 = modeler.script.stream().createAt("simgen", u"Sim Gen", 200, 200)
simgen_node1.setPropertyValue("fields", [["Education", "Real", False, ["Binomial", [["n", 32],
 ["prob", 0.7]]], "", ""]])
```

La distribuzione Binomial utilizza due parametri: n e prob. Poiché Binomial non supporta i valori minimo e massimo, questi vengono forniti da una stringa vuota.

Nota: Se non è possibile impostare direttamente distribution, lo si usa insieme alla proprietà fields.

I seguenti esempi mostrano tutti i tipi di distribuzione possibili. Notare che la soglia viene inserita come thresh in NegativeBinomialFailures e NegativeBinomialTrial.

```
stream = modeler.script.stream()

simgennode = stream.createAt("simgen", u"Sim Gen", 200, 200)

beta_dist = ["Field1", "Real", False, ["Beta",[["shape1","1"],["shape2","2"]]], "", ""]
binomial_dist = ["Field2", "Real", False, ["Binomial",[["n" ,"1"],["prob","1"]]], "", ""]
categorical_dist = ["Field3", "String", False, ["Categorical", [["A",0.3],["B",0.5],["C",0.2]]], "", ""]
dice_dist = ["Field4", "Real", False, ["Dice", [["1" ,"0.5"],["2","0.5"]]], "", ""]
exponential_dist = ["Field5", "Real", False, ["Exponential", [["scale","1"]]], "", ""]
fixed_dist = ["Field6", "Real", False, ["Fixed", [["value","1" ]]], "", ""]
gamma_dist = ["Field7", "Real", False, ["Gamma", [["scale","1"],["shape"," 1"]]], "", ""]
lognormal_dist = ["Field8", "Real", False, ["Lognormal", [["a","1"],["b","1" ]]], "", ""]
```

```
negbinomialfailures_dist = ["Field9", "Real", False, ["NegativeBinomialFailures",[["prob","0.5"],["thresh","1"]]], "", ""]
negbinomialtrial_dist = ["Field10", "Real", False, ["NegativeBinomialTrials",[["prob","0.2"],["thresh","1"]]], "", ""]
normal_dist = ["Field11", "Real", False, ["Normal", [["mean","1"] ,["stddev","2"]]], "", ""]
poisson_dist = ["Field12", "Real", False, ["Poisson", [["mean","1"]]], "", ""]
range_dist = ["Field13", "Real", False, ["Range", [["BEGIN","[1,3]"] ,["END","[2,4]"],["PROB","[[0.5],[0.5]]"]]], "", ""]
triangular_dist = ["Field14", "Real", False, ["Triangular", [["min","0"],["max","1"],["mode","1"]]], "", ""]
uniform_dist = ["Field15", "Real", False, ["Uniform", [["min","1"],["max","2"]]], "", ""]
weibull_dist = ["Field16", "Real", False, ["Weibull", [["a","0"],["b","1 "],["c","1"]]], "", ""]

simgennode.setPropertyValue("fields", [\
beta_dist, \
binomial_dist, \
categorical_dist, \
dice_dist, \
exponential_dist, \
fixed_dist, \
gamma_dist, \
lognormal_dist, \
negbinomialfailures_dist, \
negbinomialtrial_dist, \
normal_dist, \
poisson_dist, \
range_dist, \
triangular_dist, \
uniform_dist, \
weibull_dist
])
```

Esempio di correlations

Questo è un parametro di slot strutturato con la seguente sintassi:

```
simgennode.setPropertyValue("correlations", [
    [field1, field2, correlation],
    [field1, field3, correlation],
    [field2, field3, correlation]
])
```

La correlazione può essere qualsiasi numero compreso tra +1 e -1. È possibile specificare tutte le correlazioni desiderate. Tutte le correlazioni non specificate vengono impostate su zero. Se alcuni campi sono sconosciuti, il valore di correlazione deve essere impostato sulla matrice (o tabella) di correlazione e viene visualizzato in rosso. Quando sono presenti campi sconosciuti, non è possibile eseguire il nodo.

Proprietà statisticsimportnode

Il nodo File IBM SPSS Statistics legge i dati dal formato di file *.sav* utilizzato da IBM SPSS Statistics, nonché da file della cache salvati in IBM SPSS Modeler, che utilizzano lo stesso formato.

Le proprietà di questo nodo sono descritte in “Proprietà statisticsimportnode” a pagina 347.

Proprietà del nodo tm1odataimport

Il nodo origine IBM Cognos TM1 importa i dati dai database Cognos TM1.

Tabella 56. Proprietà del nodo tm1odataimport

Proprietà del nodo tm1odataimport	Tipo di dati	Descrizione proprietà
admin_host	*stringa*	L'URL per il nome host dell'API REST.
server_name	*stringa*	Il nome del server TM1 selezionato da admin_host.
credential_type	*inputCredential* o *storedCredential*	Utilizzata per indicare il tipo di credenziale.
input_credential	*elenco*	Quando credential_type è inputCredential; specificare il dominio, il nome utente e la password.
stored_credential_name	*stringa*	Quando credential_type è storedCredential; specificare il nome della credenziale sul server C&DS.
selected_view	*["campo" "campo"]*	Una proprietà elenco che contiene i dettagli del cubo TM1 selezionato ed il nome della vista cubo da cui i dati verranno importati in SPSS. Ad esempio: TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])
is_private_view	*indicatore*	Specifica se selected_view è una vista privata. Il valore di default è false.
selected_columns	*["campo"]*	Specifica la colonna selezionata; è possibile specificare solo un elemento. Ad esempio: setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["campo" "campo"]*	Specifica le righe selezionate. Ad esempio: setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])

Proprietà del nodo tm1import (obsoleto)

Il nodo origine IBM Cognos TM1 importa i dati dai database Cognos TM1.

Nota: Questo nodo è stato dichiarato obsoleto in Modeler 18.0. Il nome dello script del nodo sostitutivo è *tm1odataimport*.

Tabella 57. Proprietà del nodo tm1import.

Proprietà del nodo tm1import	Tipo di dati	Descrizione proprietà
pm_host	*stringa*	**Nota:** Solo per le versioni 16.0 e 17.0 Il nome host. Ad esempio: TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')

Tabella 57. Proprietà del nodo tm1import (Continua).

Proprietà del nodo tm1import	Tipo di dati	Descrizione proprietà
tm1_connection	*["campo","campo", ... ,"campo"]*	**Nota:** Solo per le versioni 16.0 e 17.0 Una proprietà elenco che contiene i dettagli di connessione per il server TM1. Il formato è: ["TM1_Server_Name","tm1_ username","tm1_ password"] Ad esempio: TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])
selected_view	*["campo" "campo"]*	Una proprietà elenco che contiene i dettagli del cubo TM1 selezionato ed il nome della vista cubo da cui i dati verranno importati in SPSS. Ad esempio: TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])
selected_column	*["campo"]*	Specifica la colonna selezionata; è possibile specificare solo un elemento. Ad esempio: setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["campo" "campo"]*	Specifica le righe selezionate. Ad esempio: setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])

Proprietà del nodo twcimport

Il nodo origine TWC importa i dati del meteo da The Company Weather, un IBM Business. È possibile utilizzarlo per ottenere dati meteorologici storici o previsionali per una località. Ciò consente di sviluppare soluzioni di business basate sul meteo risultato di un migliore processo decisionale che utilizza i dati meteorologici più accurati e precisi.

Tabella 58. Proprietà del nodo twcimport

Proprietà del nodo twcimport	Tipo di dati	Descrizione proprietà
TWCDataImport.latitude	*Reale*	Specifica un valore latitudine nel formato [-90.0⌂90.0]
TWCDataImport.longitude	*Reale*	Specifica un valore longitudine nel formato [-180.0⌂180.0].
TWCDataImport.licenseKey	*stringa*	Specifica la chiave di licenza ottenuta da The Weather Company.
TWCDataImport.measurmentUnit	English Metric Hybrid	Specifica l'unità di misurazione. I valori possibili sono English, Metric o Hybrid. Metric è il valore predefinito.

Tabella 58. Proprietà del nodo twcimport (Continua)

Proprietà del nodo twcimport	Tipo di dati	Descrizione proprietà
TWCDataImport.dataType	Historical Forecast	Specifica il tipo di dati meteorologici da immettere. I valori possibili sono Historical o Forecast. Historical è il valore predefinito.
TWCDataImport.startDate	*Numero intero*	Se si specifica Historical per TWCDataImport.dataType, specificare una data di inizio nel formato aaaaMMgg.
TWCDataImport.endDate	*Numero intero*	Se si specifica Historical per TWCDataImport.dataType, specificare una data di fine nel formato aaaaMMgg.
TWCDataImport.forecastHour	6 12 24 48	Se si specifica Forecast per TWCDataImport.dataType, specificare 6, 12, 24 o 48 per l'ora.

proprietà userinputnode

Il nodo Input utente consente di ottenere in modo semplice dati sintetici creandoli ex-novo oppure modificando dati esistenti. È utile, per esempio, quando si desidera creare un insieme di dati di test per la modellazione.

Esempio

```
node = stream.create("userinput", "My node")
node.setPropertyValue("names", ["test1", "test2"])
node.setKeyedPropertyValue("data", "test1", "2, 4, 8")
node.setKeyedPropertyValue("custom_storage", "test1", "Integer")
node.setPropertyValue("data_mode", "Ordered")
```

Tabella 59. proprietà userinputnode

proprietà userinputnode	Tipo di dati	Descrizione proprietà
data		
names		Slot strutturato che imposta o restituisce un elenco di nomi di campi generati dal nodo.
custom_storage	Unknown String Integer Real Time Date Timestamp	Slot basato su chiavi che imposta o restituisce l'archiviazione per un campo.

Tabella 59. proprietà userinputnode (Continua)

proprietà userinputnode	Tipo di dati	Descrizione proprietà
data_mode	Combined Ordered	Se è specificato Combined, i record vengono generati per ogni combinazione di valori di insieme e valori minimi e massimi. Il numero di record generati è uguale al prodotto del numero di valori in ogni campo. Se è specificato Ordered, viene preso un solo valore da ogni colonna per ogni record allo scopo di generare una riga di dati. Il numero di record generati è uguale al numero di valori più grande associato a un campo. Tutti i campi con valori di dati inferiori verranno riempiti con valori null.
values		**Nota:** Questa proprietà è obsoleta, non viene più utilizzata ed è stata sostituita dalla proprietà userinputnode.data.

Proprietà variablefilenode

Il nodo Testo variabile legge dati da file di testo a campi liberi, ovvero file i cui record contengono un numero costante di campi e un numero variabile di caratteri. Questo nodo può essere utilizzato per file con testo di intestazione a lunghezza fissa e alcuni tipi di annotazioni.

Esempio

```
node = stream.create("variablefile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("read_field_names", True)
node.setPropertyValue("delimit_other", True)
node.setPropertyValue("other", ",")
node.setPropertyValue("quotes_1", "Discard")
node.setPropertyValue("decimal_symbol", "Comma")
node.setPropertyValue("invalid_char_mode", "Replace")
node.setPropertyValue("invalid_char_replacement", "|")
node.setKeyedPropertyValue("use_custom_values", "Age", True)
node.setKeyedPropertyValue("direction", "Age", "Input")
node.setKeyedPropertyValue("type", "Age", "Range")
node.setKeyedPropertyValue("values", "Age", [1, 100])
```

Tabella 60. proprietà variablefilenode

proprietà variablefilenode	Tipo di dati	Descrizione proprietà
skip_header	*number*	Specifica il numero di caratteri che si desidera ignorare all'inizio del primo record.
num_fields_auto	*indicatore*	Determina automaticamente il numero di campi in ogni record. I record devono terminare con un carattere di nuova riga.
num_fields	*number*	Specifica manualmente il numero di campi in ogni record.
delimit_space	*indicatore*	Specifica il carattere utilizzato per delimitare i delimitatori di campo nel file.
delimit_tab	*indicatore*	

Tabella 60. proprietà variablefilenode (Continua)

proprietà variablefilenode	Tipo di dati	Descrizione proprietà
delimit_new_line	*indicatore*	
delimit_non_printing	*indicatore*	
delimit_comma	*indicatore*	Nei casi in cui la virgola è sia il delimitatore di campo che il separatore decimale dei flussi, impostare delimit_other su *true* e specificare una virgola come delimitatore utilizzando la proprietà other.
delimit_other	*indicatore*	Consente di specificare un delimitatore personalizzato utilizzando la proprietà other.
other	*stringa*	Specifica il delimitatore utilizzato quando delimit_other è *vero*.
decimal_symbol	Default Comma Period	Specifica il separatore decimale utilizzato nella sorgente dati.
multi_blank	*indicatore*	Considera più caratteri di delimitazione vuoti adiacenti come un delimitatore singolo.
read_field_names	*indicatore*	Considera la prima riga del file di dati come etichette per la colonna.
strip_spaces	None Left Right Both	Scarta gli spazi iniziali e finali nelle stringhe durante l'importazione.
invalid_char_mode	Discard Replace	Rimuove i caratteri non validi (null, 0 o qualsiasi carattere non esistente nella codifica corrente) dall'input dei dati o sostituisce i caratteri non validi con il simbolo a un carattere specificato.
invalid_char_replacement	*stringa*	
break_case_by_newline	*indicatore*	Specifica che il delimitatore di riga è un carattere di nuova riga.
lines_to_scan	*number*	Indica il numero di righe da esaminare per i tipi di dati specificati.
auto_recognize_datetime	*indicatore*	Specifica se data e ora vengono identificate automaticamente nei dati di origine.
quotes_1	Discard PairAndDiscard IncludeAsText	Specifica il trattamento delle virgolette singole durante l'importazione.
quotes_2	Discard PairAndDiscard IncludeAsText	Specifica il trattamento delle virgolette durante l'importazione.
full_filename	*stringa*	Nome completo del file da leggere, inclusa la directory.
use_custom_values	*indicatore*	

Tabella 60. proprietà variablefilenode (Continua)

proprietà variablefilenode	Tipo di dati	Descrizione proprietà
custom_storage	Unknown String Integer Real Time Date Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "GG-MM-AA" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	Valida solo se è stata specificata un'archiviazione personalizzata.
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Valida solo se è stata specificata un'archiviazione personalizzata.
custom_decimal_symbol	*campo*	Valida solo se è stata specificata un'archiviazione personalizzata.

Tabella 60. proprietà variablefilenode (Continua)

proprietà variablefilenode	Tipo di dati	Descrizione proprietà
encoding	StreamDefault SystemDefault "UTF-8"	Specifica il metodo di codifica del testo.

Proprietà xmlimportnode

Il nodo origine XML importa i dati in formato XML nel flusso. È possibile importare un singolo file oppure tutti i file in una directory. Come opzione, è possibile specificare un file schema da cui leggere la struttura XML.

Esempio

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

Tabella 61. proprietà xmlimportnode.

proprietà xmlimportnode	Tipo di dati	Descrizione proprietà
read	single directory	Legge un singolo file di dati (default), oppure tutti i file XML in una directory.
recurse	*indicatore*	Specifica se leggere anche i file XML da tutte le sottodirectory della directory specificata.
full_filename	*stringa*	(obbligatorio) Percorso e nome file completi del file XML da importare (se read = single).
directory_name	*stringa*	(obbligatorio) Percorso completo e nome della directory dalla quale importare i file XML (se read = directory).
full_schema_filename	*stringa*	Percorso e nome file completi del file XSD o DTD dal quale leggere la struttura XML. Se si omette questo parametro, la struttura viene letta dal file di origine XML.
records	*stringa*	Espressione XPath (ad esempio, /autore/nome) che indica i limiti del record. Ogni volta che si incontra questo elemento nel file di origine, viene creato un nuovo record.
mode	read specify	Leggere tutti i dati (default), oppure specificare gli elementi da leggere.
fields		Elenco di voci (elementi e attributi) da importare. Ogni voce dell'elenco è un'espressione XPath.

Capitolo 10. Proprietà dei nodi Operazioni su record

proprietà appendnode

Il nodo Accodamento concatena insiemi di record. Può essere utilizzato per combinare insiemi di dati con strutture simili ma dati diversi.

Esempio

```
node = stream.create("append", "My node")
node.setPropertyValue("match_by", "Name")
node.setPropertyValue("match_case", True)
node.setPropertyValue("include_fields_from", "All")
node.setPropertyValue("create_tag_field", True)
node.setPropertyValue("tag_field_name", "Append_Flag")
```

Tabella 62. proprietà appendnode.

Proprietà appendnode	Tipo di dati	Descrizione proprietà
match_by	Position Name	È possibile accodare insiemi di dati in base alla posizione dei campi nella sorgente dati principale o al nome dei campi nei dataset di input.
match_case	*indicatore*	Attiva la distinzione tra caratteri maiuscoli/minuscoli quando si esegue la corrispondenza tra i nomi dei campi.
include_fields_from	Main All	
create_tag_field	*indicatore*	
tag_field_name	*stringa*	

Proprietà aggregatenode

Il nodo Aggregazione sostituisce una sequenza di record di input con record di output aggregati di riepilogo.

Esempio

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
```

```
node.setPropertyValue("inc_record_count", True)
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

Tabella 63. Proprietà aggregatenode.

Proprietà aggregatenode	Tipo di dati	Descrizione proprietà
keys	*elenco*	Elenca i campi che possono essere utilizzati come chiavi per l'aggregazione. Per esempio, se Sesso e Regione sono i campi chiave disponibili, verrà generato un record aggregato per ogni combinazione univoca di M e F con le aree N e S (ovvero quattro combinazioni univoche).
contiguous	*indicatore*	Selezionare questa opzione se si sa che tutti i record con gli stessi valori chiave sono raggruppati insieme nell'input (per esempio, se l'input è ordinato in base ai campi chiave). In questo modo si migliorano le prestazioni.
aggregates		Proprietà strutturata che elenca i campi numerici i cui valori verranno aggregati e le modalità di aggregazione selezionate.
aggregate_exprs		Una proprietà le cui chiavi vengono utilizzate dal nome campo derivato con l'espressione aggregato per eseguirne il calcolo. Ad esempio: `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*stringa*	Specifica un prefisso o suffisso per campi aggregati duplicati (vedere l'esempio seguente).
add_as	Suffix Prefix	
inc_record_count	*indicatore*	Crea un campo aggiuntivo che specifica quanti record di input sono stati aggregati per formare ogni record aggregato.
count_field	*stringa*	Specifica il nome del campo conteggio record.
allow_approximation	*Booleano*	Consente l'approssimazione delle statistiche di ordinamento quando viene eseguita l'aggregazione in Analytic Server
bin_count	*numero intero*	Specifica il numero di bin da utilizzare nell'approssimazione

proprietà balancenode

Il nodo bilanciamento corregge squilibri in un insieme di dati in modo che soddisfi una determinata condizione. La direttiva di bilanciamento regola la proporzione di record in cui una condizione è vera in base al fattore specificato.

Esempio

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

Tabella 64. proprietà balancenode

Proprietà `balancenode`	Tipo di dati	Descrizione proprietà
`directives`		Proprietà strutturata per il bilanciamento della proporzione dei valori del campo in base al numero specificato (vedere l'esempio seguente).
`training_data_only`	*indicatore*	Specifica che devono essere bilanciati solo i dati di addestramento. Se nel flusso non è presente alcun campo partizione, tale opzione viene ignorata.

La proprietà di questo nodo utilizza il formato:

[[*numero, stringa*] \ [*numero, stringa*] \ ... [*numero, stringa*]].

Nota: se nell'espressione sono integrate delle stringhe (che utilizzano le virgolette), tali stringhe devono essere precedute dal carattere di escape" \ ". Il carattere " \ " è anche il carattere di continuazione della riga, che può essere utilizzato per allineare gli argomenti in modo da migliorare la leggibilità.

Proprietà cplexoptnode

Il nodo Ottimizzazione CPLEX consente di utilizzare l'ottimizzazione basata su CPLEX (complex mathematical) mediante un file di modello OPL (Optimization Programming Language). Questa funzionalità è disponibile nel prodotto IBM Analytical Decision Management, ma ora è possibile utilizzare il nodo CPLEX anche in SPSS Modeler sena richiedere IBM Analytical Decision Management.

Per ulteriori informazioni relative ad OPL ed all'ottimizzazione CPLEX, consultare la documentazione IBM Analytical Decision Management.

Tabella 65. Proprietà cplexoptnode

Proprietà `cplexoptnode`	Tipo di dati	Descrizione proprietà
`opl_model_text`	*stringa*	Il programma script OPL (Optimization Programming Language) che verrà eseguito dal nodo Ottimizzazione CPLEX e quindi genera il risultato di ottimizzazione.
`opl_tuple_set_name`	*stringa*	Il nome dell'insieme di tuple nel modello OPL che corrisponde ai dati in entrata. Non è richiesto e generalmente non è impostato mediante uno script. Dovrebbe essere utilizzato solo per modificare i mapping dei campi di un'origine dati selezionata.
`data_input_map`	*Elenco di proprietà strutturate*	I mapping del campo di input per un'origine dati. Con non è richiesto e generalmente non è impostato mediante uno script. Dovrebbe essere utilizzato solo per modificare i mapping dei campi di un'origine dati selezionata.

Tabella 65. Proprietà cplexoptnode (Continua)

Proprietà cplexoptnode	Tipo di dati	Descrizione proprietà
md_data_input_map	*Elenco di proprietà strutturate*	I mapping del campo tra ogni tupla definita in OPL con ogni origine dati del campo corrispondente (dati in ingresso). Gli utenti possono modificarli ciascuno singolarmente per l'origine dati. Con questo script, è possibile impostare la proprietà direttamente per impostare tutti i mapping in una volta. Questa impostazione non è mostrata nell'interfaccia utente. Ogni entità nell'elenco è un dato strutturato: **Tag origine dati.** La tag dell'origine dati che è possibile trovare nell'elenco a discesa delle origini dati. Ad esempio, per `0_Products_Type` la tag è `0`. **Indice origine dati.** La sequenza fisica (indice) dell'origine dati. Questa viene determinata dall'ordine di connessione. **Nodo origine.** Il nodo origine (annotazione) dell'origine dati. È possibile trovarlo nell'elenco a discesa delle origini dati. Ad esempio, per `0_Products_Type` il nodo origine è `Prodotti`. **Nodo connesso.** Il nodo precedente (annotazione) che connette il nodo Ottimizzazione CPLEX corrente. È possibile trovarlo nell'elenco a discesa delle origini dati. Ad esempio, per `0_Products_Type` il nodo connesso è `Tipo`. **Nome insieme di tuple.** Il nome dell'insieme di tuple dell'origine dati. Deve corrispondere a quello definito in OPL. **Nome campo tuple.** Il nome del campo dell'insieme di tuple dell'origine dati. Deve corrispondere a quello definito nella definizione dell'insieme di tuple. **Tipo di archiviazione.** Il tipo di archiviazione del campo. I valori possibili sono `int`, `float`,o `string`.

		Nome campo dati. Il nome campo dell'origine dati.**Esempio:**
		`[[0,0,'Product','Type','Products','prod_id_tup','int','prod_`
		`[0,0,'Product','Type','Products','prod_name_tup','string',`
		`'prod_name'],[1,1,'Components','Type','Components',`
		`'comp_id_tup','int','comp_id'],[1,1,'Components','Type',`
		`'Components','comp_name_tup','string','comp_name']]`

opl_data_text	*stringa*	La definizione di alcune variabili o dati utilizzati per OPL.

Tabella 65. Proprietà cplexoptnode (Continua)

Proprietà cplexoptnode	Tipo di dati	Descrizione proprietà
output_value_mode	*stringa*	I valori possibili sono raw o dvar. Se viene specificato If dvar, nella scheda Output l'utente deve specificare il nome della variabile della funzione oggetto in OPL per l'output. Se viene specificato raw, l'output della funzione obiettivo verrà eseguito direttamente, indipendentemente dal nome.
decision_variable_name	*stringa*	Il nome variabile della funzione obiettivo definito in OPL. È abilitata solo quando la proprietà output_value_mode è impostata su dvar.
objective_function_value_fieldname	*stringa*	Il nome campo per il valore della funzione obiettivo da utilizzare nell'output. Il valore predefinito è _OBJECTIVE.
output_tuple_set_names	*stringa*	Il nome delle tuple predefinite dai dati in ingresso. Funge da indici della variabile di decisione ed è previsto che sia l'output con gli output della variabile. La tupla di output deve essere congruente con la definizione di variabile di decisione in OPL. Se esistono più indici, i nomi tupla devono essere uniti da una virgola ,). Un esempio di una singola tupla è Prodotti, con la corrispondente definizione OPL dvar float+ Production[Products]; Un esempio di più tuple è Prodotti, componenti, con la corrispondente definizione OPL dvar float+ Production[Products][Components];
decision_output_map	*Elenco di proprietà strutturate*	La mappatura dei campi tra le variabili definite in OPL che genererà l'output e i campi di output. Ogni entità nell'elenco è un dato strutturato. **Nome variabile** Il nome variabile in OPL da generare. **Tipo di archiviazione** I valori possibili sono int, float, or string. **Nome campo di output.** Il nome campo previsto nei risultati (output o esportazione).**Esempio:** [['Production','int','res'],['Remark','string','res_1'][' 'float','res_2']]

Proprietà derive_stbnode

Il nodo STB (Space-Time-Boxes) determina le STB dai campi latitudine, longitudine e data/ora. È possibile anche identificare frequenti STB (Space-Time-Boxes) come hangout.

Esempio

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Individual Records mode
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")

# Hangouts mode
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

Tabella 66. Proprietà del nodo STB (Space-Time-Boxes)

Proprietà derive_stbnode	**Tipo di dati**	**Descrizione proprietà**
mode	IndividualRecords Hangouts	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	
hangout_density	*densità*	Una singola densità. Vedere densities per valori validi di densità.
densities	[*densità*,*densità*,..., *densità*]	Ogni densità è una stringa, ad esempio STB_GH8_1DAY. **Nota:** Vi sono limiti ai quali le densità sono valide. Per la geohash, possono essere utilizzati valori da GH1 a GH15. Per la parte temporale, possono essere utilizzati i seguenti valori: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC

Tabella 66. Proprietà del nodo STB (Space-Time-Boxes) (Continua)

Proprietà derive_stbnode	Tipo di dati	Descrizione proprietà
id_field	*campo*	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Deve essere una stringa.
min_events	*numero intero*	Il valore del numero intero minimo valido è 2.
qualifying_pct	*numero intero*	Deve essere in un intervallo tra 1 e 100.
add_extension_as	Prefix Suffix	
name_extension	*stringa*	

proprietà distinctnode

Il nodo Elimina duplicati rimuove record duplicati passando il primo record distinto nello stream di dati oppure scartando il primo record e passando nello stream tutti i duplicati.

Esempio

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

Tabella 67. proprietà distinctnode.

Proprietà distinctnode	Tipo di dati	Descrizione proprietà
mode	Include Discard	È possibile includere il primo record distinto nel flusso di dati oppure scartare il primo record distinto e passare invece tutti i record duplicati al flusso di dati.
grouping_fields	*elenco*	Elenca i campi utilizzati per stabilire se i record sono identici. **Nota:** Questa proprietà è obsoleta a partire da IBM SPSS Modeler 16 in poi.
composite_value	Slot strutturato	Vedere l'esempio seguente.

Tabella 67. proprietà distinctnode (Continua).

Proprietà distinctnode	Tipo di dati	Descrizione proprietà
composite_values	Slot strutturato	Vedere l'esempio seguente.
inc_record_count	*indicatore*	Crea un campo aggiuntivo che specifica quanti record di input sono stati aggregati per formare ogni record aggregato.
count_field	*stringa*	Specifica il nome del campo conteggio record.
sort_keys	Slot strutturato.	**Nota:** Questa proprietà è obsoleta a partire da IBM SPSS Modeler 16 in poi.
default_ascending	*indicatore*	
low_distinct_key_count	*indicatore*	Indica che i record e/o i valori univoci dei campi chiave sono in numero ridotto.
keys_pre_sorted	*indicatore*	Specifica che tutti i record con gli stessi valori chiave sono raggruppati insieme nell'input.
disable_sql_generation	*indicatore*	

Esempio per la proprietà composite_value

Di seguito è riportato il formato generale della proprietà composite_value:

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION ha il formato [FillType, Option1, Option2, ...].

Esempi:

```
node.setKeyedPropertyValue("composite_value", "Age", ["First"])
node.setKeyedPropertyValue("composite_value", "Age", ["last"])
node.setKeyedPropertyValue("composite_value", "Age", ["Total"])
node.setKeyedPropertyValue("composite_value", "Age", ["Average"])
node.setKeyedPropertyValue("composite_value", "Age", ["Min"])
node.setKeyedPropertyValue("composite_value", "Age", ["Max"])
node.setKeyedPropertyValue("composite_value", "Date", ["Earliest"])
node.setKeyedPropertyValue("composite_value", "Date", ["Latest"])
node.setKeyedPropertyValue("composite_value", "Code", ["FirstAlpha"])
node.setKeyedPropertyValue("composite_value", "Code", ["LastAlpha"])
```

Le opzioni personalizzate richiedono più di un argomento; tali argomenti vengono aggiungi come elenco, ad esempio:

```
node.setKeyedPropertyValue("composite_value", "Name", ["MostFrequent", "FirstRecord"])
node.setKeyedPropertyValue("composite_value", "Date", ["LeastFrequent", "LastRecord"])
node.setKeyedPropertyValue("composite_value", "Pending", ["IncludesValue", "T", "F"])
node.setKeyedPropertyValue("composite_value", "Marital", ["FirstMatch", "Married", "Divorced", "Separated"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Space"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Comma"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "UnderScore"])
```

Esempio per la proprietà composite_values

Di seguito è riportato il formato generale della proprietà composite_values:

```
node.setPropertyValue("composite_values", [
                      [FIELD1, [FILLOPTION1]],
                      [FIELD2, [FILLOPTION2]],
.
.
])
```

Esempio:

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

Proprietà extensionprocessnode

Con il nodo Trasforma estensione, è possibile estrarre i dati da un flusso e applicare le trasformazioni ai dati utilizzando script R o Python for Spark.

Esempio Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_process", "extension_process")
node.setPropertyValue("syntax_type", "Python")

process_script = """
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = StructType([StructField("Age", LongType(), nullable=True), \
                      StructField("Sex", StringType(), nullable=True), \
                      StructField("BP", StringType(), nullable=True), \
                      StructField("Na", DoubleType(), nullable=True), \
                      StructField("K", DoubleType(), nullable=True), \
                      StructField("Drug", StringType(), nullable=True)])
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    print df.dtypes[:]
    _newDF = df.select("Age","Sex","BP","Na","K","Drug")
    print _newDF.dtypes[:]
    cxt.setSparkOutputData(_newDF)
"""

node.setPropertyValue("python_syntax", process_script)
```

Esempio R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabella 68. Proprietà extensionprocessnode

Proprietà extensionprocessnode	Tipo di dati	Descrizione proprietà
syntax_type	*R* *Python*	Specifica quale script viene eseguito, R o Python (R è il valore predefinito).
r_syntax	*stringa*	La sintassi di script R da eseguire.
python_syntax	*stringa*	La sintassi di script Python da eseguire.
use_batch_size	*indicatore*	Attiva l'utilizzo dell'elaborazione batch.
batch_size	*numero intero*	Specifica il numero di record di dati da includere in ciascun batch.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.

proprietà mergenode

Il nodo Unione prende più record di input e crea un singolo record di output contenente tutti o alcuni campi di input. È utile per unire dati da sorgenti diverse, per esempio dati interni sui clienti e dati demografici acquistati.

Esempio

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

Tabella 69. proprietà mergenode

Proprietà **mergenode**	Tipo di dati	Descrizione proprietà

`method`	`Order`
	`Keys`
	`Condition`
	`Rankedcondition`
		Specifica se i record vengono uniti nell'ordine secondo cui sono elencati nei file di dati o se verranno utilizzati uno o più campi chiave per unire i record con lo stesso valore nei campi chiave, se i record verranno uniti nel caso venga soddisfatta una specifica condizione, oppure se è necessario unire ciascuna coppia di righe nel dataset primario ed in tutti i dataset secondari; utilizzare l'espressione di classificazione per ordinare le corrispondenze multiple dal valore più basso a quello più alto.

`condition`	*stringa*	Se `method` è impostato su `Condition`, specifica la condizione per includere o scartare i record.
`key_fields`	*elenco*	
`common_keys`	*indicatore*	

`join`	`Inner`
	`FullOuter`
	`PartialOuter`
	`Anti`	

`outer_join_tag.n`	*indicatore*	In questa proprietà, *n* è il nome del tag come viene visualizzato nella finestra di dialogo Seleziona insieme di dati. Si noti che è possibile specificare più nomi di tag, in quanto qualsiasi numero di insiemi di dati può contribuire con record incompleti.
`single_large_input`	*indicatore*	Specifica se verrà utilizzata l'ottimizzazione per avere un input relativamente grande rispetto agli altri input.
`single_large_input_tag`	*stringa*	Specifica il nome del tag come viene visualizzato nella finestra di dialogo Seleziona insieme di dati grande. Si noti che l'utilizzo di questa proprietà differisce leggermente rispetto alla proprietà `outer_join_tag` (flag e stringa) perché è possibile specificare solo un dataset di input.
`use_existing_sort_keys`	*indicatore*	Specifica se gli input sono già ordinati in base a uno o più campi chiave.
`existing_sort_keys`	[[*'stringa'*, `'Ascending'`] \ [*'stringa''*, `'Descending'`]]	Specifica i campi già ordinati e la direzione nella quale sono ordinati.
`primary_dataset`	*stringa*	Se `method` è `Rankedcondition`, selezionare il dataset principale nell'unione. Questo può essere considerato il lato sinistro di un'unione esterna.
`rename_duplicate_fields`	*Booleano*	Se `method` è `Rankedcondition`, e questo è impostato su `Y`, se il dataset risultante dall'unione contiene più campi con lo stesso nome provenienti da origini dati diverse, all'inizio delle intestazioni delle colonna del campo vengono aggiunti i tag relativi provenienti dalle origini dati.
`merge_condition`	*stringa*	
`ranking_expression`	*stringa*	

Tabella 69. proprietà mergenode (Continua)

Proprietà mergenode	Tipo di dati	Descrizione proprietà
Num_matches	*numero intero*	Il numero di corrispondenze da restituire, in base a merge_condition e ranking_expression. Minimo 1, massimo 100.

proprietà rfmaggregatenode

Il nodo Aggregazione RFM (Recency, Frequency, Monetary, Passato recente, Frequenza, Monetario) consente di prendere in considerazione i dati storici delle transazioni dei clienti, eliminare i dati non utilizzati e combinare tutti i dati delle transazioni rimanenti in un'unica riga che indica quanto tempo è trascorso dall'ultima transazione, il numero di transazioni effettuate e il valore monetario totale delle transazioni.

Esempio

```
node = stream.create("rfmaggregate", "My node")
node.setPropertyValue("relative_to", "Fixed")
node.setPropertyValue("reference_date", "2007-10-12")
node.setPropertyValue("id_field", "CardID")
node.setPropertyValue("date_field", "Date")
node.setPropertyValue("value_field", "Amount")
node.setPropertyValue("only_recent_transactions", True)
node.setPropertyValue("transaction_date_after", "2000-10-01")
```

Tabella 70. proprietà rfmaggregatenode

Proprietà rfmaggregatenode	Tipo di dati	Descrizione proprietà
relative_to	Fixed Today	Specifica la data a partire dalla quale verrà calcolato il passato recente delle transazioni.
reference_date	*data*	Disponibile solo se per relative_to viene scelto Fissa.
contiguous	*indicatore*	Se i dati sono preordinati in modo che tutti i record con lo stesso ID appaiano insieme nel flusso di dati, selezionare questa opzione per accelerare l'elaborazione.
id_field	*campo*	Specifica il campo da utilizzare per identificare il cliente e le relative transazioni.
date_field	*campo*	Specifica il campo data da utilizzare per calcolare il passato recente.
value_field	*campo*	Specifica il campo da utilizzare per calcolare il valore monetario.
extension	*stringa*	Specifica un prefisso o suffisso per campi aggregati duplicati.
add_as	Suffix Prefisso	Specifica se extension viene aggiunta come suffisso o prefisso.
discard_low_value_records	*indicatore*	Attiva l'utilizzo dell'impostazione discard_records_below.
discard_records_below	*number*	Specifica un valore minimo al di sotto del quale non vengono utilizzati i dettagli delle transazioni nel calcolo dei totali RFM. Le unità di valore sono relative al campo valore selezionato.

Tabella 70. proprietà rfmaggregatenode (Continua)

Proprietà rfmaggregatenode	Tipo di dati	Descrizione proprietà
only_recent_transactions	*indicatore*	Attiva l'utilizzo dell'impostazione specify_transaction_date o transaction_within_last.
specify_transaction_date	*indicatore*	
transaction_date_after	*data*	Disponibile solo se è selezionata specify_transaction_date. Specificare la data della transazione dopo la quale i record verranno inclusi nell'analisi.
transaction_within_last	*number*	Disponibile solo se è selezionata transaction_within_last. Specifica il numero e il tipo di periodi (giorni, settimane, mesi o anni) dalla data di Calcola passato recente relativo a dopo la quale i record saranno inclusi nell'analisi.
transaction_scale	Days Weeks Mesi Years	Disponibile solo se è selezionata transaction_within_last. Specifica il numero e il tipo di periodi (giorni, settimane, mesi o anni) dalla data di Calcola passato recente relativo a dopo la quale i record saranno inclusi nell'analisi.
save_r2	*indicatore*	Visualizza la data della seconda transazione più recente per ogni cliente.
save_r3	*indicatore*	Disponibile solo se è selezionata save_r2. Visualizza la data della terza transazione più recente per ogni cliente.

Proprietà Rprocessnode

Il nodo Trasformazioni R consente di estrarre i dati da un flusso IBM(r) SPSS(r) Modeler e di modificarli utilizzando il proprio script R personalizzato. Una volta modificati, i dati vengono restituiti al flusso.

Esempio

```
node = stream.create("rprocess", "My node")
node.setPropertyValue("custom_name", "my_node")
node.setPropertyValue("syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
node.setPropertyValue("convert_datetime", "POSIXct")
```

Tabella 71. Proprietà Rprocessnode.

Proprietà Rprocessnode	Tipo di dati	Descrizione proprietà
syntax	*stringa*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*indicatore*	

Tabella 71. Proprietà Rprocessnode (Continua).

Proprietà Rprocessnode	Tipo di dati	Descrizione proprietà
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*indicatore*	
use_batch_size	*indicatore*	Attiva l'utilizzo dell'elaborazione batch
batch_size	*numero intero*	Specifica il numero di record di dati da includere in ciascun batch

proprietà samplenode

Il nodo Campione seleziona un sottoinsieme di record. Sono supportati vari tipi di campioni, inclusi campioni stratificati, raggruppati e non casuali (strutturati). Il campionamento può essere utile per migliorare le prestazioni e per selezionare gruppi di record correlati o transazioni per un'analisi.

Esempio

```
/* Create two Sample nodes to extract
   different samples from the same data */

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"], ["M", "Normal", "Default"],
 ["F", "High", 0.3], ["F", "Normal", 0.3]])
```

Tabella 72. proprietà samplenode

Proprietà samplenode	Tipo di dati	Descrizione proprietà
method	Semplice Complesso	
mode	Include Discard	Include o scarta i record che soddisfano la condizione specificata.
sample_type	First OneInN RandomPct	Specifica il metodo di campionamento.
first_n	*numero intero*	I record fino al punto di interruzione specificato verranno inclusi o scartati.
one_in_n	*number*	Include o scarta ogni *n* record.
rand_pct	*number*	Specifica la percentuale di record da includere o scartare.
use_max_size	*indicatore*	Attiva l'utilizzo dell'impostazione maximum_size.

Tabella 72. proprietà samplenode (Continua)

Proprietà **samplenode**	Tipo di dati	Descrizione proprietà
maximum_size	*numero intero*	Specifica la dimensione massima del campione da includere nel flusso di dati o da scartare. Questa opzione è ridondante e risulta pertanto disattivata se vengono specificati Primi e Includi.
set_random_seed	*indicatore*	Attiva l'utilizzo dell'impostazione del seme random.
random_seed	*numero intero*	Specifica il valore utilizzato come seme random.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*numero intero*	
variable_proportions	*campo*	
variable_counts	*campo*	
use_min_stratum_size	*indicatore*	
minimum_stratum_size	*numero intero*	Questa opzione è valida solo quando con sample_units=Proportions viene acquisito un campione Complesso.
use_max_stratum_size	*indicatore*	
maximum_stratum_size	*numero intero*	Questa opzione è valida solo quando con sample_units=Proportions viene acquisito un campione Complesso.
clusters	*campo*	
stratify_by	*[campo1 ... campoN]*	
specify_input_weight	*indicatore*	
input_weight	*campo*	
new_output_weight	*stringa*	
sizes_proportions	[[string *valore stringa*][string *valore stringa*]...]	Se sample_units=proportions e sample_size_proportions=Custom, specifica un valore per ogni possibile combinazione di valori di campi di stratificazione.
default_proportion	*number*	
sizes_counts	[[string *valore stringa*][string *valore stringa*]...]	Specifica un valore per ogni possibile combinazione di valori di campi di stratificazione. L'utilizzo è simile a quello della proprietà sizes_proportions, con la differenza che viene specificato un numero intero anziché una proporzione.
default_count	*number*	

proprietà selectnode

Il nodo Seleziona consente di selezionare o scartare un sottoinsieme di record dal flusso dei dati basato su una condizione specifica. Per esempio, è possibile selezionare i record relativi a una determinata area vendite.

Esempio

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

Tabella 73. proprietà selectnode

Proprietà selectnode	Tipo di dati	Descrizione proprietà
mode	Include Discard	Indica se includere o scartare i record selezionati.
condition	*stringa*	Condizione per includere o scartare i record.

proprietà sortnode

Il nodo Ordina ordina record in ordine crescente o decrescente in base ai valori di uno o più campi.

Esempio

```
node = stream.create("sort", "My node")
node.setPropertyValue("keys", [["Age", "Ascending"], ["Sex", "Descending"]])
node.setPropertyValue("default_ascending", False)
node.setPropertyValue("use_existing_keys", True)
node.setPropertyValue("existing_keys", [["Age", "Ascending"]])
```

Tabella 74. proprietà sortnode

Proprietà sortnode	Tipo di dati	Descrizione proprietà
keys	*elenco*	Specifica i campi in base ai quali si desidera eseguire l'ordinamento. Se non viene specificata una direzione, viene utilizzata quella di default.
default_ascending	*indicatore*	Specifica il criterio di ordinamento di default.
use_existing_keys	*indicatore*	Specifica se l'ordinamento è ottimizzato utilizzando il criterio di ordinamento precedente per i campi già ordinati.
existing_keys		Specifica i campi già ordinati e la direzione nella quale sono ordinati. Utilizza lo stesso formato della proprietà keys.

Proprietà spacetimeboxes

Gli STB (Space-Time-Box) sono un'estensione delle posizioni di spazio con Geohash. Più specificatamente, un STB è una stringa alfanumerica che rappresenta un'area delimitata regolarmente di spazio e tempo.

Tabella 75. Proprietà spacetimeboxes

Proprietà spacetimeboxes	Tipo di dati	Descrizione proprietà
mode	*IndividualRecords* *Hangouts*	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	
densities	*[densità, densità, densità...]*	Ciascuna densità è una stringa. Ad esempio: STB_GH8_1DAY Notare che esistono limiti per cui le densità sono valide. Per geohash, è possibile utilizzare i valori da GH1-GH15. Per la parte temporale, possono essere utilizzati i seguenti valori: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2 MINS 1 MIN 30SECS 15SECS 10SECS 5 SECS 2 SECS 1SEC
field_name_extension	*stringa*	
add_extension_as	*Prefix* *Suffix*	
hangout_density	*densità*	Densità singola (vedere sopra)
id_field	*campo*	

Tabella 75. Proprietà spacetimeboxes (Continua)

Proprietà spacetimeboxes	Tipo di dati	Descrizione proprietà
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 2HOURS 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Questa deve essere una stringa.
min_events	*numero intero*	Il valore minimo è 2
qualifying_pct	*numero intero*	Deve essere compreso tra 1 e 100

proprietà dello spectralcluster

L'algoritmo Spectral Clustering © utilizza diversi autovettori per proiettare i dati in uno spazio con meno dimensioni. Quindi un algoritmo di cluster k-means viene applicato nel nuovo spazio per separare i dati in cluster. È ragionevolmente veloce per i piccoli record con molti campi e molto dispendioso dal punto di vista dei costi per i grandi set di dati. Il nodo Spectral Clustering in SPSS Modeler espone le funzioni principali e i parametri comunemente utilizzati della libreria Spectral Clustering. Il nodo è implementato in Python.

Tabella 76. proprietà dello spectralcluster

proprietà di spectralcluster	Tipo di dati	Descrizione proprietà
InputFields	*campo*	Tutti i predittori per il raggruppamento.
nCluster	*numero intero*	La dimensione del sottospazio di proiezione. Specificare un valore di 2 o più. Il valore predefinito è 8.
nInit	*numero intero*	Numero di volte in cui l'algoritmo k-means verrà eseguito con diversi semi centroidi. I risultati finali saranno il miglior risultato di n_init esecuzioni consecutive in termini di inerzia. Specificare un valore di 1 o più. Il valore predefinito è 10.
assignLabels	*stringa*	La strategia da utilizzare per assegnare le etichette nello spazio di incorporamento. Ci sono due modi per assegnare le etichette dopo l'incorporamento laplaciano. K-means può essere applicato ed è una scelta popolare – ma può anche essere sensibile all'inizializzazione. La discretizzazione è un altro approccio, che è meno sensibile all'inizializzazione casuale. Specificare kmeans o discretize. Il valore predefinito è kmeans.

Tabella 76. proprietà dello spectralcluster (Continua)

proprietà di spectralcluster	Tipo di dati	Descrizione proprietà
useStringLabel	*booleano*	Indica se utilizzare un'etichetta di stringa per il campo dell'etichetta del cluster. Se impostato su true, il valore di stringa nel parametro stringLabelPrefix verrà utilizzato come prefisso del campo dell'etichetta del cluster. Ad esempio, se il parametro stringLabelPrefix è impostato su Cluster, 'etichetta del cluster nel campo dell'etichetta del cluster sarà Cluster-1, Cluster-2, e così via.
stringLabelPrefix	*stringa*	Specificare il formato per il campo di appartenenza del cluster generato. L'appartenenza al cluster può essere indicata come una stringa con il prefisso dell'etichetta specificato (ad esempio, Cluster-1, Cluster-2, e così via) o come numero. Il valore predefinito è cluster.
randomSeed	*numero intero*	Specificare un valore per il seme casuale. Oppure utilizzare il valore predefinito di 0, che usa lo stato random globale da numpy.random.
affinity	*stringa*	Affinità è il metodo per valutare la distanza a coppie o l'affinità di serie di campioni. Specificare RBF, Nearest_neighbors, Sigmoid, Polynomial, Linear, Cosine, Laplacian, o Chi2. Il valore predefinito è RBF.
gamma	*double*	Coefficiente del Kernel per kernel RBF, Polynomial, Sigmoid, Laplacian e Chi2. Ignorato per il kernel Nearest_neighbors. L'impostazione predefinita è 1.0.
degree	*double*	Grado del kernel Polynomial. Ignorato da altri kernel. Il valore predefinito è 3.0.
coef0	*double*	Coefficiente zero per i kernel Polynomial and Sigmoid. Ignorato da altri kernel. L'impostazione predefinita è 1.0.
nNeighbors	*numero intero*	Numero di neighbors da utilizzare quando si costruisce la matrice di affinità usando il metodo dei neighbors più vicini. Ignorato per il kernel RBF. Il valore predefinito è 10.
eigenSolver	*stringa*	La strategia di scomposizione degli autovalori da utilizzare. Specificare None, Arpack, Lobpcg o Amg. Il valore predefinito è None. Amg richiede l'installazione di pyamg. Può essere più veloce su problemi molto grandi, sparsi, ma può anche portare a instabilità.
eigenTol	*double*	Criterio di arresto per la eigendecomposizione della matrice Laplacian quando si usa Arpack per eigenSolver.

Proprietà streamingtimeseries

Il nodo serie temporale streaming crea e calcola il punteggio dei modelli delle serie temporali in un'unica fase.
Nota: Questo nodo serie temporali streaming vengono sostituti nella versione 18 di SPSS Modeler.

Tabella 77. Proprietà streamingtimeseries

Proprietà streamingtimeseries	Valori	Descrizione proprietà
targets	*campo*	Il nodo Serie temporali streaming prevede uno o più obiettivi, utilizzando in via facoltativa uno o più campi di input come predittori. I campi frequenza e peso non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
candidate_inputs	[*campo1 ... campoN*]	I campi di input o predittore utilizzati dal modello.
use_period	*indicatore*	
date_time_field	*campo*	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*campo*	
period_start_value	*numero intero*	
num_days_per_week	*numero intero*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*numero intero*	
start_hour_of_day	*numero intero*	
timestamp_increments	*numero intero*	
cyclic_increments	*numero intero*	
cyclic_periods	*elenco*	

Tabella 77. Proprietà streamingtimeseries (Continua)

Proprietà streamingtimeseries	Valori	Descrizione proprietà
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*indicatore*	
cross_hour	*indicatore*	
aggregate_and_distribute	*elenco*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*numero intero*	
use_estimation_period	*indicatore*	
estimation_period	Observations Times	
date_estimation	*elenco*	Disponibile solo se si utilizza date_time_field
period_estimation	*elenco*	Disponibile solo se si utilizza use_period
observations_type	Latest Earliest	
observations_num	*numero intero*	
observations_exclude	*numero intero*	
method	ExpertModeler Exsmooth Arima	
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*indicatore*	
detect_outliers	*indicatore*	

Tabella 77. Proprietà streamingtimeseries (Continua)

Proprietà **streamingtimeseries**	Valori	Descrizione proprietà
expert_outlier_additive	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_innovational	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_transient	*indicatore*	
expert_outlier_seasonal_additive	*indicatore*	
expert_outlier_local_trend	*indicatore*	
expert_outlier_additive_patch	*indicatore*	
consider_newesmodels	*indicatore*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	
futureValue_type_method	Compute specify	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*numero intero*	
arima.d	*numero intero*	
arima.q	*numero intero*	
arima.sp	*numero intero*	
arima.sd	*numero intero*	
arima.sq	*numero intero*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*indicatore*	
tf_arima.p. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.d. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.q. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.sp. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.sd. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.

Tabella 77. Proprietà streamingtimeseries (Continua)

Proprietà streamingtimeseries	Valori	Descrizione proprietà
tf_arima.sq. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.delay. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.transformation_type. *nomecampo*	None SquareRoot NaturalLog	Per le funzioni di trasferimento.
arima_detect_outliers	*indicatore*	
arima_outlier_additive	*indicatore*	
arima_outlier_level_shift	*indicatore*	
arima_outlier_innovational	*indicatore*	
arima_outlier_transient	*indicatore*	
arima_outlier_seasonal_additive	*indicatore*	
arima_outlier_local_trend	*indicatore*	
arima_outlier_additive_patch	*indicatore*	
conf_limit_pct	*reale*	
events	*campi*	
forecastperiods	*numero intero*	
extend_records_into_future	*indicatore*	
conf_limits	*indicatore*	
noise_res	*indicatore*	

Proprietà streamingts (obsoleto)

Nota: Questo nodo Streaming serie temporali è stato dichiarato obsoleto nella versione 18 di SPSS Modeler e sostituito dal nuovo nodo Streaming serie temporali progettato per sfruttare la potenza di IBM SPSS Analytic Server ed elaborare dati di grandi dimensioni.Il nodo Streaming TS crea e calcola il punteggio dei modelli delle serie temporali in un'unica fase, senza dover utilizzare un nodo Intervalli di tempo.

Esempio

```
node = stream.create("streamingts", "My node")
node.setPropertyValue("deployment_force_rebuild", True)
node.setPropertyValue("deployment_rebuild_mode", "Count")
node.setPropertyValue("deployment_rebuild_count", 3)
node.setPropertyValue("deployment_rebuild_pct", 11)
node.setPropertyValue("deployment_rebuild_field", "Year")
```

Tabella 78. Proprietà streamingts.

Proprietà streamingts	Tipo di dati	Descrizione proprietà
custom_fields	*indicatore*	Se custom_fields=false, vengono usate le impostazione dal nodo di tipo upstream. Se custom_fields=true, targets e inputs devono essere specificati.
targets	[*campo1...campoN*]	
inputs	[*campo1...campoN*]	

Tabella 78. Proprietà streamingts (Continua).

Proprietà streamingts	Tipo di dati	Descrizione proprietà
method	ExpertModeler Exsmooth Arima	
calculate_conf	*indicatore*	
conf_limit_pct	*reale*	
use_time_intervals_node	*indicatore*	Se use_time_intervals_node=true, vengono utilizzate le impostazioni del nodo Intervalli di tempo upstream. Se use_time_intervals_node=false, è necessario specificare interval_offset_position, interval_offset e interval_type.
interval_offset_position	LastObservation LastRecord	LastObservation si riferisce alla **ultima osservazione valida**. LastRecord si riferisce a **ripresa conteggio dall'ultimo record**.
interval_offset	*number*	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*campi*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*indicatore*	
detect_outliers	*indicatore*	
expert_outlier_additive	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_innovational	*indicatore*	
expert_outlier_transient	*indicatore*	
expert_outlier_seasonal_additive	*indicatore*	
expert_outlier_local_trend	*indicatore*	
expert_outlier_additive_patch	*indicatore*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*numero intero*	Stessa proprietà del nodo modelli serie temporali

Tabella 78. Proprietà streamingts (Continua).

Proprietà streamingts	Tipo di dati	Descrizione proprietà
arima_d	*numero intero*	Stessa proprietà del nodo modelli serie temporali
arima_q	*numero intero*	Stessa proprietà del nodo modelli serie temporali
arima_sp	*numero intero*	Stessa proprietà del nodo modelli serie temporali
arima_sd	*numero intero*	Stessa proprietà del nodo modelli serie temporali
arima_sq	*numero intero*	Stessa proprietà del nodo modelli serie temporali
arima_transformation_type	None SquareRoot NaturalLog	Stessa proprietà del nodo modelli serie temporali
arima_include_constant	*indicatore*	Stessa proprietà del nodo modelli serie temporali
tf_arima_p.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_d.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_q.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_sp.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_sd.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_sq.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_delay.*fieldname*	*numero intero*	Stessa proprietà del nodo modelli serie temporali. Per le funzioni di trasferimento.
tf_arima_transformation_type.*fieldname*	None SquareRoot NaturalLog	
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*indicatore*	
arima_outlier_level_shift	*indicatore*	
arima_outlier_innovational	*indicatore*	
arima_outlier_transient	*indicatore*	
arima_outlier_seasonal_additive	*indicatore*	
arima_outlier_local_trend	*indicatore*	
arima_outlier_additive_patch	*indicatore*	
deployment_force_rebuild	*indicatore*	
deployment_rebuild_mode	Count Percent	
deployment_rebuild_count	*number*	
deployment_rebuild_pct	*number*	

Tabella 78. Proprietà streamingts (Continua).

Proprietà streamingts	Tipo di dati	Descrizione proprietà
deployment_rebuild_field	*<field>*	

Capitolo 11. Proprietà dei nodi Operazioni su campi

proprietà anonymizenode

Il nodo Anonimizza consente di mascherare i nomi o i valori dei campi, quando si utilizzano dati da includere in un modello a valle del nodo, permettendo di nascondere i dati originali. Questa funzionalità può essere utile se si desidera consentire ad altri utenti di creare modelli utilizzando dati riservati, quali nomi di clienti o altri dettagli.

Esempio

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

Tabella 79. proprietà anonymizenode

Proprietà anonymizenode	Tipo di dati	Descrizione proprietà
enable_anonymize	*indicatore*	Se impostata su True, attiva l'anonimizzazione dei valori dei campi (equivale alla selezione di **Sì** per tale campo nella colonna Anonimizza valori).
use_prefix	*indicatore*	Quando è impostata su True, viene utilizzato un prefisso personalizzato, se ne è stato definito uno. È valida per i campi che saranno anonimizzati con il metodo hash ed equivale alla selezione del pulsante di scelta **Personalizzato** nella finestra di dialogo Sostituisci valori di quel campo.
prefix	*stringa*	Equivale alla digitazione di un prefisso nella casella di testo della finestra di dialogo Sostituisci valori. Se non sono stati specificati altri valori, il prefisso di default è il valore di default.
transformation	Random Fixed	Determina se i parametri di trasformazione di un campo anonimizzato con il metodo Trasformazioni saranno casuali o fissi.
set_random_seed	*indicatore*	Quando è impostata su True, viene utilizzato il valore di seed specificato (se trasformation è impostato su Random).
random_seed	*numero intero*	Quando set_random_seed è impostata su True, questo è il seed del numero casuale.
scale	*number*	Quando transformation è impostata su Fixed, questo valore viene utilizzato per "scale by". Il valore di scala massimo in genere è 10, ma può essere ridotto per evitare l'overflow.

Tabella 79. proprietà anonymizenode (Continua)

Proprietà anonymizenode	Tipo di dati	Descrizione proprietà
translate	*number*	Quando transformation è impostata su Fixed, questo valore viene utilizzato per "translate". Il valore di translate massimo in genere è 1000, ma può essere ridotto per evitare l'overflow.

proprietà autodataprepnode

Il nodo Preparazione automatica dati (ADP) può analizzare i dati e individuare le correzioni, escludere i campi problematici o probabilmente inutili e derivare all'occorrenza nuovi attributi, migliorando le performance grazie allo screening intelligente e alle tecniche di campionamento. Il nodo si può utilizzare in modo completamente automatico, permettendogli di scegliere e di applicare le correzioni, oppure visualizzando in anteprima le modifiche prima dell'applicazione e accettandole, respingendole o modificandole a seconda dei casi.

Esempio

```
node = stream.create("autodataprep", "My node")
node.setPropertyValue("objective", "Balanced")
node.setPropertyValue("excluded_fields", "Filter")
node.setPropertyValue("prepare_dates_and_times", True)
node.setPropertyValue("compute_time_until_date", True)
node.setPropertyValue("reference_date", "Today")
node.setPropertyValue("units_for_date_durations", "Automatic")
```

Tabella 80. proprietà autodataprepnode

Proprietà autodataprepnode	Tipo di dati	Descrizione proprietà
objective	Balanced Speed Accuracy Custom	
custom_fields	*indicatore*	Se vera, consente di specificare i campi obiettivo, di input e di altro tipo per il nodo corrente. Se falsa, vengono utilizzate le impostazioni correnti di un nodo Tipo a monte.
target	*campo*	Specifica un singolo campo obiettivo.
inputs	[*campo1 ... campoN*]	I campi di input o predittore utilizzati dal modello.
use_frequency	*indicatore*	
frequency_field	*campo*	
use_weight	*indicatore*	
weight_field	*campo*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*indicatore*	Controllo dell'accesso a tutti i campi data e ora
compute_time_until_date	*indicatore*	

Tabella 80. proprietà autodataprepnode (Continua)

Proprietà autodataprepnode	Tipo di dati	Descrizione proprietà
reference_date	Today Fixed	
fixed_date	*data*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Mesi Days	
compute_time_until_time	*indicatore*	
reference_time	CurrentTime Fixed	
fixed_time	*ora*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*indicatore*	
extract_month_from_date	*indicatore*	
extract_day_from_date	*indicatore*	
extract_hour_from_time	*indicatore*	
extract_minute_from_time	*indicatore*	
extract_second_from_time	*indicatore*	
exclude_low_quality_inputs	*indicatore*	
exclude_too_many_missing	*indicatore*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*indicatore*	
maximum_number_categories	*number*	
exclude_if_large_category	*indicatore*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*indicatore*	
adjust_type_inputs	*indicatore*	
adjust_type_target	*indicatore*	
reorder_nominal_inputs	*indicatore*	
reorder_nominal_target	*indicatore*	
replace_outliers_inputs	*indicatore*	
replace_outliers_target	*indicatore*	
replace_missing_continuous_inputs	*indicatore*	
replace_missing_continuous_target	*indicatore*	
replace_missing_nominal_inputs	*indicatore*	
replace_missing_nominal_target	*indicatore*	
replace_missing_ordinal_inputs	*indicatore*	

Tabella 80. proprietà autodataprepnode (Continua)

Proprietà autodataprepnode	Tipo di dati	Descrizione proprietà
replace_missing_ordinal_target	*indicatore*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*indicatore*	
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*indicatore*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*indicatore*	
maximize_association_with_target	*indicatore*	
p_value_for_merging	*number*	
merge_ordinal_features	*indicatore*	
merge_nominal_features	*indicatore*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*indicatore*	
p_value_for_binning	*number*	
perform_feature_selection	*indicatore*	
p_value_for_selection	*number*	
perform_feature_construction	*indicatore*	
transformed_target_name_extension	*stringa*	
transformed_inputs_name_extension	*stringa*	
constructed_features_root_name	*stringa*	
years_duration_ name_extension	*stringa*	
months_duration_ name_extension	*stringa*	
days_duration_ name_extension	*stringa*	
hours_duration_ name_extension	*stringa*	
minutes_duration_ name_extension	*stringa*	
seconds_duration_ name_extension	*stringa*	
year_cyclical_name_extension	*stringa*	
month_cyclical_name_extension	*stringa*	
day_cyclical_name_extension	*stringa*	
hour_cyclical_name_extension	*stringa*	

Tabella 80. proprietà autodataprepnode (Continua)

Proprietà autodataprepnode	Tipo di dati	Descrizione proprietà
minute_cyclical_name_extension	*stringa*	
second_cyclical_name_extension	*stringa*	

Proprietà astimeintervalsnode

Utilizzare il nodo Intervalli di tempo per specificare gli intervalli e derivare un nuovo campo ora per effettuare stime o previsioni. È supportata una gamma completa di intervalli temporali, dai secondi agli anni.

Tabella 81. Proprietà astimeintervalsnode

Proprietà astimeintervalsnode	Tipo di dati	Descrizione proprietà
time_field	*campo*	Prevede sono un solo campo continuo. Tale campo viene utilizzato dal nodo come chiave di aggregazione per la conversione dell'intervallo. Se viene utilizzato un campo intero, viene considerato come un indice temporale.
dimensions	*[campo1 campo2 ... campon]*	Questi campi vengono utilizzati per creare singole serie temporali basate sui valori del campo.
fields_to_aggregate	*[campo1 campo2 ... campon]*	Questi campi vengono aggregati come parte della modifica del campo periodo di tempo. Tutti i campi non inclusi in questa selezione vengono filtrati dai dati che escono dal il nodo.

proprietà binningnode

Il nodo Discretizza crea automaticamente nuovi campi nominali (insieme) basati sui valori di uno o più campi continui (intervallo numerico) esistenti. Per esempio, è possibile trasformare un campo continuo relativo al reddito in campo categoriale contenente gruppi di reddito come deviazioni dalla media. Dopo aver creato bin per il nuovo campo, è possibile generare un nodo Ricava basato sui punti di divisione.

Esempio

```
node = stream.create("binning", "My node")
node.setPropertyValue("fields", ["Na", "K"])
node.setPropertyValue("method", "Rank")
node.setPropertyValue("fixed_width_name_extension", "_binned")
node.setPropertyValue("fixed_width_add_as", "Suffix")
node.setPropertyValue("fixed_bin_method", "Count")
node.setPropertyValue("fixed_bin_count", 10)
node.setPropertyValue("fixed_bin_width", 3.5)
node.setPropertyValue("tile10", True)
```

Tabella 82. proprietà binningnode

Proprietà binningnode	Tipo di dati	Descrizione proprietà
fields	*[campo1 campo2 ... campon]*	Campi continui (intervalli numerici) in attesa di trasformazione. È possibile eseguire la discretizzazione di più campi contemporaneamente.
method	FixedWidth EqualCount Rank SDev Optimal	Metodo utilizzato per determinare i punti di divisione per i nuovi bin di campo (categorie).
rcalculate_bins	Always IfNecessary	Specifica se i bin vengono ricalcolati e i dati collocati nel bin corrispondente ogni volta che viene eseguito il nodo o se i dati vengono semplicemente inseriti nei bin esistenti e negli eventuali nuovi bin aggiunti.
fixed_width_name_extension	*stringa*	L'estensione di default è *_BIN*.
fixed_width_add_as	Suffix Prefisso	Specifica se l'estensione viene aggiunta alla fine (suffisso) del nome del campo oppure all'inizio (prefisso). L'estensione di default è *income_BIN*.
fixed_bin_method	Width Count	
fixed_bin_count	*numero intero*	Specifica un numero intero utilizzato per determinare il numero di bin a larghezza fissa (categorie) per i nuovi campi.
fixed_bin_width	*reale*	Valore (numero intero o reale) utilizzato per calcolare la larghezza del bin.
equal_count_name_ extension	*stringa*	L'estensione di default è *_TILE*.
equal_count_add_as	Suffix Prefisso	Specifica un'estensione, un suffisso o un prefisso, utilizzata per il nome del campo generato con p-tili standard. L'estensione di default è *_TILE* preceduta da *N*, dove *N* è il numero percentile.
tile4	*indicatore*	Genera quattro bin quantile, ognuno contenente il 25% dei casi.
tile5	*indicatore*	Genera cinque bin quintile.
tile10	*indicatore*	Genera 10 bin decile.
tile20	*indicatore*	Genera 20 bin ventile.
tile100	*indicatore*	Genera 100 bin percentile.
use_custom_tile	*indicatore*	
custom_tile_name_extension	*stringa*	L'estensione di default è *_TILEN*.
custom_tile_add_as	Suffix Prefisso	
custom_tile	*numero intero*	
equal_count_method	RecordCount ValueSum	Il metodo RecordCount cerca di assegnare un numero uguale di record a ciascun bin, mentre ValueSum assegna i record in modo che la somma dei valori in ogni bin sia uguale.

Tabella 82. proprietà binningnode (Continua)

Proprietà binningnode	Tipo di dati	Descrizione proprietà
tied_values_method	Next Current Random	Specifica quali dati relativi ai valori pari merito dei bin devono essere inseriti.
rank_order	Crescente Decrescente	Questa proprietà include Ascending (il valore più basso viene indicato con 1) o Descending (il valore più alto viene indicato con 1).
rank_add_as	Suffix Prefisso	Questa opzione è applicabile a rango, rango frazionario e percentuale rango.
rango	*indicatore*	
rank_name_extension	*stringa*	L'estensione di default è *_RANK*.
rank_fractional	*indicatore*	Opzioni dei ranghi in cui il valore del nuovo campo equivale al rango diviso per la somma dei pesi dei casi non mancanti. I ranghi frazionari sono compresi nell'intervallo tra 0 e 1.
rank_fractional_name_ extension	*stringa*	L'estensione di default è *_F_RANK*.
rank_pct	*indicatore*	Ogni rango è diviso in base al numero di record con valori validi e moltiplicato per 100. I ranghi frazionari in percentuale sono compresi nell'intervallo tra 1 e 100.
rank_pct_name_extension	*stringa*	L'estensione di default è *_P_RANK*.
sdev_name_extension	*stringa*	
sdev_add_as	Suffix Prefisso	
sdev_count	One Two Three	
optimal_name_extension	*stringa*	L'estensione di default è *_OPTIMAL*.
optimal_add_as	Suffix Prefisso	
optimal_supervisor_field	*campo*	Campo scelto come campo supervisore a cui sono correlati i campi selezionati per la discretizzazione.
optimal_merge_bins	*indicatore*	Specifica che tutti i bin con conteggi di casi ridotti vengono aggiunti a bin più grandi adiacenti.
optimal_small_bin_threshold	*numero intero*	
optimal_pre_bin	*indicatore*	Indica che deve essere eseguita la discretizzazione preventiva del dataset.
optimal_max_bins	*numero intero*	Specifica un limite superiore per evitare di creare un numero eccessivamente elevato di bin.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	

Tabella 82. proprietà binningnode (Continua)

Proprietà binningnode	Tipo di dati	Descrizione proprietà
optimal_last_bin	Unbounded Bounded	

Proprietà derivenode

Il nodo Ricava modifica valori di dati o crea nuovi campi da uno o più campi esistenti. Crea campi di tipo Formula, Flag, Nominale, Stato, Conteggio e Condizionale.

Esempio 1

```
# Create and configure a Flag Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Create and configure a Conditional Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" >< @INDEX")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

Esempio 2

In questo script, si suppone siano disponibili due colonne numeriche denominate `XPos` e `YPos` che rappresentano le coordinate X e Y di un punto (ad esempio, il punto in cui si è verificato un evento). Lo script crea un nodo Ricava che calcola una colonna geospaziale dalle coordinate X e Y che rappresentano quel punto in uno specifico sistema di coordinate:

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabella 83. Proprietà derivenode

Proprietà derivenode	Tipo di dati	Descrizione proprietà
new_name	*stringa*	Nome del nuovo campo.
mode	Single Multiple	Specifica campi singoli o multipli.
fields	*elenco*	Utilizzata nella modalità Multiple solo per selezionare più campi.

Tabella 83. Proprietà derivenode (Continua)

Proprietà derivenode	Tipo di dati	Descrizione proprietà
name_extension	*stringa*	Specifica l'estensione del nome del nuovo campo.
add_as	Suffix Prefisso	Aggiunge l'estensione come prefisso (all'inizio) o come suffisso (alla fine) del nome del campo.
result_type	Formula Flag Set State Count Conditional	Sei tipi di nuovi campi che è possibile creare.
formula_expr	*stringa*	Espressione per il calcolo del nuovo valore del campo in qualsiasi nodo Ricava.
flag_expr	*stringa*	
flag_true	*stringa*	
flag_false	*stringa*	
set_default	*stringa*	
set_value_cond	*stringa*	Strutturata per fornire la condizione associata a un valore specificato.
state_on_val	*stringa*	Specifica il valore per il nuovo campo quando viene soddisfatta la condizione Attivato.
state_off_val	*stringa*	Specifica il valore per il nuovo campo quando viene soddisfatta la condizione Disattivato.
state_on_expression	*stringa*	
state_off_expression	*stringa*	
state_initial	On Off	Assegna ad ogni record del nuovo campo un valore iniziale di On o Off. Questo valore può cambiare quando viene soddisfatta ciascuna condizione.
count_initial_val	*stringa*	
count_inc_condition	*stringa*	
count_inc_expression	*stringa*	
count_reset_condition	*stringa*	
cond_if_cond	*stringa*	
cond_then_expr	*stringa*	
cond_else_expr	*stringa*	
formula_measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Questa proprietà può essere utilizzata per definire la misurazione associata al campo derivato. La funzione setter può essere passata come stringa o come uno dei valori MeasureType. La funzione getter viene sempre restituita sui valori MeasureType.

Tabella 83. Proprietà derivenode (Continua)

Proprietà derivenode	Tipo di dati	Descrizione proprietà
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Per i campi di raccolta (elenchi con profondità uguale a 0), questa proprietà definisce il tipo di misurazione associato ai valori sottostanti.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Per i campi geospaziali, questa proprietà definisce il tipo di oggetto geospaziale rappresentato da questo campo. Questo valore deve essere coerente con la profondità di elenco dei valori
has_coordinate_system	*booleano*	Per i campi geospaziali, questa proprietà definisce se questo campo dispone di un sistema di coordinate
coordinate_system	*stringa*	Per i campi geospaziali, questa proprietà definisce il sistema di coordinate per questo campo

proprietà ensemblenode

Il nodo dell'insieme combina due o più nugget del modello al fine di ottenere previsioni più precise di quelle ricavabili dai singoli modelli.

Esempio

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

Tabella 84. proprietà ensemblenode

Proprietà ensemblenode	Tipo di dati	Descrizione proprietà
ensemble_target_field	*campo*	Specifica il campo obiettivo per tutti i modelli utilizzati nell'insieme.
filter_individual_model_output	*indicatore*	Specifica se i risultati del calcolo del punteggio dei singoli modelli devono essere esclusi.
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Specifica il metodo utilizzato per determinare il punteggio dell'insieme. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo flag.

Tabella 84. proprietà ensemblenode (Continua)

Proprietà ensemblenode	Tipo di dati	Descrizione proprietà
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Specifica il metodo utilizzato per determinare il punteggio dell'insieme. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo nominale.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Se è selezionato un metodo del confronto, specifica le modalità di risoluzione delle situazioni di pari merito. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo flag.
set_voting_tie_selection	Random HighestConfidence	Se è selezionato un metodo del confronto, specifica le modalità di risoluzione delle situazioni di pari merito. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo nominale.
calculate_standard_error	*indicatore*	Se il campo obiettivo è continuo viene eseguito per default il calcolo dell'errore standard per calcolare la differenza fra i valori misurati o stimati e i valori veri e per evidenziare il grado di corrispondenza di tali stime.

proprietà fillernode

Il nodo Riempimento sostituisce valori di campo e modifica l'archiviazione. È possibile scegliere di sostituire i valori in base a una condizione CLEM, per esempio `@BLANK(@FIELD)`. In alternativa, si può scegliere di sostituire tutti i valori null o vuoti con un valore specifico. Il nodo Riempimento è utilizzato spesso in combinazione con il nodo Tipo per sostituire valori mancanti.

Esempio

```
node = stream.create("filler", "My node")
node.setPropertyValue("fields", ["Age"])
node.setPropertyValue("replace_mode", "Always")
node.setPropertyValue("condition", "(\"Age\" > 60) and (\"Sex\" = \"M\"")
node.setPropertyValue("replace_with", "\"old man\"")
```

Tabella 85. proprietà fillernode

Proprietà fillernode	Tipo di dati	Descrizione proprietà
fields	*elenco*	Campi dell'insieme di dati i cui valori saranno esaminati e sostituiti.
replace_mode	Always Conditional Blank Null BlankAndNull	È possibile sostituire tutti i valori, i valori vuoti, i valori null oppure sostituire i valori basati su una condizione specifica.
condition	*stringa*	
replace_with	*stringa*	

proprietà filternode

Il nodo Filtro filtra (ignora) campi, rinomina campi e mappa campi tra i nodi origine.

Esempio

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

Utilizzo della proprietà default_include. Si noti che l'impostazione del valore della proprietà `default_include` non include o esclude automaticamente tutti i campi, ma determina semplicemente l'impostazione di default della selezione corrente. Dal punto di vista funzionale, equivale a fare clic sul pulsante **Include i campi per default** nella finestra di dialogo Nodo Filtro. Per esempio, si supponga di eseguire lo script seguente:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

Il nodo passerà i campi *Età* e *Sesso* e scarterà tutti gli altri. Si supponga ora di eseguire di nuovo lo stesso script ma indicando due campi diversi:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

Verranno aggiunti altri due campi al filtro, per un totale di quattro campi passati (*Età*, *Sesso*, *Pressione*, *Na*). In altre parole, il fatto di reimpostare il valore di `default_include` su `Falso` non reimposta automaticamente tutti i campi.

In alternativa, se si cambia `default_include` in `Vero` utilizzando uno script o dalla finestra di dialogo Nodo Filtro, si inverte il comportamento in modo che i quattro campi sopraindicati vengano scartati anziché inclusi. In caso di dubbio, potrebbe essere utile sperimentare con i controlli della finestra di dialogo Nodo Filtro per capire questa interazione.

Tabella 86. proprietà filternode

Proprietà `filternode`	Tipo di dati	Descrizione proprietà
`default_include`	*indicatore*	Proprietà basata su chiavi utilizzata per specificare se il comportamento di default determina il passaggio o il filtro di campi: Si noti che l'impostazione di questa proprietà non include o esclude automaticamente tutti i campi, ma determina semplicemente se i campi selezionati sono inclusi o esclusi per default. Per commenti aggiuntivi, vedere l'esempio seguente.
`include`	*indicatore*	Proprietà basata su chiavi utilizzata per l'inclusione e la rimozione.
`new_name`	*stringa*	

proprietà historynode

Il nodo Cronologia crea nuovi campi contenenti dati dei campi di record precedenti. I nodi Cronologia sono utilizzati in genere per dati sequenziali, per esempio per dati di serie temporali. Prima di utilizzare un nodo Cronologia, può essere utile ordinare i dati con un nodo Ordina.

Esempio

```
node = stream.create("history", "My node")
node.setPropertyValue("fields", ["Drug"])
node.setPropertyValue("offset", 1)
node.setPropertyValue("span", 3)
node.setPropertyValue("unavailable", "Discard")
node.setPropertyValue("fill_with", "undef")
```

Tabella 87. proprietà historynode

Proprietà `historynode`	Tipo di dati	Descrizione proprietà
`fields`	*elenco*	Campi di cui si desidera creare una cronologia.
`offset`	*number*	Specifica il record che precede quello corrente dal quale si desidera estrarre i valori di campo cronologici.
`span`	*number*	Specifica il numero di record precedenti a partire dal quale si desidera estrarre i valori.

`unavailable`	`Discard`
	`Leave`
	`Fill`
		Per la gestione di record senza valori di cronologia, in genere riferiti ai primi record (all'inizio dell'insieme di dati), per i quali non esistono record precedenti da utilizzare come cronologia.

`fill_with`	`String` `Number`	Specifica il valore o la stringa da utilizzare per i record in cui non sia disponibile un valore cronologico.

proprietà partitionnode

Il nodo Partizione genera un campo partizione che suddivide i dati in sottoinsiemi separati per le fasi di addestramento, verifica e convalida della creazione del modello.

Esempio

```
node = stream.create("partition", "My node")
node.setPropertyValue("create_validation", True)
node.setPropertyValue("training_size", 33)
node.setPropertyValue("testing_size", 33)
node.setPropertyValue("validation_size", 33)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 123)
node.setPropertyValue("value_mode", "System")
```

Tabella 88. proprietà partitionnode

Proprietà partitionnode	Tipo di dati	Descrizione proprietà
new_name	*stringa*	Nome del campo partizione generato dal nodo.
create_validation	*indicatore*	Specifica se deve essere creata una partizione di convalida.
training_size	*numero intero*	Percentuale di record (0–100) da allocare alla partizione di addestramento.
testing_size	*numero intero*	Percentuale di record (0-100) da allocare alla partizione di test.
validation_size	*numero intero*	Percentuale di record (0-100) da allocare alla partizione di convalida. Viene ignorata se non viene creata alcuna partizione di convalida.
training_label	*stringa*	Etichetta per la partizione di convalida.
testing_label	*stringa*	Etichetta per la partizione di test.
validation_label	*stringa*	Etichetta per la partizione di convalida. Viene ignorata se non viene creata alcuna partizione di convalida.
value_mode	System SystemAndLabel Label	Specifica i valori utilizzati per rappresentare ogni partizione nei dati. Per esempio, il campione di addestramento può essere rappresentato dal valore intero di sistema 1, dall'etichetta Training o da una combinazione dei due 1_Training.
set_random_seed	*Booleano*	Specifica se è necessario utilizzare un seme random definito dall'utente.
random_seed	*numero intero*	Valore del seme random definito dall'utente. Per l'utilizzo di questo valore è necessario che set_random_seed sia impostata su True.
enable_sql_generation	*Booleano*	Specifica se utilizzare il push back SQL per assegnare i record alle partizioni.
unique_field		Specifica il campo di input utilizzato per garantire che i record vengano assegnati alle partizioni in modo casuale ma ripetibile. Per l'utilizzo di questo valore è necessario che enable_sql_generation sia impostata su True.

proprietà del nodo Ricodifica

Il nodo Ricodifica trasforma un insieme di valori categoriali in un altro. L'operazione di ricodifica consente di comprimere categorie o raggruppare dati per l'analisi.

Esempio

```
node = stream.create("reclassify", "My node")
node.setPropertyValue("mode", "Multiple")
node.setPropertyValue("replace_field", True)
node.setPropertyValue("field", "Drug")
node.setPropertyValue("new_name", "Chemical")
node.setPropertyValue("fields", ["Drug", "BP"])
node.setPropertyValue("name_extension", "reclassified")
node.setPropertyValue("add_as", "Prefix")
node.setKeyedPropertyValue("reclassify", "drugA", True)
node.setPropertyValue("use_default", True)
node.setPropertyValue("default", "BrandX")
node.setPropertyValue("pick_list", ["BrandX", "Placebo", "Generic"])
```

Tabella 89. proprietà del nodo Ricodifica

Proprietà reclassifynode	**Tipo di dati**	**Descrizione proprietà**
mode	Single Multiple	La modalità Single ricodifica le categorie per un campo. La modalità Multiple attiva le opzioni che consentono la trasformazione di più campi contemporaneamente.
replace_field	*indicatore*	
campo	*stringa*	Utilizzata solo in modalità Singola.
new_name	*stringa*	Utilizzata solo in modalità Singola.
fields	*[campo1 campo2 ... campon]*	Utilizzata solo in modalità Multipla.
name_extension	*stringa*	Utilizzata solo in modalità Multipla.
add_as	Suffix Prefisso	Utilizzata solo in modalità Multipla.
ricodifica	*stringa*	Proprietà strutturata per i valori dei campi.
use_default	*indicatore*	Utilizza il valore di default.
default	*stringa*	Specificare un valore di default.
pick_list	*[stringa stringa ... stringa]*	Consente all'utente di importare un elenco di nuovi valori noti per popolare l'elenco a discesa nella tabella.

proprietà reordernode

Il nodo Ordina campi definisce l'ordine naturale utilizzato per visualizzare i campi a valle. Tale ordine incide sulla visualizzazione dei campi in vari contesti, quali tabelle, elenchi e Selettore di campo. Questa operazione risulta utile se si desidera rendere più visibili i campi interessanti in insiemi di dati di grandi dimensioni.

Esempio

```
node = stream.create("reorder", "My node")
node.setPropertyValue("mode", "Custom")
node.setPropertyValue("sort_by", "Storage")
node.setPropertyValue("ascending", False)
node.setPropertyValue("start_fields", ["Age", "Cholesterol"])
node.setPropertyValue("end_fields", ["Drug"])
```

Tabella 90. proprietà reordernode

Proprietà reordernode	Tipo di dati	Descrizione proprietà
mode	Custom Auto	È possibile ordinare i valori automaticamente oppure specificare un ordine personalizzato.
sort_by	Name Type Storage	
ascending	*indicatore*	
start_fields	*[campo1 campo2 ... campon]*	Dopo questi campi vengono inseriti altri nuovi campi.
end_fields	*[campo1 campo2 ... campon]*	Prima di questi campi vengono inseriti altri nuovi campi.

Proprietà reprojectnode

In SPSS Modeler, gli elementi come le funzioni spaziali del Builder di espressioni, il nodo STP (Spatio-Temporal Prediction) ed il nodo Visualizzazione mappa utilizzano il sistema di coordinate proiettate. Utilizzare il nodo Riproiezione per modificare il sistema di coordinate dei dati importati che utilizzano un sistema di coordinate geografiche.

Tabella 91. Proprietà reprojectnode

Proprietà reprojectnode	Tipo di dati	Descrizione proprietà
reproject_fields	*[campo1 campo2 ... campon]*	Elenca tutti i campi riproiettati.
reproject_type	Streamdefault Specify	Definisce come vengono riproiettati i campi.
coordinate_system	*stringa*	Il nome del sistema di coordinate da applicare ai campi. Esempio: `set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"`

proprietà restructurenode

Il nodo Riorganizza converte un campo nominale o flag in un gruppo di campi in cui è possibile inserire i valori di un altro campo. Per esempio, dato un campo denominato *tipo di pagamento*, con valori di *credito*, *contanti* e *debito*, verrebbero creati tre nuovi campi (*credito*, *contanti*, *debito*), ognuno dei quali può contenere il valore del pagamento effettuato.

Esempio

```
node = stream.create("restructure", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("include_field_name", True)
node.setPropertyValue("value_mode", "OtherFields")
node.setPropertyValue("value_fields", ["Age", "BP"])
```

Tabella 92. proprietà restructurenode

Proprietà **restructurenode**	Tipo di dati	Descrizione proprietà
`fields_from`	[*categoria categoria categoria*] `all`	
`include_field_name`	*indicatore*	Indica se utilizzare il nome del campo nel nome del campo riorganizzato.
`value_mode`	`OtherFields` `Flags`	Indica la modalità per specificare i valori dei campi riorganizzati. Con `OtherFields`, è necessario specificare quali campi utilizzare (vedere sezione seguente). Con `Flags`, i valori sono flag numerici.
`value_fields`	*elenco*	Necessario se `value_mode` è `OtherFields`. Specifica quali campi utilizzare come campi valore.

proprietà rfmanalysisnode

Il nodo Analisi RFM (Recency, Frequency, Monetary, Passato recente, Frequenza, Monetario) consente di determinare in modo quantitativo i clienti potenzialmente migliori verificando quanto tempo è trascorso dal loro ultimo acquisto (passato recente), con quale frequenza hanno effettuato acquisti (frequenza) e quanto hanno speso per tutte le transazioni (monetario).

Esempio

```
node = stream.create("rfmanalysis", "My node")
node.setPropertyValue("recency", "Recency")
node.setPropertyValue("frequency", "Frequency")
node.setPropertyValue("monetary", "Monetary")
node.setPropertyValue("tied_values_method", "Next")
node.setPropertyValue("recalculate_bins", "IfNecessary")
node.setPropertyValue("recency_thresholds", [1, 500, 800, 1500, 2000, 2500])
```

Tabella 93. proprietà rfmanalysisnode

Proprietà **rfmanalysisnode**	Tipo di dati	Descrizione proprietà
`attualità`	*campo*	Specifica il campo Passato recente, il cui valore può essere una data, un timestamp o un semplice numero.
`frequency`	*campo*	Specifica il campo Frequenza.
`monetary`	*campo*	Specifica il campo Monetario.
`recency_bins`	*numero intero*	Specifica il numero di bin di passato recente da generare.
`recency_weight`	*number*	Specifica la ponderazione da applicare ai dati di passato recente. Il valore di default è 100.
`frequency_bins`	*numero intero*	Specifica il numero di bin di frequenza da generare.

Tabella 93. proprietà rfmanalysisnode (Continua)

Proprietà rfmanalysisnode	Tipo di dati	Descrizione proprietà
frequency_weight	*number*	Specifica la ponderazione da applicare ai dati di frequenza. Il valore di default è 10.
monetary_bins	*numero intero*	Specifica il numero di bin monetari da generare.
monetary_weight	*number*	Specifica la ponderazione da applicare ai dati monetari. L'impostazione di default è 1.
tied_values_method	Next Current	Specifica quali dati relativi ai valori pari merito dei bin devono essere inseriti.
recalculate_bins	Always IfNecessary	
add_outliers	*indicatore*	Disponibile solo se recalculate_bins è impostata su IfNecessary. Se la proprietà è impostata, i record di valore inferiore a quello del bin più basso vengono aggiunti a tale bin e quelli di valore superiore a quello del bin più alto vengono aggiunti a tale bin.
binned_field	Recency Frequency Monetary	
recency_thresholds	*valore valore*	Disponibile solo se recalculate_bins è impostata su Always. Specifica la soglia superiore e inferiore per i bin di passato recente. La soglia superiore di un bin viene utilizzata come soglia inferiore del bin successivo, esempio, [10 30 60] definirebbe due bin, il primo con soglia superiore e inferiore rispettivamente di 10 e 30 e il secondo con soglia superiore e inferiore rispettivamente di 30 e 60.
frequency_thresholds	*valore valore*	Disponibile solo se recalculate_bins è impostata su Always.
monetary_thresholds	*valore valore*	Disponibile solo se recalculate_bins è impostata su Always.

proprietà settoflagnode

Il nodo Crea flag crea campi flag in base ai valori categoriali di uno o più campi nominali.

Esempio

```
node = stream.create("settoflag", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("true_value", "1")
node.setPropertyValue("false_value", "0")
node.setPropertyValue("use_extension", True)
```

```
node.setPropertyValue("extension", "Drug_Flag")
node.setPropertyValue("add_as", "Suffix")
node.setPropertyValue("aggregate", True)
node.setPropertyValue("keys", ["Cholesterol"])
```

Tabella 94. proprietà settoflagnode

Proprietà **settoflagnode**	Tipo di dati	Descrizione proprietà
`fields_from`	[*categoria categoria categoria*] `all`	
`true_value`	*stringa*	Specifica il valore vero utilizzato dal nodo durante l'impostazione di un flag. L'impostazione di default è `T`.
`false_value`	*stringa*	Specifica il valore falso utilizzato dal nodo durante l'impostazione di un flag. L'impostazione di default è `F`.
`use_extension`	*indicatore*	Utilizzare un'estensione come suffisso o prefisso al nuovo campo flag.
`extension`	*stringa*	
`add_as`	`Suffix` `Prefisso`	Specifica se l'estensione viene aggiunta come suffisso o prefisso.
`aggregate`	*indicatore*	Raggruppa i record in base ai campi chiave. Tutti i campi flag di un gruppo vengono attivati se un record è impostato su vero.
`keys`	*elenco*	Campi chiave.

proprietà statisticstransformnode

Il nodo Trasformazioni Statistics esegue una selezione di comandi di sintassi IBM SPSS Statistics rispetto alle sorgenti dati in IBM SPSS Modeler. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Le proprietà di questo nodo sono descritte in “proprietà statisticstransformnode” a pagina 347.

Proprietà timeintervalsnode (obsoleto)

Nota: Questo nodo è stato dichiarato obsoleto nella versione 18 di SPSS Modeler e sostituito dal nuovo nodo Serie temporali.Il nodo Intervalli di tempo specifica intervalli e, se necessario, crea etichette per la modellazione di dati di serie temporali. Se i valori non sono spaziati in modo uniforme, il nodo può riempire o aggregare i valori in base alle proprie esigenze per generare un intervallo uniforme tra i record.

Esempio

```
node = stream.create("timeintervals", "My node")
node.setPropertyValue("interval_type", "SecondsPerDay")
node.setPropertyValue("days_per_week", 4)
node.setPropertyValue("week_begins_on", "Tuesday")
node.setPropertyValue("hours_per_day", 10)
node.setPropertyValue("day_begins_hour", 7)
```

```
node.setPropertyValue("day_begins_minute", 5)
node.setPropertyValue("day_begins_second", 17)
node.setPropertyValue("mode", "Label")
node.setPropertyValue("year_start", 2005)
node.setPropertyValue("month_start", "January")
node.setPropertyValue("day_start", 4)
node.setKeyedPropertyValue("pad", "AGE", "MeanOfRecentPoints")
node.setPropertyValue("agg_mode", "Specify")
node.setPropertyValue("agg_set_default", "Last")
```

Tabella 95. proprietà timeintervalsnode.

Proprietà **timeintervalsnode**	Tipo di dati	Descrizione proprietà
interval_type	None Periodi CyclicPeriods Years Trimestri Mesi DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Label Create	Specifica se si desidera identificare i record consecutivamente o creare le serie in base a un campo data, timestamp o ora specifico.
campo	*campo*	Quando si creano le serie a partire dai dati, specifica il campo che indica la data o l'ora di ciascun record.
period_start	*numero intero*	Specifica l'intervallo iniziale dei periodi o dei periodi ciclici.
cycle_start	*numero intero*	Ciclo iniziale dei periodi ciclici.
year_start	*numero intero*	Per i tipi di intervalli, ove applicabile, anno nel quale cade il primo intervallo.
quarter_start	*numero intero*	Per i tipi di intervalli, ove applicabile, trimestre nel quale cade il primo intervallo.
month_start	gennaio febbraio marzo aprile maggio giugno luglio agosto settembre ottobre novembre Dicembre	
day_start	*numero intero*	
hour_start	*numero intero*	
minute_start	*numero intero*	
second_start	*numero intero*	

Tabella 95. proprietà timeintervalsnode (Continua).

Proprietà timeintervalsnode	Tipo di dati	Descrizione proprietà
periods_per_cycle	*numero intero*	Per i periodi ciclici, numero entro ciascun ciclo.
fiscal_year_begins	gennaio febbraio marzo aprile maggio giugno luglio agosto settembre ottobre novembre Dicembre	Per gli intervalli trimestrali, specifica il mese nel quale inizia l'anno fiscale.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	Per gli intervalli periodici (giorni alla settimana, ore al giorno, minuti al giorno e secondi al giorno), specifica il giorno in cui inizia la settimana.
day_begins_hour	*numero intero*	Per gli intervalli periodici (ore al giorno, minuti al giorno e secondi al giorno), specifica l'ora in cui inizia il giorno. Può essere utilizzata in combinazione con day_begins_minute e day_begins_second per specificare un orario esatto, per esempio *8:05:01*. Vedere l'esempio di utilizzo seguente.
day_begins_minute	*numero intero*	Per gli intervalli periodici (ore al giorno, minuti al giorno e secondi al giorno), specifica il minuto in cui inizia il giorno (per esempio, il *5* in *8:05*).
day_begins_second	*numero intero*	Per gli intervalli periodici (ore al giorno, minuti al giorno e secondi al giorno), specifica il secondo in cui inizia il giorno (per esempio, il *17* in *8:05:17*).
days_per_week	*numero intero*	Per gli intervalli periodici (giorni alla settimana, ore al giorno, minuti al giorno e secondi al giorno), specifica il numero di giorni per settimana.
hours_per_day	*numero intero*	Per gli intervalli periodici (ore al giorno, minuti al giorno e secondi al giorno), specifica il numero di ore del giorno.
interval_increment	1 2 3 4 5 6 10 15 20 30	Per i minuti al giorno e i secondi al giorno, specifica il numero di minuti o secondi da incrementare per ogni record.

Tabella 95. proprietà timeintervalsnode (Continua).

Proprietà **timeintervalsnode**	Tipo di dati	Descrizione proprietà
field_name_extension	*stringa*	
field_name_extension_as_prefix	*indicatore*	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	Specifica il metodo di aggregazione per un campo.
pad	Blank MeanOfRecentPoints True False	Specifica il metodo di padding per un campo.

Tabella 95. proprietà timeintervalsnode (Continua).

Proprietà timeintervalsnode	Tipo di dati	Descrizione proprietà
agg_mode	All Specify	Specifica se aggregare o riempire tutti i campi con le funzioni di default quando necessario oppure specifica i campi e le funzioni da utilizzare.
agg_range_default	Mean Sum Mode Min Max	Specifica la funzione di default da utilizzare durante l'aggregazione dei campi continui.
agg_set_default	Mode First Last	Specifica la funzione di default da utilizzare durante l'aggregazione dei campi nominali.
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	Blank MeanOfRecentPoints	Specifica la funzione di default da utilizzare durante il padding dei campi continui.
pad_set_default	Blank MostRecentValue	
pad_flag_default	Blank True False	
max_records_to_create	*numero intero*	Specifica il numero massimo di record da creare durante il riempimento delle serie.
estimation_from_beginning	*indicatore*	
estimation_to_end	*indicatore*	
estimation_start_offset	*numero intero*	
estimation_num_holdouts	*numero intero*	
create_future_records	*indicatore*	
num_future_records	*numero intero*	
create_future_field	*indicatore*	
future_field_name	*stringa*	

proprietà transposenode

Il nodo Trasponi scambia i dati delle righe e delle colonne in modo da trasporre i campi in record e i record in campi.

Esempio

```
node = stream.create("transpose", "My node")
node.setPropertyValue("transposed_names", "Read")
node.setPropertyValue("read_from_field", "TimeLabel")
node.setPropertyValue("max_num_fields", "1000")
node.setPropertyValue("id_field_name", "ID")
```

Tabella 96. proprietà transposenode

Proprietà transposenode	Tipo di dati	Descrizione proprietà
transpose_method	*enum*	Specifica il metodo di trasposizione: Normale (normal), CASE a VAR (casetovar), o VAR a CASE (vartocase).
transposed_names	Prefix Read	Proprietà per il metodo di trasposizione Normal. È possibile generare automaticamente i nomi dei nuovi campi in base a un prefisso specificato, oppure leggerli da un campo esistente nei dati.
prefix	*stringa*	Proprietà per il metodo di trasposizione Normal.
num_new_fields	*numero intero*	Proprietà per il metodo di trasposizione Normal. Quando si utilizza un prefisso, specifica il numero massimo di nuovi campi da creare.
read_from_field	*campo*	Proprietà per il metodo di trasposizione Normal. Campo dal quale vengono letti i nomi. Deve essere un campo istanziato, altrimenti si verificherà un errore al momento dell'esecuzione del nodo.
max_num_fields	*numero intero*	Proprietà per il metodo di trasposizione Normal. Quando si leggono i nomi da un campo, specifica un limite superiore per evitare di creare un numero eccessivamente elevato di campi.
transpose_type	Numeric String Custom	Proprietà per il metodo di trasposizione Normal. Per default vengono trasposti solo i campi continui (intervalli numerici), ma è possibile scegliere un sottoinsieme personalizzato di campi numerici oppure trasporre tutti i campi stringa.
transpose_fields	*elenco*	Proprietà per il metodo di trasposizione Normal. Specifica i campi da trasporre quando viene utilizzata l'opzione Personalizzato.
id_field_name	*campo*	Proprietà per il metodo di trasposizione Normal.
index	*campo*	Proprietà per il metodo di trasposizione CASE a VAR (casetovar). Accetta più campi da utilizzare come campi di indicizzazione. field1 ... fieldN
column	*campo*	Proprietà per il metodo di trasposizione CASE a VAR (casetovar). Accetta più campi da utilizzare come campi colonna. field1 ... fieldN
valore	*campo*	Proprietà per il metodo di trasposizione CASE a VAR (casetovar). Accetta più campi da utilizzare come campi valore. field1 ... fieldN
id_variables	*campo*	Proprietà per il metodo di trasposizione VAR a CASE (vartocase). Accetta più campi da utilizzare come campi variabile ID. field1 ... fieldN
value_variables	*campo*	Proprietà per il metodo di trasposizione VAR a CASE (vartocase). Accetta più campi da utilizzare come campi variabile valore. field1 ... fieldN

proprietà typenode

Il nodo Tipo specifica proprietà e metadati di campo. Per esempio, è possibile specificare un livello di misurazione (continuo, nominale, ordinale o flag) per ogni campo, impostare opzioni relative alla gestione dei valori mancanti e dei valori null di sistema, impostare il ruolo di un campo per la modellazione, specificare le etichette di campo e valore e specificare i valori per un campo.

Esempio

```
node = stream.createAt("type", "My node", 50, 50)
node.setKeyedPropertyValue("check", "Cholesterol", "Coerce")
node.setKeyedPropertyValue("direction", "Drug", "Input")
node.setKeyedPropertyValue("type", "K", "Range")
node.setKeyedPropertyValue("values", "Drug", ["drugA", "drugB", "drugC", "drugD", "drugX",
 "drugY", "drugZ"])
node.setKeyedPropertyValue("null_missing", "BP", False)
node.setKeyedPropertyValue("whitespace_missing", "BP", False)
node.setKeyedPropertyValue("description", "BP", "Blood Pressure")
node.setKeyedPropertyValue("value_labels", "BP", [["HIGH", "High Blood Pressure"],
 ["NORMAL", "normal blood pressure"]])
```

Si noti che in alcuni casi potrebbe essere necessario istanziare il nodo Tipo per consentire il corretto funzionamento di altri nodi, quali la proprietà `fields_from` del nodo Crea flag. È possibile collegare semplicemente un nodo Tabella ed eseguirlo per istanziare i campi:

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

Tabella 97. proprietà typenode

Proprietà typenode	**Tipo di dati**	**Descrizione proprietà**

`direzione`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Proprietà basata su chiavi per i ruoli del campo.
		Nota: i valori `In` e `Out` sono obsoleti. Nelle versioni future potrebbero non essere più supportati.

`type`	`Range`
	`Flag`
	`Set`
	`Typeless`
	`Discrete`
	`OrderedSet`
	`Default`
		Livello di misurazione del campo (precedentemente definito "tipo" di campo). Se si imposta il `type` su `Default`, eventual Se `value_mode` è impostato su `Pass` o `Read`, l'impostazione di `type` non influirà su `value_mode`.
		Nota: I tipi di dati utilizzati internamente sono diversi da quelli visibili nel nodo tipologia. La corrispondenza è riportata di seguito: Range -> Continuous Set - > Nominal OrderedSet -> Ordinal Discrete- > Categorical

Tabella 97. proprietà typenode (Continua)

Proprietà typenode	Tipo di dati	Descrizione proprietà
storage	Unknown String Integer Real Time Date Timestamp	Proprietà basata su chiavi in sola lettura per il tipo di archiviazione del campo.
check	None Annulla Coerce Discard Warn Abort	Proprietà basata su chiavi per il controllo del tipo di campo e dell'intervallo.
values	[*valore valore*]	Per un campo continuo, il primo valore corrisponde al minimo e l'ultimo valore al massimo. Per i campi nominali, specificare tutti i valori. Nel caso dei campi flag, il primo valore rappresenta *falso* e l'ultimo valore rappresenta *vero*. L'impostazione automatica di questa proprietà consente di impostare la proprietà value_mode su Specify.
value_mode	Read Pass Leggi+ Current Specify	Determina la modalità di impostazione dei valori. Si noti che non è possibile impostare questa proprietà direttamente su Specify. Per utilizzare valori specifici, impostare la proprietà values.
extend_values	*indicatore*	Viene applicato quando value_mode è impostata su Read. Per aggiungere valori appena letti a eventuali valori esistenti per il campo, impostare su T. Per scartare i valori esistenti e sostituirli con i valori appena letti, impostare su F.
enable_missing	*indicatore*	Se impostato su V, attiva la registrazione dei valori mancanti per il campo.
missing_values	[*valore valore ...*]	Specifica i valori dei dati che indicano dati mancanti.
range_missing	*indicatore*	Specifica se viene definito un intervallo di valori mancanti (vuoti) per un campo.
missing_lower	*stringa*	Se range_missing è impostata su vero, specifica il limite inferiore dell'intervallo di valori mancanti.
missing_upper	*stringa*	Se range_missing è impostata su vero, specifica il limite superiore dell'intervallo di valori mancanti.
null_missing	*indicatore*	Se impostata su T, i valori *null* (valori non definiti, visualizzati come $null$ nel software) vengono considerati valori mancanti.
whitespace_ missing	*indicatore*	Se impostata su T, i valori contenenti solo uno spazio vuoto (spazi, tabulazioni e nuove righe) vengono considerati valori mancanti.
descrizione	*stringa*	Specifica la descrizione di un campo.

Tabella 97. proprietà typenode (Continua)

Proprietà typenode	Tipo di dati	Descrizione proprietà
value_labels	*[[Valore EtichettaStringa] [Valore EtichettaStringa] ...]*	Utilizzata per specificare etichette per coppie di valori.
display_places	*numero intero*	Imposta il numero di decimali del campo per la visualizzazione (valida solo per campi con archiviazione di tipo REAL). Se viene specificato il valore –1, verrà utilizzata l'impostazione di default del flusso.
export_places	*numero intero*	Imposta il numero di decimali del campo per l'esportazione (valida solo per campi con archiviazione di tipo REAL). Se viene specificato il valore –1, verrà utilizzata l'impostazione di default del flusso.
decimal_separator	DEFAULT PERIOD COMMA	Imposta il separatore decimale per il campo (valida solo per i campi con archiviazione di tipo REAL).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	Imposta il formato di data per il campo (valida solo per campi con archiviazione di tipoDATE o TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Imposta il formato di ora per il campo (valida solo per campi con archiviazione di tipo TIME o TIMESTAMP).

Tabella 97. proprietà typenode (Continua)

Proprietà typenode	Tipo di dati	Descrizione proprietà
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Imposta il formato di visualizzazione dei numeri per il campo.
standard_places	*numero intero*	Imposta il numero di decimali del campo per la visualizzazione in formato standard. Se viene specificato il valore –1, verrà utilizzata l'impostazione di default del flusso. Si noti che lo slot display_places esistente consente di ottenere lo stesso risultato, tuttavia tale configurazione è obsoleta in questa versione.
scientific_places	*numero intero*	Imposta il numero di decimali del campo per la visualizzazione in formato scientifico. Se viene specificato il valore –1, verrà utilizzata l'impostazione di default del flusso.
currency_places	*numero intero*	Imposta il numero di decimali del campo per la visualizzazione in formato valuta. Se viene specificato il valore –1, verrà utilizzata l'impostazione di default del flusso.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Imposta il simbolo di raggruppamento per il campo.
column_width	*numero intero*	Imposta la larghezza delle colonne per il campo. Se viene specificato il valore –1, la larghezza delle colonne verrà impostata su Auto.
justify	AUTO CENTER LEFT RIGHT	Imposta la giustificazione delle colonne per il campo.
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Questa proprietà basata su chiavi è simile a type in quanto può essere utilizzata per definire la misurazione associata al campo. La differenza consiste nel fatto che, negli script Python, alla funzione setter può essere passato anche uno dei valori MeasureType mentre getter viene restituito sempre sui valori MeasureType.
collection_ measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Per i campi di raccolta (elenchi con profondità uguale a 0), questa proprietà basata su chiavi definisce il tipo di misurazione associato ai valori sottostanti.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Per i campi geospaziali, questa proprietà basata su chiavi definisce il tipo di oggetto geospaziale rappresentato da questo campo. Questo valore deve essere coerente con la profondità di elenco dei valori.

Tabella 97. proprietà typenode (Continua)

Proprietà typenode	Tipo di dati	Descrizione proprietà
`has_coordinate_system`	*booleano*	Per i campi geospaziali, questa proprietà definisce se questo campo dispone di un sistema di coordinate
`coordinate_system`	*stringa*	Per i campi geospazialli, questa proprietà basata su chiavi definisce il sistema di coordinate per questo campo.

`custom_storage_type`	`Unknown / MeasureType.UNKNOWN`
	`String / MeasureType.STRING`
	`Integer / MeasureType.INTEGER`
	`Real / MeasureType.REAL`
	`Time / MeasureType.TIME`
	`Date / MeasureType.DATE`
	`Timestamp / MeasureType.TIMESTAMP`
	`List / MeasureType.LIST`
		Questa proprietà basata su chaiavi è simile a `custom_storage` in quanto può essere utilizzata per definire l'archiviazione di sostituzione per il campo. La differenza consiste nel fatto che, negli script Python, alla funzione setter può essere passato anche uno dei valori `StorageType` mentre getter viene restituito sempre sui valori `StorageType`.

`custom_list_storage_type`	`String / MeasureType.STRING`
	`Integer / MeasureType.INTEGER`
	`Real / MeasureType.REAL`
	`Time / MeasureType.TIME`
	`Date / MeasureType.DATE`
	`Timestamp / MeasureType.TIMESTAMP`
		Per i campi di elenco, questa proprietà basata su chiavi specifica il tipo di archiviazione dei valori sottostanti.

`custom_list_depth`	*numero intero*	Per i campi di elenco, questa proprietà basata su chiavi specifica la profondità del campo
`max_list_length`	*numero intero*	Disponibile solo per i dati con livello di misurazione *Geospaziale* o *Raccolta*. Impostare la lunghezza massima dell'elenco specificando il numero di elementi che l'elenco può contenere.
`max_string_length`	*numero intero*	Disponibile solo per i dati *typeless* ed utilizzato quando viene generato il codice SQL per creare una tabella. Immettere il valore della stringa di dimensioni maggiori nei propri dati; in questo modo, nella tabella viene generata una colonna di grandezza sufficiente per contenere la stringa.

Capitolo 12. Proprietà dei nodi Grafici

Proprietà comuni dei nodi Grafici

In questa sezione vengono illustrate le proprietà disponibili per i nodi Grafici, incluse le proprietà comuni e quelle specifiche per ogni tipo di nodo.

Tabella 98. Proprietà comuni dei nodi Grafici

Proprietà comuni dei nodi Grafici	Tipo di dati	Descrizione proprietà
`title`	*stringa*	Specifica il titolo. Esempio: "Questo è un titolo".
`caption`	*stringa*	Specifica la didascalia. Esempio: "Questa è una didascalia".
`output_mode`	`Screen` `File`	Specifica se l'output del nodo Grafico viene visualizzato o scritto su un file.

`output_format`	`BMP`
	`JPEG`
	`PNG`
	`HTML`
	`output` (*.cou*)
		Specifica il tipo di output. Il tipo di output consentito varia da nodo a nodo.

`full_filename`	*stringa*	Specifica il percorso di destinazione e il nome file per l'output generato dal nodo Grafico.
`use_graph_size`	*indicatore*	Controlla se le dimensioni del grafico vengono indicate esplicitamente, utilizzando le proprietà di larghezza e altezza seguenti. Influisce solo sui grafici per i quali viene visualizzato l'output su schermo. Non disponibile per il nodo distribuzione.
`graph_width`	*number*	Quando `use_graph_size` è `Vero`, imposta la larghezza del grafico in pixel.
`graph_height`	*number*	Quando `use_graph_size` è `Vero`, imposta l'altezza del grafico in pixel.

Disattivazione dei campi facoltativi

È possibile disattivare i campi facoltativi, per esempio un campo di sovrapposizione per i plot, impostando il valore della proprietà su " " (stringa vuota), come illustrato nell'esempio seguente.

```
plotnode.setPropertyValue("color_field", "")
```

Specifica dei colori

Il colore dei titoli, delle didascalie, degli sfondi e delle etichette può essere specificato utilizzando le stringhe esadecimali che iniziano con il simbolo del cancelletto (#). Per esempio, per impostare uno sfondo di colore azzurro per il grafico è possibile utilizzare l'istruzione seguente:

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

Le prime due cifre, 87, specificano il contenuto rosso; le due cifre intermedie, CE, specificano il contenuto verde e le ultime due cifre, EB, specificano il contenuto blu. Ogni cifra può avere un valore compreso nell'intervallo 0-9 o A-F. Utilizzando la combinazione di questi valori è possibile specificare un colore RGB (red-green-blue).

Nota: quando si specificano colori RGB, è possibile utilizzare il selettore di campo disponibile nell'interfaccia utente per definire il codice di colore corretto. È sufficiente posizionare il puntatore del mouse sul colore per visualizzare una descrizione contenente le informazioni desiderate.

Proprietà collectionnode

Il nodo Raccolta mostra la distribuzione dei valori di un campo numerico in relazione ai valori di un altro, ovvero crea grafici simili a istogrammi. È utile per illustrare una variabile o un campo i cui valori vengono modificati nel tempo. La grafica 3-D consente inoltre di includere un asse simbolico che visualizza le distribuzioni per categoria.

Esempio

```
node = stream.create("collection", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

Tabella 99. proprietà collectionnode

Proprietà **collectionnode**	Tipo di dati	Descrizione proprietà
over_field	*campo*	
over_label_auto	*indicatore*	
over_label	*stringa*	
collect_field	*campo*	
collect_label_auto	*indicatore*	
collect_label	*stringa*	
three_D	*indicatore*	
by_field	*campo*	
by_label_auto	*indicatore*	
by_label	*stringa*	
operation	Sum Mean Min Max SDev	
color_field	*stringa*	
panel_field	*stringa*	

Tabella 99. proprietà collectionnode (Continua)

Proprietà collectionnode	Tipo di dati	Descrizione proprietà
animation_field	*stringa*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*indicatore*	
graph_background	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
page_background	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.

Proprietà distributionnode

Il nodo distribuzione mostra l'occorrenza di valori simbolici (categoriali), per esempio tipo o genere di ipoteca. In genere è possibile utilizzare un nodo distribuzione per mostrare squilibri nei dati, che possono essere successivamente corretti con un nodo bilanciamento prima di creare un modello.

Esempio

```
node = stream.create("distribution", "My node")
# "Plot" tab
node.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

Tabella 100. proprietà distributionnode

Proprietà distributionnode	Tipo di dati	Descrizione proprietà
plot	SelectedFields Flags	
x_field	*campo*	
color_field	*campo*	Campo sovrapposto.
normalize	*indicatore*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*indicatore*	

Proprietà evaluationnode

Il nodo Valutazione facilita la valutazione e il confronto di modelli predittivi. Il grafico di valutazione mostra il comportamento dei modelli nella previsione di particolari risultati. Ordina i record in base al valore previsto e alla confidenza della previsione, quindi li suddivide in gruppi di uguale dimensione (**quantili**) e infine rappresenta il valore del criterio di business per ciascun quantile, dal più alto al più basso. I modelli multipli sono mostrati nel grafico come linee separate.

Esempio

```
node = stream.create("evaluation", "My node")
# "Plot" tab
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

Tabella 101. proprietà evaluationnode.

Proprietà evaluationnode	Tipo di dati	Descrizione proprietà
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*indicatore*	
field_detection_method	Metadata Name	
use_fixed_cost	*indicatore*	
cost_value	*numero*	
cost_field	*stringa*	
use_fixed_revenue	*indicatore*	
revenue_value	*numero*	
revenue_field	*stringa*	
use_fixed_weight	*indicatore*	
weight_value	*numero*	
weight_field	*campo*	

Tabella 101. proprietà evaluationnode (Continua).

Proprietà evaluationnode	Tipo di dati	Descrizione proprietà
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*indicatore*	
style	Line Point	
point_type	Rettangolo Punto Triangolo Esagono Segno più Pentagono Stella Farfallino Trattino orizzontale Trattino verticale Croce Fabbrica Casa Cattedrale Cupola Triangolo concavo Geoide Occhio di gatto Cuscino quadrato Rettangolo arrotondato Ventaglio	
export_data	*indicatore*	
data_filename	*stringa*	
delimiter	*stringa*	
new_line	*indicatore*	
inc_field_names	*indicatore*	
inc_best_line	*indicatore*	
inc_business_rule	*indicatore*	
business_rule_condition	*stringa*	
plot_score_fields	*indicatore*	
score_fields	*[campo1 ... campoN]*	
target_field	*campo*	
use_hit_condition	*indicatore*	
hit_condition	*stringa*	
use_score_expression	*indicatore*	
score_expression	*stringa*	
caption_auto	*indicatore*	

Proprietà graphboardnode

Il nodo Lavagna grafica offre numerosi tipi di grafici diversi in un unico nodo. Con questo nodo è possibile scegliere i campi di dati da esplorare e selezionare quindi un grafico fra quelli disponibili per i dati selezionati. Il nodo esclude automaticamente tutti i tipi di grafici non adatti ai campi selezionati.

Nota: se si imposta una proprietà non valida per il tipo di grafico, per esempio specificando `y_field` per un istogramma, tale proprietà viene ignorata.

Nota: Nell'interfaccia utente, nella scheda Dettagliato di molti tipi di grafici differenti, è presente un campo **Riepilogo**; tale campo non è attualmente supportato dagli script.

Esempio

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

Tabella 102. proprietà graphboardnode

Proprietà graphboard	Tipo di dati	Descrizione proprietà
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Barre BarCounts BarCountsMap BarMap BinnedScatter Grafico a scatole Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues	Identifica il tipo di grafico.

Tabella 102. proprietà graphboardnode (Continua)

Proprietà graphboard	Tipo di dati	Descrizione proprietà
	ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallelo Percorso Pie PieCountMap PieCounts PieMap	
	PointOverlayMap PolygonOverlayMap Nastro Grafico a dispersione SPLOM Superficie	
x_field	*campo*	Specifica un'etichetta personalizzata per l'asse *x*. Disponibile solo per le etichette.
y_field	*campo*	Specifica un'etichetta personalizzata per l'asse *y*. Disponibile solo per le etichette.
z_field	*campo*	Utilizzata in alcuni grafici 3D.
color_field	*campo*	Utilizzata nelle mappe termiche.
size_field	*campo*	Utilizzata nei grafici a bolle.
categories_field	*campo*	
values_field	*campo*	
rows_field	*campo*	
columns_field	*campo*	
fields	*campo*	
start_longitude_field	*campo*	Utilizzata con le frecce su una mappa di riferimento.
end_longitude_field	*campo*	
start_latitude_field	*campo*	
end_latitude_field	*campo*	
data_key_field	*campo*	Utilizzata in varie mappe.
panelrow_field	*stringa*	
panelcol_field	*stringa*	
animation_field	*stringa*	

Tabella 102. proprietà graphboardnode (Continua)

Proprietà graphboard	Tipo di dati	Descrizione proprietà
longitude_field	*campo*	Utilizzata con le coordinate sulle mappe.
latitude_field	*campo*	
map_color_field	*campo*	

Proprietà histogramnode

Il nodo Istogramma mostra l'occorrenza dei valori per i campi numerici. Viene spesso utilizzato per analizzare i dati prima delle manipolazioni e della creazione del modello. Come il nodo distribuzione, anche il nodo Istogramma viene frequentemente utilizzato per rivelare squilibri nei dati.

Esempio

```
node = stream.create("histogram", "My node")
# "Plot" tab
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

Tabella 103. proprietà histogramnode

Proprietà histogramnode	Tipo di dati	Descrizione proprietà
campo	*campo*	
color_field	*campo*	
panel_field	*campo*	
animation_field	*campo*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*indicatore*	
separate_bands	*indicatore*	
x_label_auto	*indicatore*	
x_label	*stringa*	

Tabella 103. proprietà histogramnode (Continua)

Proprietà `histogramnode`	Tipo di dati	Descrizione proprietà
`y_label_auto`	*indicatore*	
`y_label`	*stringa*	
`use_grid`	*indicatore*	
`graph_background`	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
`page_background`	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
`normal_curve`	*indicatore*	Indica se la curva di distribuzione normale deve essere visualizzata nell'output.

Proprietà mapvisualization

Il nodo Visualizzazione della mappa può accettare più connessioni di input e visualizzare i dati geospaziali su una mappa come una serie di livelli. Ciascun livello è un singolo campo geospaziale; ad esempio, il livello di base potrebbe essere la mappa di un paese, al di sopra della quale potrebbero essere presenti un livello per le strade, uno per i fiumi ed uno per le città.

Tabella 104. Proprietà mapvisualization

Proprietà `mapvisualization`	Tipo di dati	Descrizione proprietà
`tag`	*stringa*	Imposta il nome del tag per l'input. La tag predefinito è un numero basato sull'ordine in cui gli input sono stati collegati al nodo (il primo tag di connessione è `1`, il secondo tag di connessione è `2`, ecc.)
`layer_field`	*campo*	Seleziona quale campo geo del dataset viene visualizzato come livello della mappa. La selezione predefinita si basa sul seguente criterio di ordinamento: • Primo - Punto • Linestring • Poligono • Multipunto • MultiLinestring • Ultimo - MultiPoligono Se sono presenti due campi con lo stesso tipo di misurazione, il primo campo in ordine alfabetico (in base al nome) viene selezionato per impostazione predefinita.
`color_type`	*booleano*	Specifica se a tutte le funzioni di un campo geo viene applicato un colore standard, oppure un campo di sovrapposizione, che colora le funzioni in base ai valori di un altro campo nel dataset. I valori possibili sono `standard` o `overlay`. Il valore predefinito è `standard`.

Tabella 104. Proprietà mapvisualization (Continua)

Proprietà `mapvisualization`	Tipo di dati	Descrizione proprietà
`colore`	*stringa*	Se per `color_type` viene selezionato `standard`, l'elenco a discesa contiene la stessa tavolozza di colori dell'ordine di colori della categoria del grafico nella scheda Visualizza delle opzioni utente. Il valore predefinito è colore categoria grafico 1.
`color_field`	*campo*	Se per `color_type` viene selezionato `overlay`, l'elenco a discesa contiene tutti i campi dello stesso dataset del campo geo selezionato come livello.
`symbol_type`	*booleano*	Specifica se a tutti i record del campo geo viene applicato un simbolo standard, oppure un simbolo di sovrapposizione, che cambia l'icona del simbolo per i punti in base ai valori di un altro campo nel dataset. I valori possibili sono `standard` o `overlay`. Il valore predefinito è `standard`.
`symbol`	*stringa*	Se per `symbol_type` viene selezionato `standard`, l'elenco a discesa contiene una selezione di simboli che possono essere utilizzati per visualizzare i punti sulla mappa.
`symbol_field`	*campo*	Se per `symbol_type` viene selezionato `overlay`, l'elenco a discesa contiene tutti i campi nominali, ordinali o categoriali dello stesso dataset del campo geo selezionato come livello.
`size_type`	*booleano*	Specifica se a tutti i record del campo geo viene applicata una dimensione standard, oppure una dimensione di sovrapposizione, che cambia la dimensione dell'icona del simbolo o lo spessore della linea in base ai valori di un altro campo nel dataset. I valori possibili sono `standard` o `overlay`. Il valore predefinito è `standard`.
`size`	*stringa*	Seper `size_type` viene selezionato `standard`, per `point` o `multipoint`, l'elenco a discesa contiene una selezione di dimensioni per il simbolo selezionato. Per `linestring` o `multilinestring`, l'elenco a discesa contiene una selezione di spessori della linea.
`size_field`	*campo*	Se per `size_type` viene selezionato `overlay`, l'elenco a discesa contiene tutti i campi dello stesso dataset del campo geo selezionato come livello.
`transp_type`	*booleano*	Specifica se a tutti i record del campo geo viene applicata una trasparenza standard, oppure una trasparenza di sovrapposizione, che cambia il livello di trasparenza del simbolo, della linea o del poligono in base ai valori di un altro campo nel dataset. I valori possibili sono `standard` o `overlay`. Il valore predefinito è `standard`.

Tabella 104. Proprietà mapvisualization (Continua)

Proprietà `mapvisualization`	Tipo di dati	Descrizione proprietà
`transp`	*numero intero*	Se per `transp_type` viene selezionato `standard`, l'elenco a discesa contiene una selezione di livelli di trasparenza che parte da `0%` (opaco) ed aumenta fino a `100%` (trasparente) con incrementi del 10%. Imposta la trasparenza dei punti, delle linee, o dei poligoni sulla mappa. Se per `size_type` viene selezionato `overlay`, l'elenco a discesa contiene tutti i campi dello stesso dataset del campo geo selezionato come livello. Per `point`, `multipoint`, `linestring`, e `multilinestring`, `polygon` e `multipolygon` (che si trovano nel livello più basso), il valore predefinito è `0%`. Per `polygons` e `multipolygons` che non sono al livello più basso, il valore predefinito è `50%` (per evitare livelli oscuranti sotto tali poligoni).
`transp_field`	*campo*	Se per `transp_type` viene selezionato `overlay`, l'elenco a discesa contiene tutti i campi dello stesso dataset del campo geo selezionato come livello.
`data_label_field`	*campo*	Specifica il campo da utilizzare come etichetta dati sulla mappa. Ad esempio, se il livello a cui è applicata questa impostazione è un livello poligono, l'etichetta dati potrebbe essere il campo `nome`, che contiene il nome di ciascun poligono. Quindi la selezione del campo `nome` in questo punto determinerà la visualizzazione di tali nomi sulla mappa.
`use_hex_binning`	*booleano*	Abilita la raccolta hex ed abilita tutti gli elenchi a discesa dell'aggregazione. Questa opzione è disattivata per impostazione predefinita.

Tabella 104. Proprietà mapvisualization (Continua)

Proprietà `mapvisualization`	Tipo di dati	Descrizione proprietà
`color_aggregation` e `transp_aggregation`	*stringa*	Se viene selezionato un campo Sovrapposizione per un livello point che utilizza la raccolta hex, tutti i valori per tale campo devono essere aggregati per tutti i punti all'interno dell'esagono. Specificare quindi una funzione di aggregazione per tutti i campi di sovrapposizione che si desidera applicare alla mappa. Le funzioni di aggregazione disponibili sono: Continua (archiviazione di numeri reali o interi): • Sum • Mean • Min • Max • Median • 1° quartile • 3° quartile Continua (archiviazione Ora, Data o Data/Ora): • Mean • Min • Max Nominale o Categoriale: • Mode • Min • Max Indicatore: • True se uno è true • False se uno è falso
`custom_storage`	*stringa*	Imposta il tipo di archiviazione generale del campo. L'impostazione perdefinita è `List`. Se si specifica `List`, i seguenti controlli `custom_value_storage` e `list_depth` sono disabilitati.
`custom_value_storage`	*stringa*	Imposta i tipi di archiviazione degli elementi nell'elenco anziché nel campo. L'impostazione predefinita è `Real`.

Tabella 104. Proprietà mapvisualization (Continua)

Proprietà `mapvisualization`	Tipo di dati	Descrizione proprietà
`list_depth`	*numero intero*	Imposta la profondità del campo dell'elenco. La profondità richiesta dipende dal tipo di campo geo che rispetta i criteri riportati di seguito: • Point - 0 • LineString - 1 • Polygon - 2 • Multipoint - 1 • MultiLineString - 2 • Multipolygon - 3 È necessario conoscere il tipo di campo geospaziale che si sta convertendo in elenco e la profondità richiesta per tale tipo di campo. Se impostato in modo non corretto, il campo non può essere utilizzato. Il valore predefinito è `0`, il minimo è `0` ed il massimo è `10`.

Proprietà multiplotnode

Un nodo Multiplot crea un grafico che consente di visualizzare più campi *Y* in un singolo campo *X*. I campi *Y* sono rappresentati come linee colorate e ognuno di essi equivale a un nodo Plot con lo Stile impostato su **Linea** e la Modalità X impostata su **Ordina**. I multiplot sono utili quando si desidera esplorare la fluttuazione di numerose variabili nel tempo.

Esempio

```
node = stream.create("multiplot", "My node")
# "Plot" tab
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# "Overlay" section
node.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

Tabella 105. proprietà multiplotnode

Proprietà `multiplotnode`	Tipo di dati	Descrizione proprietà
`x_field`	*campo*	
`y_fields`	*elenco*	
`panel_field`	*campo*	
`animation_field`	*campo*	
`normalize`	*indicatore*	
`use_overlay_expr`	*indicatore*	

Tabella 105. proprietà multiplotnode (Continua)

Proprietà `multiplotnode`	Tipo di dati	Descrizione proprietà
`overlay_expression`	*stringa*	
`records_limit`	*number*	

`if_over_limit`	`PlotBins`
	`PlotSample`
	`PlotAll`	

`x_label_auto`	*indicatore*	
`x_label`	*stringa*	
`y_label_auto`	*indicatore*	
`y_label`	*stringa*	
`use_grid`	*indicatore*	
`graph_background`	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
`page_background`	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.

Proprietà plotnode

Il nodo Plot mostra la relazione tra campi numerici. È possibile creare un grafico utilizzando punti (un grafico a dispersione) oppure linee.

Esempio

```
node = stream.create("plot", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# "Output" tab
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/plot_output.jpeg")
```

Tabella 106. proprietà plotnode

Proprietà `plotnode`	Tipo di dati	Descrizione proprietà
`x_field`	*campo*	Specifica un'etichetta personalizzata per l'asse *x*. Disponibile solo per le etichette.
`y_field`	*campo*	Specifica un'etichetta personalizzata per l'asse *y*. Disponibile solo per le etichette.

Tabella 106. proprietà plotnode (Continua)

Proprietà plotnode	Tipo di dati	Descrizione proprietà
three_D	*indicatore*	Specifica un'etichetta personalizzata per l'asse *y*. Disponibile solo per le etichette nei grafici 3-D.
z_field	*campo*	
color_field	*campo*	Campo sovrapposto.
size_field	*campo*	
shape_field	*campo*	
panel_field	*campo*	Specifica un campo nominale o flag da utilizzare per creare un grafico distinto per ciascuna categoria. I grafici vengono rappresentati in pannelli in un'unica finestra di output.
animation_field	*campo*	Specifica un campo nominale o flag per rappresentare le categorie per i valori dei dati creando una serie di grafici visualizzati in sequenza tramite l'animazione.
transp_field	*campo*	Specifica un campo per rappresentare le categorie per i valori dei dati utilizzando un livello di trasparenza diverso per ciascuna categoria. Non disponibile per i grafici a linee.
overlay_type	None Smoother Function	Specifica se viene visualizzata una funzione sovrapposta o un livellamento di LOESS.
overlay_expression	*stringa*	Specifica l'espressione utilizzata quando overlay_type è impostata su Function.
style	Point Line	
point_type	Rettangolo Punto Triangolo Esagono Segno più Pentagono Stella Farfallino Trattino orizzontale Trattino verticale Croce Fabbrica Casa Cattedrale Cupola Triangolo concavo Geoide Occhio di gatto Cuscino quadrato Rettangolo arrotondato Ventaglio	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*number*	
x_range_max	*number*	

Tabella 106. proprietà plotnode (Continua)

Proprietà plotnode	Tipo di dati	Descrizione proprietà
y_range_mode	Automatic UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*indicatore*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*indicatore*	
x_label	*stringa*	
y_label_auto	*indicatore*	
y_label	*stringa*	
z_label_auto	*indicatore*	
z_label	*stringa*	
use_grid	*indicatore*	
graph_background	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
page_background	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
use_overlay_expr	*indicatore*	Reso obsoleto a favore di overlay_type.

Proprietà timeplotnode

Il nodo del grafico temporale visualizza uno o più insiemi di dati di serie temporali. In genere, si utilizza prima un nodo Intervalli di tempo per creare un campo *EtichettaTempo*, che viene utilizzato per attribuire un'etichetta all'asse *x*.

Esempio

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Appearance settings
node.setPropertyValue("symbol_size", 2.0)
```

Tabella 107. proprietà timeplotnode

Proprietà **timeplotnode**	Tipo di dati	Descrizione proprietà
plot_series	Series Models	
use_custom_x_field	*indicatore*	
x_field	*campo*	
y_fields	*elenco*	
panel	*indicatore*	
normalize	*indicatore*	
line	*indicatore*	
points	*indicatore*	
point_type	Rettangolo Punto Triangolo Esagono Segno più Pentagono Stella Farfallino Trattino orizzontale Trattino verticale Croce Fabbrica Casa Cattedrale Cupola Triangolo concavo Geoide Occhio di gatto Cuscino quadrato Rettangolo arrotondato Ventaglio	
smoother	*indicatore*	È possibile aggiungere livellamenti al plot solo se si imposta panel su True.
use_records_limit	*indicatore*	
records_limit	*numero intero*	
symbol_size	*number*	Specifica la dimensione di un simbolo.
panel_layout	Horizontal Vertical	

Proprietà eplotnode

Il nodo E-Plot (Beta) mostra la relazione tra i campi numerici. È simile al nodo Plot, ma differiscono le relative opzioni e l'output utilizza una nuova interfaccia grafica specifica per questo nodo. Utilizzare il nodo a livello beta per divertirsi con le nuove funzioni grafiche.

Tabella 108. Proprietà eplotnode

Proprietà eplotnode	Tipo di dati	Descrizione proprietà
x_field	*stringa*	Specifica il campo da visualizzare sull'asse orizzontale X.
y_field	*stringa*	Specifica il campo da visualizzare sull'asse verticale Y.
color_field	*stringa*	Specifica il campo da utilizzare per la sovrapposizione della mappa colori nel'output, se si desidera.
size_field	*stringa*	Specifica il campo da utilizzare per la sovrapposizione della mappa dimensioni nell'output, se si desidera.
shape_field	*stringa*	Specifica il campo da utilizzare per la sovrapposizione della mappa forme nell'output, se si desidera.
interested_fields	*stringa*	Specifica i campi che si desidera includere nell'output.
records_limit	*numero intero*	Specifica un numero per il numero massimo di record da rappresentare nell'output. 2000 è l'impostazione predefinita.
if_over_limit	*Booleano*	Specifica se utilizzare l'opzione Campione o l'opzione Utilizza tutti i dati se è eliminato il records_limit. Campione è l'impostazione predefinita e campiona casualmente i dati fino a quando non si raggiunge il records_limit. Se si specifica Utilizza tutti i dati ignorare records_limit e rappresentare tutti i punti dati, considerare che ciò può ridurre drammaticamente le prestazioni.

Proprietà tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) è uno strumento per la visualizzazione dei dati altamente dimensionali. Converte le affinità dei punti dati in probabilità. Questo nodo t-SNE in SPSS Modeler è implementato in Python e richiede la libreria scikit-learn©.

Tabella 109. Proprietà tsnenode

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
mode_type	*stringa*	Specificare la modalità simple o expert.
n_components	*stringa*	Dimensione dello spazio integrato (2D or 3D). Specificare 2 o 3. Il valore predefinito è 2.
method	*stringa*	Specificare barnes_hut o exact. Il valore predefinito è barnes_hut.
init	*stringa*	Inizializzazione dell'integrazione. Specificare random o pca. Il valore predefinito è random.

Tabella 109. Proprietà tsnenode (Continua)

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
target_field	*stringa*	Il nome del campo di destinazione. Sarà una mappa colorata nel grafico di output. Il grafico utilizzerà un colore se viene specificato il campo di destinazione.
perplexity	*float*	La perplessità è correlata al numero di elementi adiacenti più vicini utilizzati in altri algoritmi di apprendimento. Generalmente, i dataset di dimensioni maggiori richiedono una perplessità maggiore. Considerare la selezione di un valore compreso tra 5 e 50. Il valore predefinito è 30.
early_exaggeration	*float*	Questa impostazione controlla quanto i cluster naturali nello spazio originale saranno vicini nello spazio integrato e la quantità di spazio tra di loro. Il valore predefinito è 12.0.
learning_rate	*float*	Il valore predefinito è 200.
n_iter	*numero intero*	Il numero massimo di iterazioni per l'ottimizzazione. Impostare ad almeno 250. Il valore predefinito è 1000.
angle	*float*	La dimensione angolare di un nodo distante misurata da un punto. Specificare un valore nell'intervallo compreso tra 0 e 1. Il valore predefinito è 0.5.
enable_random_seed	*Booleano*	Impostare su true per abilitare il parametro random_seed. Il valore predefinito è false.
random_seed	*numero intero*	Il seed random da utilizzare. Il valore predefinito è None.
n_iter_without_progress	*numero intero*	Numero massimo di iterazioni senza avanzamento. Il valore predefinito è 300.
min_grad_norm	*stringa*	Se la norma del gradiente è inferiore a questa soglia, l'ottimizzazione viene arrestata. Il valore predefinito è 1.0E-7. I valori possibili sono: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*Booleano*	Impostare su true per eseguire t-SNE con diverse perplessità differenti. Il valore predefinito è false.
output_Rename	*Booleano*	Specificare true se si desidera fornire un nome personalizzato o su false per denominare l'output in modo automatico. Il valore predefinito è false.
output_to	*stringa*	Specificare Screen o Output. Il valore predefinito è Screen.

Tabella 109. Proprietà tsnenode (Continua)

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Specificare il nome del file di output.
output_file_type	*stringa*	Formato del file di output. Specificare HTML o Output object. Il valore predefinito è HTML.

Proprietà webnode

Il nodo Web illustra l'intensità della relazione tra valori di due o più campi simbolici (categoriali). Il grafico utilizza linee di spessore diverso per indicare l'intensità della connessione. Un nodo Web può essere utilizzato, per esempio, per analizzare la relazione tra l'acquisto di vari oggetti in un sito di e-commerce.

Esempio

```
node = stream.create("web", "My node")
# "Plot" tab
node.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# "Options" tab
node.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

Tabella 110. proprietà webnode

Proprietà webnode	Tipo di dati	Descrizione proprietà
use_directed_web	*indicatore*	
fields	*elenco*	
to_field	*campo*	
from_fields	*elenco*	
true_flags_only	*indicatore*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*indicatore*	
num_links	ShowMaximum ShowLinksAbove ShowAll	

Tabella 110. proprietà webnode (Continua)

Proprietà **webnode**	Tipo di dati	Descrizione proprietà
max_num_links	*number*	
links_above	*number*	
discard_links_min	*indicatore*	
links_min_records	*number*	
discard_links_max	*indicatore*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*indicatore*	
web_display	Circular Network Directed Grid	
graph_background	*colore*	I colori standard dei grafici sono descritti all'inizio di questa sezione.
symbol_size	*number*	Specifica la dimensione di un simbolo.

Capitolo 13. Proprietà dei nodi Modelli

Proprietà comuni nodi modellazione

Le seguenti proprietà sono comuni ad alcuni o a tutti i nodi Modelli. Le eventuali eccezioni sono segnalate, ove necessario, nella documentazione relativa ai singoli nodi Modelli.

Tabella 111. Proprietà comuni nodo modellazione

Proprietà	Valori	Descrizione proprietà
custom_fields	*indicatore*	Se vera, consente di specificare i campi obiettivo, di input e di altro tipo per il nodo corrente. Se falsa, vengono utilizzate le impostazioni correnti di un nodo Tipo a monte.
obiettivo o targets	*campo* o [*campo1 ... campoN*]	Specifica un unico campo obiettivo o più campi obiettivo, a seconda del tipo di modello.
inputs	[*campo1 ... campoN*]	I campi di input o predittore utilizzati dal modello.
partition	*campo*	
use_partitioned_data	*indicatore*	Se è definito un campo partizione, questa opzione garantisce che per la creazione del modello verranno utilizzati solo i dati della partizione di addestramento.
use_split_data	*indicatore*	
splits	*[campo1 ... campoN]*	Specifica il campo o i campi da usare per la creazione di modelli suddivisi. Efficace solo se use_split_data è impostato su True.
use_frequency	*indicatore*	I campi peso e frequenza vengono utilizzati da determinati modelli, come riportato per ogni tipo di modello.
frequency_field	*campo*	
use_weight	*indicatore*	
weight_field	*campo*	
use_model_name	*indicatore*	
model_name	*stringa*	Nome personalizzato per il nuovo modello.
mode	Simple Expert	

proprietà anomalydetectionnode

Il nodo Rilevamento anomalie identifica casi insoliti, o valori anomali, non conformi a schemi di dati "normali". Con questo nodo è possibile identificare valori anomali anche se questi non rientrano in schemi precedentemente conosciuti e anche se l'utente non sa esattamente ciò che sta cercando.

Esempio

```
node = stream.create("anomalydetection", "My node")
node.setPropertyValue("anomaly_method", "PerRecords")
node.setPropertyValue("percent_records", 95)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("peer_group_num_auto", True)
node.setPropertyValue("min_num_peer_groups", 3)
node.setPropertyValue("max_num_peer_groups", 10)
```

Tabella 112. proprietà anomalydetectionnode

Proprietà **anomalydetectionnode**	Valori	Descrizione proprietà
inputs	*[campo1 ... campoN]*	I modelli Rilevamento anomalie effettuano lo screening dei record in base ai campi di input specificati. Non utilizzano un campo obiettivo, né i campi peso e frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Specifica il metodo utilizzato per determinare il valore di interruzione per contrassegnare i record come anomali.
index_level	*number*	Specifica il valore di interruzione minimo per contrassegnare le anomalie.
percent_records	*number*	Imposta la soglia per contrassegnare i record in base alla percentuale di record nei dati di addestramento.
num_records	*number*	Imposta la soglia per contrassegnare i record in base al numero di record nei dati di addestramento.
num_fields	*numero intero*	Numero di campi da segnalare per ciascun record anomalo.
impute_missing_values	*indicatore*	
adjustment_coeff	*number*	Valore utilizzato per bilanciare il peso relativo attribuito ai campi continui e categoriali nel calcolo della distanza.
peer_group_num_auto	*indicatore*	Calcola automaticamente il numero dei gruppi di peer.
min_num_peer_groups	*numero intero*	Specifica il numero minimo di gruppi di peer utilizzati quando peer_group_num_auto è impostata su True.
max_num_per_groups	*numero intero*	Specifica il numero massimo di gruppi di peer.
num_peer_groups	*numero intero*	Specifica il numero di gruppi di peer utilizzati quando peer_group_num_auto è impostata su False.
noise_level	*number*	Determina come trattare i valori anomali durante il raggruppamento tramite cluster. Specificare un valore compreso tra 0 e 0.5.

Tabella 112. proprietà anomalydetectionnode (Continua)

Proprietà anomalydetectionnode	Valori	Descrizione proprietà
noise_ratio	*number*	Specifica la parte di memoria allocata per il componente da utilizzare per la memorizzazione del rumore. Specificare un valore compreso tra 0 e 0.5.

proprietà apriorinode

Il nodo Apriori estrae un insieme di regole dai dati, estrapolando le regole con il più alto contenuto di informazioni. Apriori offre cinque diversi metodi per la selezione delle regole e utilizza uno schema di indicizzazione sofisticato per elaborare in modo efficiente insiemi di dati di grandi dimensioni. In caso di problemi complessi, l'addestramento di Apriori è in genere più rapido. Apriori non ha un limite arbitrario per quanto riguarda il numero di regole che possono essere mantenute e può gestire regole con un massimo di 32 precondizioni. Apriori richiede che tutti i campi di input e output siano categoriali ma garantisce prestazioni migliori perché è ottimizzato per questo tipo di dati.

Esempio

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Tabella 113. proprietà apriorinode

Proprietà apriorinode	Valori	Descrizione proprietà
consequents	*campo*	I modelli Apriori utilizzano antecedenti e conseguenti al posto dei campi obiettivo e di input standard. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
antecedents	[*campo1 ... campoN*]	
min_supp	*number*	

Tabella 113. proprietà apriorinode (Continua)

Proprietà apriorinode	Valori	Descrizione proprietà
min_conf	*number*	
max_antecedents	*number*	
true_flags	*indicatore*	
optimize	Speed Memory	
use_transactional_data	*indicatore*	
contiguous	*indicatore*	
id_field	*stringa*	
content_field	*stringa*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Speed Memory	Utilizzare per specificare se ottimizzare la velocità o la memoria durante la creazione del modello.

Proprietà associationrulesnode

Il Nodo Regole di associazione è simile al Nodo Apriori; tuttavia, a differenza del nodo Apriori, il Nodo Regole di associazione è in grado di elaborare i dati dell'elenco. Inoltre, il Nodo Regole di associazione può essere utilizzato con IBM SPSS Analytic Server per elaborare dati di quantità elevata e sfruttare una più rapida elaborazione parallela.

Tabella 114. Proprietà associationrulesnode

Proprietà associationrulesnode	Tipo di dati	Descrizione proprietà
predictions	*campo*	I campi possono essere visualizzati in questo elenco solo come predittori di una regola
conditions	*[campo1...campoN]*	I campi possono essere visualizzati in questo elenco solo come condizione di una regola
max_rule_conditions	*numero intero*	Il numero massimo di condizioni che possono essere incluse in una regola singola. Minimo 1, massimo 9.
max_rule_predictions	*numero intero*	Il numero massimo di previsioni che possono essere incluse in una regola singola. Minimo 1, massimo 5.
max_num_rules	*numero intero*	Il numero massimo di regole che possono essere considerate come parte della creazione della regola. Minimo 1, massimo 10.000.

Tabella 114. Proprietà associationrulesnode (Continua)

Proprietà associationrulesnode	Tipo di dati	Descrizione proprietà
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Distribuibilità	Il criterio della regola che determina il valore in base al quale vengono scelte le prime "N" regole nel modello.
true_flags	*Booleano*	Impostando *Y* si determina che durante la creazione della regola vengono considerati solo i valori true per i campi indicatore.
rule_criterion	*Booleano*	Impostando *Y* si determina che durante la creazione del modello vengono considerati solo i valori di creazione della regola per l'esclusione delle regole.
min_confidence	*number*	Da 0,1 a 100 - il valore percentuale per il livello di confidenza minimo richiesto per una regola prodotta da un modello. Se il modello produce una regola con un livello di confidenza inferiore al valore specificato, la regola viene eliminata.
min_rule_support	*number*	Da 0,1 a 100 - il valore percentuale per il supporto della regola minimo richiesto per una regola prodotta da un modello. Se il modello produce una regola con un livello di supporto della regola inferiore al valore specificato, la regola viene eliminata.
min_condition_support	*number*	Da 0,1 a 100 - il valore percentuale per il supporto della condizione minimo richiesto per una regola prodotta da un modello. Se il modello produce una regola con un livello di supporto della condizione inferiore al valore specificato, la regola viene eliminata.
min_lift	*numero intero*	Da 1 a 10 – rappresenta il minimo guadagno cumulativo richiesto per una regola generata dal modello. Se il modello produce una regola con un livello di guadagno cumulativo inferiore al valore specificato, la regola viene eliminata.
exclude_rules	*Booleano*	Utilizzata per selezionare un elenco di campi correlati da cui non si desidera che il modello crei regole. Esempio: set :gsarsnode.exclude_rules = [[[field1,field2, field3]],[[field4, field5]]] - dove ciascun elenco di campi separati da [] è una riga nella tabella.
num_bins	*numero intero*	Impostare il numero di bin automatici a cui i campi continui vengono discretizzati. Minimo 2, massimo 10.
max_list_length	*numero intero*	Si applica a tutti i campi dell'elenco di cui è sconosciuta la lunghezza massima. Gli elementi dell'elenco vengono inseriti nella creazione del modello fino al raggiungimento del numero qui specificato; eventuali ulteriori elementi vengono eliminati. Minimo 1, massimo 100.
output_confidence	*Booleano*	

Tabella 114. Proprietà associationrulesnode (Continua)

Proprietà associationrulesnode	Tipo di dati	Descrizione proprietà
output_rule_support	*Booleano*	
output_lift	*Booleano*	
output_condition_support	*Booleano*	
output_deployability	*Booleano*	
rules_to_display	upto all	Il numero massimo di regole da visualizzare nelle tabelle di output.
display_upto	*numero intero*	Se si imposta upto in rules_to_display, impostare il numero di regole da visualizzare nelle tabelle di output. Minimo 1.
field_transformations	*Booleano*	
records_summary	*Booleano*	
rule_statistics	*Booleano*	
most_frequent_values	*Booleano*	
most_frequent_fields	*Booleano*	
word_cloud	*Booleano*	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Distribuibilità	
word_cloud_display	*numero intero*	Minimo 1, massimo 20
max_predictions	*numero intero*	Il numero massimo di regole che possono essere applicate a ciascun input nel punteggio.
criterion	Confidence Rulesupport Lift Conditionsupport Distribuibilità	Selezionare la misura che determina l'efficacia delle regole.
allow_repeats	*Booleano*	Determina se nel punteggio vengono incluse le regole con la stessa previsione.
check_input	NoPredictions Predictions NoCheck	

proprietà autoclassifiernode

Il nodo Classificatore automatico crea e confronta svariati tipi di modelli per risultati binari (sì o no, abbandono oppure no e così via), consentendo di scegliere l'approccio migliore per una determinata analisi. Sono supportati numerosi algoritmi di modellazione ed è possibile selezionare i metodi da utilizzare, le opzioni specifiche per ognuno di essi e i criteri per confrontare i risultati. Il nodo genera un insieme di modelli basato sulle opzioni specificate e classifica i candidati migliori in base ai criteri indicati.

Esempio

```
node = stream.create("autoclassifier", "My node")
node.setPropertyValue("ranking_measure", "Accuracy")
node.setPropertyValue("ranking_dataset", "Training")
```

```
node.setPropertyValue("enable_accuracy_limit", True)
node.setPropertyValue("accuracy_limit", 0.9)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("use_costs", True)
node.setPropertyValue("svm", False)
```

Tabella 115. proprietà autoclassifiernode

Proprietà **autoclassifiernode**	Valori	Descrizione proprietà
target	*campo*	Per i campi obiettivo, il nodo Classificatore automatico richiede un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare i campi frequenza e peso. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*numero intero*	Numero dei modelli da includere nel nugget del modello. Specificare un numero intero compreso fra 1 e 100.
calculate_variable_importance	*indicatore*	
enable_accuracy_limit	*indicatore*	
accuracy_limit	*numero intero*	Numero intero compreso fra 0 e 100.
enable_ area_under_curve _limit	*indicatore*	
area_under_curve_limit	*number*	Numero reale compreso tra 0.0 e 1.0.
enable_profit_limit	*indicatore*	
profit_limit	*number*	Numero intero maggiore di 0.
enable_lift_limit	*indicatore*	
lift_limit	*number*	Numero reale maggiore di 1.0.
enable_number_of_variables_limit	*indicatore*	
number_of_variables_limit	*number*	Numero intero maggiore di 0.
use_fixed_cost	*indicatore*	
fixed_cost	*number*	Numero reale maggiore di 0.0.
variable_cost	*campo*	
use_fixed_revenue	*indicatore*	
fixed_revenue	*number*	Numero reale maggiore di 0.0.
variable_revenue	*campo*	
use_fixed_weight	*indicatore*	
fixed_weight	*number*	Numero reale maggiore di 0,0.
variable_weight	*campo*	
lift_percentile	*number*	Numero intero compreso fra 0 e 100.
enable_model_build_time_limit	*indicatore*	

Tabella 115. proprietà autoclassifiernode (Continua)

Proprietà autoclassifiernode	Valori	Descrizione proprietà
model_build_time_limit	*number*	Numero intero impostato sul numero di minuti per limitare il tempo impiegato per creare ogni singolo modello.
enable_stop_after_time_limit	*indicatore*	
stop_after_time_limit	*number*	Numero reale impostato sul numero di ore per limitare il tempo complessivo impiegato per l'esecuzione di un classificatore automatico.
enable_stop_after_valid_model_produced	*indicatore*	
use_costs	*indicatore*	
<algorithm>	*indicatore*	Abilita o disabilita l'utilizzo di un algoritmo specifico.
<algorithm>.<property>	*stringa*	Imposta il valore di una proprietà di un algoritmo specifico. Per ulteriori informazioni, consultare l'argomento "Impostazione delle proprietà degli algoritmi".

Impostazione delle proprietà degli algoritmi

Per i nodi Classificatore automatico, Numerico automatico e Cluster automatico, le proprietà degli algoritmi specifici utilizzati dal nodo si possono impostare utilizzando il formato generico:

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

Ad esempio:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

I nomi degli algoritmi per il nodo Classificatore automatico sono `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` e `knn`.

I nomi degli algoritmi per il nodo Numerico automatico sono `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` e `knn`.

I nomi degli algoritmi per il nodo Cluster automatico sono `twostep`, `Medie K` e `kohonen`.

I nomi delle proprietà sono standard, come documentato per i nodi dei singoli algoritmi.

Le proprietà degli algoritmi che contengono punti o altri tipi di punteggiatura devono essere racchiuse tra virgolette singole, per esempio:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

Come proprietà è possibile assegnare anche valori multipli, per esempio:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

Per attivare o disattivare l'utilizzo di un algoritmo specifico:

```
node.setPropertyValue("chaid", True)
```

Nota: Nei casi in cui determinate opzioni di algoritmi non siano disponibili nel nodo Classificatore automatico o quando è possibile specificare un solo valore anziché un intervallo di valori, per gli script si applicano gli stessi limiti validi per l'accesso al nodo con la normale procedura.

proprietà autoclusternode

Il nodo Cluster automatico stima e confronta i modelli di cluster che identificano gruppi di record con caratteristiche simili. Il nodo funziona in modo analogo ad altri nodi Modelli automatici e consente di sperimentare varie combinazioni di opzioni in un singolo passaggio di modellazione. I modelli si possono confrontare utilizzando misure di base con cui tentare di filtrare e classificare l'utilità dei modelli di cluster e fornire una misura in base all'importanza di determinati campi.

Esempio

```
node = stream.create("autocluster", "My node")
node.setPropertyValue("ranking_measure", "Silhouette")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_silhouette_limit", True)
node.setPropertyValue("silhouette_limit", 5)
```

Tabella 116. proprietà autoclusternode

Proprietà **autoclusternode**	Valori	Descrizione proprietà
evaluation	*campo*	**Nota:** solo nodo Cluster automatico. Identifica il campo per cui verrà calcolato un valore di importanza. In alternativa, può essere utilizzato per identificare il modo in cui il cluster distingue il valore di questo campo e, pertanto, la validità della previsione di questo campo da parte del modello.
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Training Test	
summary_limit	*numero intero*	Numero dei modelli da elencare nel report. Specificare un numero intero compreso fra 1 e 100.
enable_silhouette_limit	*indicatore*	
silhouette_limit	*numero intero*	Numero intero compreso fra 0 e 100.
enable_number_less_limit	*indicatore*	
number_less_limit	*number*	Numero reale compreso tra 0.0 e 1.0.
enable_number_greater_limit	*indicatore*	
number_greater_limit	*number*	Numero intero maggiore di 0.
enable_smallest_cluster_limit	*indicatore*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*numero intero*	Numero intero maggiore di 0.

Tabella 116. proprietà autoclusternode (Continua)

Proprietà autoclusternode	Valori	Descrizione proprietà
enable_largest_cluster_limit	*indicatore*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percentage	*number*	
largest_cluster_limit_count	*numero intero*	
enable_smallest_largest_limit	*indicatore*	
smallest_largest_limit	*number*	
enable_importance_limit	*indicatore*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	Numero intero compreso fra 0 e 100.
importance_limit_less_than	*number*	Numero intero compreso fra 0 e 100.
<algorithm>	*indicatore*	Abilita o disabilita l'utilizzo di un algoritmo specifico.
<algorithm>.<property>	*stringa*	Imposta il valore di una proprietà di un algoritmo specifico. Per ulteriori informazioni, consultare l'argomento "Impostazione delle proprietà degli algoritmi" a pagina 192.

proprietà autonumericnode

Il nodo Numerico automatico stima e confronta i modelli per i risultati di intervalli numerici continui utilizzando svariati metodi. Il nodo funziona in modo analogo al nodo Classificatore automatico e consente di scegliere gli algoritmi da utilizzare e di sperimentare più combinazioni di opzioni in un singolo passaggio di modellazione. Gli algoritmi supportati includono reti neurali, C&R Tree, CHAID, regressione lineare, regressione lineare generalizzata e SVM (Support Vector Machine). I modelli si possono confrontare in base a correlazione, errore relativo o numero di variabili utilizzato.

Esempio

```
node = stream.create("autonumeric", "My node")
node.setPropertyValue("ranking_measure", "Correlation")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_correlation_limit", True)
node.setPropertyValue("correlation_limit", 0.8)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("neuralnetwork", True)
node.setPropertyValue("chaid", False)
```

Tabella 117. proprietà autonumericnode

Proprietà autonumericnode	Valori	Descrizione proprietà
custom_fields	*indicatore*	Se True (vero), saranno utilizzate le impostazioni personalizzate dei campi anziché le impostazioni del nodo Tipo.

Tabella 117. proprietà autonumericnode (Continua)

Proprietà autonumericnode	Valori	Descrizione proprietà
target	*campo*	Il nodo Numerico automatico richiede un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare i campi frequenza e peso. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
inputs	*[campo1 ... campo2]*	
partition	*campo*	
use_frequency	*indicatore*	
frequency_field	*campo*	
use_weight	*indicatore*	
weight_field	*campo*	
use_partitioned_data	*indicatore*	Se è definito un campo partizione, per la creazione del modello verranno utilizzati solo i dati di addestramento.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*numero intero*	Numero dei modelli da includere nel nugget del modello. Specificare un numero intero compreso fra 1 e 100.
calculate_variable_importance	*indicatore*	
enable_correlation_limit	*indicatore*	
correlation_limit	*numero intero*	
enable_number_of_fields_limit	*indicatore*	
number_of_fields_limit	*numero intero*	
enable_relative_error_limit	*indicatore*	
relative_error_limit	*numero intero*	
enable_model_build_time_limit	*indicatore*	
model_build_time_limit	*numero intero*	
enable_stop_after_time_limit	*indicatore*	
stop_after_time_limit	*numero intero*	
stop_if_valid_model	*indicatore*	
<algorithm>	*indicatore*	Abilita o disabilita l'utilizzo di un algoritmo specifico.
<algorithm>.<property>	*stringa*	Imposta il valore di una proprietà di un algoritmo specifico. Per ulteriori informazioni, consultare l'argomento "Impostazione delle proprietà degli algoritmi" a pagina 192.

Proprietà bayesnetnode

Il nodo Rete bayesiana consente di generare un modello di probabilità combinando elementi osservati e registrati con conoscenze del mondo reale per stabilire la probabilità di occorrenze. Il nodo si concentra sulle reti TAN (Tree Augmented Naïve Bayes) e coperta di Markov, che sono prevalentemente utilizzate a scopo di classificazione.

Esempio

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Tabella 118. Proprietà bayesnetnode

Proprietà bayesnetnode	Valori	Descrizione proprietà
inputs	*[campo1 ... campoN]*	I modelli di rete bayesiana utilizzano un solo campo obiettivo e uno o più campi di input. I campi continui vengono automaticamente discretizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
continue_training_existing_model	*indicatore*	
structure_type	TAN MarkovBlanket	Selezionare la struttura da utilizzare per la creazione della rete bayesiana.
use_feature_selection	*indicatore*	
parameter_learning_method	Likelihood Bayes	Specifica il metodo utilizzato per stimare le tabelle di probabilità condizionale tra i nodi in cui i valori dei nodi padre sono noti.
mode	Expert Simple	
missing_values	*indicatore*	
all_probabilities	*indicatore*	
independence	Likelihood Pearson	Specifica il metodo utilizzato per determinare se le osservazioni accoppiate su due variabili sono indipendenti tra loro.
significance_level	*number*	Specifica il valore di interruzione per determinare l'indipendenza.
maximal_conditioning_set	*number*	Imposta il numero massimo di variabili di condizionamento da utilizzare per il test dell'indipendenza.
inputs_always_selected	*[campo1 ... campoN]*	Specifica quali campi dell'insieme di dati devono sempre essere utilizzati nella creazione della rete bayesiana. **Nota:** il campo obiettivo è sempre selezionato.

Tabella 118. Proprietà bayesnetnode (Continua)

Proprietà bayesnetnode	Valori	Descrizione proprietà
maximum_number_inputs	*number*	Specifica il numero massimo di campi di input da utilizzare nella creazione della rete bayesiana.
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

Proprietà buildr

Il nodo Creazione R consente di immettere uno script R personalizzato per eseguire la creazione del modello e il calcolo del punteggio del modello distribuito in IBM SPSS Modeler.

Esempio

```
node = stream.create("buildr", "My node")
node.setPropertyValue("score_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabella 119. Proprietà buildr.

Proprietà buildr	Valori	Descrizione proprietà
build_syntax	*stringa*	Sintassi degli script R per la creazione del modello.
score_syntax	*stringa*	Sintassi degli script R per il calcolo del punteggio del modello.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
output_html	*indicatore*	Opzione per visualizzare grafici su una scheda nel nugget del modello R.
output_text	*indicatore*	Opzione per scrivere l'output di testo della console R in una scheda nel nugget del modello R.

proprietà c50node

Il nodo C5.0 crea una struttura ad albero delle decisioni o un insieme di regole. Il modello suddivide il campione in base al campo che fornisce il massimo guadagno di informazioni a ogni livello. Il campo obiettivo deve essere categoriale. Sono consentite suddivisioni multiple in più di due sottogruppi.

Esempio

```
node = stream.create("c50", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tab
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Tabella 120. proprietà c50node

Proprietà c50node	Valori	Descrizione proprietà
target	*campo*	I modelli C50 utilizzano un solo campo obiettivo e uno o più campi di input. È anche possibile specificare un campo peso. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
output_type	DecisionTree RuleSet	
group_symbolics	*indicatore*	
use_boost	*indicatore*	
boost_num_trials	*number*	
use_xval	*indicatore*	
xval_num_folds	*number*	
mode	Simple Expert	
favor	Accuracy Generality	Precisione o generalità della preferenza.
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*indicatore*	
costs	*strutturato*	Si tratta di una proprietà strutturata.
use_winnowing	*indicatore*	
use_global_pruning	*indicatore*	Impostato su (True) per default.

Tabella 120. proprietà c50node (Continua)

Proprietà c50node	Valori	Descrizione proprietà
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

proprietà carmanode

Il modello CARMA estrae un insieme di regole dai dati senza che venga richiesto all'utente di specificare i campi di input o obiettivo. A differenza di Apriori , il nodo CARMA fornisce le impostazioni di creazione per il supporto delle regole (sia per l'antecedente che per il conseguente) anziché solo per il supporto antecedente. Pertanto, le regole generate possono essere utilizzate per una gamma più vasta di applicazioni, ad esempio per trovare un elenco di prodotti o di servizi (antecedenti) il cui conseguente è rappresentato dall'articolo che si desidera promuovere per le festività correnti.

Esempio

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
node.setPropertyValue("vary_support", True)
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

Tabella 121. proprietà carmanode

Proprietà carmanode	Valori	Descrizione proprietà
inputs	*[campo1 ... campon]*	I modelli CARMA utilizzano un elenco di campi di input, ma nessun campo obiettivo. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
id_field	*campo*	Campo utilizzato come campo ID per la creazione del modello.
contiguous	*indicatore*	Utilizzata per specificare se gli ID del campo ID sono contigui.
use_transactional_data	*indicatore*	

Tabella 121. proprietà carmanode (Continua)

Proprietà carmanode	Valori	Descrizione proprietà
content_field	*campo*	
min_supp	*numero (percentuale)*	Si riferisce al supporto della regola più che al supporto antecedente. Il valore predefinito è 20%.
min_conf	*numero (percentuale)*	Il valore predefinito è 20%.
max_size	*number*	Il valore di default è 10.
mode	Simple Expert	L'impostazione di default è Simple.
exclude_multiple	*indicatore*	Esclude le regole con più conseguenti. L'impostazione predefinita è False.
use_pruning	*indicatore*	L'impostazione predefinita è False.
pruning_value	*number*	Il valore predefinito è 500.
vary_support	*indicatore*	
estimated_transactions	*numero intero*	
rules_without_antecedents	*indicatore*	

proprietà cartnode

Il nodo Struttura ad albero di classificazione e regressione (C&R) genera una struttura ad albero delle decisioni che consente di prevedere o classificare osservazioni future. Il metodo utilizza partizionamento ricorsivo per suddividere i record di addestramento in segmenti, riducendo l'impurità ad ogni passaggio. Un nodo della struttura ad albero è considerato "puro" quando il 100% dei casi nel nodo fa parte di una categoria specifica del campo obiettivo. I campi obiettivo e di input possono essere intervalli numerici o categoriali (nominali, ordinali o flag); tutte le suddivisioni sono binarie (solo due sottogruppi).

Esempio

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4""")
# "Build Options" tab, "Basics" panel
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
```

```
# "Model Options" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Tabella 122. proprietà cartnode

Proprietà **cartnode**	Valori	Descrizione proprietà
target	*campo*	I modelli C&R Tree richiedono un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
continue_training_existing_model	*indicatore*	
objective	Standard Boosting Bagging psm	psm viene utilizzato per insiemi di dati di grandi dimensioni e richiede una connessione Server.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*indicatore*	
tree_directives	*stringa*	Specifica le direttive per l'ingrandimento di strutture ad albero. È possibile racchiudere le direttive tra virgolette triple per evitare il carattere escape in nuove righe o virgolette. Si noti che le direttive possono essere molto sensibili a piccole modifiche nelle opzioni di modellazione e nei dati e possono non essere generalizzate ad altri insiemi di dati.
use_max_depth	Default Custom	
max_depth	*numero intero*	Profondità massima della struttura ad albero, da 0 a 1000. Valore utilizzato solo se use_max_depth = Custom.
prune_tree	*indicatore*	Taglia struttura ad albero per evitare sovradattamento.
use_std_err	*indicatore*	Utilizza differenza massima di rischio (in errori standard).
std_err_multiplier	*number*	Differenza massima.
max_surrogates	*number*	Numero massimo surrogati.
use_percentage	*indicatore*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*indicatore*	
costs	*strutturato*	Proprietà strutturata.

Tabella 122. proprietà cartnode (Continua)

Proprietà cartnode	Valori	Descrizione proprietà
priors	Data Equal Custom	
custom_priors	*strutturato*	Proprietà strutturata.
adjust_priors	*indicatore*	
trails	*number*	Numero di modelli di componenti per boosting o bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regola di combinazione di default per obiettivi categoriali.
range_ensemble_method	Mean Median	Regola di combinazione di default per target continui.
large_boost	*indicatore*	Applica il boosting a insiemi di dati di grandi dimensioni.
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	
train_pct	*number*	Insieme di prevenzione del sovradattamento.
set_random_seed	*indicatore*	Opzione Replica risultati.
seed	*number*	
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

proprietà chaidnode

Il nodo CHAID genera una struttura ad albero delle decisioni utilizzando statistiche chi-quadrato per identificare suddivisioni ottimali. A differenza dei nodi C&R Tree e QUEST, il nodo CHAID può generare strutture ad albero non binarie e pertanto alcune suddivisioni possono avere più di due rami. I campi obiettivo e di input possono essere intervallo numerico (continui) o categoriali. Un CHAID completo è una modificazione di CHAID che esegue operazioni avanzate per l'analisi di tutte le suddivisioni possibili, ma richiede tempi di elaborazione maggiori.

Esempio

```
filenode = stream.createAt("variablefile", "My node", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("chaid", "My node", 200, 100)
stream.link(filenode, node)

node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("use_model_name", True)
```

```
node.setPropertyValue("model_name", "CHAID")
node.setPropertyValue("method", "Chaid")
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "Test")
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("merge_alpha", 0.04)
node.setPropertyValue("chi_square", "Pearson")
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("epsilon", 0.003)
node.setPropertyValue("max_iterations", 75)
node.setPropertyValue("split_merged_categories", True)
node.setPropertyValue("bonferroni_adjustment", True)
```

Tabella 123. proprietà chaidnode

Proprietà chaidnode	**Valori**	**Descrizione proprietà**
target	*campo*	I modelli CHAID richiedono un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
continue_training_existing_model	*indicatore*	
objective	Standard Boosting Bagging psm	psm viene utilizzato per insiemi di dati di grandi dimensioni e richiede una connessione Server.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*indicatore*	
tree_directives	*stringa*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*numero intero*	Profondità massima della struttura ad albero, da 0 a 1000. Valore utilizzato solo se use_max_depth = Custom.
use_percentage	*indicatore*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*indicatore*	
costs	*strutturato*	Proprietà strutturata.
trails	*number*	Numero di modelli di componenti per boosting o bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regola di combinazione di default per obiettivi categoriali.

Tabella 123. proprietà chaidnode (Continua)

Proprietà chaidnode	Valori	Descrizione proprietà
range_ensemble_method	Mean Median	Regola di combinazione di default per target continui.
large_boost	*indicatore*	Applica il boosting a insiemi di dati di grandi dimensioni.
split_alpha	*number*	Livello di significatività per suddivisione.
merge_alpha	*number*	Livello di significatività per unione.
bonferroni_adjustment	*indicatore*	Adegua valori di significatività tramite il metodo di Bonferroni.
split_merged_categories	*indicatore*	Consenti risuddivisione di categorie unite.
chi_square	Pearson LR	Metodo utilizzato per calcolare la statistica chi-quadrato: Pearson o Rapporto di verosimiglianza
epsilon	*number*	Modifica minima nelle frequenze di cella previste.
max_iterations	*number*	Numero massimo di iterazioni per la convergenza.
set_random_seed	*numero intero*	
seed	*number*	
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*numero intero*	

proprietà coxregnode

Il nodo Regressione di Cox consente di generare un modello di sopravvivenza per i dati della relazione tempo-evento in presenza di record censurati. Il modello produce una funzione di sopravvivenza che prevede la probabilità che l'evento di interesse si sia verificato a una determinata ora (*t*) per i valori dati delle variabili di input.

Esempio

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Tabella 124. proprietà coxregnode

Proprietà coxregnode	Valori	Descrizione proprietà
survival_time	*campo*	I modelli di regressione di Cox richiedono un solo campo contenente i tempi di sopravvivenza.

Tabella 124. proprietà coxregnode (Continua)

Proprietà coxregnode	Valori	Descrizione proprietà
target	*campo*	I modelli di regressione di Cox richiedono un solo campo obiettivo e uno o più campi di input. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
method	Enter Stepwise BackwardsStepwise	
groups	*campo*	
model_type	MainEffects Custom	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*indicatore*	
ci_value	90 95 99	
correlation	*indicatore*	
display_baseline	*indicatore*	
survival	*indicatore*	
hazard	*indicatore*	

Tabella 124. proprietà coxregnode (Continua)

Proprietà coxregnode	Valori	Descrizione proprietà
log_minus_log	*indicatore*	
one_minus_survival	*indicatore*	
separate_line	*campo*	
valore	*numero* o *stringa*	Se per un campo non viene specificato alcun valore sarà utilizzata l'opzione di default "Media".

Proprietà decisionlistnode

Il nodo Elenco di decisioni identifica i sottogruppi o i segmenti che mostrano una probabilità maggiore o minore che si verifichi un determinato risultato binario rispetto alla popolazione globale. Per esempio, è possibile che si cerchino i clienti non a rischio di abbandono o quelli che più probabilmente rispondano in modo favorevole a una campagna. È possibile incorporare le proprie conoscenze di business nel modello aggiungendo propri segmenti personalizzati e visualizzando in anteprima modelli alternativi uno accanto all'altro per confrontarne i risultati. I modelli Elenco di decisioni consistono in un elenco di regole in cui ogni regola ha una condizione e un risultato. Le regole vengono applicate in ordine e la prima regola corrispondente determina il risultato.

Esempio

```
node = stream.create("decisionlist", "My node")
node.setPropertyValue("search_direction", "Down")
node.setPropertyValue("target_value", 1)
node.setPropertyValue("max_rules", 4)
node.setPropertyValue("min_group_size_pct", 15)
```

Tabella 125. Proprietà decisionlistnode

Proprietà decisionlistnode	Valori	Descrizione proprietà
target	*campo*	I modelli Elenco di decisioni utilizzano un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento “Proprietà comuni nodi modellazione” a pagina 185.
model_output_type	Modello InteractiveBuilder	
search_direction	Up Down	Si riferisce alla ricerca di segmenti, dove Up è l'equivalente di Probabilità elevata e Down è l'equivalente di Probabilità bassa.
target_value	*stringa*	Se non specificata, presupporrà il valore vero per i flag.
max_rules	*numero intero*	Il numero massimo di segmenti escluso il resto.
min_group_size	*numero intero*	Dimensione minima del segmento.
min_group_size_pct	*number*	Dimensioni minime del segmento espresse come percentuale.

Tabella 125. Proprietà decisionlistnode (Continua)

Proprietà decisionlistnode	Valori	Descrizione proprietà
confidence_level	*number*	Soglia minima di cui un campo di input deve migliorare la probabilità di risposta (guadagno cumulativo) perché valga la pena aggiungerlo alla definizione di un segmento.
max_segments_per_rule	*numero intero*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*numero intero*	Larghezza di ricerca per gli elenchi.
max_rules_per_cycle	*numero intero*	Larghezza di ricerca per le regole di segmento.
segment_growth	*number*	
include_missing	*indicatore*	
final_results_only	*indicatore*	
reuse_fields	*indicatore*	Consente il riutilizzo degli attributi (campi di input che compaiono nelle regole).
max_alternatives	*numero intero*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

proprietà discriminantnode

L'analisi discriminante prevede presupposti più rigidi rispetto alla regressione logistica, ma può essere una valida alternativa o un complemento dell'analisi di regressione logistica quando vengono soddisfatti tali presupposti.

Esempio

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tabella 126. proprietà discriminantnode

Proprietà discriminantnode	Valori	Descrizione proprietà
target	*campo*	I modelli Discriminante richiedono un solo campo obiettivo e uno o più campi di input. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.

Tabella 126. proprietà discriminantnode (Continua)

Proprietà discriminantnode	Valori	Descrizione proprietà
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*indicatore*	Opzioni relative alle statistiche nella finestra di dialogo Output avanzato.
univariate_anovas	*indicatore*	
box_m	*indicatore*	
within_group_covariance	*indicatore*	
within_groups_correlation	*indicatore*	
separate_groups_covariance	*indicatore*	
total_covariance	*indicatore*	
fishers	*indicatore*	
unstandardized	*indicatore*	
casewise_results	*indicatore*	Opzioni relative alla classificazione nella finestra di dialogo Output avanzato.
limit_to_first	*number*	Il valore di default è 10.
summary_table	*indicatore*	
leave_one_classification	*indicatore*	
combined_groups	*indicatore*	
separate_groups_covariance	*indicatore*	Opzione delle matrici **Covarianza per gruppi separati**.
territorial_map	*indicatore*	
combined_groups	*indicatore*	Opzione del nodo Plot **Gruppi combinati**.
separate_groups	*indicatore*	Opzione del nodo Plot **Gruppi separati**.
summary_of_steps	*indicatore*	
F_pairwise	*indicatore*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	
F_value_entry	*number*	Il valore di default è 3.84.
F_value_removal	*number*	Il valore di default è 2,71.
probability_entry	*number*	Il valore di default è 0.05.
probability_removal	*number*	Il valore di default è 0.10.

Tabella 126. proprietà discriminantnode (Continua)

Proprietà discriminantnode	Valori	Descrizione proprietà
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

Proprietà extensionmodelnode

Con il nodo Modello estensione, è possibile eseguire script R o Python for spark per creare ed assegnare punteggi ai risultati.

Esempio Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_build", "extension_build")
node.setPropertyValue("syntax_type", "Python")

build_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTree

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]

target = "Drug"
predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

def metaMap(row,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(set([row[col]]))
        else:
            meta.append((row[col],row[col]))
        col += 1
    return meta

def metaReduce(meta1,meta2,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(meta1[col].union(meta2[col]))
        else:
            meta.append((min(meta1[col][0],meta2[col][0]),max(meta1[col][1],meta2[col][1])))
```

```
        col += 1
    return meta

metadata = df.rdd.map(lambda row: metaMap(row,schema)).reduce(lambda x,y:metaReduce(x,y,schema))

def setToList(v):
    if isinstance(v,set):
        return list(v)
    return v

metadata = map(lambda x: setToList(x), metadata)
print metadata

lookup = {}
for i in range(0,len(schema)):
    lookup[schema[i][0]] = i

def row2LabeledPoint(dm,lookup,target,predictors,row):
    target_index = lookup[target]
    tval = dm[target_index].index(row[target_index])
    pvals = []
    for predictor in predictors:
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            pval = dm[predictor_index].index(row[predictor_index])
        else:
            pval = row[predictor_index]
        pvals.append(pval)
    return LabeledPoint(tval,DenseVector(pvals))

# count number of target classes
predictorClassCount = len(metadata[lookup[target]])

# define function to extract categorical predictor information from datamodel
def getCategoricalFeatureInfo(dm,lookup,predictors):
    info = {}
    for i in range(0,len(predictors)):
        predictor = predictors[i]
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            info[i] = len(dm[predictor_index])
    return info

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))

treeModel = DecisionTree.trainClassifier(
    lps,
    numClasses=predictorClassCount,
    categoricalFeaturesInfo=getCategoricalFeatureInfo(metadata, lookup, predictors),
    impurity='gini',
    maxDepth=5,
    maxBins=100)

_outputPath = cxt.createTemporaryFolder()
treeModel.save(cxt.getSparkContext(), _outputPath)
cxt.setModelContentFromPath("TreeModel", _outputPath)
cxt.setModelContentFromString("model.dm",json.dumps(metadata), mimeType="application/json")\
   .setModelContentFromString("model.structure",treeModel.toDebugString())

"""

node.setPropertyValue("python_build_syntax", build_script)
```

Esempio R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <- lm(modelerData$Na~modelerData$K,modelerData)
modelerDataModel
modelerModel
""")
```

Tabella 127. Proprietà extensionmodelnode

Proprietà extensionmodelnode	Valori	Descrizione proprietà
syntax_type	*R* *Python*	Specifica quale script viene eseguito, R o Python (R è il valore predefinito).
r_build_syntax	*stringa*	Sintassi degli script R per la creazione del modello.
r_score_syntax	*stringa*	Sintassi degli script R per il calcolo del punteggio del modello.
python_build_syntax	*stringa*	Sintassi degli script Python per la creazione del modello.
python_score_syntax	*stringa*	Sintassi degli script Python per il calcolo del punteggio del modello.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.
output_html	*indicatore*	Opzione per visualizzare grafici su una scheda nel nugget del modello R.
output_text	*indicatore*	Opzione per scrivere l'output di testo della console R in una scheda nel nugget del modello R.

proprietà factornode

Il nodo fattoriale/PCA offre potenti tecniche di riduzione dei dati che consentono di diminuirne la complessità. L'analisi dei componenti principali (PCA, Principal Components Analysis) trova le combinazioni lineari dei campi di input che catturano meglio la varianza nell'intero insieme di campi, dove i componenti sono ortogonali (perpendicolari) l'uno rispetto all'altro. L'analisi fattoriale tenta di identificare i concetti sottostanti, o fattori, che spiegano lo schema delle correlazioni all'interno dell'insieme di campi osservati. Entrambi gli approcci mirano a trovare un numero ridotto di campi derivati che riassumono in modo efficace le informazioni presenti nell'insieme originale di campi.

Esempio

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
```

```
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" section
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Tabella 128. proprietà factornode

Proprietà **factornode**	Valori	Descrizione proprietà
inputs	[*campo1 ... campoN*]	I modelli fattoriali/PCA utilizzano un elenco di campi di input, ma nessun campo obiettivo. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*indicatore*	
matrix	Correlation Covarianza	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	

Tabella 128. proprietà factornode (Continua)

Proprietà **factornode**	Valori	Descrizione proprietà
`delta`	*number*	Se si seleziona `DirectOblimin` come tipo di dati di rotazione, è possibile specificare un valore per `delta`. Se non si specifica un valore, per `delta` verrà utilizzato il valore di default.
`kappa`	*number*	Se si seleziona `Promax` come tipo di dati di rotazione, è possibile specificare un valore per `kappa`. Se non si specifica un valore, per `kappa` verrà utilizzato il valore di default.
`sort_values`	*indicatore*	
`hide_values`	*indicatore*	
`hide_below`	*number*	

proprietà featureselectionnode

Il nodo Selezione funzioni effettua lo screening dei campi di input, rimuovendoli in base a un insieme di criteri quali la percentuale di valori mancanti. Classifica quindi gli input restanti in ordine di importanza rispetto a un determinato obiettivo. Per esempio, dato un insieme di dati con centinaia di input potenziali, quali sono quelli con la maggiore probabilità di essere utili nella modellazione di risultati clinici?

Esempio

```
node = stream.create("featureselection", "My node")
node.setPropertyValue("screen_single_category", True)
node.setPropertyValue("max_single_category", 95)
node.setPropertyValue("screen_missing_values", True)
node.setPropertyValue("max_missing_values", 80)
node.setPropertyValue("criteria", "Likelihood")
node.setPropertyValue("unimportant_below", 0.8)
node.setPropertyValue("important_above", 0.9)
node.setPropertyValue("important_label", "Check Me Out!")
node.setPropertyValue("selection_mode", "TopN")
node.setPropertyValue("top_n", 15)
```

Per un esempio più dettagliato di creazione e applicazione di un modello di selezione funzioni, vedere in in.

Tabella 129. proprietà featureselectionnode

Proprietà **featureselectionnode**	Valori	Descrizione proprietà
`target`	*campo*	I modelli di selezione funzioni classificano i predittori rispetto all'obiettivo specificato. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento “Proprietà comuni nodi modellazione” a pagina 185.

Tabella 129. proprietà featureselectionnode (Continua)

Proprietà **featureselectionnode**	Valori	Descrizione proprietà
screen_single_category	*indicatore*	Se True, esegue lo screening dei campi che hanno troppi record che rientrano nella stessa categoria rispetto al numero totale di record.
max_single_category	*number*	Specifica la soglia utilizzata quando screen_single_category è True.
screen_missing_values	*indicatore*	Se True, esegue lo screening dei campi con troppi valori mancanti, espressi come percentuale del numero totale di record.
max_missing_values	*number*	
screen_num_categories	*indicatore*	Se True, esegue lo screening dei campi con troppe categorie rispetto al numero totale di record.
max_num_categories	*number*	
screen_std_dev	*indicatore*	Se True, esegue lo screening dei campi con una deviazione standard inferiore o uguale al minimo specificato.
min_std_dev	*number*	
screen_coeff_of_var	*indicatore*	Se True, esegue lo screening dei campi con un coefficiente di varianza inferiore o uguale al minimo specificato.
min_coeff_of_var	*number*	
criteria	Pearson Likelihood CramersV Lambda	Quando si classificano i predittori categoriali rispetto a un obiettivo categoriale, specifica la misura sulla quale si basa il valore di importanza.
unimportant_below	*number*	Specifica i valori *p* di soglia utilizzati per classificare variabili quali importante, marginale o non importante. Accetta i valori compresi fra 0.0 e 1.0.
important_above	*number*	Accetta i valori compresi fra 0.0 e 1.0.
unimportant_label	*stringa*	Specifica l'etichetta per la classificazione non importante.
marginal_label	*stringa*	
important_label	*stringa*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi importanti.
select_marginal	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi marginali.
select_unimportant	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi non importanti.

Tabella 129. proprietà featureselectionnode (Continua)

Proprietà **featureselectionnode**	Valori	Descrizione proprietà
importance_value	*number*	Quando selection_mode è impostata su ImportanceValue, specifica il valore di interruzione da utilizzare. Accetta i valori compresi tra 0 e 100.
top_n	*numero intero*	Quando selection_mode è impostata su TopN, specifica il valore di interruzione da utilizzare. Accetta i valori compresi tra 0 e 1000.

proprietà genlinnode

Il modello Lineare generalizzato amplia il modello lineare generale in modo che la variabile dipendente venga linearmente correlata ai fattori e alle covariate tramite una funzione di collegamento specifica. Inoltre, il modello consente alla variabile dipendente di avere una distribuzione non normale. Copre la funzionalità di un grande numero di modelli statistici, inclusi modelli di regressione lineare, modelli di regressione logistica, modelli loglineari per dati dei conteggi e modelli di sopravvivenza censurati per intervallo.

Esempio

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabella 130. proprietà genlinnode

Proprietà **genlinnode**	Valori	Descrizione proprietà
target	*campo*	I modelli lineari generalizzati richiedono un solo campo obiettivo (che deve essere nominale o flag) e uno o più campi di input. È anche possibile specificare un campo peso. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
use_weight	*indicatore*	
weight_field	*campo*	Il tipo di campo è solo continuo.
target_represents_trials	*indicatore*	
trials_type	Variable FixedValue	
trials_field	*campo*	Il tipo di campo è continuo, flag o ordinale.
trials_number	*number*	Il valore di default è 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*campo*	Il tipo di campo è solo continuo.
offset_value	*number*	Deve essere un numero reale.

Tabella 130. proprietà genlinnode (Continua)

Proprietà genlinnode	Valori	Descrizione proprietà
base_category	Last First	
include_intercept	*indicatore*	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: gaussiana inversa. NEGBIN: binomiale negativa.
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	Il valore di default è 1. Deve contenere un numero reale non negativo.
tweedie_parameter	*number*	
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: doppia logaritmica complementare. LOGC: log-complemento. NEGBIN: binomiale negativa. NLOGLOG: doppia logaritmica negativa. CUMCAUCHIT: cauchit cumulativa. CUMCLOGLOG: log-log complementare cumulativa. CUMLOGIT: logit cumulativa. CUMNLOGLOG: log-log negativa cumulativa. CUMPROBIT: probit cumulativa.
power	*number*	Il valore deve essere un numero reale diverso da zero.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*number*	Il valore di default è 1; sono consentiti solo numeri interi positivi.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*number*	Il valore di default è 1; deve essere maggiore di 0.
covariance_matrix	ModelEstimator RobustEstimator	

Tabella 130. proprietà genlinnode (Continua)

Proprietà genlinnode	Valori	Descrizione proprietà
max_iterations	*number*	Il valore di default è 100; solo numeri interi non negativi.
max_step_halving	*number*	Il valore di default è 5; solo numeri interi positivi.
check_separation	*indicatore*	
start_iteration	*number*	Il valore di default è 20; sono consentiti solo numeri interi positivi.
estimates_change	*indicatore*	
estimates_change_min	*number*	Il valore di default è 1E-006; sono consentiti solo numeri positivi.
estimates_change_type	Absolute Relative	
loglikelihood_change	*indicatore*	
loglikelihood_change_min	*number*	Sono consentiti solo numeri positivi.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*indicatore*	
hessian_convergence_min	*number*	Sono consentiti solo numeri positivi.
hessian_convergence_type	Absolute Relative	
case_summary	*indicatore*	
contrast_matrices	*indicatore*	
descriptive_statistics	*indicatore*	
estimable_functions	*indicatore*	
model_info	*indicatore*	
iteration_history	*indicatore*	
goodness_of_fit	*indicatore*	
print_interval	*number*	Il valore di default è 1; deve essere un numero intero positivo.
model_summary	*indicatore*	
lagrange_multiplier	*indicatore*	
parameter_estimates	*indicatore*	
include_exponential	*indicatore*	
covariance_estimates	*indicatore*	
correlation_estimates	*indicatore*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*number*	Il valore di default è 0.0001.

Tabella 130. proprietà genlinnode (Continua)

Proprietà genlinnode	Valori	Descrizione proprietà
confidence_interval	*number*	Il valore di default è 95.
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Crescente Decrescente DataOrder	
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

Proprietà glmmnode

Un modello misto lineare generalizzato (GLMM) estende il modello lineare in modo che l'obiettivo possa avere una distribuzione non normale, sia linearmente correlato ai fattori e alle covariate tramite una funzione di collegamento specifica e in modo che le osservazioni possano essere correlate. I modelli misti lineari generalizzati includono un'ampia gamma di modelli, dalla regressione lineare semplice ai modelli multilivello complessi per i dati longitudinali non normali.

Tabella 131. Proprietà glmmnode

Proprietà glmmnode	Valori	Descrizione proprietà
residual_subject_spec	*strutturato*	La combinazione di valori dei campi categoriali specificati che definisce in modo univoco i soggetti all'interno dell'insieme di dati
repeated_measures	*strutturato*	I campi utilizzati per identificare le osservazioni ripetute.
residual_group_spec	[*campo1 ... campoN*]	I campi che definiscono insiemi indipendenti di parametri di covarianza a effetti ripetuti.
residual_covariance_type	Diagonale AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Specifica la struttura di covarianza per i residui.

Tabella 131. Proprietà glmmnode (Continua)

Proprietà glmmnode	Valori	Descrizione proprietà
custom_target	*indicatore*	Indica se utilizzare la destinazione definita nel nodo upstream (false) o la destinazione personalizzata specificata da target_field (true).
target_field	*campo*	Il campo da utilizzare come destinazione se custom_target è true.
use_trials	*indicatore*	Indica se un campo o valore aggiuntivo che specifica il numero di prove deve essere utilizzato quando la risposta obiettivo rappresenta un numero di eventi che si verificano in un insieme di prove. Il valore di default è false.
use_field_or_value	Campo Valore	Indica se il campo (default) o valore viene utilizzato per specificare il numero di prove.
trials_field	*campo*	Campo da utilizzare per specificare il numero di prove.
trials_value	*numero intero*	Valore da utilizzare per specificare il numero di prove. Se specificato, il valore minimo è 1.
use_custom_target_reference	*indicatore*	Indica se la categoria di riferimento personalizzata deve essere utilizzata per un target di categoria. Il valore di default è false.
target_reference_value	*stringa*	La categoria di riferimento da utilizzare se use_custom_target_reference è true.
dist_link_combination	Nominale Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	I modelli comuni per la distribuzione dei valori dell'obiettivo. Scegliere Custom per specificare una distribuzione dall'elenco fornito da target_distribution.
target_distribution	Normal Binomial Multinomial Gamma Inverso NegativeBinomial Poisson	Distribuzione dei valori per l'obiettivo quando dist_link_combination è Custom.

Tabella 131. Proprietà glmmnode (Continua)

Proprietà glmmnode	Valori	Descrizione proprietà
link_function_type	Identità LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Funzione di collegamento per correlare i valori obiettivo per Se target_distribution è Binomial è possibile utilizzare una qualsiasi delle funzioni di collegamento elencate. Se target_distribution è Multinomial è possibile utilizzare CLOGLOG, CAUCHIT, LOGIT, NLOGLOG oppure PROBIT. Se target_distribution è diverso da Binomial o Multinomial è possibile utilizzare IDENTITY, LOG oppure POWER.
link_function_param	*number*	Il valore del parametro della funzione di collegamento da utilizzare. Applicabile solo se normal_link_function o link_function_type è POWER.
use_predefined_inputs	*indicatore*	Indica se i campi a effetto fisso devono essere quelli definiti a monte come campi di input (true) o quelli di fixed_effects_list (false). Il valore di default è false.
fixed_effects_list	*strutturato*	Se use_predefined_inputs è false, specifica i campi di input da utilizzare come campi a effetto fisso.
use_intercept	*indicatore*	Se true (default), include l'intercettazione nel modello.
random_effects_list	*strutturato*	Elenco dei campi da specificare come effetti random.
regression_weight_field	*campo*	Campo da utilizzare come campo del peso dell'analisi.
use_offset	None offset_value offset_field	Indica il modo in cui viene specificato l'offset. Il valore None indica che non viene utilizzato nessun offset.
offset_value	*number*	Il valore da utilizzare per l'offset se use_offset è impostato su offset_value.
offset_field	*campo*	Il campo da utilizzare per il valore offset se use_offset è impostato su offset_field.
target_category_order	Crescente Descending Data	Criterio di ordinamento per i target di categoria. Il valore Data specifica l'utilizzo del criterio di ordinamento trovato nei dati. L'impostazione di default è Ascending.
inputs_category_order	Crescente Descending Data	Criterio di ordinamento per i predittori di categoria. Il valore Data specifica l'utilizzo del criterio di ordinamento trovato nei dati. L'impostazione di default è Ascending.
max_iterations	*numero intero*	Numero massimo di iterazioni che l'algoritmo eseguirà. Un numero intero non negativo; l'impostazione di default è 100.

Tabella 131. Proprietà glmmnode (Continua)

Proprietà glmmnode	Valori	Descrizione proprietà
confidence_level	*numero intero*	Livello di confidenza utilizzato per calcolare le stime di intervallo dei coefficienti del modello. Un numero intero non negativo; il massimo è 100, l'impostazione di default è 95.
degrees_of_freedom_method	Fixed Varied	Specifica la modalità di calcolo dei gradi di libertà per i test di significatività.
test_fixed_effects_coeffecients	Modello Robust	Il metodo per il calcolo della matrice di covarianza delle stime dei parametri.
use_p_converge	*indicatore*	Opione per la convergenza dei parametri.
p_converge	*number*	Vuoto, o qualsiasi valore positivo.
p_converge_type	Assoluti Relative	
use_l_converge	*indicatore*	Opzione per la convergenza di verosimiglianza logaritmica.
l_converge	*number*	Vuoto, o qualsiasi valore positivo.
l_converge_type	Assoluti Relative	
use_h_converge	*indicatore*	Opzione per la convergenza hessiana.
h_converge	*number*	Vuoto, o qualsiasi valore positivo.
h_converge_type	Assoluti Relative	
max_fisher_steps	*numero intero*	
singularity_tolerance	*number*	
use_model_name	*indicatore*	Indica se specificare un nome personalizzato per il modello (true) o utilizzare il nome generato dal sistema (false). Il valore predefinito è false.
model_name	*stringa*	Se use_model_name è true, specifica il nome del modello da utilizzare.
confidence	onProbability onIncrease	Base per il calcolo del valore di confidenza del punteggio: probabilità prevista più alta o differenza tra le probabilità più alte e la seconda massima prevista.
score_category_probabilities	*indicatore*	Se true, produce le probabilità previste per i target di categoria. Il valore di default è false.
max_categories	*numero intero*	Se score_category_probabilities è true, specifica il numero massimo di categorie da salvare.
score_propensity	*indicatore*	Se true, produce punteggi di propensione per i campi obiettivo di tipo indicatore che indicano la probabilità del risultato "true" per il campo.
emeans	*struttura*	Per ogni campo relativo alla categoria dall'elenco a effetti fissi, specifica se produrre le medie marginali stimate.

Tabella 131. Proprietà glmmnode (Continua)

Proprietà glmmnode	Valori	Descrizione proprietà
covariance_list	*struttura*	Per ogni campo continuo dall'elenco effetti fissi, specifica se utilizzare la media o un valore personalizzato quando si calcola le medie marginali stimate.
mean_scale	Original Transformed	Specifica se calcolare le medie marginali stimate in base alla scala originale dell'obiettivo (default) o in base alla trasformazione della funzione di collegamento.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Metodo di regolazione da utilizzare quando si esegue il test sull'ipotesi con più contrasti.

Proprietà gle

Un nodo GLE estende il modello lineare in modo che l'obiettivo possa avere una distribuzione non normale, sia linearmente correlato ai fattori ed alle covariate mediante una funzione di collegamento specificata e in modo che le osservazioni possano essere correlate. I modelli misti lineari generalizzati includono un'ampia gamma di modelli, dalla regressione lineare semplice ai modelli multilivello complessi per i dati longitudinali non normali.

Tabella 132. Proprietà gle

Proprietà gle	Valori	Descrizione proprietà
custom_target	*indicatore*	Indica se utilizzare la destinazione definita nel nodo upstream (false) o la destinazione personalizzata specificata da target_field (true).
target_field	*campo*	Il campo da utilizzare come destinazione se custom_target è true.
use_trials	*indicatore*	Indica se un campo o valore aggiuntivo che specifica il numero di prove deve essere utilizzato quando la risposta obiettivo rappresenta un numero di eventi che si verificano in un insieme di prove. Il valore di default è false.
use_trials_field_or_value	Campo Valore	Indica se il campo (default) o valore viene utilizzato per specificare il numero di prove.
trials_field	*campo*	Campo da utilizzare per specificare il numero di prove.
trials_value	*numero intero*	Valore da utilizzare per specificare il numero di prove. Se specificato, il valore minimo è 1.
use_custom_target_reference	*indicatore*	Indica se la categoria di riferimento personalizzata deve essere utilizzata per un target di categoria. Il valore predefinito è false.
target_reference_value	*stringa*	La categoria di riferimento da utilizzare se use_custom_target_reference è true.

Tabella 132. Proprietà gle (Continua)

Proprietà gle	Valori	Descrizione proprietà
dist_link_combination	NormalIdentity GammaLog PoissonLog NegbinLog TweedieIdentity NominalLogit BinomialLogit BinomialProbit BinomialLogC CUSTOM	I modelli comuni per la distribuzione dei valori dell'obiettivo. Scegliere CUSTOM per specificare una distribuzione dall'elenco fornito da target_distribution.
target_distribution	Normal Binomial Multinomial Gamma INVERSE_GAUSS NEG_BINOMIAL Poisson TWEEDIE UNKNOWN	Distribuzione dei valori per l'obiettivo quando dist_link_combination è Custom.
link_function_type	UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER NEG_BINOMIAL GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT	Funzione di collegamento per correlare i valori obiettivo per i predittori. Se target_distribution è Binomial è possibile utilizzare: UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER Se target_distribution è NEG_BINOMIAL è possibile utilizzare: NEG_BINOMIAL. Se target_distribution è UNKNOWN è possibile utilizzare: GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT
link_function_param	*number*	Valore del parametro Tweedie da utilizzare. Applicabile solo se normal_link_function o link_function_type è POWER.
tweedie_param	*number*	Il valore del parametro della funzione di collegamento da utilizzare. Applicabile solo se dist_link_combination è impostata su TweedieIdentity o link_function_type è TWEEDIE.
use_predefined_inputs	*indicatore*	Indica se i campi a effetto del modello devono essere quelli definiti a monte come campi di input (true) o quelli di fixed_effects_list (false).

Tabella 132. Proprietà gle (Continua)

Proprietà gle	Valori	Descrizione proprietà
model_effects_list	*strutturato*	Se use_predefined_inputs è false, specifica i campi di input da utilizzare come campi a effetto del modello.
use_intercept	*indicatore*	Se true (default), include l'intercettazione nel modello.
regression_weight_field	*campo*	Campo da utilizzare come campo del peso dell'analisi.
use_offset	None Value Variable	Indica il modo in cui viene specificato l'offset. Il valore None indica che non viene utilizzato nessun offset.
offset_value	*number*	Il valore da utilizzare per l'offset se use_offset è impostato su offset_value.
offset_field	*campo*	Il campo da utilizzare per il valore offset se use_offset è impostato su offset_field.
target_category_order	Crescente Decrescente	Criterio di ordinamento per i target di categoria. L'impostazione di default è Ascending.
inputs_category_order	Crescente Decrescente	Criterio di ordinamento per i predittori di categoria. L'impostazione di default è Ascending.
max_iterations	*numero intero*	Numero massimo di iterazioni che l'algoritmo eseguirà. Un numero intero non negativo; l'impostazione di default è 100.
confidence_level	*number*	Livello di confidenza utilizzato per calcolare le stime di intervallo dei coefficienti del modello. Un numero intero non negativo; il massimo è 100, l'impostazione di default è 95.
test_fixed_effects_coeffecients	Modello Robust	Il metodo per il calcolo della matrice di covarianza delle stime dei parametri.
detect_outliers	*indicatore*	Quando è impostata su true, l'algoritmo trova i valori anomali influenti per tutte le distribuzioni ad eccezione delle distribuzioni multinomiali.
conduct_trend_analysis	*indicatore*	Quando è impostata su true, l'algoritmo effettua l'analisi delle tendenze per il grafico a dispersione.
estimation_method	FISHER_SCORING NEWTON_RAPHSON HYBRID	Specificare l'algoritmo di stima della verosimiglianza massima.
max_fisher_iterations	*numero intero*	Se si utilizza FISHER_SCORING estimation_method, il numero massimo di iterazioni. Minimum 0, maximum 20.
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	Specificare il metodo da utilizzare per la stima del parametro di scala.
scale_value	*number*	Disponibile solo se scale_parameter_method è impostata su Fixed.
negative_binomial_method	MLE FIXED	Specificare il metodo per la stima del parametro ausiliario binomiale negativo.
negative_binomial_value	*number*	Disponibile solo se negative_binomial_method è impostato su Fixed.

Tabella 132. Proprietà gle (Continua)

Proprietà gle	Valori	Descrizione proprietà
use_p_converge	*indicatore*	Opzione per la convergenza dei parametri.
p_converge	*number*	Vuoto, o qualsiasi valore positivo.
p_converge_type	*indicatore*	True = Absolute, False = Relative
use_l_converge	*indicatore*	Opzione per la convergenza di verosimiglianza logaritmica.
l_converge	*number*	Vuoto, o qualsiasi valore positivo.
l_converge_type	*indicatore*	True = Absolute, False = Relative
use_h_converge	*indicatore*	Opzione per la convergenza hessiana.
h_converge	*number*	Vuoto, o qualsiasi valore positivo.
h_converge_type	*indicatore*	True = Absolute, False = Relative
max_iterations	*numero intero*	Numero massimo di iterazioni che l'algoritmo eseguirà. Un numero intero non negativo; l'impostazione di default è 100.
sing_tolerance	*numero intero*	
use_model_selection	*indicatore*	Abilita i controlli del metodo di selezione del modello e della soglia del parametro.
method	LASSO ELASTIC_NET FORWARD_STEPWISE RIDGE	Determina il metodo di selezione del modello oppure, se si utilizza Ridge, il metodo di regolarizzazione utilizzato.
detect_two_way_interactions	*indicatore*	Quando è impostata su True, il modello rileva automaticamente le interazioni a due vie tra i campi di input. Questo controllo dovrebbe essere abilitato solo se il modello è solo di effetti principali (cioè in cui l'utente non ha creato altri effetti di ordine superiore) e se il Metodo selezionato è Forward Stepwise, Lasso o Elastic Net.
automatic_penalty_params	*indicatore*	Disponibile solo se il method di selezione del modello è Lasso o Elastic Net. Utilizzare questa funzione per immettere i parametri di penalità associati ai metodi di selezione delle variabili Lasso o Elastic Net. Se è impostata su True, vengono utilizzati i valori predefiniti. Se è impostata su False, i parametri di penalità sono abilitati ed è possibile immettere valori personalizzati.
lasso_penalty_param	*number*	Disponibile solo se il method di selezione del modello è Lasso o Elastic Net e automatic_penalty_params è False. Specificare il valore del parametro di penalità per Lasso.
elastic_net_penalty_param1	*number*	Disponibile solo se il method di selezione del modello è Lasso o Elastic Net e automatic_penalty_params è False. Specificare il valore del parametro di penalità per il parametro Elastic Net 1.

Tabella 132. Proprietà gle (Continua)

Proprietà gle	Valori	Descrizione proprietà
elastic_net_penalty_param2	*number*	Disponibile solo se il method di selezione del modello è Lasso o Elastic Net e automatic_penalty_params è False. Specificare il valore del parametro di penalità per il parametro Elastic Net 2.
probability_entry	*number*	Disponibile solo se il method selezionato è Forward Stepwise. Specificare il livello di significatività del criterio statistica f per l'inserimento degli effetti.
probability_removal	*number*	Disponibile solo se il method selezionato è Forward Stepwise. Specificare il livello di significatività del criterio statistica f per la rimozione degli effetti.
use_max_effects	*indicatore*	Disponibile solo se il method selezionato è Forward Stepwise. Abilita il controllo max_effects. Quando è impostata su False il numero predefinito di effetti deve essere uguale al numero totale di effetti forniti al modello, meno l'intercettazione.
max_effects	*numero intero*	Specificare il numero massimo di effetti quando si utilizza il metodo di creazione Stepwise in avanti.
use_max_steps	*indicatore*	Abilita il controllo max_steps. Quando è impostata su False, il numero predefinito di passi deve essere uguale al triplo del numero di effetti forniti al modello, esclusa l'intercettazione.
max_steps	*numero intero*	Specificare il numero massimo di fasi da compiere quando si utilizza il method di creazione Forward Stepwise.
use_model_name	*indicatore*	Indica se specificare un nome personalizzato per il modello (true) o utilizzare il nome generato dal sistema (false). Il valore di default è false.
model_name	*stringa*	Se use_model_name è true, specifica il nome del modello da utilizzare.
usePI	*indicatore*	Se true, l'importanza predittore viene calcolata.

proprietà kmeansnode

Il nodo Medie K raggruppa l'insieme di dati in gruppi distinti (o cluster). Il metodo definisce un numero fisso di cluster, esegue un'assegnazione iterativa dei record ai cluster e modifica i centri di cluster finché un'ulteriore ridefinizione non consente più un miglioramento del modello. Invece di tentare di prevedere un risultato, il nodo *K*-medie utilizza un processo denominato apprendimento non supervisionato per scoprire gli schemi nell'insieme di campi di input.

Esempio

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K", "Age"])
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

Tabella 133. proprietà kmeansnode

Proprietà kmeansnode	Valori	Descrizione proprietà
inputs	[*campo1 ... campoN*]	I modelli Medie K eseguono l'analisi dei cluster su un insieme di campi di input ma non utilizzano un campo obiettivo. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
num_clusters	*number*	
gen_distance	*indicatore*	
cluster_label	String Number	
label_prefix	*stringa*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	
optimize	Speed Memory	Utilizzare per specificare se ottimizzare la velocità o la memoria durante la creazione del modello.

Proprietà kmeansasnode

K-Means è uno degli algoritmi di clustering più comunemente usati. Raggruppa i punti dati in un numero predefinito di cluster. Il nodo K-Means-AS in SPSS Modeler è implementato in Spark. Per i dettagli relativi agli algoritmi K-Means, consultare https://spark.apache.org/docs/2.2.0/ml-clustering.html. Si noti che il nodo K-Means-AS esegue automaticamente una codifica one-hot per le variabili categoriali.

Tabella 134. proprietà kmeansasnode

Proprietàkmeansasnode	Valori	Descrizione proprietà
roleUse	*stringa*	Specificare predefined per utilizzare i ruoli predefiniti oppure custom per utilizzare le assegnazioni di campo personalizzate. Il valore predefinito è predefined.
autoModel	*Booleano*	Specificare true per utilizzare il nome predefinito ($S-prediction) per il nuovo campo di calcolo del punteggio generato, oppure false per utilizzare un nome personalizzato. Il valore predefinito è true.
features	*campo*	Elenco dei nomi dei campi per l'input quando la proprietà roleUse è impostata su custom.
name	*stringa*	Il nome del nuovo campo di calcolo del punteggio generato quando la proprietà autoModel è impostata su false.
clustersNum	*numero intero*	Il numero di cluster da creare. Il valore predefinito è 5.
initMode	*stringa*	L'algoritmo di inizializzazione. I valori possibili sono k-means|| o random. Il valore predefinito è k-means||.
initSteps	*numero intero*	Il numero di fasi di inizializzazione quando initMode è impostato su k-means||. Il valore predefinito è 2.
advancedSettings	*Booleano*	Specificare true per rendere disponibili le seguenti quattro proprietà. Il valore predefinito è false.
maxIteration	*numero intero*	Numero massimo di iterazioni per l'ottimizzazione. Il valore predefinito è 20.
tolerance	*stringa*	La tolleranza per arrestare le iterazioni. Le impostazioni possibili sono 1.0E-1, 1.0E-2, ..., 1.0E-6. Il valore predefinito è 1.0E-4.
setSeed	*Booleano*	Specificare true per utilizzare un seed random. Il valore predefinito è false.
randomSeed	*numero intero*	Il seed random personalizzato quando la proprietà setSeed è true.

proprietà knnnode

Il nodo Elemento vicino più prossimo *K* (KNN) associa un nuovo caso alla categoria o valore degli oggetti *K* più vicini ad esso nello spazio predittore, dove *K* è un numero intero. I casi simili sono vicini gli uni agli altri, mentre i casi dissimili sono distanti gli uni dagli altri.

Esempio

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
```

```
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

Tabella 135. proprietà knnnode

Proprietà knnnode	Valori	Descrizione proprietà
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*indicatore*	
use_case_labels	*indicatore*	Selezionare la casella per abilitare l'opzione successiva.
case_labels_field	*campo*	
identify_focal_cases	*indicatore*	Selezionare la casella per abilitare l'opzione successiva.
focal_cases_field	*campo*	
automatic_k_selection	*indicatore*	
fixed_k	*numero intero*	Attiva solo se automatic_k_selection è False.
minimum_k	*numero intero*	Attiva solo se automatic_k_selection è True.
maximum_k	*numero intero*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*indicatore*	
range_predictions	Mean Median	
perform_feature_selection	*indicatore*	
forced_entry_inputs	[*campo1 ... campoN*]	
stop_on_error_ratio	*indicatore*	
number_to_select	*numero intero*	
minimum_change	*number*	
validation_fold_assign_by_field	*indicatore*	
number_of_folds	*numero intero*	Attiva solo se validation_fold_assign_by_field è False
set_random_seed	*indicatore*	
random_seed	*number*	
folds_field	*campo*	Attiva solo se validation_fold_assign_by_field è True
all_probabilities	*indicatore*	
save_distances	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Tabella 135. proprietà knnnode (Continua)

Proprietà knnnode	Valori	Descrizione proprietà
adjusted_propensity_partition	Test Validation	

proprietà kohonennode

Il nodo Kohonen genera un tipo di rete neurale che può essere utilizzato per raggruppare l'insieme di dati in gruppi distinti. Al termine dell'apprendimento della rete, i record analoghi dovranno essere vicini nella mappa di output, mentre i record diversi saranno a notevole distanza. Per identificare le unità forti, è possibile controllare il numero di osservazioni catturate da ciascuna unità nel nugget del modello. In questo modo è possibile avere un'idea del numero appropriato di cluster.

Esempio

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

Tabella 136. proprietà kohonennode

Proprietà kohonennode	Valori	Descrizione proprietà
inputs	[*campo1 ... campoN*]	I modelli Kohonen utilizzano un elenco di campi di input, ma nessun campo obiettivo. I campi frequenza e peso non sono utilizzati. Per ulteriori informazioni, consultare l'argomento “Proprietà comuni nodi modellazione” a pagina 185.
continue	*indicatore*	
show_feedback	*indicatore*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	Utilizzare per specificare se ottimizzare la velocità o la memoria durante la creazione del modello.
cluster_label	*indicatore*	

Tabella 136. proprietà kohonennode (Continua)

Proprietà kohonennode	Valori	Descrizione proprietà
mode	Simple Expert	
width	*number*	
length	*number*	
decay_style	Linear Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

Proprietà linearnode

I modelli di regressione lineare prevedono un target continuo basato sulle relazioni lineari tra l'obiettivo e uno o più predittori.

Esempio

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabella 137. Proprietà linearnode

Proprietà linearnode	Valori	Descrizione proprietà
target	*campo*	Specifica un singolo campo obiettivo.
inputs	[*campo1 ... campoN*]	I campi predittore utilizzati dal modello.
continue_training_existing_model	*indicatore*	
objective	Standard Bagging Boosting psm	psm viene utilizzato per insiemi di dati di grandi dimensioni e richiede una connessione Server.
use_auto_data_preparation	*indicatore*	
confidence_level	*number*	
model_selection	ForwardStepwise BestSubsets None	

Tabella 137. Proprietà linearnode (Continua)

Proprietà linearnode	Valori	Descrizione proprietà
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*indicatore*	
max_effects	*number*	
use_max_steps	*indicatore*	
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
use_random_seed	*indicatore*	
random_seed	*number*	
use_custom_model_name	*indicatore*	
custom_model_name	*stringa*	
use_custom_name	*indicatore*	
custom_name	*stringa*	
tooltip	*stringa*	
keywords	*stringa*	
annotation	*stringa*	

Proprietà linearasnode

I modelli di regressione lineare prevedono un target continuo basato sulle relazioni lineari tra l'obiettivo e uno o più predittori.

Tabella 138. Proprietà linearasnode

Proprietà linearasnode	Valori	Descrizione proprietà
obiettivo	*campo*	Specifica un singolo campo obiettivo.
inputs	[*campo1 ... campoN*]	I campi predittore utilizzati dal modello.
weight_field	*campo*	Campo di analisi utilizzato dal modello.
custom_fields	*indicatore*	Il valore predefinito è TRUE.
intercept	*indicatore*	Il valore predefinito è TRUE.

Tabella 138. Proprietà linearasnode (Continua)

Proprietà linearasnode	Valori	Descrizione proprietà
detect_2way_interaction	*indicatore*	Indica se considerare o meno un'interazione a due vie. Il valore predefinito è TRUE.
cin	*number*	L'intervallo di confidenza utilizzato per calcolare le stime dei coefficienti del modello. Specificare un valore maggiore di 0 e minore di 100. Il valore predefinito è 95.
factor_order	ascending descending	Il criterio di ordinamento per i predittori di categoria. Il valore predefinito è ascending.
var_select_method	ForwardStepwise BestSubsets none	Il metodo di selezione del modello da utilizzare. Il valore predefinito è ForwardStepwise.
criteria_for_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	La statistica utilizzata per determinare se un effetto deve essere aggiunto o eliminato dal modello. Il valore predefinito è AdjustedRSquare.
pin	*number*	L'effetto con il valore P più piccolo e minore di quello specificato nella soglia pin viene aggiunto al modello. Il valore predefinito è 0.05.
pout	*number*	Tutti gli effetti presenti nel modello che hanno un valore p superiore alla soglia pout specificata vengono eliminati. Il valore predefinito è 0.10.
use_custom_max_effects	*indicatore*	Indica se utilizzare il numero massimo di effetti nel modello finale. Il valore predefinito è FALSE.
max_effects	*number*	Numero massimo di effetti da utilizzare nel modello finale. Il valore predefinito è 1.
use_custom_max_steps	*indicatore*	Indica se utilizzare il numero massimo di fasi. Il valore predefinito è FALSE.
max_steps	*number*	Il numero massimo di fasi prima che l'algoritmo stepwise venga arrestato. Il valore predefinito è 1.
criteria_for_best_subsets	AICC AdjustedRSquare ASE	La modalità del criterio da utilizzare. Il valore predefinito è AdjustedRSquare.

proprietà logregnode

La regressione logistica, una tecnica statistica che consente di classificare i record in base ai valori dei campi di input, è analoga alla regressione lineare ma, al posto di un intervallo numerico, prende un campo obiettivo categoriale.

Esempio multinomiale

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
```

```
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" options
node.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
node.setPropertyValue("max_terms", 10)
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

Esempio binomiale

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
```

```
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Tabella 139. proprietà logregnode

Proprietà logregnode	Valori	Descrizione proprietà
target	*campo*	I modelli di regressione logistica richiedono un solo campo obiettivo e uno o più campi di input. I campi frequenza e peso non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
logistic_procedure	Binomial Multinomial	
include_constant	*indicatore*	
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	
model_type	MainEffects FullFactorial Custom	Se FullFactorial è specificato come tipo di modello, i criteri di controllo non verranno eseguiti, anche se sono specificati. Verrà invece utilizzato il metodo Enter. Se il tipo di modello è impostato su Custom, ma non sono stati specificati campi personalizzati, verrà creato un modello effetti principali.
custom_terms	[*[BP Sex][BP][Age]*]	

Tabella 139. proprietà logregnode (Continua)

Proprietà logregnode	Valori	Descrizione proprietà
multinomial_base_category	*stringa*	Specifica come viene determinata la categoria di riferimento.
binomial_categorical_input	*stringa*	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Proprietà basata su chiavi per input categoriali che indica come viene determinato il confronto.
binomial_input_category	First Last	Proprietà basata su chiavi per input categoriali che indica come viene determinata la categoria di riferimento.
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*indicatore*	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	*indicatore*	
max_terms	*number*	
entry_criterion	Punteggio LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requisiti	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*number*	
max_steps	*number*	

Tabella 139. proprietà logregnode (Continua)

Proprietà logregnode	Valori	Descrizione proprietà
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	*indicatore*	
history_steps	*number*	
summary	*indicatore*	
likelihood_ratio	*indicatore*	
asymptotic_correlation	*indicatore*	
goodness_fit	*indicatore*	
parametri	*indicatore*	
confidence_interval	*number*	
asymptotic_covariance	*indicatore*	
classification_table	*indicatore*	
stepwise_summary	*indicatore*	
info_criteria	*indicatore*	
monotonicity_measures	*indicatore*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*indicatore*	
binomial_parameters	*indicatore*	
binomial_iteration_history	*indicatore*	
binomial_classification_plots	*indicatore*	
binomial_ci_enable	*indicatore*	
binomial_ci	*number*	
binomial_residual	valori anomali all	
binomial_residual_enable	*indicatore*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	

Proprietà lsvmnode

Il nodo LSVM (Linear Support Vector Machine) consente di classificare i dati in uno di due gruppi senza sovradattamento. Il nodo LSVM è lineare e particolarmente indicato con dataset di grandi dimensioni, come quelli con un numero elevato di record.

Tabella 140. Proprietà lsvmnode

Proprietà `lsvmnode`	Valori	Descrizione proprietà
`intercept`	*indicatore*	Include l'intercettazione nel modello. Il valore predefinito è `True`.
`target_order`	`Crescente` `Decrescente`	Specifica il criterio di ordinamento per l'obiettivo categoriale. Ignorato per gli obiettivi continui. L'impostazione di default è `Ascending`.
`precision`	*number*	Utilizzata solo se il livello di misurazione del campo obiettivo è `Continuous`. Specifica il parametro correlato alla sensibilità della perdita di regressione. Il valore minimo è `0` e non è previsto un valore massimo. Il valore predefinito è `0.1`.
`exclude_missing_values`	*indicatore*	Quando è impostata su `True`, un record viene escluso se manca un singolo valore qualsiasi. Il valore predefinito è `False`.
`penalty_function`	`L1` `L2`	Specifica il tipo di funzione di penalità utilizzata. Il valore predefinito è `L2`.
`lambda`	*number*	Parametro di penalità (regolarizzazione).
`calculate_variable_importance`	*indicatore*	Per i modelli che producono un'appropriata misura dell'importanza, questa opzione visualizza un grafico che indica l'importanza relativa di ciascun predittore nella stima del modello. Notare che per alcuni modelli, il calcolo dell'importanza delle variabili può richiedere più tempo, in particolare quando si utilizzano dataset di grandi dimensioni, e che, come risultato, la funzione è disattivata per impostazione predefinita per alcuni modelli. L'importanza delle variabili non è disponibile per i modelli di elenco delle decisioni.

proprietà neuralnetnode

Importante: in questa versione è disponibile una nuova versione del modello Rete neurale con funzionalità avanzate, descritta nella sezione che segue (*neuralnetwork*). Sebbene sia ancora possibile creare e calcolare il punteggio di un modello con la versione precedente, si consiglia di aggiornare gli script in modo da utilizzare la nuova versione. I dettagli della versione precedente sono riportati a scopo informativo.

Esempio

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Tabella 141. proprietà neuralnetnode

Proprietà neuralnetnode	Valori	Descrizione proprietà
targets	[*campo1 ... campoN*]	Il nodo Rete neurale richiede uno o più campi obiettivo e uno o più campi di input. I campi frequenza e peso vengono ignorati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*indicatore*	
train_pct	*number*	
set_random_seed	*indicatore*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Time	Modalità di arresto.

Tabella 141. proprietà neuralnetnode (Continua)

Proprietà neuralnetnode	Valori	Descrizione proprietà
accuracy	*number*	Precisione di arresto.
cycles	*number*	Cicli di apprendimento.
time	*number*	Tempo di addestramento (minuti).
continue	*indicatore*	
show_feedback	*indicatore*	
binary_encode	*indicatore*	
use_last_model	*indicatore*	
gen_logfile	*indicatore*	
logfile_name	*stringa*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*stringa*	
m_non_pyramids	*indicatore*	
m_persistence	*number*	
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*indicatore*	
r_alpha	*number*	
r_eta	*number*	

Tabella 141. proprietà neuralnetnode (Continua)

Proprietà neuralnetnode	Valori	Descrizione proprietà
optimize	Speed Memory	Utilizzare per specificare se ottimizzare la velocità o la memoria durante la creazione del modello.
calculate_variable_importance	*indicatore*	Nota: la proprietà sensitivity_analysis utilizzata nelle versioni precedenti è obsoleta ed è stata sostituita da questa proprietà. La vecchia proprietà è ancora supportata, ma si consiglia di utilizzare calculate_variable_importance.
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

proprietà neuralnetworknode

Il nodo Rete neurale utilizza un modello semplificato del modo in cui il cervello umano elabora le informazioni. Funziona simulando un elevato numero di semplici unità di elaborazione interconnesse che assomigliano a versioni astratte di neuroni. Le reti neurali sono potenti strumenti di valutazione delle funzioni generali e richiedono una conoscenza statistica o matematica minima per l'addestramento o l'applicazione.

Esempio

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabella 142. proprietà neuralnetworknode

Proprietà neuralnetworknode	Valori	Descrizione proprietà
targets	[*campo1 ... campoN*]	Specifica i campi obiettivo.
inputs	[*campo1 ... campoN*]	I campi predittore utilizzati dal modello.
splits	*[campo1 ... campoN*	Specifica il campo o i campi da usare per la creazione di modelli suddivisi.
use_partition	*indicatore*	Se è definito un campo partizione, questa opzione garantisce che per la creazione del modello verranno utilizzati solo i dati della partizione di addestramento.
continue	*indicatore*	Addestramento continuo modello esistente.
objective	Standard Bagging Boosting psm	psm viene utilizzato per insiemi di dati di grandi dimensioni e richiede una connessione Server.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*indicatore*	
first_layer_units	*number*	

Tabella 142. proprietà neuralnetworknode (Continua)

Proprietà neuralnetworknode	Valori	Descrizione proprietà
second_layer_units	*number*	
use_max_time	*indicatore*	
max_time	*number*	
use_max_cycles	*indicatore*	
max_cycles	*number*	
use_min_accuracy	*indicatore*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*indicatore*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*booleano*	
model_name	*stringa*	
confidence	onProbability onIncrease	
score_category_probabilities	*indicatore*	
max_categories	*number*	
score_propensity	*indicatore*	
use_custom_name	*indicatore*	
custom_name	*stringa*	
tooltip	*stringa*	
keywords	*stringa*	
annotation	*stringa*	

proprietà questnode

Il nodo QUEST offre un metodo di classificazione binario per la creazione di strutture ad albero delle decisioni, progettato per ridurre i tempi di elaborazione necessari per le analisi C&R Tree più complesse, riducendo inoltre la tendenza dei metodi per le strutture ad albero di classificazione a favorire gli input che consentono un numero maggiore di suddivisioni. I campi di input possono essere intervalli numerici (continui), ma il campo obiettivo deve essere categoriale. Tutte le suddivisioni sono binarie.

Esempio

```
node = stream.create("quest", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("max_surrogates", 5)
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("prune_tree", True)
node.setPropertyValue("use_std_err", True)
node.setPropertyValue("std_err_multiplier", 3)
```

Tabella 143. proprietà questnode

Proprietà questnode	**Valori**	**Descrizione proprietà**
target	*campo*	I modelli QUEST richiedono un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
continue_training_existing_model	*indicatore*	
objective	Standard Boosting Bagging psm	psm viene utilizzato per insiemi di dati di grandi dimensioni e richiede una connessione Server.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*indicatore*	
tree_directives	*stringa*	
use_max_depth	Default Custom	
max_depth	*numero intero*	Profondità massima della struttura ad albero, da 0 a 1000. Valore utilizzato solo se use_max_depth = Custom.
prune_tree	*indicatore*	Taglia struttura ad albero per evitare sovradattamento.
use_std_err	*indicatore*	Utilizza differenza massima di rischio (in errori standard).
std_err_multiplier	*number*	Differenza massima.
max_surrogates	*number*	Numero massimo surrogati.
use_percentage	*indicatore*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*indicatore*	
costs	*strutturato*	Proprietà strutturata.

Tabella 143. proprietà questnode (Continua)

Proprietà questnode	Valori	Descrizione proprietà
priors	Data Equal Custom	
custom_priors	*strutturato*	Proprietà strutturata.
adjust_priors	*indicatore*	
trails	*number*	Numero di modelli di componenti per boosting o bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regola di combinazione di default per obiettivi categoriali.
range_ensemble_method	Mean Median	Regola di combinazione di default per target continui.
large_boost	*indicatore*	Applica il boosting a insiemi di dati di grandi dimensioni.
split_alpha	*number*	Livello di significatività per suddivisione.
train_pct	*number*	Insieme di prevenzione del sovradattamento.
set_random_seed	*indicatore*	Opzione Replica risultati.
seed	*number*	
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

Proprietà randomtrees

Il nodo Random Trees è simile al nodo &RT esistente; tuttavia il nodo Random Trees è progettato per elaborare i dati di notevoli dimensioni per creare una singola struttura ad albero e visualizza il modello risultante nel visualizzatore di output che è stato aggiunto a SPSS Modeler versione 17. Il nodo Random Trees genera una struttura ad albero delle decisioni che viene utilizzata per la previsione o la classificazione delle osservazioni future. Il metodo utilizza l'esecuzione ricorsiva di partizioni per suddividere i record di addestramento in segmenti riducendo l'impurità ad ogni passaggio. Un nodo della struttura ad albero è considerato *puro* quando il 100% dei casi nel nodo fa parte di una categoria specifica del campo obiettivo. I campi obiettivo e di input possono essere intervalli numerici o categoriali (nominali, ordinali o flag); tutte le suddivisioni sono binarie (solo due sottogruppi).

Tabella 144. Proprietà randomtrees

Proprietà randomtrees	Valori	Descrizione proprietà
target	*campo*	Nel nodo Random Trees, i modelli richiedono un singolo obiettivo ed uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.

Tabella 144. Proprietà randomtrees (Continua)

Proprietà randomtrees	Valori	Descrizione proprietà
number_of_models	*numero intero*	Determina il numero di modelli da creare come parte della modellazione dell'insieme.
use_number_of_predictors	*indicatore*	Determina se viene utilizzato number_of_predictors.
number_of_predictors	*numero intero*	Specifica il numero di predittori da utilizzare quando si creano modelli di suddivisione.
use_stop_rule_for_accuracy	*indicatore*	Determina se la creazione del modello si arresta quando non è possibile migliorare la precisione.
sample_size	*number*	Ridurre questo valore per migliorare le prestazioni durante l'elaborazione di dataset di grandi dimensioni.
handle_imbalanced_data	*indicatore*	Se l'obiettivo del modello è un particolare risultato flag ed il rapporto tra il risultato desiderato ed un risultato non desiderato è molto piccolo, i dati sono sbilanciati ed il campionamento bootstrap eseguito dal modello può avere effetti sulla precisione del modello. Abilitare la gestione dei dati sbilanciati in modo che il modello catturerà una proporzione maggiore del risultato desiderato e potrà generare un modello più forte.
use_weighted_sampling	*indicatore*	Quando è impostata su *False*, le variabili per ciascun nodo vengono selezionate casualmente con la stessa probabilità. Quando è impostata su *True*, le variabili vengono ponderate e selezionate di conseguenza.
max_node_number	*numero intero*	Il numero massimo di nodi consentiti nelle singole strutture ad albero. Se il numero viene superato alla suddivisione successiva, l'accrescimento della struttura ad albero viene arrestato.
max_depth	*numero intero*	Profondità massima della struttura ad albero prima dell'arresto dell'accrescimento.
min_child_node_size	*numero intero*	Determina il numero minimo di record consentiti in un nodo figlio dopo la suddivisione del nodo padre. Se un nodo figlio contiene un numero di record inferiore a quello specificato in questo punto, il nodo padre non viene suddiviso
use_costs	*indicatore*	
costs	*strutturato*	Proprietà strutturata. Il formato è un elenco di 3 valori: il valore effettivo, il valore previsto ed il costo nel caso di previsione errata. Ad esempio: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])

Tabella 144. Proprietà randomtrees (Continua)

Proprietà randomtrees	Valori	Descrizione proprietà
default_cost_increase	none linear square custom	**Nota:** abilitata solo per obiettivi ordinali. Impostare i valori predefiniti nelle matrice costi.
max_pct_missing	*numero intero*	Se la percentuale di valori mancanti in un input è maggior del valore specificato in questo punto, l'input viene escluso. Minimo 0, massimo 100.
exclude_single_cat_pct	*numero intero*	Se un valore di categoria rappresenta una percentuale di record più alta rispetto a quanto specificato in questo punto, l'intero campo viene escluso dalla creazione del modello. Minimo 1, massimo 99.
max_category_number	*numero intero*	Se il numero di categorie in un campo supera questo valore, il campo viene escluso dalla creazione del modello. Minimo 2.
min_field_variation	*number*	Se il coefficiente di variazione di un campo continuo è più piccolo di questo valore, il campo viene escluso dalla creazione del modello.
num_bins	*numero intero*	Utilizzata solo se i dati sono costituiti da input continui. Impostare il numero di bin di frequenza da utilizzare per gli input; le opzioni sono: 2, 4, 5, 10, 20, 25, 50 o 100.
topN	*numero intero*	Specifica il numero di regole da inserire nel report. Il valore predefinito è 50, con valore minimo 1 e valore massimo 1000.

proprietà regressionnode

La regressione lineare è una tecnica statistica molto comune per riassumere i dati ed eseguire previsioni individuando un'area o una linea retta in grado di ridurre le discrepanze tra i valori di output previsti e quelli osservati.

Nota: il nodo Regressione verrà sostituito dal nodo Lineare nella prossima versione. Da questo momento si consiglia di utilizzare i modelli lineari per la regressione lineare.

Esempio

```
node = stream.create("regression", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
node.setPropertyValue("weight_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
```

```
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

Tabella 145. proprietà regressionnode

Proprietà **regressionnode**	Valori	Descrizione proprietà
target	*campo*	I modelli di regressione richiedono un solo campo obiettivo e uno o più campi di input. È anche possibile specificare un campo peso. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
method	Enter Stepwise Backwards Forwards	
include_constant	*indicatore*	
use_weight	*indicatore*	
weight_field	*campo*	
mode	Simple Expert	
complete_records	*indicatore*	
tolerance	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10 1.0E-11 1.0E-12	Utilizzare le virgolette per gli argomenti.
stepping_method	useP useF	useP : usa probabilità di F useF: usa valore F

Tabella 145. proprietà regressionnode (Continua)

Proprietà regressionnode	Valori	Descrizione proprietà
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*indicatore*	
confidence_interval	*indicatore*	
covariance_matrix	*indicatore*	
collinearity_diagnostics	*indicatore*	
regression_coefficients	*indicatore*	
exclude_fields	*indicatore*	
durbin_watson	*indicatore*	
model_fit	*indicatore*	
r_squared_change	*indicatore*	
p_correlations	*indicatore*	
descriptives	*indicatore*	
calculate_variable_importance	*indicatore*	

proprietà sequencenode

Il nodo Sequenza consente di scoprire le regole di associazione nei dati sequenziali o basati su valori temporali. Per sequenza si intende un elenco di serie di elementi che tendono a ricorrere secondo un ordine prevedibile. Ad esempio, un cliente che acquista un rasoio e la lozione dopobarba potrebbe in seguito acquistare la schiuma da barba. Il nodo Sequenza si basa sull'algoritmo delle regole di associazione CARMA, che utilizza un metodo efficiente in due passaggi per trovare le sequenze.

Esempio

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
node.setPropertyValue("use_pruning", True)
```

```
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

Tabella 146. proprietà sequencenode

Proprietà sequencenode	Valori	Descrizione proprietà
id_field	*campo*	Per creare un modello Sequenza, è necessario specificare un campo ID, un campo ora facoltativo e uno o più campi contenuto. I campi peso e frequenza non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
time_field	*campo*	
use_time_field	*indicatore*	
content_fields	[*campo1 ... campon*]	
contiguous	*indicatore*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*indicatore*	
max_duration	*number*	
use_gaps	*indicatore*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*indicatore*	
pruning_value	*number*	
set_mem_sequences	*indicatore*	
mem_sequences	*numero intero*	

proprietà slrmnode

Il nodo Modello risposta autoapprendimento consente di creare un modello in cui è possibile utilizzare un unico nuovo caso oppure un numero limitato di nuovi casi per eseguire una nuova stima del modello senza doverlo riaddestrare con tutti i dati.

Esempio

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tabella 147. proprietà slrmnode

Proprietà slrmnode	Valori	Descrizione proprietà
target	*campo*	Il campo obiettivo deve essere nominale o flag. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
target_response	*campo*	Il tipo deve essere Flag.
continue_training_existing_model	*indicatore*	
target_field_values	*indicatore*	Utilizza tutto: utilizza tutti i valori dalla sorgente. Specifica: è necessario selezionare valori.
target_field_values_specify	*[campo1 ... campoN]*	
include_model_assessment	*indicatore*	
model_assessment_random_seed	*number*	Deve essere un numero reale.
model_assessment_sample_size	*number*	Deve essere un numero reale.
model_assessment_iterations	*number*	Numero di iterazioni.
display_model_evaluation	*indicatore*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Crescente Decrescente	Specifica se verranno visualizzate per prime le offerte con i punteggi più alti o più bassi.
model_reliability	*indicatore*	
calculate_variable_importance	*indicatore*	

proprietà statisticsmodelnode

Il nodo Modello Statistics consente di analizzare e operare con i dati eseguendo le procedure IBM SPSS Statistics che generano PMML. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Le proprietà di questo nodo sono descritte in "proprietà statisticsmodelnode" a pagina 348.

Proprietà stpnode

Il nodo di previsione spazio temporale (STP, Spatio-Temporal Prediction) utilizza dati contenenti informazioni sull'ubicazione, campi di input per la previsione (predittori), e un campo di destinazione. Ciascuna ubicazione ha numerose righe nei dati che rappresentano i valori di ciascun predittore in ogni momento di misurazione. Una volta analizzati i dati, essi possono essere utilizzati per prevedere i valori di destinazione in qualsiasi ubicazione all'interno dei dati di forma utilizzati nell'analisi.

Tabella 148. Proprietà stpnode

Proprietà stpnode	Tipo di dati	Descrizione proprietà
Scheda **Campi**		
target	*campo*	Questo è il campo obiettivo.
location	*campo*	Il campo ubicazione del modello. Sono consentiti solo campi geospaziali.
location_label	*campo*	Il campo categoriale da utilizzare nell'output per apporre un'etichetta alle ubicazioni scelte in location
time_field	*campo*	Il campo ora per il modello. Sono consentiti solo campi con una misurazione continua, mentre il tipo di archiviazione deve essere time, date, timestamp o integer.
inputs	*[campo1 ... campoN]*	Un elenco di campi di input.
Scheda **Intervalli di tempo**		
interval_type_timestamp	Years Trimestri Mesi Weeks Days Hours Minutes Seconds	
interval_type_date	Years Trimestri Mesi Weeks Days	
interval_type_time	Hours Minutes Seconds	Riduce il numero di giorni a settimana presi in considerazione durante la creazione dell'indice ora utilizzato da STP per il calcolo
interval_type_integer	Periods (solo campi indice Time, archiviazione Integer)	L'intervallo in cui il dataset verrà convertito. La selezione disponibile è dipendente dal tipo di archiviazione del campo scelto come time_field per il modello.
period_start	*numero intero*	
start_month	Gennaio Febbraio Marzo Aprile Maggio Giugno Luglio Agosto Settembre Ottobre Novembre Dicembre	Il mese da cui il modello inizierà ad eseguire l'indicizzazione (ad esempio, se impostato su Marzo ma il primo record del dataset è Gennaio, il modello ignora i primi due record ed inizia l'indicizzazione da Marzo).

Tabella 148. Proprietà stpnode (Continua)

Proprietà stpnode	Tipo di dati	Descrizione proprietà
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	Il punto iniziale dell'indice ora creato da STP dai dati
days_per_week	*numero intero*	Minimo 1, massimo 7, con incrementi di 1
hours_per_day	*numero intero*	Il numero di ore di cui il modello tiene conto in un giorno. Se questo valore è impostato su 10, il modello inizierà l'indicizzazione all'ora indicata da day_begins_at e continuerà l'indicizzazione per 10 ore, quindi passerà al valore successivo che corrisponde al valore day_begins_at e così via.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Imposta il valore dell'ora a partire dal quale il modello inizia l'indicizzazione.
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Questa impostazione di incremento è relativa a minuti o secondi. Determina il punto in cui il modello crea gli indici dai dati. Pertanto, con un incremento uguale a 30 e tipo di intervallo secondi, il modello creerà un indice dai dati ogni 30 secondi.
data_matches_interval	*Booleano*	Se impostato su N, la conversione dei dati nel tipo interval_type regolare si verifica prima che il modello venga creato. Se i propri dati sono già nel formato corretto, e interval_type e tutte le impostazioni associate corrispondono ai propri dati, impostare questo valore su Y per impedire la conversione o l'aggregazione dei propri dati. L'impostazione di questo valore su Y disabilita tutti i controlli Aggregazione.

Tabella 148. Proprietà stpnode (Continua)

Proprietà stpnode	Tipo di dati	Descrizione proprietà
agg_range_default	Sum Mean Min Max Median 1stQuartile 3rdQuartile	Determina il metodo di aggregazione predefinito utilizzato per i campi continui. Tutti i campi continui non inclusi in modo specifico nell'aggregazione personalizzata verranno aggregati utilizzando il metodo specificato in questo punto.
custom_agg	[[field, aggregation method],[]..] Demo: [['x5' 'FirstQuartile']['x4' 'Sum']]	Proprietà strutturata: Parametro script: custom_agg Ad esempio: set :stpnode.custom_agg = [[field1 function] [field2 function]] Dove function è la funzione di aggregazione da utilizzare con tale campo.
Scheda **Di base**		
include_intercept	*indicatore*	
max_autoregressive_lag	*numero intero*	Valore minimo 1, valore massimo 5, con incrementi di 1. Questo è il numero di record precedenti richiesti per una previsione. Quindi, ad esempio, se impostato su 5, per creare una nuova previsione vengono utilizzati i 5 record precedenti. Il numero di record specificati in questo punto dai dati di creazione viene incorporato nel modello e, pertanto, l'utente non deve fornire nuovamente i dati durante il calcolo del punteggio del modello.
estimation_method	Parametric Nonparametric	Il metodo per la modellazione della matrice di conversione spaziale
parametric_model	Gaussian Exponential PoweredExponential	Parametro ordine per il modello di covarianza spaziale Parametric
exponential_power	*number*	Livello di potenza per il modello PoweredExponential. Valore minimo 1, valore massimo 2.
Scheda **Avanzate**		
max_missing_values	*numero intero*	La percentuale massima di record con valori mancanti consentita nel modello.

Tabella 148. Proprietà stpnode (Continua)

Proprietà stpnode	Tipo di dati	Descrizione proprietà
significance	*number*	Il livello di significatività per il test delle ipotesi nella creazione del modello. Specifica il valore di significatività per tutti i test nella valutazione del modello STP, inclusi due test Bontà di adattamento, test Effect F e Coefficient t.
Scheda **Output**		
model_specifications	*indicatore*	
temporal_summary	*indicatore*	
location_summary	*indicatore*	Determina se la tabella Riepilogo ubicazione è inclusa nell'output del modello.
model_quality	*indicatore*	
test_mean_structure	*indicatore*	
mean_structure_coefficients	*indicatore*	
autoregressive_coefficients	*indicatore*	
test_decay_space	*indicatore*	
parametric_spatial_covariance	*indicatore*	
correlations_heat_map	*indicatore*	
correlations_map	*indicatore*	
location_clusters	*indicatore*	
similarity_threshold	*number*	La soglia alla quale i cluster di output vengono considerati abbastanza simili per essere uniti in un singolo cluster.
max_number_clusters	*numero intero*	Il limite superiore per il numero di cluster che è possibile includere nell'output del modello.
Scheda **Opzioni modello**		
use_model_name	*indicatore*	
model_name	*stringa*	
uncertainty_factor	*number*	Valore minimo 0, valore massimo 100. Determina l'incremento dell'incertezza (errore) applicata alle previsioni nel futuro. Rappresenta il limite superiore ed inferiore per le previsioni.

proprietà svmnode

Il nodo SVM (Support Vector Machine) consente di classificare i dati in uno di due gruppi senza sovradattamento. Il nodo SVM è particolarmente indicato per l'utilizzo con insiemi di dati di grandi dimensioni, cioè quelli con un elevato numero di campi di input.

Esempio

```
node = stream.create("svm", "My node")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Tabella 149. proprietà svmnode.

Proprietà svmnode	Valori	Descrizione proprietà
all_probabilities	*indicatore*	
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (default) 1.0E-4 1.0E-5 1.0E-6	Determina quando interrompere l'algoritmo di ottimizzazione.
regularization	*numero*	Nota anche come parametro C.
precision	*numero*	Utilizzata solo se il livello di misurazione del campo obiettivo è Continuous.
kernel	RBF(default) Polynomial Sigmoid Linear	Tipo di funzione Kernel utilizzata per la trasformazione.
rbf_gamma	*numero*	Utilizzata solo se kernel è RBF.
gamma	*numero*	Utilizzata solo se kernel è Polynomial o Sigmoid.
bias	*numero*	
degree	*numero*	Utilizzata solo se kernel è Polynomial.
calculate_variable_importance	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_ propensities	*indicatore*	
adjusted_propensity_partition	Test Validation	

Proprietà tcmnode

La modellazione causale temporale tenta di rilevare le relazioni causali principali nei dati di serie temporali. Nella modellazione causale temporale, vengono specificati un insieme di serie di destinazione e un insieme di immissioni candidati per tali destinazioni. La procedura quindi crea un modello di serie temporale autoregressivo per ciascuna destinazione ed include solo gli input che hanno una relazione causale più significativa con la destinazione.

Tabella 150. Proprietà tcmnode

Proprietà tcmnode	Valori	Descrizione proprietà
custom_fields	*Booleano*	
dimensionlist	[*dimension1 ... dimensionN*]	

Tabella 150. Proprietà tcmnode (Continua)

Proprietà tcmnode	Valori	Descrizione proprietà
data_struct	Multiple Single	
metric_fields	*campi*	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Timestamp Period	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*stringa*	
period_start_value	*numero intero*	
num_days_per_week	*numero intero*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*numero intero*	
start_hour_of_day	*numero intero*	
timestamp_increments	*numero intero*	
cyclic_increments	*numero intero*	
cyclic_periods	*elenco*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	Stesso Notsame	
cross_hour	*Booleano*	

Tabella 150. Proprietà tcmnode (Continua)

Proprietà tcmnode	Valori	Descrizione proprietà
aggregate_and_distribute	*elenco*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend None	
k_mean_param	*numero intero*	
k_median_param	*numero intero*	
missing_value_threshold	*numero intero*	
conf_level	*numero intero*	
max_num_predictor	*numero intero*	
max_lag	*numero intero*	
epsilon	*number*	
threshold	*numero intero*	
is_re_est	*Booleano*	
num_targets	*numero intero*	
percent_targets	*numero intero*	
fields_display	*elenco*	
series_display	*elenco*	
network_graph_for_target	*Booleano*	
sign_level_for_target	*number*	
fit_and_outlier_for_target	*Booleano*	
sum_and_para_for_target	*Booleano*	
impact_diag_for_target	*Booleano*	
impact_diag_type_for_target	Effect Cause Both	
impact_diag_level_for_target	*numero intero*	
series_plot_for_target	*Booleano*	
res_plot_for_target	*Booleano*	
top_input_for_target	*Booleano*	
forecast_table_for_target	*Booleano*	

Tabella 150. Proprietà tcmnode (Continua)

Proprietà tcmnode	Valori	Descrizione proprietà
same_as_for_target	*Booleano*	
network_graph_for_series	*Booleano*	
sign_level_for_series	*number*	
fit_and_outlier_for_series	*Booleano*	
sum_and_para_for_series	*Booleano*	
impact_diagram_for_series	*Booleano*	
impact_diagram_type_for_series	Effect Cause Both	
impact_diagram_level_for_series	*numero intero*	
series_plot_for_series	*Booleano*	
residual_plot_for_series	*Booleano*	
forecast_table_for_series	*Booleano*	
outlier_root_cause_analysis	*Booleano*	
causal_levels	*numero intero*	
outlier_table	Interactive Pivot Both	
rmsp_error	*Booleano*	
bic	*Booleano*	
r_square	*Booleano*	
outliers_over_time	*Booleano*	
series_transormation	*Booleano*	
use_estimation_period	*Booleano*	
estimation_period	Times Observation	
observations	*elenco*	
observations_type	Latest Earliest	
observations_num	*numero intero*	
observations_exclude	*numero intero*	
extend_records_into_future	*Booleano*	
forecastperiods	*numero intero*	
max_num_distinct_values	*numero intero*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*Booleano*	
aic	*Booleano*	
rmse	*Booleano*	

Proprietà ts

Il nodo Serie temporali stima i modelli di livellamento esponenziale, i modelli ARIMA (Autoregressive Integrated Moving Average, autoregressivi integrati a media mobile) univariati e ARIMA (o a funzione di trasferimento) multivariati per i dati di serie temporali e genera previsioni di prestazioni future. Questo nodo serie temporale è simile al nodo serie temporale precedente che viene reso obsoleto nella versione 18 SPSS Modeler. Tuttavia, questo nuovo nodo Serie temporali è progettato in modo da sfruttare la potenza di IBM SPSS Analytic Server per l'elaborazione di dati di grandi dimensioni, e visualizzare il modello risultante nel visualizzatore di output aggiunto in SPSS Modeler versione 17.

Tabella 151. Proprietà ts

Proprietà ts	Valori	Descrizione proprietà
targets	*campo*	Il nodo Serie temporali prevede uno o più obiettivi, utilizzando in via facoltativa uno o più campi di input come predittori. I campi frequenza e peso non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
candidate_inputs	[*campo1 ... campoN*]	I campi di input o predittore utilizzati dal modello.
use_period	*indicatore*	
date_time_field	*campo*	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*campo*	
period_start_value	*numero intero*	
num_days_per_week	*numero intero*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*numero intero*	
start_hour_of_day	*numero intero*	

Tabella 151. Proprietà ts (Continua)

Proprietà ts	Valori	Descrizione proprietà
timestamp_increments	*numero intero*	
cyclic_increments	*numero intero*	
cyclic_periods	*elenco*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*indicatore*	
cross_hour	*indicatore*	
aggregate_and_distribute	*elenco*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*numero intero*	
use_estimation_period	*indicatore*	
estimation_period	Osservazioni Times	
date_estimation	*elenco*	Disponibile solo se si utilizza date_time_field
period_estimation	*elenco*	Disponibile solo se si utilizza use_period
observations_type	Latest Earliest	
observations_num	*numero intero*	
observations_exclude	*numero intero*	
method	ExpertModeler Exsmooth Arima	

Tabella 151. Proprietà ts (Continua)

Proprietà ts	Valori	Descrizione proprietà
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*indicatore*	
detect_outliers	*indicatore*	
expert_outlier_additive	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_innovational	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_transient	*indicatore*	
expert_outlier_seasonal_additive	*indicatore*	
expert_outlier_local_trend	*indicatore*	
expert_outlier_additive_patch	*indicatore*	
consider_newesmodels	*indicatore*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	Specifica il metodo di livellamento esponenziale. L'impostazione perdefinita è Simple.

futureValue_type_method	Compute
	specify
		Se viene utilizzato Compute il sistema calcola i valori futuri per il periodo di previsione per ogni predittore.
		
		Per ogni predittore, è possibile scegliere da un elenco di funzioni (vuoto, media dei punti recenti, il valore più recente) o utilizzare specify per immettere i valori manualmente. Per specificare singoli campi e proprietà, utilizzare la proprietà extend_metric_values. Ad esempio:
		
		`set :ts.futureValue_type_method="specify"`
		`set :ts.extend_metric_values=[{'Market_1'`
		`{'Market_2','MOST_RECENT_VALUE', ''},{'Ma`

exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*numero intero*	

Tabella 151. Proprietà ts (Continua)

Proprietà ts	Valori	Descrizione proprietà
arima.d	*numero intero*	
arima.q	*numero intero*	
arima.sp	*numero intero*	
arima.sd	*numero intero*	
arima.sq	*numero intero*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*indicatore*	
tf_arima.p. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.d. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.q. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.sp. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.sd. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.sq. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.delay. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima.transformation_type. *nomecampo*	None SquareRoot NaturalLog	Per le funzioni di trasferimento.
arima_detect_outliers	*indicatore*	
arima_outlier_additive	*indicatore*	
arima_outlier_level_shift	*indicatore*	
arima_outlier_innovational	*indicatore*	
arima_outlier_transient	*indicatore*	
arima_outlier_seasonal_additive	*indicatore*	
arima_outlier_local_trend	*indicatore*	
arima_outlier_additive_patch	*indicatore*	
max_lags	*numero intero*	
cal_PI	*indicatore*	
conf_limit_pct	*reale*	
events	*campi*	
continue	*indicatore*	
scoring_model_only	*indicatore*	Utilizzato per i modelli con grandi quantità (decine di migliaia) di serie temporali.
forecastperiods	*numero intero*	
extend_records_into_future	*indicatore*	

Tabella 151. Proprietà ts (Continua)

Proprietà ts	Valori	Descrizione proprietà
extend_metric_values	*campi*	Consente di fornire valori futuri per i predittori.
conf_limits	*indicatore*	
noise_res	*indicatore*	
max_models_output	*numero intero*	Controlla il numero di modelli visualizzati nell'output. Il valore predefinito è 10. I modelli non vengono visualizzati nell'output se il numero totale di modelli creati supera questo valore. I modelli sono ancora disponibili per il calcolo del punteggio.

Proprietà timeseriesnode (obsoleto)

Nota: Questo nodo Serie temporali originale è stato dichiarato obsoleto nella versione 18 di SPSS Modeler e sostituito dal nuovo nodo Serie temporali progettato per sfruttare la potenza di IBM SPSS Analytic Server ed elaborare dati di grandi dimensioni.Il nodo Serie temporali stima i modelli di livellamento esponenziale, i modelli ARIMA (Autoregressive Integrated Moving Average, autoregressivi integrati a media mobile) univariati e ARIMA (o a funzione di trasferimento) multivariati per i dati di serie temporali e genera previsioni di prestazioni future. Il nodo Serie temporali deve sempre essere preceduto da un nodo Intervalli di tempo.

Esempio

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tabella 152. proprietà timeseriesnode

Proprietà timeseriesnode	Valori	Descrizione proprietà
targets	*campo*	Il nodo Serie temporali prevede uno o più obiettivi, utilizzando in via facoltativa uno o più campi di input come predittori. I campi frequenza e peso non sono utilizzati. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
continue	*indicatore*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*indicatore*	

Tabella 152. proprietà timeseriesnode (Continua)

Proprietà **timeseriesnode**	Valori	Descrizione proprietà
consider_seasonal	*indicatore*	
detect_outliers	*indicatore*	
expert_outlier_additive	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_innovational	*indicatore*	
expert_outlier_level_shift	*indicatore*	
expert_outlier_transient	*indicatore*	
expert_outlier_seasonal_additive	*indicatore*	
expert_outlier_local_trend	*indicatore*	
expert_outlier_additive_patch	*indicatore*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*numero intero*	
arima_d	*numero intero*	
arima_q	*numero intero*	
arima_sp	*numero intero*	
arima_sd	*numero intero*	
arima_sq	*numero intero*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*indicatore*	
tf_arima_p. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_d. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_q. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_sp. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_sd. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_sq. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.
tf_arima_delay. *nomecampo*	*numero intero*	Per le funzioni di trasferimento.

Tabella 152. proprietà timeseriesnode (Continua)

Proprietà timeseriesnode	Valori	Descrizione proprietà
tf_arima_transformation_type. *nomecampo*	None SquareRoot NaturalLog	Per le funzioni di trasferimento.
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*indicatore*	
arima_outlier_level_shift	*indicatore*	
arima_outlier_innovational	*indicatore*	
arima_outlier_transient	*indicatore*	
arima_outlier_seasonal_additive	*indicatore*	
arima_outlier_local_trend	*indicatore*	
arima_outlier_additive_patch	*indicatore*	
conf_limit_pct	*reale*	
max_lags	*numero intero*	
events	*campi*	
scoring_model_only	*indicatore*	Utilizzato per i modelli con grandi quantità (decine di migliaia) di serie temporali.

Proprietà treeas

Il nodo Tree-AS è simile al nodo CHAID esistente; tuttavia, il nodo Tree-AS è progettato per elaborare dati di quantità elevata per creare una singola struttura ad albero e visualizza il modello risultante nel Visualizzatore output aggiunto in SPSS Modeler versione 17. Il nodo genera una struttura ad albero delle decisioni utilizzando statistiche chi-quadrato (CHAID) per identificare le suddivisioni ottimali. Tale utilizzo di CHAID può generare strutture ad albero non binarie; ciò significa che alcune suddivisioni dispongono di più di due rami. I campi obiettivo e di input possono essere intervallo numerico (continui) o categoriali. Un CHAID completo è una modificazione di CHAID che esegue operazioni avanzate per l'analisi di tutte le suddivisioni possibili, ma richiede tempi di elaborazione maggiori.

Tabella 153. Proprietà treeas

Proprietà treeas	Valori	Descrizione proprietà
target	*campo*	Nel nodo Tree-AS, i modelli CHAID richiedono un solo campo obiettivo e uno o più campi di input. È inoltre possibile specificare un campo frequenza. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
method	chaid exhaustive_chaid	
max_depth	*numero intero*	Profondità massima della struttura ad albero, da 0 a 20. Il valore predefinito è 5.

Tabella 153. Proprietà treeas (Continua)

Proprietà treeas	Valori	Descrizione proprietà
num_bins	*numero intero*	Utilizzata solo se i dati sono costituiti da input continui. Impostare il numero di bin di frequenza da utilizzare per gli input; le opzioni sono: 2, 4, 5, 10, 20, 25, 50 o 100.
record_threshold	*numero intero*	Il numero di record a cui il modello passerà dall'utilizzo dei valori P all'utilizzo delle dimensioni degli effetti durante la creazione della struttura ad albero. Il valore predefinito è 1.000.000; aumentare o diminuire questo valore con incrementi di 10.000.
split_alpha	*number*	Livello di significatività per suddivisione. Il valore deve essere compreso tra 0,01 e 0,99.
merge_alpha	*number*	Livello di significatività per unione. Il valore deve essere compreso tra 0,01 e 0,99.
bonferroni_adjustment	*indicatore*	Adegua valori di significatività tramite il metodo di Bonferroni.
effect_size_threshold_cont	*number*	Impostare la soglia della dimensione degli effetti durante la suddivisione dei nodi e l'unione delle categorie quando si utilizza un obiettivo continuo. Il valore deve essere compreso tra 0,01 e 0,99.
effect_size_threshold_cat	*number*	Impostare la soglia della dimensione degli effetti durante la suddivisione dei nodi e l'unione delle categorie quando si utilizza un obiettivo categoriale. Il valore deve essere compreso tra 0,01 e 0,99.
split_merged_categories	*indicatore*	Consenti risuddivisione di categorie unite.
grouping_sig_level	*number*	Utilizzato per determinare il modo in cui i gruppi di nodi sono costituiti o come vengono identificati i nodi inusuali.
chi_square	pearson likelihood_ratio	Metodo utilizzato per calcolare la statistica chi-quadrato: Pearson o Rapporto di verosimiglianza
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*number*	Il valore predefinito è 2. Valore minimo 1, valore massimo 100, con incrementi di 1. Il valore del ramo principale deve essere superiore al valore del ramo secondario.
min_child_records_pc	*number*	Il valore predefinito è 1. Valore minimo 1, valore massimo 100, con incrementi di 1.
min_parent_records_abs	*number*	Il valore predefinito è 100. Valore minimo 1, valore massimo 100, con incrementi di 1. Il valore del ramo principale deve essere superiore al valore del ramo secondario.
min_child_records_abs	*number*	Il valore predefinito è 50. Valore minimo 1, massimo 100, con incrementi di 1.
epsilon	*number*	Modifica minima nelle frequenze di cella previste.

Tabella 153. Proprietà treeas (Continua)

Proprietà treeas	Valori	Descrizione proprietà
max_iterations	*number*	Numero massimo di iterazioni per la convergenza.
use_costs	*indicatore*	
costs	*strutturato*	Proprietà strutturata. Il formato è un elenco di 3 valori: il valore effettivo, il valore previsto ed il costo nel caso di previsione errata. Ad esempio: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square custom	**Nota:** abilitata solo per obiettivi ordinali. Impostare i valori predefiniti nelle matrice costi.
calculate_conf	*indicatore*	
display_rule_id	*indicatore*	Aggiunge un campo all'output del calcolo del punteggio che indica l'ID del nodo terminale al quale è assegnato ogni record.

Proprietà twostepnode

Il nodo TwoStep è un metodo di raggruppamento tramite cluster in due fasi. La prima fase esegue un singolo passaggio nei dati per comprimere i dati di input non elaborati in un insieme gestibile di cluster secondari. Nella seconda fase viene utilizzato un metodo di raggruppamento tramite cluster gerarchico per unire progressivamente i cluster secondari in cluster sempre più grandi. Il nodo TwoStep offre il vantaggio di stimare automaticamente il numero ottimale di cluster per i dati di addestramento. Può gestire in modo efficiente tipi di campo misti e insiemi di dati di grandi dimensioni.

Esempio

```
node = stream.create("twostep", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Age", "K", "Na", "BP"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "TwoStep_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("exclude_outliers", True)
node.setPropertyValue("cluster_label", "String")
node.setPropertyValue("label_prefix", "TwoStep_")
node.setPropertyValue("cluster_num_auto", False)
node.setPropertyValue("max_num_clusters", 9)
node.setPropertyValue("min_num_clusters", 3)
node.setPropertyValue("num_clusters", 7)
```

Tabella 154. proprietà twostepnode

Proprietà twostepnode	Valori	Descrizione proprietà
inputs	[*campo1 ... campoN*]	I modelli TwoStep utilizzano un elenco di campi di input, ma nessun campo obiettivo. I campi peso e frequenza non vengono riconosciuti. Per ulteriori informazioni, consultare l'argomento "Proprietà comuni nodi modellazione" a pagina 185.
standardize	*indicatore*	
exclude_outliers	*indicatore*	
percentage	*number*	
cluster_num_auto	*indicatore*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String Number	
label_prefix	*stringa*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Proprietà twostepAS

TwoStep Cluster è uno strumento esplorativo progettato per rivelare i raggruppamenti naturali (o cluster) all'interno di un dataset, che altrimenti non sarebbero evidenti. L'algoritmo impiegato da questa procedura ha diverse funzioni interessanti che lo differenziano dalle tecniche tradizionali di clustering, quali la gestione delle variabili continue e categoriali, la selezione automatica del numero di cluster e la scalabilità.

Tabella 155. Proprietà twostepAS

Proprietà twostepAS	Valori	Descrizione proprietà
inputs	[*f1 ... fN*]	I modelli TwoStepAS utilizzano un elenco di campi di input, ma nessun campo obiettivo. I campi peso e frequenza non vengono riconosciuti.
use_predefined_roles	Booleana	Valore predefinito=True
use_custom_field_assignments	Booleana	Valore predefinito=False
cluster_num_auto	Booleana	Valore predefinito=True
min_num_clusters	integer	Valore predefinito=2
max_num_clusters	integer	Valore predefinito=15
num_clusters	integer	Valore predefinito=5
clustering_criterion	AIC BIC	

Tabella 155. Proprietà twostepAS (Continua)

Proprietà **twostepAS**	Valori	Descrizione proprietà
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Minimo Massimo	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	Booleana	
random_seed	integer	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	Booleana	Valore predefinito=True
num_cases_in_feature_tree_leaf_is_less_than	integer	Valore predefinito=10
top_perc_outliers	integer	Valore predefinito=5
initial_dist_change_threshold	integer	Valore predefinito=0
leaf_node_maximum_branches	integer	Valore predefinito=8
non_leaf_node_maximum_branches	integer	Valore predefinito=8
max_tree_depth	integer	Valore predefinito=3
adjustment_weight_on_measurement_level	integer	Valore predefinito=6
memory_allocation_mb	number	Valore predefinito=512
delayed_split	Booleana	Valore predefinito=True
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	Booleana	Valore predefinito=True
featureMisPercent	integer	Valore predefinito=70
coefRange	number	Valore predefinito=0.05
percCasesSingleCategory	integer	Valore predefinito=95
numCases	integer	Valore predefinito=24
include_model_specifications	Booleana	Valore predefinito=True
include_record_summary	Booleana	Valore predefinito=True
include_field_transformations	Booleana	Valore predefinito=True
excluded_inputs	Booleana	Valore predefinito=True
evaluate_model_quality	Booleana	Valore predefinito=True
show_feature_importance bar chart	Booleana	Valore predefinito=True
show_feature_importance_ word_cloud	Booleana	Valore predefinito=True
show_outlier_clusters interactive_table_and_chart	Booleana	Valore predefinito=True
show_outlier_clusters_pivot_table	Booleana	Valore predefinito=True
across_cluster_feature_importance	Booleana	Valore predefinito=True
across_cluster_profiles_pivot_table	Booleana	Valore predefinito=True
withinprofiles	Booleana	Valore predefinito=True
cluster_distances	Booleana	Valore predefinito=True

Tabella 155. Proprietà twostepAS (Continua)

Proprietà **twostepAS**	Valori	Descrizione proprietà
cluster_label	String Number	
label_prefix	Stringa	

Capitolo 14. Proprietà del nodo del nugget del modello

I nodi dei nugget del modello condividono le stesse proprietà comuni agli altri nodi. Per ulteriori informazioni, consultare l'argomento “Proprietà comuni dei nodi” a pagina 71.

Proprietà applyanomalydetectionnode

I nodi Modelli Rilevamento anomalie si possono utilizzare per generare un nugget del modello Rilevamento anomalie. Il nome di script di questo nugget del modello è *applyanomalydetectionnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere “proprietà anomalydetectionnode” a pagina 185

Tabella 156. proprietà applyanomalydetectionnode

Proprietà **applyanomalydetectionnode**	Valori	Descrizione proprietà
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Determina quali output sono creati per il calcolo del punteggio.
num_fields	*numero intero*	Campi da inserire nel report.
discard_records	*indicatore*	Indica se i record sono scartati o meno dall'output.
discard_anomalous_records	*indicatore*	Indica se scartare i record anomali o *non* anomali. L'impostazione di default è off, ad indicare che i record *non* anomali vengono scartati. Altrimenti, se l'impostazione è su on, verranno scartati i record anomali. Questa proprietà è attivata solo se è attivata la proprietà discard_records.

Proprietà applyapriorinode

I nodi Modelli Apriori si possono utilizzare per generare un nugget del modello Apriori. Il nome di script di questo nugget del modello è *applyapriorinode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere “proprietà apriorinode” a pagina 187

Tabella 157. proprietà applyapriorinode

Proprietà **applyapriorinode**	Valori	Descrizione proprietà
max_predictions	*numero (intero)*	
ignore_unmatached	*indicatore*	
allow_repeats	*indicatore*	
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Distribuibilità	

Proprietà applyassociationrulesnode

È possibile utilizzare il nodo modellazione Regole di associazione per generare un nugget del modello Regole di associazione. Il nome di script di questo nugget del modello è *applyassociationrulesnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà associationrulesnode" a pagina 188.

Tabella 158. Proprietà applyassociationrulesnode

Proprietà applyassociationrulesnode	Tipo di dati	Descrizione proprietà
max_predictions	*numero intero*	Il numero massimo di regole che possono essere applicate a ciascun input nel punteggio.
criterion	Confidence Rulesupport Lift Conditionsupport Distribuibilità	Selezionare la misura che determina l'efficacia delle regole.
allow_repeats	*Booleano*	Determina se nel punteggio vengono incluse le regole con la stessa previsione.
check_input	NoPredictions Predictions NoCheck	

Proprietà applyautoclassifiernode

I nodi Modelli Classificatore automatico si possono utilizzare per generare un nugget del modello Classificatore automatico. Il nome di script di questo nugget del modello è *applyautoclassifiernode*. Per ulteriori informazioni sugli script per il nodo Modelli, vedere "proprietà autoclassifiernode" a pagina 190

Tabella 159. proprietà applyautoclassifiernode

Proprietà applyautoclassifiernode	Valori	Descrizione proprietà
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Specifica il metodo utilizzato per determinare il punteggio dell'insieme. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo flag.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Se è selezionato un metodo del confronto, specifica le modalità di risoluzione delle situazioni di pari merito. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo flag.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Specifica il metodo utilizzato per determinare il punteggio dell'insieme. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo insieme.
set_voting_tie_selection	Random HighestConfidence	Se è selezionato un metodo del confronto, specifica le modalità di risoluzione delle situazioni di pari merito. Questa impostazione è valida solamente se l'obiettivo selezionato è un campo nominale.

Proprietà applyautoclusternode

I nodi Modelli Cluster automatico si possono utilizzare per generare un nugget del modello Cluster automatico. Il nome di script di questo nugget del modello è *applyautoclusternode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà autoclusternode" a pagina 193

Proprietà applyautonumericnode

I nodi Modelli Numerico automatico si possono utilizzare per generare un nugget del modello Numerico automatico. Il nome di script di questo nugget del modello è *applyautonumericnode*. Per ulteriori informazioni sugli script per il nodo Modelli, vedere "proprietà autonumericnode" a pagina 194

Tabella 160. proprietà applyautonumericnode

Proprietà applyautonumericnode	Valori	Descrizione proprietà
calculate_standard_error	*indicatore*	

Proprietà applybayesnetnode

I nodi Modelli Rete bayesiana si possono utilizzare per generare un nugget del modello Rete bayesiana. Il nome di script di questo nugget del modello è *applybayesnetnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà bayesnetnode" a pagina 196.

Tabella 161. proprietà applybayesnetnode.

Proprietà applybayesnetnode	Valori	Descrizione proprietà
all_probabilities	*indicatore*	
raw_propensity	*indicatore*	
adjusted_propensity	*indicatore*	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applyc50node

I nodi Modelli C5.0 si possono utilizzare per generare un nugget del modello C5.0. Il nome di script di questo nugget del modello è *applyc50node*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà c50node" a pagina 198.

Tabella 162. proprietà applyc50node

Proprietà applyc50node	Valori	Descrizione proprietà
sql_generate	udf Never NoMissingValues	Consente di impostare le opzioni di generazione SQL durante l'esecuzione dell'insieme di regole. Il valore predefinito è udf.
calculate_conf	*indicatore*	Disponibile quando è attivata la generazione SQL, questa proprietà include i calcoli di confidenza nella struttura ad albero generata.
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applycarmanode

I nodi Modelli CARMA si possono utilizzare per generare un nugget del modello CARMA. Il nome di script di questo nugget del modello è *applycarmanode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà carmanode" a pagina 199.

Proprietà applycartnode

I nodi Modelli C&R possono essere utilizzati per generare un nugget del modello C&R Tree. Il nome di script di questo nugget del modello è *applycartnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà cartnode" a pagina 200.

Tabella 163. proprietà applycartnode

Proprietà applycartnode	Valori	Descrizione proprietà
enable_sql_generation	Never MissingValues NoMissingValues	Consente di impostare le opzioni di generazione SQL durante l'esecuzione dell'insieme di regole.
calculate_conf	*indicatore*	Disponibile quando è attivata la generazione SQL, questa proprietà include i calcoli di confidenza nella struttura ad albero generata.
display_rule_id	*indicatore*	Aggiunge un campo all'output del calcolo del punteggio che indica l'ID del nodo terminale al quale è assegnato ogni record.
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applychaidnode

I nodi Modelli CHAID si possono utilizzare per generare un nugget del modello CHAID. Il nome di script di questo nugget del modello è *applychaidnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà chaidnode" a pagina 202.

Tabella 164. proprietà applychaidnode

Proprietà applychaidnode	Valori	Descrizione proprietà
enable_sql_generation	Never MissingValues	Consente di impostare le opzioni di generazione SQL durante l'esecuzione dell'insieme di regole.
calculate_conf	*indicatore*	
display_rule_id	*indicatore*	Aggiunge un campo all'output del calcolo del punteggio che indica l'ID del nodo terminale al quale è assegnato ogni record.
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applycoxregnode

I nodi Modelli Cox si possono utilizzare per generare un nugget del modello Cox. Il nome di script di questo nugget del modello è *applycoxregnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà coxregnode" a pagina 204.

Tabella 165. proprietà applycoxregnode

Proprietà **applycoxregnode**	Valori	Descrizione proprietà
future_time_as	Intervals Campi	
time_interval	*number*	
num_future_times	*numero intero*	
time_field	*campo*	
past_survival_time	*campo*	
all_probabilities	*indicatore*	
cumulative_hazard	*indicatore*	

Proprietà applydecisionlistnode

I nodi Modelli Elenco di decisioni si possono utilizzare per generare un nugget del modello Elenco di decisioni. Il nome di script di questo nugget del modello è *applydecisionlistnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà decisionlistnode" a pagina 206.

Tabella 166. proprietà applydecisionlistnode

Proprietà **applydecisionlistnode**	Valori	Descrizione proprietà
enable_sql_generation	*indicatore*	Se questa proprietà è vera, IBM SPSS Modeler cerca di rinviare il modello Elenco di decisioni a SQL.
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applydiscriminantnode

I nodi Modelli Discriminante si possono utilizzare per generare un nugget del modello Discriminante. Il nome di script di questo nugget del modello è *applydiscriminantnode*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "proprietà discriminantnode" a pagina 207.

Tabella 167. proprietà applydiscriminantnode

Proprietà **applydiscriminantnode**	Valori	Descrizione proprietà
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applyextension

I nodi modello Estensione possono essere utilizzati per generare un nugget del modello Estensione. Il nome di script di questo nugget del modello è *applyextension*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà extensionmodelnode" a pagina 209.

Esempio Python for Spark

```
#### script example for Python for Spark
applyModel = stream.findByType("extension_apply", None)

score_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTreeModel
from pyspark.sql.types import StringType, StructField

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()

    _modelPath = cxt.getModelContentToPath("TreeModel")
    metadata = json.loads(cxt.getModelContentToString("model.dm"))

    schema = df.dtypes[:]
    target = "Drug"
    predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

    lookup = {}
    for i in range(0,len(schema)):
        lookup[schema[i][0]] = i

    def row2LabeledPoint(dm,lookup,target,predictors,row):
        target_index = lookup[target]
        tval = dm[target_index].index(row[target_index])
        pvals = []
        for predictor in predictors:
            predictor_index = lookup[predictor]
            if isinstance(dm[predictor_index],list):
                pval = row[predictor_index] in dm[predictor_index] and dm[predictor_index].index(row[predictor_index]) or -
            else:
                pval = row[predictor_index]
            pvals.append(pval)
        return LabeledPoint(tval, DenseVector(pvals))

    # convert dataframe to an RDD containing LabeledPoint
    lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
    treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
    # score the model, produces an RDD containing just double values
    predictions = treeModel.predict(lps.map(lambda lp: lp.features))

    def addPrediction(x,dm,lookup,target):
        result = []
```

```
        for _idx in range(0, len(x[0])):
            result.append(x[0][_idx])
        result.append(dm[lookup[target]][int(x[1])])
        return result

    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    rdd2 = df.rdd.zip(predictions).map(lambda x:addPrediction(x, metadata, lookup, target))
    outDF = cxt.getSparkSQLContext().createDataFrame(rdd2, _schema)

    cxt.setSparkOutputData(outDF)
"""
applyModel.setPropertyValue("python_syntax", score_script)
```

Esempio R

```
#### script example for R
applyModel.setPropertyValue("r_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabella 168. Proprietà applyextension

Proprietà applyextension	Valori	Descrizione proprietà
r_syntax	*stringa*	Sintassi degli script R per il calcolo del punteggio del modello.
python_syntax	*stringa*	Sintassi degli script Python per il calcolo del punteggio del modello.
use_batch_size	*indicatore*	Attiva l'utilizzo dell'elaborazione batch.
batch_size	*numero intero*	Specifica il numero di record di dati da includere in ciascun batch.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.

Proprietà applyfactornode

I nodi modelli fattoriale/PCA si possono utilizzare per generare un nugget del modello fattoriale/PCA. Il nome di script di questo nugget del modello è *applyfactornode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni, sugli script del nodo Modelli specifico, vedere “proprietà factornode” a pagina 211.

Proprietà applyfeatureselectionnode

I nodi Modelli Selezione funzioni si possono utilizzare per generare un nugget del modello Selezione funzioni. Il nome di script di questo nugget del modello è *applyfeatureselectionnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà featureselectionnode" a pagina 213.

Tabella 169. proprietà applyfeatureselectionnode

Proprietà `applyfeatureselectionnode`	Valori	Descrizione proprietà
`selected_ranked_fields`		Specifica quali campi classificati sono selezionati nel browser dei modelli.
`selected_screened_fields`		Specifica quali campi sottoposti a screening sono selezionati nel browser dei modelli.

Proprietà applygeneralizedlinearnode

I nodi Modelli lineari generalizzati (GenLin) si possono utilizzare per generare un nugget del modello Lineare generalizzato. Il nome di script di questo nugget del modello è *applygeneralizedlinearnode*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "proprietà genlinnode" a pagina 215.

Tabella 170. proprietà applygeneralizedlinearnode

Proprietà `applygeneralizedlinearnode`	Valori	Descrizione proprietà
`calculate_raw_propensities`	*indicatore*	
`calculate_adjusted_propensities`	*indicatore*	

Proprietà applyglmmnode

I nodi Modelli GLMM si possono utilizzare per generare un nugget del modello GLMM. Il nome di script di questo nugget del modello è *applyglmmnode*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "Proprietà glmmnode" a pagina 218.

Tabella 171. proprietà applyglmmnode

Proprietà `applyglmmnode`	Valori	Descrizione proprietà
`confidence`	`onProbability` `onIncrease`	Base per il calcolo del valore di confidenza del punteggio: probabilità prevista più alta o differenza tra le probabilità più alte e la seconda massima prevista.
`score_category_probabilities`	*indicatore*	Se impostato su `True`, produce le probabilità previste per i target di categoria. Viene creato un campo per ogni categoria. Il valore di default è `False`.
`max_categories`	*numero intero*	Numero massimo di categorie per cui prevedere le probabilità. Utilizzata solo se `score_category_probabilities` è `True`.
`score_propensity`	*indicatore*	Se impostato su `True`, produce punteggi di propensione grezza (verosimiglianza del risultato "True") per i modelli con obiettivi flag. Se le partizioni sono attive, producono anche punteggi di propensione regolata basati sulla partizione di test. Il valore di default è `False`.

Tabella 171. proprietà applyglmmnode (Continua)

Proprietà applyglmmnode	Valori	Descrizione proprietà
enable_sql_generation	udf native	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Scegliere di eseguire il pushback verso il database e il calcolo del punteggio utilizzando un adattatore per calcolo punteggio SPSS® Modeler Server (se connessi a un database con un adattatore per calcolo punteggio installato), oppure eseguire il calcolo all'interno di SPSS Modeler. Il valore predefinito è udf.

Proprietà applygle

Il nodo Modelli GLE può essere utilizzato per generare un nugget del modello GLE. Il nome di script di questo nugget del modello è *applygle*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà gle" a pagina 222.

Tabella 172. Proprietà applygle

Proprietà applygle	Valori	Descrizione proprietà
enable_sql_generation	udf native	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Scegliere di eseguire il pushback verso il database e il calcolo del punteggio utilizzando un adattatore per calcolo punteggio SPSS Modeler Server (se connessi a un database con un adattatore per calcolo punteggio installato), oppure eseguire il calcolo all'interno di SPSS Modeler.

Proprietà applygmm

Il nodo Gaussian Mixture può essere utilizzato per generare un nugget del modello Gaussian Mixture. Il nome script di questo nugget del modello è *applygmm*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà gmm" a pagina 351.

Proprietà applykmeansnode

I nodi Modelli Medie K si possono utilizzare per generare un nugget del modello Medie K. Il nome di script di questo nugget del modello è *applykmeansnode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà kmeansnode" a pagina 226.

Proprietà applyknnnode

I nodi Modelli KNN possono essere utilizzati per generare un nugget del modello KNN. Il nome di script di questo nugget del modello è *applyknnnode*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "proprietà knnnode" a pagina 228.

Tabella 173. proprietà applyknnnode

Proprietà applyknnnode	Valori	Descrizione proprietà
all_probabilities	*indicatore*	
save_distances	*indicatore*	

Proprietà applykohonennode

I nodi Modelli Kohonen si possono utilizzare per generare un nugget del modello Kohonen. Il nome di script di questo nugget del modello è *applykohonennode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà c50node" a pagina 198.

Proprietà applylinearnode

I nodi Modelli lineari si possono utilizzare per generare un nugget del modello lineari. Il nome di script di questo nugget del modello è *applylinearnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà linearnode" a pagina 231.

Tabella 174. Proprietà applylinearnode

Proprietà linear	Valori	Descrizione proprietà
use_custom_name	*indicatore*	
custom_name	*stringa*	
enable_sql_generation	udf native puresql	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Le opzioni consentono di eseguire il pushback verso il database ed il calcolo del punteggio utilizzando un adattatore per calcolo del punteggio SPSS® Modeler Server (se collegato ad un database con un adattatore per calcolo del punteggio installato), di calcolare il punteggio in SPSS Modeler o di eseguire il pushback verso il database e calcolare il punteggio utilizzando SQL. Il valore predefinito è udf.

Proprietà applylinearasnode

I nodi Modelli Linear-AS possono essere utilizzati per generare un nugget del modello Linear-AS. Il nome di script di questo nugget del modello è *applylinearasnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà linearasnode" a pagina 232.

Tabella 175. Proprietà applylinearasnode

Proprietà applylinearasnode	Valori	Descrizione proprietà
enable_sql_generation	udf native	Il valore predefinito è udf.

Proprietà applylogregnode

I nodi Modelli Regressione logistica si possono utilizzare per generare un nugget del modello Regressione logistica. Il nome di script di questo nugget del modello è*applylogregnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà logregnode" a pagina 233.

Tabella 176. proprietà applylogregnode

Proprietà applylogregnode	Valori	Descrizione proprietà
calculate_raw_propensities	*indicatore*	
calculate_conf	*indicatore*	
enable_sql_generation	*indicatore*	

Proprietà applylsvmnode

I nodi modello LSVM possono essere utilizzati per generare un nugget del modello LSVM. Il nome di script di questo nugget del modello è *applylsvmnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà lsvmnode" a pagina 238.

Tabella 177. Proprietà applylsvmnode

Proprietà applylsvmnode	Valori	Descrizione proprietà
calculate_raw_propensities	*indicatore*	Specifica se calcolare i punteggi di propensione grezza.
enable_sql_generation	udf native	Specifica se calcolare il punteggio utilizzando l'Adattatore per calcolo del punteggio (se installato) o nel processo oppure se calcolare il punteggio all'esterno del database.

Proprietà applyneuralnetnode

I nodi Modelli Rete neurale si possono utilizzare per generare un nugget del modello Rete neurale. Il nome di script di questo nugget del modello è *applyneuralnetnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà neuralnetnode" a pagina 239.

Attenzione: in questa release è disponibile una nuova versione del nugget Rete neurale con funzionalità avanzate, descritta nella sezione che segue (*applyneuralnetwork*). Sebbene la versione precedente sia ancora disponibile, si consiglia di aggiornare gli script in modo da utilizzare la nuova versione. I dettagli della versione precedente vengono mantenuti in questa sezione per riferimento, ma nelle versioni future non sarà più supportata.

Tabella 178. proprietà applyneuralnetnode

Proprietà applyneuralnetnode	Valori	Descrizione proprietà
calculate_conf	*indicatore*	Disponibile quando è attivata la generazione SQL, questa proprietà include i calcoli di confidenza nella struttura ad albero generata.
enable_sql_generation	*indicatore*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

proprietà applyneuralnetworknode

I nodi Modelli Rete neurale si possono utilizzare per generare un nugget del modello Rete neurale. Il nome di script di questo nugget del modello è *applyneuralnetworknode*. Per ulteriori informazioni sugli script del nodo Modelli, consultare Proprietà neuralnetworknode.

Tabella 179. proprietà applyneuralnetworknode

Proprietà applyneuralnetworknode	Valori	Descrizione proprietà
use_custom_name	*indicatore*	
custom_name	*stringa*	
confidence	onProbability onIncrease	
score_category_probabilities	*indicatore*	
max_categories	*number*	
score_propensity	*indicatore*	
enable_sql_generation	udf native puresql	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Le opzioni consentono di eseguire il pushback verso il database ed il calcolo del punteggio utilizzando un adattatore per calcolo del punteggio SPSS® Modeler Server (se collegato ad un database con un adattatore per calcolo del punteggio installato), di calcolare il punteggio in SPSS Modeler o di eseguire il pushback verso il database e calcolare il punteggio utilizzando SQL. Il valore predefinito è udf.

Proprietà di applyocsvmnode

I nodi SVM a una classe possono essere utilizzati per generare un nugget del modello SVM a una classe. Il nome di script di questo nugget del modello è *applyocsvmnode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà di ocsvmnode" a pagina 356.

Proprietà applyquestnode

I nodi Modelli QUEST si possono utilizzare per generare un nugget del modello QUEST. Il nome di script di questo nugget del modello è *applyquestnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà questnode" a pagina 242.

Tabella 180. proprietà applyquestnode

Proprietà applyquestnode	Valori	Descrizione proprietà
enable_sql_generation	Never MissingValues NoMissingValues	Consente di impostare le opzioni di generazione SQL durante l'esecuzione dell'insieme di regole.
calculate_conf	*indicatore*	
display_rule_id	*indicatore*	Aggiunge un campo all'output del calcolo del punteggio che indica l'ID del nodo terminale al quale è assegnato ogni record.

Tabella 180. proprietà applyquestnode (Continua)

Proprietà applyquestnode	Valori	Descrizione proprietà
calculate_raw_propensities	*indicatore*	
calculate_adjusted_propensities	*indicatore*	

Proprietà applyr

I nodi di creazione R possono essere utilizzati per generare un nugget del modello R. Il nome di script di questo nugget del modello è *applyr*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "Proprietà buildr" a pagina 197.

Tabella 181. proprietà di applyr

Proprietà applyr	Valori	Descrizione proprietà
score_syntax	*stringa*	Sintassi degli script R per il calcolo del punteggio del modello.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
use_batch_size	*indicatore*	Attiva l'utilizzo dell'elaborazione batch
batch_size	*numero intero*	Specifica il numero di record di dati da includere in ciascun batch

Proprietà applyrandomtrees

Il nodo modelli Random Trees può essere utilizzato per generare un nugget del modello Random Trees. Il nome di script di questo nugget del modello è *applyrandomtrees*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà randomtrees" a pagina 244.

Tabella 182. Proprietà applyrandomtrees

Proprietà applyrandomtrees	Valori	Descrizione proprietà
calculate_conf	*indicatore*	Questa proprietà include i calcoli di confidenza nella struttura ad albero generata.
enable_sql_generation	udf native	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Scegliere di eseguire il pushback verso il database e il calcolo del punteggio utilizzando un adattatore per calcolo punteggio SPSS Modeler Server (se connessi a un database con un adattatore per calcolo punteggio installato), oppure eseguire il calcolo all'interno di SPSS Modeler.

Proprietà applyregressionnode

I nodi Modelli Regressione lineare si possono utilizzare per generare un nugget del modello Regressione lineare. Il nome di script di questo nugget del modello è *applyregressionnode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà regressionnode" a pagina 246.

proprietà applyselflearningnode

I nodi Modelli SLRM (Risposta autoapprendimento) si possono utilizzare per generare un nugget del modello SLRM. Il nome di script di questo nugget del modello è *applyselflearningnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà slrmnode" a pagina 249.

Tabella 183. proprietà applyselflearningnode

Proprietà `applyselflearningnode`	Valori	Descrizione proprietà
`max_predictions`	*number*	
`randomization`	*number*	
`scoring_random_seed`	*number*	
`sort`	`ascending` `descending`	Specifica se verranno visualizzate per prime le offerte con i punteggi più alti o più bassi.
`model_reliability`	*indicatore*	Tiene conto dell'opzione di affidabilità del modello inclusa nella scheda Impostazioni.

Proprietà applysequencenode

I nodi Modelli Sequenza si possono utilizzare per generare un nugget del modello Sequenza. Il nome di script di questo nugget del modello è *applysequencenode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà sequencenode" a pagina 248.

Proprietà applysvmnode

I nodi Modelli SVM si possono utilizzare per generare un nugget del modello SVM. Il nome di script di questo nugget del modello è *applysvmnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "proprietà svmnode" a pagina 254.

Tabella 184. proprietà applysvmnode

Proprietà `applysvmnode`	Valori	Descrizione proprietà
`all_probabilities`	*indicatore*	
`calculate_raw_propensities`	*indicatore*	
`calculate_adjusted_propensities`	*indicatore*	

Proprietà applystpnode

È possibile utilizzare il nodo di modellazione STP per generare un nugget del modello associato, che visualizzi l'output del modello nel visualizzatore output. Il nome di script di questo nugget del modello è *applystpnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà stpnode" a pagina 250.

Tabella 185. proprietà applystpnode

Proprietà applystpnode	Tipo di dati	Descrizione proprietà
uncertainty_factor	*Booleano*	Minimo 0, massimo 100.

Proprietà applytcmnode

I nodi di modeling TCM (Temporal Causal Modeling) possono essere utilizzati per generare un nugget del modello TCM. Il nome di script di questo nugget del modello è *applytcmnode*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà tcmnode" a pagina 255.

Tabella 186. Proprietà applytcmnode

Proprietà applytcmnode	Valori	Descrizione proprietà
ext_future	*booleano*	
ext_future_num	*numero intero*	
noise_res	*booleano*	
conf_limits	*booleano*	
target_fields	*elenco*	
target_series	*elenco*	

Proprietà applyts

Il nodo Modelli serie temporali possono essere utilizzati per generare un nugget del modello serie temporali. Il nome di script di questo nugget del modello è *applyts*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà ts" a pagina 259.

Tabella 187. Proprietà applyts

Proprietà applyts	Valori	Descrizione proprietà
extend_records_into_future	*Booleano*	
ext_future_num	*numero intero*	
compute_future_values_input	*Booleano*	
forecastperiods	*numero intero*	
noise_res	*booleano*	
conf_limits	*booleano*	
target_fields	*elenco*	
target_series	*elenco*	
includeTargets	*campo*	

Proprietà applytimeseriesnode (obsoleto)

Il nodo Modelli serie temporali possono essere utilizzati per generare un nugget del modello serie temporali. Il nome di script di questo nugget del modello è *applytimeseriesnode*. Per ulteriori informazioni sugli script del nodo Modelli, vedere "Proprietà timeseriesnode (obsoleto)" a pagina 263.

Tabella 188. proprietà applytimeseriesnode.

Proprietà applytimeseriesnode	Valori	Descrizione proprietà
calculate_conf	*indicatore*	
calculate_residuals	*indicatore*	

Proprietà applytreeas

I nodi di modeling Tree-AS si possono utilizzare per generare un nugget del modello Tree-AS. Il nome di script di questo nugget del modello è *applytreenas*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà treeas" a pagina 265.

Tabella 189. Proprietà applytreeas

Proprietà applytreeas	Valori	Descrizione proprietà
calculate_conf	*indicatore*	Questa proprietà include i calcoli di confidenza nella struttura ad albero generata.
display_rule_id	*indicatore*	Aggiunge un campo all'output del calcolo del punteggio che indica l'ID del nodo terminale al quale è assegnato ogni record.
enable_sql_generation	udf native	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Scegliere di eseguire il pushback verso il database e il calcolo del punteggio utilizzando un adattatore per calcolo punteggio SPSS Modeler Server (se connessi a un database con un adattatore per calcolo punteggio installato), oppure eseguire il calcolo all'interno di SPSS Modeler.

Proprietà applytwostepnode

I nodi Modelli TwoStep si possono utilizzare per generare un nugget del modello TwoStep. Il nome di script di questo nugget del modello è *applytwostepnode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà twostepnode" a pagina 267.

Proprietà applytwostepAS

I nodi modelli TwoStep AS possono essere utilizzati per generare un nugget del modello TwoStep AS. Il nome di script di questo nugget del modello è *applytwostepAS*. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà twostepAS" a pagina 268.

Tabella 190. Proprietà applytwostepAS

Proprietà applytwostepAS	Valori	Descrizione proprietà
enable_sql_generation	udf native	Utilizzato per impostare le opzioni di generazione SQL durante l'esecuzione del flusso. Scegliere di eseguire il pushback verso il database e il calcolo del punteggio utilizzando un adattatore per calcolo punteggio SPSS® Modeler Server (se connessi a un database con un adattatore per calcolo punteggio installato), oppure eseguire il calcolo all'interno di SPSS Modeler. Il valore predefinito è udf.

Proprietà di applyxgboosttreenode

È possibile utilizzare il nodo XGBoost Tree per generare un nugget del modello XGBoost Tree. Il nome di script di questo nugget del modello è *applyxgboosttreenode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà di xgboosttreenode" a pagina 365.

Proprietà di applyxgboostlinearnode

È possibile utilizzare i nodi XGBoost Linear per generare un nugget del modello XGBoost Linear. Il nome di script di questo nugget del modello è *applyxgboostlinearnode*. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà xgboostlinearnode" a pagina 364.

Proprietà hdbscannugget

È possibile utilizzare il nodo HDBSCAN per generare un nugget del modello HDBSCAN. Il nome script di questo nugget del modello è `hdbscannugget`. Per questo nugget del modello non esistono altre proprietà. Per ulteriori informazioni sugli script del nodo Modelli specifico, vedere "Proprietà hdbscannode" a pagina 352.

Proprietà kdeapply

Il nodo di modellazione KDE può essere utilizzato per generare un nugget del modello KDE. Il nome script di questo nugget del modello è `kdeapply`. Per informazioni relative allo script del nodo di modellazione stesso, consultare "Proprietà kdemodel" a pagina 353.

Tabella 191. proprietà kdeapply

Proprietà `kdeapply`	Tipo di dati	Descrizione proprietà
`outLogDensity`	*booleano*	Specificare `True` o `False` per includere o escludere il valore di densità del log nell'output. Il valore di default è `False`.

Capitolo 15. Proprietà nodo di modellazione del database

IBM SPSS Modeler supporta l'integrazione con gli strumenti di modellazione e di data mining offerti da diversi fornitori di database, quali Microsoft SQL Server Analysis Services, Oracle Data Mining, e IBM Netezza Analytics. Operando all'interno dell'applicazione IBM SPSS Modeler, è possibile creare modelli e calcolarne il punteggio mediante algoritmi nativi del database. I modelli di database possono essere creati e manipolati anche tramite script utilizzando le proprietà descritte in questa sezione.

Ad esempio, il seguente estratto di script illustra la creazione di un modello Strutture ad albero delle decisioni Microsoft utilizzando l'interfaccia script di IBM SPSS Modeler:

```
stream = modeler.script.stream()
msbuilder = stream.createAt("mstreenode", "MSBuilder", 200, 200)

msbuilder.setPropertyValue("analysis_server_name", 'localhost')
msbuilder.setPropertyValue("analysis_database_name", 'TESTDB')
msbuilder.setPropertyValue("mode", 'Expert')
msbuilder.setPropertyValue("datasource", 'LocalServer')
msbuilder.setPropertyValue("target", 'Drug')
msbuilder.setPropertyValue("inputs", ['Age', 'Sex'])
msbuilder.setPropertyValue("unique_field", 'IDX')
msbuilder.setPropertyValue("custom_fields", True)
msbuilder.setPropertyValue("model_name", 'MSDRUG')

typenode = stream.findByType("type", None)
stream.link(typenode, msbuilder)
results = []
msbuilder.run(results)
msapplier = stream.createModelApplierAt(results[0], "Drug", 200, 300)
tablenode = stream.createAt("table", "Results", 300, 300)
stream.linkBetween(msapplier, typenode, tablenode)
msapplier.setPropertyValue("sql_generate", True)
tablenode.run([])
```

Proprietà dei nodi Modelli Microsoft

Proprietà dei nodi Modelli Microsoft

Proprietà comuni

Le seguenti proprietà sono comuni ai nodi Modelli database Microsoft.

Tabella 192. Proprietà comuni dei nodi Microsoft

Proprietà comuni dei nodi Microsoft	Valori	Descrizione proprietà
analysis_database_name	*stringa*	Nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Services.
use_transactional_data	*indicatore*	Specifica se i dati di input sono in formato tabulare o transazionale.
inputs	*elenco*	Campi di input per dati in formato tabellare.
target	*campo*	Campo predittore (non applicabile al nodo Raggruppamento cluster MS o Cluster di sequenze MS).
unique_field	*campo*	Campo chiave.

Tabella 192. Proprietà comuni dei nodi Microsoft (Continua)

Proprietà comuni dei nodi Microsoft	Valori	Descrizione proprietà
msas_parameters	*strutturato*	Parametri degli algoritmi. Per ulteriori informazioni, consultare l'argomento "Parametri degli algoritmi" a pagina 291.
with_drillthrough	*indicatore*	Opzione Con funzione drill-through.

Struttura ad albero delle decisioni MS

Per i nodi di tipo mstreenode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Raggruppamento cluster MS

Per i nodi di tipo msclusternode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Regole di associazione MS

Per i nodi di tipo msassocnode sono disponibili le seguenti proprietà specifiche:

Tabella 193. proprietà msassocnode

Proprietà msassocnode	Valori	Descrizione proprietà
id_field	*campo*	Identifica le singole transazioni nei dati.
trans_inputs	*elenco*	Campi di input per dati transazionali.
transactional_target	*campo*	Campo predittore (dati transazionali).

Naive Bayes MS

Per i nodi di tipo msbayesnode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Regressione lineare MS

Per i nodi di tipo msregressionnode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Rete neurale MS

Per i nodi di tipo msneuralnetworknode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Regressione logistica MS

Per i nodi di tipo mslogisticnode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

Serie temporali MS

Per i nodi di tipo mstimeseriesnode non sono definite proprietà specifiche. Vedere le proprietà comuni di Microsoft all'inizio di questa sezione.

MS Sequence Clustering

Per i nodi di tipo mssequenceclusternode sono disponibili le seguenti proprietà specifiche:

Tabella 194. proprietà mssequenceclusternode

Proprietà mssequenceclusternode	Valori	Descrizione proprietà
id_field	*campo*	Identifica le singole transazioni nei dati.
input_fields	*elenco*	Campi di input per dati transazionali.
sequence_field	*campo*	Identificativo sequenza.
target_field	*campo*	Campo predittore (dati in formato tabellare).

Parametri degli algoritmi

Ogni tipo di modello di database Microsoft possiede parametri specifici che è possibile impostare mediante la proprietà msas_parameters, per esempio:

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters", [["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Tali parametri sono derivati da SQL Server. Per visualizzare i parametri relativi ai singoli nodi:

1. Collocare un nodo origine del database nell'area.
2. Aprire il nodo origine del database.
3. Selezionare un'origine valida dall'elenco a discesa **Sorgente dati**.
4. Selezionare una tabella valida dall'elenco **Nome tabella**.
5. Fare clic su **OK** per chiudere il nodo origine del database.
6. Collegare il nodo Modelli database Microsoft di cui si desiderano elencare le proprietà.
7. Aprire il nodo Modelli database.
8. Selezionare la scheda **Livello avanzato**.

Vengono visualizzate le proprietà msas_parameters disponibili per quel nodo.

Proprietà dei nugget del modello Microsoft

Le seguenti proprietà sono relative ai nugget del modello creati mediante i nodi Modelli database Microsoft.

Struttura ad albero delle decisioni MS

Tabella 195. Proprietà della struttura ad albero delle decisioni MS

Proprietà applymstreenode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.
datasource	*stringa*	Nome del DSN (nome sorgente dati, Data Source Name) ODBC SQL Server.
sql_generate	*indicatore* udf	Attiva la generazione SQL.

Regressione lineare MS

Tabella 196. Proprietà della regressione lineare MS

Proprietà applymsregressionnode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.

Rete neurale MS

Tabella 197. Proprietà della rete neurale MS

Proprietà applymsneuralnetworknode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.

Regressione logistica MS

Tabella 198. Proprietà della regressione logistica MS

Proprietà applymslogisticnode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.

Serie temporali MS

Tabella 199. Proprietà delle serie temporali MS

proprietà applymstimeseriesnode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.
start_from	new_prediction historical_prediction	Specifica se effettuare previsioni future o storiche.
new_step	*number*	Definisce il periodo di tempo iniziale per le previsioni future.
historical_step	*number*	Definisce il periodo di tempo iniziale per le previsioni storiche.

Tabella 199. Proprietà delle serie temporali MS (Continua)

proprietà applymstimeseriesnode	Valori	Descrizione
end_step	*number*	Definisce il periodo di tempo finale per le previsioni.

MS Sequence Clustering

Tabella 200. Proprietà del cluster di sequenze MS

proprietà applymssequenceclusternode	Valori	Descrizione
analysis_database_name	*stringa*	Il calcolo del punteggio di questo nodo può essere eseguito direttamente in un flusso. Questa proprietà consente di identificare il nome del database di Analysis Services.
analysis_server_name	*stringa*	Nome dell'host di Analysis Server.

Proprietà dei nodi Modelli Oracle

Proprietà dei nodi Modelli Oracle

Le seguenti proprietà sono comuni ai nodi Modelli database Oracle.

Tabella 201. Proprietà comuni dei nodi Oracle

Proprietà comuni dei nodi Oracle	Valori	Descrizione proprietà
target	*campo*	
inputs	*Elenco di campi*	
partition	*campo*	Campo utilizzato per partizionare i dati in campioni distinti per le fasi di addestramento, di test e di convalida della creazione del modello.
datasource		
username		
password		
epassword		
use_model_name	*indicatore*	
model_name	*stringa*	Nome personalizzato per il nuovo modello.
use_partitioned_data	*indicatore*	Se è definito un campo partizione, questa opzione garantisce che per la creazione del modello verranno utilizzati solo i dati della partizione di addestramento.
unique_field	*campo*	
auto_data_prep	*indicatore*	Attiva o disattiva la funzione di preparazione dei dati automatici di Oracle (solo database 11g).
costs	*strutturato*	Proprietà strutturata nel formato: `[[drugA drugB 1.5] [drugA drugC 2.1]]`, dove gli argomenti racchiusi tra [] rappresentano i costi previsti effettivi.
mode	`Simple` `Expert`	Consente di ignorare determinate proprietà se impostate su `Simple`, come illustrato nelle proprietà dei singoli nodi.

Tabella 201. Proprietà comuni dei nodi Oracle (Continua)

Proprietà comuni dei nodi Oracle	Valori	Descrizione proprietà
use_prediction_probability	*indicatore*	
prediction_probability	*stringa*	
use_prediction_set	*indicatore*	

Naive Bayes Oracle

Per i nodi di tipo oranbnode sono disponibili le seguenti proprietà.

Tabella 202. Proprietà oranbnode

Proprietà oranbnode	Valori	Descrizione proprietà
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
distribuzioni di probabilità a priori	Data Equal Custom	
custom_priors	*strutturato*	Proprietà strutturata nel formato: set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Proprietà ignorata se mode è impostata su Simple.

Bayes adattivo Oracle

Per i nodi di tipo oraabnnode sono disponibili le seguenti proprietà.

Tabella 203. Proprietà oraabnnode

Proprietà oraabnnode	Valori	Descrizione proprietà
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*indicatore*	*
execution_time_limit	*numero intero*	Il valore deve essere maggiore di 0.*
max_naive_bayes_predictors	*numero intero*	Il valore deve essere maggiore di 0.*
max_predictors	*numero intero*	Il valore deve essere maggiore di 0.*
distribuzioni di probabilità a priori	Data Equal Custom	
custom_priors	*strutturato*	Proprietà strutturata nel formato: set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Proprietà ignorata se mode è impostata su Simple.

SVM Oracle

Per i nodi di tipo orasvmnode sono disponibili le seguenti proprietà.

Tabella 204. proprietà orasvmnode

Proprietà orasvmnode	Valori	Descrizione proprietà
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*numero intero*	Solo kernel gaussiano. Il valore deve essere maggiore di 0.*
convergence_tolerance	*number*	Il valore deve essere maggiore di 0.*
use_standard_deviation	*indicatore*	Solo kernel gaussiano.*
standard_deviation	*number*	Il valore deve essere maggiore di 0.*
use_epsilon	*indicatore*	Solo modelli di regressione.*
epsilon	*number*	Il valore deve essere maggiore di 0.*
use_complexity_factor	*indicatore*	*
complexity_factor	*number*	*
use_outlier_rate	*indicatore*	Solo variante a una classe.*
outlier_rate	*number*	Solo variante a una classe. 0.0–1.0.*
weights	Data Equal Custom	
custom_weights	*strutturato*	Proprietà strutturata nel formato: set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Proprietà ignorata se mode è impostata su Simple.

Modelli lineari generalizzati Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo oraglmnode.

Tabella 205. proprietà oraglmnode

Proprietà oraglmnode	Valori	Descrizione proprietà
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*indicatore*	*
row_weights_field	*campo*	*
save_row_diagnostics	*indicatore*	*

Tabella 205. proprietà oraglmnode (Continua)

Proprietà oraglmnode	Valori	Descrizione proprietà
row_diagnostics_table	*stringa*	*
coefficient_confidence	*number*	*
use_reference_category	*indicatore*	*
reference_category	*stringa*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*indicatore*	*

* Proprietà ignorata se mode è impostata su Simple.

struttura ad albero delle decisioni Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo oradecisiontreenode.

Tabella 206. proprietà oradecisiontreenode

Proprietà oradecisiontreenode	Valori	Descrizione proprietà
use_costs	*indicatore*	
impurity_metric	Entropy Gini	
term_max_depth	*numero intero*	2–20.*
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*
term_minrec_node	*numero intero*	Il valore deve essere maggiore di 0.*
term_minrec_split	*numero intero*	Il valore deve essere maggiore di 0.*
display_rule_ids	*indicatore*	*

* Proprietà ignorata se mode è impostata su Simple.

O-Cluster Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo oraoclusternode.

Tabella 207. proprietà oraoclusternode

Proprietà oraoclusternode	Valori	Descrizione proprietà
max_num_clusters	*numero intero*	Il valore deve essere maggiore di 0.
max_buffer	*numero intero*	Il valore deve essere maggiore di 0.*
sensitivity	*number*	0.0–1.0.*

* Proprietà ignorata se mode è impostata su Simple.

Medie K Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo orakmeansnode.

Tabella 208. proprietà orakmeansnode

Proprietà orakmeansnode	Valori	Descrizione proprietà
num_clusters	*numero intero*	Il valore deve essere maggiore di 0.
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
iterations	*numero intero*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	L'impostazione di default è Variance.*
num_bins	*numero intero*	Il valore deve essere maggiore di 0.*
block_growth	*numero intero*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* Proprietà ignorata se mode è impostata su Simple.

NMF Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo oranmfnode.

Tabella 209. proprietà oranmfnode

Proprietà oranmfnode	Valori	Descrizione proprietà
normalization_method	minmax none	
use_num_features	*indicatore*	*
num_features	*numero intero*	0–1. Il valore di default viene stimato dall'algoritmo in base ai dati.*
random_seed	*number*	*
num_iterations	*numero intero*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*indicatore*	*

* Proprietà ignorata se mode è impostata su Simple.

Apriori Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo oraapriorinode.

Tabella 210. proprietà oraapriorinode

Proprietà oraapriorinode	Valori	Descrizione proprietà
content_field	*campo*	

Tabella 210. proprietà oraapriorinode (Continua)

Proprietà oraapriorinode	Valori	Descrizione proprietà
id_field	*campo*	
max_rule_length	*numero intero*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*indicatore*	

Oracle MDL (Lunghezza descrizione minima)

Per i nodi di tipo oramdlnode non sono definite proprietà specifiche. Vedere le proprietà comuni di Oracle all'inizio di questa sezione.

Importanza attributo Oracle (AI)

Le seguenti proprietà sono disponibili per i nodi di tipo oraainode.

Tabella 211. proprietà oraainode

proprietà oraainode	Valori	Descrizione proprietà
custom_fields	*indicatore*	Se vera, consente di specificare i campi obiettivo, di input e di altro tipo per il nodo corrente. Se falsa, vengono utilizzate le impostazioni correnti di un nodo Tipo a monte.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi importanti.
important_label	*stringa*	Specifica l'etichetta per la classificazione "importante".
select_marginal	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi marginali.
marginal_label	*stringa*	Specifica l'etichetta per la classificazione "marginale".
important_above	*number*	0.0–1.0.
select_unimportant	*indicatore*	Quando selection_mode è impostata su ImportanceLevel, specifica se selezionare i campi non importanti.
unimportant_label	*stringa*	Specifica l'etichetta per la classificazione "non importante".
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	Quando selection_mode è impostata su ImportanceValue, specifica il valore di interruzione da utilizzare. Accetta i valori compresi tra 0 e 100.
top_n	*number*	Quando selection_mode è impostata su TopN, specifica il valore di interruzione da utilizzare. Accetta i valori compresi tra 0 e 1000.

Proprietà dei nugget del modello Oracle

Le seguenti proprietà sono relative ai nugget del modello creati mediante i modelli Oracle.

Naive Bayes Oracle

Per i nodi di tipo `applyoranbnode` non sono definite proprietà specifiche.

Bayes adattivo Oracle

Per i nodi di tipo `applyoraabnnode` non sono definite proprietà specifiche.

SVM Oracle

Non vi sono proprietà specifiche definite per i nodi di tipo `applyorasvmnode`.

struttura ad albero delle decisioni Oracle

Le seguenti proprietà sono disponibili per i nodi di tipo `applyoradecisiontreenode`.

Tabella 212. proprietà applyoradecisiontreenode

Proprietà `applyoradecisiontreenode`	Valori	Descrizione proprietà
`use_costs`	*indicatore*	
`display_rule_ids`	*indicatore*	

O-Cluster Oracle

Non vi sono proprietà specifiche definite per i nodi di tipo `applyoraoclusternode`.

Medie K Oracle

Non vi sono proprietà specifiche definite per i nodi di tipo `applyorakmeansnode`.

NMF Oracle

Per i nodi di tipo `applyoranmfnode` è disponibile la seguente proprietà:

Tabella 213. proprietà applyoranmfnode

proprietà `applyoranmfnode`	Valori	Descrizione proprietà
`display_all_features`	*indicatore*	

Apriori Oracle

Questo nugget del modello non può essere applicato negli script.

MDL Oracle

Questo nugget del modello non può essere applicato negli script.

Proprietà dei nodi Modelli IBM Netezza Analytics

Proprietà dei nodi Modelli Netezza

Le seguenti proprietà sono comuni ai nodi Modelli database IBM Netezza.

Tabella 214. Proprietà comuni dei nodi Netezza

Proprietà comuni dei nodi Netezza	Valori	Descrizione proprietà
custom_fields	*indicatore*	Se vera, consente di specificare i campi obiettivo, di input e di altro tipo per il nodo corrente. Se falsa, vengono utilizzate le impostazioni correnti di un nodo Tipo a monte.
inputs	*[campo1 ... campoN]*	I campi di input o predittore utilizzati dal modello.
target	*campo*	Campo obiettivo (continuo o categoriale).
record_id	*campo*	Campo da utilizzare come identificatore univoco del record.
use_upstream_connection	*indicatore*	Se vera (default) usa i dettagli di connessione specificati in un nodo a monte. Non utilizzato se viene specificato move_data_to_connection.
move_data_connection	*indicatore*	Se vera, sposta i dati nel database specificato da connessione. Non utilizzato se viene specificato use_upstream_connection.
connection	*strutturato*	La stringa di connessione per il database Netezza in cui è archiviato il modello. Proprietà strutturata nel formato: ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' [true|false]] dove: <dsn> è il nome dell'origine dati <username> e <psw> sono il nome utente e la password del database <catname> è il nome del catalogo <conn_attribs> sono gli attributi di connessione true | false indica se la password è necessaria.
table_name	*stringa*	Nome della tabella di database in cui sarà archiviato il modello.
use_model_name	*indicatore*	Se vera, utilizza il nome specificato da model_name come nome del modello, in caso contrario il nome del modello viene creato dal sistema.
model_name	*stringa*	Nome personalizzato per il nuovo modello.
include_input_fields	*indicatore*	Se vera, passa tutti i campi di input a valle, in caso contrario passa solo record_id e i campi generati dal modello.

Struttura ad albero delle decisioni di Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo `netezzadectreenode`.

Tabella 215. proprietà netezzadectreenode

Proprietà `netezzadectreenode`	Valori	Descrizione proprietà
`impurity_measure`	`Entropy` `Gini`	La misurazione dell'impurità utilizzata per valutare il punto migliore in cui suddividere la struttura ad albero.
`max_tree_depth`	*numero intero*	Numero massimo di livelli di cui una struttura ad albero può crescere. Il valore di default è 62 (il massimo possibile).
`min_improvement_splits`	*number*	Miglioramento minimo in impurità perché si verifichi una suddivisione. L'impostazione di default è 0.01.
`min_instances_split`	*numero intero*	Numero minimo di record ancora da suddividere perché sia possibile effettuare una suddivisione. Il valore di default è 2 (il minimo possibile).
`weights`	*strutturato*	Ponderazioni relative per le classi. Proprietà strutturata nel formato: `set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]]` L'impostazione di default è il peso 1 per tutte le classi.
`pruning_measure`	`Acc` `wAcc`	L'impostazione di default è `Acc` (accuracy). In alternativa `wAcc` (precisione ponderata) tiene conto dei pesi delle classi durante l'applicazione del taglio.

`prune_tree_options`	`allTrainingData`
	`partitionTrainingData`
	`useOtherTable`
		L'impostazione di default da utilizzare è `allTrainingData` per valutare la precisione del modello. Utilizzare `partitionTrainingData` èer specificare una percentuale dei dati di addestramento da utilizzare o `useOtherTable` per utilizzare un insieme di dati di addestramento di una tabella di database specifica.

`perc_training_data`	*number*	Se `prune_tree_options` è impostato su `partitionTrainingData`, specifica la percentuale di dati da utilizzare per l'addestramento.
`prune_seed`	*numero intero*	Seme random da utilizzare per replicare i risultati delle analisi quando `prune_tree_options` è impostato su `partitionTrainingData`; il valore di default è 1.
`pruning_table`	*stringa*	Nome della tabella di un insieme di dati di taglio separato per la stima della precisione del modello.

Tabella 215. proprietà netezzadectreenode (Continua)

Proprietà netezzadectreenode	Valori	Descrizione proprietà
compute_probabilities	*indicatore*	Se vera, produce un campo livello di confidenza (probabilità), nonché un campo di previsione.

Medie K Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzakmeansnode.

Tabella 216. proprietà netezzakmeansnode

Proprietà netezzakmeansnode	Valori	Descrizione proprietà
distance_measure	Euclidean Manhattan Canberra maximum	Metodo da utilizzare per misurare la distanza fra punti dati.
num_clusters	*numero intero*	Numero di cluster da creare; l'impostazione di default è 3.
max_iterations	*numero intero*	Numero di iterazioni dell'algoritmo dopo cui interrompere l'addestramento del modello; l'impostazione di default è 5.
rand_seed	*numero intero*	Seme random da utilizzare per replicare i risultati delle analisi; l'impostazione di default è 12345.

Rete di Bayes Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzabayesnode.

Tabella 217. proprietà netezzabayesnode

Proprietà netezzabayesnode	Valori	Descrizione proprietà
base_index	*numero intero*	Identificatore numerico assegnato al primo campo di input per la gestione interna; l'impostazione di default è 777.
sample_size	*numero intero*	Dimensione del campione da prendere se il numero di attributi è molto elevato; l'impostazione di default è 10.000.
display_additional_information	*indicatore*	Se vera, visualizza ulteriori informazioni sull'avanzamento in una finestra di dialogo.
type_of_prediction	best neighbors nn-neighbors	Tipo di algoritmo di previsione da utilizzare: ottima (elemento adiacente con maggiore correlazione), elementi adiacenti (previsione ponderata degli elementi adiacenti) o elementi adiacenti NN (elementi adiacenti non null).

Naive Bayes Netezza

Le seguenti proprietà sono disponinili per i nodi di tipo netezzanaivebayesnode.

Tabella 218. proprietà netezzanaivebayesnode

Proprietà netezzanaivebayesnode	Valori	Descrizione proprietà
compute_probabilities	*indicatore*	Se vera, produce un campo livello di confidenza (probabilità), nonché un campo di previsione.
use_m_estimation	*indicatore*	Se vera, utilizza la tecnica della stima m per evitare le probabilità zero durante la stima.

KNN Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzaknnnode.

Tabella 219. proprietà netezzaknnnode

Proprietà netezzaknnnode	Valori	Descrizione proprietà
weights	*strutturato*	Proprietà strutturata utilizzata per assegnare i pesi alle singole classi. Esempio: set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]
distance_measure	Euclidean Manhattan Canberra Massimo	Metodo da utilizzare per misurare la distanza fra punti dati.
num_nearest_neighbors	*numero intero*	Numero di elementi adiacenti più vicini per un caso particolare; l'impostazione di default è 3.
standardize_measurements	*indicatore*	Se vera, standardizza le misurazioni per i campi di input continui prima di calcolare i valori delle distanze.
use_coresets	*indicatore*	Se vera, utilizza il campionamento degli insiemi centrali per velocizzare il calcolo per insiemi di dati di grandi dimensioni.

Raggruppamento cluster divisivo Netezza

Le seguenti proprietà sono disponinili per i nodi di tipo netezzadivclusternode.

Tabella 220. proprietà netezzadivclusternode

Proprietà netezzadivclusternode	Valori	Descrizione proprietà
distance_measure	Euclidean Manhattan Canberra Massimo	Metodo da utilizzare per misurare la distanza fra punti dati.
max_iterations	*numero intero*	Numero massimo di iterazioni dell'algoritmo da eseguire prima di interrompere l'addestramento del modello; l'impostazione di default è 5.
max_tree_depth	*numero intero*	Numero massimo di livelli in cui possono essere suddivisi gli insiemi di dati; l'impostazione di default è 3.
rand_seed	*numero intero*	Seme random, utilizzato per replicare le analisi; l'impostazione di default è 12345.

Tabella 220. proprietà netezzadivclusternode (Continua)

Proprietà netezzadivclusternode	Valori	Descrizione proprietà
min_instances_split	*numero intero*	Numero minimo di record che possono essere suddivisi; l'impostazione di default è 5.
level	*numero intero*	Livello di gerarchia a cui deve essere calcolato il punteggio dei record; l'impostazione di default è -1.

PCA Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzapcanode.

Tabella 221. proprietà netezzapcanode

Proprietà netezzapcanode	Valori	Descrizione proprietà
center_data	*indicatore*	Se vera (default), esegue la centratura dei dati (nota anche come "sottrazione delle medie") prima dell'analisi.
perform_data_scaling	*indicatore*	Se vera, esegue il ridimensionamento dei dati prima dell'analisi. Questa operazione può ridurre l'arbitrarietà dell'analisi quando vengono misurate diverse variabili in diverse unità.
force_eigensolve	*indicatore*	Se vera, utilizza un metodo meno accurato ma più veloce per trovare le componenti principali.
pc_number	*numero intero*	Numero di componenti principali a cui deve essere ridotto l'insieme di dati; l'impostazione di default è 1.

Struttura ad albero di regressione Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzaregtreenode.

Tabella 222. proprietà netezzaregtreenode

Proprietà netezzaregtreenode	Valori	Descrizione proprietà
max_tree_depth	*numero intero*	Numero massimo di livelli a cui può espandersi la struttura ad albero al di sotto del nodo root; l'impostazione di default è 10.
split_evaluation_measure	Variance	Misurazione dell'impurità delle classi, utilizzata per valutare il punto migliore in cui suddividere la struttura ad albero; l'impostazione di default (e al momento l'unica opzione) è Variance.
min_improvement_splits	*number*	Quantità minima di riduzione dell'impurità prima che venga creata una nuova suddivisione nella struttura ad albero.
min_instances_split	*numero intero*	Numero minimo di record che possono essere suddivisi.
pruning_measure	mse r2 pearson spearman	Metodo da utilizzare per il taglio.

Tabella 222. proprietà netezzaregtreenode (Continua)

Proprietà netezzaregtreenode	Valori	Descrizione proprietà
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	L'impostazione di default da utilizzare è allTrainingData per valutare la precisione del modello. Utilizzare partitionTrainingData èer specificare una percentuale dei dati di addestramento da utilizzare o useOtherTable per utilizzare un insieme di dati di addestramento di una tabella di database specifica.
perc_training_data	*number*	Se prune_tree_options è impostato su PercTrainingData, specifica la percentuale di dati da utilizzare per l'addestramento.
prune_seed	*numero intero*	Seme random da utilizzare per replicare i risultati delle analisi quando prune_tree_options è impostato su PercTrainingData; l'impostazione di default è 1.
pruning_table	*stringa*	Nome della tabella di un insieme di dati di taglio separato per la stima della precisione del modello.
compute_probabilities	*indicatore*	Se vera, specifica che le varianze delle classi assegnate devono essere incluse nell'output.

Regressione lineare Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo' netezzalineregressionnode.

Tabella 223. proprietà netezzalineregressionnode

Proprietà netezzalineregressionnode	Valori	Descrizione proprietà
use_svd	*indicatore*	Se vera, utilizza la matrice Decomposizione ai valori singolari, anziché la matrice originale, per una maggiore velocità e precisione numerica.
include_intercept	*indicatore*	Se vera (default), aumenta la precisione generale della soluzione.
calculate_model_diagnostics	*indicatore*	Se vera, calcola la diagnostica del modello.

Serie temporali Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzatimeseriesnode.

Tabella 224. proprietà netezzatimeseriesnode

Proprietà netezzatimeseriesnode	Valori	Descrizione proprietà
time_points	*campo*	Il campo di input contenente i valori di data o ora per le serie temporali.

Tabella 224. proprietà netezzatimeseriesnode (Continua)

Proprietà **netezzatimeseriesnode**	Valori	Descrizione proprietà
time_series_ids	*campo*	Campo di input contenente gli ID delle serie temporali; utilizzarlo se l'input contiene più di una serie temporali.
model_table	*campo*	Nome della tabella di database in cui sarà archiviato il modello di serie temporali Netezza.
description_table	*campo*	Nome della tabella di input che contiene i nomi e le descrizioni delle serie temporali.
seasonal_adjustment_table	*campo*	Nome della tabella di output in cui i valori corretti per stagionalità calcolati dagli algoritmi di decomposizione a tendenza stagionale o a livellamento esponenziale verranno memorizzati.
algorithm_name	SpectralAnalysis o spectral ExponentialSmoothing o esmoothing ARIMA SeasonalTrendDecomposition o std	Un algoritmo da utilizzare per la modellazione di serie temporali.
trend_name	N A DA M DM	Tipo di tendenza per il livellamento esponenziale: N - nessuno A - additivo DA - additivo smorzato M - moltiplicativo DM - moltiplicativo smorzato
seasonality_type	N A M	Tipo di stagionalità per il livellamento esponenziale: N - nessuno A - additivo M - moltiplicativo
interpolation_method	linear cubicspline exponentialspline	Metodo di interpolazione da utilizzare.
timerange_setting	SD SP	Impostazione dell'intervallo di tempo da utilizzare: SD - determinato dal sistema (utilizza la gamma completa dei dati di serie temporali) SP - specificato dall'utente tramite earliest_time e latest_time

Tabella 224. proprietà netezzatimeseriesnode (Continua)

Proprietà **netezzatimeseriesnode**	Valori	Descrizione proprietà
earliest_time	*numero intero* *data* *ora* *timestamp*	Valori di inizio e fine, se timerange_setting è SP. Il formato deve seguire il valore time_points. Ad esempio, se il campo time_points contiene una data, questo valore deve essere una data. Esempio: set NZ_DT1.timerange_setting = 'SP' set NZ_DT1.earliest_time = '1921-01-01' set NZ_DT1.latest_time = '2121-01-01'
latest_time		
arima_setting	SD SP	Impostazione per l'algoritmo ARIMA (utilizzato solo se algorithm_name è impostato su ARIMA): SD - determinato dal sistema SP - specificato dall'utente Se arima_setting = SP, utilizzare i seguenti parametri per impostare i valori stagionali e non stagionali. Esempio (non solo stagionali): set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol	less eq lesseq	ARIMA - operatore per i parametri p, d, q, sp, sd, e sq: less - minore di eq - uguale a lesseq - minore di o uguale a
d_symbol		
q_symbol		
sp_symbol		
sd_symbol		
sq_symbol		
p	*numero intero*	ARIMA - gradi non stagionali di autorcorrelazione.
q	*numero intero*	ARIMA - valore di derivazione non stagionale.
d	*numero intero*	ARIMA - numero non stagionale di ordini di media mobile nel modello.
sp	*numero intero*	ARIMA - gradi stagionali di autorcorrelazione.
sq	*numero intero*	ARIMA - valore di derivazione stagionale.

Tabella 224. proprietà netezzatimeseriesnode (Continua)

Proprietà **netezzatimeseriesnode**	Valori	Descrizione proprietà
sd	*numero intero*	ARIMA - numero stagionale di ordini di media mobile nel modello.
advanced_setting	SD SP	Determina il modo in cui le impostazioni avanzate devono essere gestite: SD - determinato dal sistema SP - specificato dall'utente tramite period , units_period e forecast_setting. Esempio: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
period	*numero intero*	Lunghezza del ciclo stagionale, specificato insieme a units_period. Non valido per l'analisi spettrale.
units_period	ms s min h d wk q y	Le unità in cui viene espresso period: ms - millisecondi s - secondi min - minuti h - ore d - giorni wk - settimane q - trimestri y - anni Ad esempio, per una serie temporale settimanale utilizzare 1 per period e wk per units_period.
forecast_setting	forecasthorizon forecasttimes	Specifica il modo in cui le previsioni devono essere eseguite.
forecast_horizon	*numero intero* *data* *ora* *timestamp*	Se forecast_setting = forecasthorizon, specifica il valore del punto finale per la previsione. Il formato deve seguire il valore time_points. Ad esempio, se il campo time_points contiene una data, questo valore deve essere una data.
forecast_times	*numero intero* *data* *ora* *timestamp*	Se forecast_setting = forecasttimes, specifica i valori da utilizzare per effettuare le previsioni. Il formato deve seguire il valore time_points. Ad esempio, se il campo time_points contiene una data, questo valore deve essere una data.
include_history	*indicatore*	Indica se i valori storici devono essere inclusi nell'output.

Tabella 224. proprietà netezzatimeseriesnode (Continua)

Proprietà netezzatimeseriesnode	Valori	Descrizione proprietà
include_interpolated_values	*indicatore*	Indica se i valori interpolari devono essere inclusi nell'output. Non valido se include_history è false.

Lineare generalizzato Netezza

Le seguenti proprietà sono disponibili per i nodi di tipo netezzaglmnode.

Tabella 225. proprietà netezzaglmnode

Proprietà netezzaglmnode	Valori	Descrizione proprietà
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	Tipo di distribuzione; l'impostazione di default è bernoulli.
dist_params	*number*	Il valore del parametro di distribuzione da utilizzare. Applicabile solo se distribution è Negativebinomial.
trials	*numero intero*	Applicabile solo se distribution è Binomial. la risposta obiettivo rappresenta il numero di eventi di un insieme di prove, il campo obiettivo contiene il numero di eventi e il campo trials contiene il numero di prove.
model_table	*campo*	Nome della tabella di database in cui sarà archiviato il modello lineare generalizzato Netezza.
maxit	*numero intero*	Numero massimo di iterazioni che possono essere eseguite dall'algoritmo; il valore di default è 20.
eps	*number*	Il valore di errore massimo (in notazione scientifica) raggiunto il quale l'algoritmo deve interrompere la ricerca del modello più adatto. Il valore di default è -3, vale a dire 1E-3 oppure 0,001.
tol	*number*	Il valore (in notazione scientifica) sotto il quale gli errori vengono trattati come se avessero valore zero. Il valore di default è -7, vale a dire che i valori inferiori a 1E-7 (o 0,0000001) sono considerati insignificanti.

Tabella 225. proprietà netezzaglmnode (Continua)

Proprietà netezzaglmnode	Valori	Descrizione proprietà
link_func	identità inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Funzione di collegamento da utilizzare; il valore di default è logit.
link_params	*number*	Il valore del parametro della funzione di collegamento da utilizzare. Applicabile solo se link_function è power o oddspower.
interaction	[[[*nomicol1*],[*livelli1*]], [[*nomicol2*],[*livelli2*]], ...,[[*nomicolN*],[*livelliN*]],]	Specifica le interazioni tra i campi. *nomicol* è un elenco di campi di input e *livello* è sempre 0 per ogni campo. Esempio: [[["K","BP","Sex","K"],[0,0,0,0]], [["Age","Na"],[0,0]]]
intercept	*indicatore*	Se true, include l'intercettazione nel modello.

Proprietà dei nugget del modello Netezza

Le seguenti proprietà sono comuni ai nugget del modello di database Netezza.

Tabella 226. Proprietà comuni dei nugget del modello Netezza

Proprietà comuni dei nugget del modello Netezza	Valori	Descrizione proprietà
connection	*stringa*	La stringa di connessione per il database Netezza in cui è archiviato il modello.
table_name	*stringa*	Nome della tabella di database in cui è archiviato il modello.

Altre proprietà dei nugget del modello sono identiche a quelle del nodo Modelli corrispondente.

Di seguito sono indicati i nomi script dei nugget del modello.

Tabella 227. I nomi degli script dei nugget del modello Netezza

Nugget del modello	Nome script
Struttura ad albero delle decisioni	applynetezzadectreenode

Tabella 227. I nomi degli script dei nugget del modello Netezza (Continua)

Nugget del modello	Nome script
Medie K	`applynetezzakmeansnode`
Rete di Bayes	`applynetezzabayesnode`
Naive Bayes	`applynetezzanaivebayesnode`
KNN	`applynetezzaknnnode`
Raggruppamento cluster divisivo	`applynetezzadivclusternode`
PCA	`applynetezzapcanode`
Struttura ad albero di regressione	`applynetezzaregtreenode`
Regressione lineare	`applynetezzalineregressionnode`
Serie temporali	`applynetezzatimeseriesnode`
Lineare generalizzato	`applynetezzaglmnode`

Capitolo 16. Proprietà del nodo di output

Le proprietà del nodo Output sono leggermente diverse da quelle di altri tipi di nodi. Anziché fare riferimento all'opzione di un nodo specifico, le proprietà dei nodi Output consentono di memorizzare un riferimento all'oggetto di output. Ciò risulta utile per recuperare un valore da una tabella e impostarlo come un parametro del flusso.

In questa sezione vengono illustrate le proprietà degli script disponibili per i nodi Output.

proprietà analysisnode

Il nodo Analisi valuta la capacità dei modelli predittivi di generare previsioni accurate. I nodi Analisi eseguono diversi confronti tra i valori previsti e i valori effettivi per uno o più nugget del modello. Possono inoltre confrontare i modelli predittivi fra loro.

Esempio

```
node = stream.create("analysis", "My node")
# "Analysis" tab
node.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# "Define User Measure..."
node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# "Output" tab
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Tabella 228. proprietà analysisnode

Proprietà analysisnode	**Tipo di dati**	**Descrizione proprietà**
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output.
by_fields	*elenco*	

Tabella 228. proprietà analysisnode (Continua)

Proprietà analysisnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Se si tratta di output su disco, di dati o HTML, rappresenta il nome del file di output.
coincidence	*indicatore*	
performance	*indicatore*	
evaluation_binary	*indicatore*	
confidence	*indicatore*	
threshold	*number*	
improve_accuracy	*number*	
field_detection_method	Metadata Name	Determina in che modo i campi previsti vengono fatti corrispondere al campo obiettivo originale. Specificare Metadata o Name.
inc_user_measure	*indicatore*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	
user_compute	[Mean Sum Min Max SDev]	

proprietà dataauditnode

Il nodo Esplora offre una prima panoramica completa dei dati, incluse statistiche riassuntive, istogrammi e distribuzione per ciascun campo, nonché informazioni su valori anomali, mancanti ed estremi. I risultati vengono visualizzati in una matrice di semplice lettura che può essere ordinata e utilizzata per generare grafici a schermo intero e nodi di preparazione dei dati.

Esempio

```
filenode = stream.createAt("variablefile", "File", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("dataaudit", "My node", 196, 100)
stream.link(filenode, node)
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("fields", ["Age", "Na", "K"])
node.setPropertyValue("display_graphs", True)
node.setPropertyValue("basic_stats", True)
node.setPropertyValue("advanced_stats", True)
node.setPropertyValue("median_stats", False)
node.setPropertyValue("calculate", ["Count", "Breakdown"])
node.setPropertyValue("outlier_detection_method", "std")
node.setPropertyValue("outlier_detection_std_outlier", 1.0)
node.setPropertyValue("outlier_detection_std_extreme", 3.0)
node.setPropertyValue("output_mode", "Screen")
```

Tabella 229. proprietà dataauditnode

Proprietà dataauditnode	Tipo di dati	Descrizione proprietà
custom_fields	*indicatore*	

Tabella 229. proprietà dataauditnode (Continua)

Proprietà dataauditnode	Tipo di dati	Descrizione proprietà
fields	*[campo1 ... campoN]*	
overlay	*campo*	
display_graphs	*indicatore*	Utilizzato per attivare o disattivare la visualizzazione di grafici nella matrice di output.
basic_stats	*indicatore*	
advanced_stats	*indicatore*	
median_stats	*indicatore*	
calculate	Count Breakdown	Utilizzato per calcolare valori mancanti. Selezionare uno, entrambi o nessun metodo di calcolo.
outlier_detection_method	std iqr	Utilizzato per specificare il metodo di rilevamento dei valori anomali ed estremi.
outlier_detection_std_outlier	*number*	Se outlier_detection_method è std, specifica il numero da utilizzare per definire i valori anormali.
outlier_detection_std_extreme	*number*	Se outlier_detection_method è std, specifica il numero da utilizzare per definire i valori estremi.
outlier_detection_iqr_outlier	*number*	Se outlier_detection_method è iqr, specifica il numero da utilizzare per definire i valori anormali.
outlier_detection_iqr_extreme	*number*	Se outlier_detection_method è iqr, specifica il numero da utilizzare per definire i valori estremi.
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output.
paginate_output	*indicatore*	Quando output_format è HTML, l'output viene separato in pagine.
lines_per_page	*number*	Se utilizzato con paginate_output, specifica le righe per pagina di output.
full_filename	*stringa*	

Proprietà extensionoutputnode

Il nodo Output estensione consente di analizzare i dati ed i risultati del calcolo del punteggio del modello utilizzando il proprio script R o Python for Spark personalizzato. L'output dell'analisi può essere grafico o di testo. L'output viene aggiunto alla scheda **Output** del riquadro dei manager; in alternativa, è possibile reindirizzare l'output in un file.

Esempio Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_output", "extension_output")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import json
import spss.pyspark.runtime

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]
print df
"""

node.setPropertyValue("python_syntax", python_script)
```

Esempio R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

Tabella 230. Proprietà extensionoutputnode

Proprietà extensionoutputnode	Tipo di dati	Descrizione proprietà
syntax_type	*R* *Python*	Specifica quale script viene eseguito, R o Python (R è il valore predefinito).
r_syntax	*stringa*	Sintassi degli script R per il calcolo del punteggio del modello.
python_syntax	*stringa*	Sintassi degli script Python per il calcolo del punteggio del modello.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.

Tabella 230. Proprietà extensionoutputnode (Continua)

Proprietà extensionoutputnode	Tipo di dati	Descrizione proprietà
output_to	Screen File	Specifica il tipo di output (Screen o File).
output_type	Graph Text	Specifica se produrre un output grafico o di testo.
full_filename	*stringa*	Il nome del file da utilizzare per l'output generato.
graph_file_type	HTML COU	Il tipo di file di output (.html o .cou).
text_file_type	HTML TEXT COU	Specifica il tipo di file per l'output di testo (.html, .txt o .cou).

Proprietà kde

KDE (Kernel Density Estimation)© utilizza gli algoritmi Ball Tree o KD Tree per query efficienti e combina i concetti di apprendimento non supervisionato, ingegneria della funzione e modellazione dei dati. Gli approcci basati su Neighbor come ad esempio KDE rappresentano alcune tra le più popolari e utili tecniche di stima della densità. I nodi di modellazione KDE e di simulazione KDE in SPSS Modeler presentano le funzioni principali e i parametri comunemente utilizzati della libreria KDE. I nodi sono implementati in Python.

Tabella 231. Proprietà kde

Proprietà kde	Tipo di dati	Descrizione proprietà
inputs	*campo*	Campi di input per il raggruppamento tramite cluster.
bandwidth	*double*	Il valore predefinito è 1.
kernel	*stringa*	Il kernel da utilizzare: gaussian o tophat. L'impostazione predefinita è gaussian.
algorithm	*stringa*	L'algoritmo della struttura ad albero da utilizzare: kd_tree, ball_tree, o auto. Il valore predefinito è auto.
metric	*stringa*	La metrica da utilizzare quando si calcola la distanza. Per l'algoritmo kd_tree scegliere tra: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2, o L1. Per l'algoritmo ball_tree, scegliere tra: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath, o Kulsinski. L'impostazione predefinita è Euclidean.
atol	*float*	La tolleranza assoluta desiderata. Una tolleranza superiore generalmente dà luogo ad un'esecuzione più veloce. L'impostazione predefinita è 0.0.
rtol	*float*	La tolleranza relativa desiderata del risultato. Una tolleranza superiore dà luogo generalmente ad un'esecuzione più veloce. Il valore predefinito è 1E-8.

Tabella 231. Proprietà kde (Continua)

Proprietà kde	Tipo di dati	Descrizione proprietà
breadthFirst	*booleano*	Impostare su True per utilizzare un approccio breadth-first. Impostare su False per utilizzare un approccio depth-first. L'impostazione predefinita è True.
LeafSize	*numero intero*	La dimensione dell'elemento foglia della struttura ad albero sottostante. IL valore predefinito è 40. La modifica di questo valore può impattare significativamente sulle prestazioni.
pValue	*double*	Specificare il valore P da utilizzare se si sta utilizzando Minkowski per la metrica. Il valore predefinito è 1.5.

proprietà matrixnode

Il nodo Matrice crea una tabella che mostra le relazioni tra i campi. In genere viene utilizzato per mostrare le relazioni tra due campi simbolici, ma è possibile avvalersene anche per mostrare le relazioni tra campi flag o numerici.

Esempio

```
node = stream.create("matrix", "My node")
# "Settings" tab
node.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
# "Appearance" tab
node.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# "Output" tab
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabella 232. proprietà matrixnode

Proprietà matrixnode	Tipo di dati	Descrizione proprietà
fields	Selected Flags Numerics	
row	*campo*	
column	*campo*	

Tabella 232. proprietà matrixnode (Continua)

Proprietà **matrixnode**	Tipo di dati	Descrizione proprietà
include_missing_values	*indicatore*	Specifica se i valori mancanti definiti dall'utente (vuoti) e i valori mancanti di sistema (null) sono inclusi nell'output delle righe e delle colonne.
cell_contents	CrossTabs Function	
function_field	*stringa*	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Crescente Decrescente	
highlight_top	*number*	Se diversa da zero, è vera.
highlight_bottom	*number*	Se diversa da zero, è vera.
display	[Counts Expected Residuals RowPct ColumnPct TotalPct]	
include_totals	*indicatore*	
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output. I formati Formatted e Delimited possono utilizzare il modificatore transposed, che traspone righe e colonne nella tabella.
paginate_output	*indicatore*	Quando output_format è HTML, l'output viene separato in pagine.
lines_per_page	*number*	Se utilizzato con paginate_output, specifica le righe per pagina di output.
full_filename	*stringa*	

proprietà meansnode

Il nodo Medie confronta le medie tra gruppi indipendenti o coppie di campi correlati per verificare se esiste una differenza significativa. Per esempio, è possibile confrontare le entrate medie prima e dopo il lancio di una promozione, oppure confrontare le entrate determinate da clienti che non hanno ricevuto la promozione con quelli che l'hanno ricevuta.

Esempio

```
node = stream.create("means", "My node")
node.setPropertyValue("means_mode", "BetweenFields")
node.setPropertyValue("paired_fields", [["OPEN_BAL", "CURR_BAL"]])
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("output_view", "Advanced")
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/means_output.html")
```

Tabella 233. proprietà meansnode

Proprietà meansnode	Tipo di dati	Descrizione proprietà
means_mode	BetweenGroups BetweenFields	Specifica il tipo di medie statistiche da eseguire sui dati.
test_fields	[campo1 ... campon]	Specifica il campo di verifica quando means_mode è impostato su BetweenGroups.
grouping_field	*campo*	Specifica il campo di raggruppamento.
paired_fields	[[campo1 campo2] [campo3 campo4] ...]	Specifica le coppie di campi da utilizzare quando means_mode è impostato su BetweenFields.
label_correlations	*indicatore*	Specifica se le etichette di correlazione devono essere visualizzate nell'output. Questa impostazione si applica solo quando means_mode è impostato su BetweenFields.
correlation_mode	Probability Assoluti	Specifica se etichettare le correlazioni per probabilità o valore assoluto.
weak_label	*stringa*	
medium_label	*stringa*	
strong_label	*stringa*	
weak_below_probability	*number*	Quando correlation_mode è impostato su Probabilità, specifica il valore di interruzione per correlazioni deboli. Questo valore deve essere compreso 0 e 1, ad esempio 0.90.
strong_above_probability	*number*	Valore di interruzione per correlazioni forti.

Tabella 233. proprietà meansnode (Continua)

Proprietà meansnode	Tipo di dati	Descrizione proprietà
weak_below_absolute	*number*	Quando correlation_mode è impostato su Absolute, specifica il valore di interruzione per correlazioni deboli. Questo valore deve essere compreso 0 e 1, ad esempio 0.90.
strong_above_absolute	*number*	Valore di interruzione per correlazioni forti.
unimportant_label	*stringa*	
marginal_label	*stringa*	
important_label	*stringa*	
unimportant_below	*number*	Valore di interruzione per importanza di campo bassa. Questo valore deve essere compreso 0 e 1, ad esempio 0.90.
important_above	*number*	
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Nome da utilizzare.
output_mode	Screen File	Specifica la posizione di destinazione per l'output generato dal nodo Output.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Specifica il tipo di output.
full_filename	*stringa*	
output_view	Simple Advanced	Specifica se l'output deve presentare la visualizzazione di base o avanzata.

proprietà reportnode

Il nodo Report crea report formattati che contengono sia testo fisso sia dati e altre espressioni derivate dai dati. Il formato del report viene specificato utilizzando modelli di testo per definire il testo fisso e costruzioni di output dei dati. È possibile fornire una formattazione personalizzata del testo utilizzando tag HTML nel modello e impostando apposite opzioni nella scheda Output. È possibile includere valori di dati e altro output condizionale utilizzando espressioni CLEM nel modello.

Esempio

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

Tabella 234. proprietà reportnode.

Proprietà reportnode	Tipo di dati	Descrizione proprietà
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Utilizzata per specificare il tipo di output del file.
format	Auto Custom	Utilizzata per decidere se l'output viene formattato automaticamente o utilizzando l'HTML incluso nel modello. Per utilizzare la formattazione HTML nel modello, specificare Custom.
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
text	*stringa*	
full_filename	*stringa*	
highlights	*indicatore*	
title	*stringa*	
lines_per_page	*numero*	

Proprietà routputnode

Il nodo Output R consente di analizzare i dati ed i risultati del calcolo del punteggio del modello utilizzando il proprio script R personalizzato. L'output dell'analisi può essere grafico o di testo. L'output viene aggiunto alla scheda **Output** del riquadro dei manager; in alternativa, è possibile reindirizzare l'output in un file.

Tabella 235. Proprietà routputnode

Proprietà routputnode	Tipo di dati	Descrizione proprietà
syntax	*stringa*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*indicatore*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*indicatore*	
output_name	Automatico Custom	
custom_name	*stringa*	
output_to	Screen File	
output_type	Graph Text	

Tabella 235. Proprietà routputnode (Continua)

Proprietà routputnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

proprietà setglobalsnode

Il nodo Calcola globali analizza i dati e calcola i valori di riepilogo che possono essere utilizzati nelle espressioni CLEM. Per esempio, è possibile utilizzare questo nodo per calcolare le statistiche di un campo denominato *età* e utilizzare quindi la media globale dell'*età* nelle espressioni CLEM inserendo la funzione `@GLOBAL_MEAN(età)`.

Esempio

```
node = stream.create("setglobals", "My node")
node.setKeyedPropertyValue("globals", "Na", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "K", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
node.setPropertyValue("clear_first", False)
node.setPropertyValue("show_preview", True)
```

Tabella 236. proprietà setglobalsnode.

Proprietà setglobalsnode	Tipo di dati	Descrizione proprietà
globals	[Sum Mean Min Max SDev]	Proprietà strutturata nella quale per fare riferimento ai campi da impostare è necessario utilizzare la sintassi seguente: `node.setKeyedPropertyValue( "globals", "Age", ["Max", "Sum", "Mean", "SDev"])`
clear_first	*indicatore*	
show_preview	*indicatore*	

Proprietà simevalnode

Il nodo Valutazione della simulazione valuta un campo obiettivo previsto specificato e visualizza le informazioni di distribuzione e correlazione relative al campo obiettivo.

Tabella 237. Proprietà simevalnode.

Proprietà simevalnode	Tipo di dati	Descrizione proprietà
target	*campo*	
iteration	*campo*	
presorted_by_iteration	*booleano*	

Tabella 237. Proprietà simevalnode (Continua).

Proprietà simevalnode	Tipo di dati	Descrizione proprietà
max_iterations	*number*	
tornado_fields	*[campo1...campoN]*	
plot_pdf	*booleano*	
plot_cdf	*booleano*	
show_ref_mean	*booleano*	
show_ref_median	*booleano*	
show_ref_sigma	*booleano*	
num_ref_sigma	*number*	
show_ref_pct	*booleano*	
ref_pct_bottom	*number*	
ref_pct_top	*number*	
show_ref_custom	*booleano*	
ref_custom_values	*[numero1...numeroN]*	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	*booleano*	
pct_table	Quartili Intervals Custom	
pct_intervals_num	*number*	
pct_custom_values	*[numero1...numeroN]*	

Proprietà simfitnode

Il nodo Adattamento simulazione esamina la distribuzione statistica dei dati in ciascun campo e genera (o aggiorna) un nodo Genera simulazione, con la migliore distribuzione di adattamento assegnata a ciascun campo. Il nodo Genera simulazione può essere quindi utilizzato per generare dati simulati.

Tabella 238. Proprietà simfitnode.

Proprietà simfitnode	Tipo di dati	Descrizione proprietà
build	Node XMLExport Both	
use_source_node_name	*booleano*	
source_node_name	*stringa*	Il nome personalizzato del nodo di origine generato o aggiornato.
use_cases	All LimitFirstN	
use_case_limit	*numero intero*	

Tabella 238. Proprietà simfitnode (Continua).

Proprietà simfitnode	Tipo di dati	Descrizione proprietà
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*numero intero*	
parameter_xml_filename	*stringa*	
generate_parameter_import	*booleano*	

proprietà statisticsnode

Il nodo Statistiche fornisce informazioni riassuntive di base su campi numerici. Calcola statistiche riassuntive per singoli campi e per correlazioni tra campi.

Esempio

```
node = stream.create("statistics", "My node")
# "Settings" tab
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
node.setPropertyValue("strong_label", "upper quartile")
# "Output" tab
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Tabella 239. proprietà statisticsnode

Proprietà statisticsnode	Tipo di dati	Descrizione proprietà
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output.
full_filename	*stringa*	
examine	*elenco*	
correlate	*elenco*	

Tabella 239. proprietà statisticsnode (Continua)

Proprietà statisticsnode	Tipo di dati	Descrizione proprietà
statistics	[count mean sum min max range variance sdev semean median mode]	
correlation_mode	Probability Assoluti	Specifica se etichettare le correlazioni per probabilità o valore assoluto.
label_correlations	*indicatore*	
weak_label	*stringa*	
medium_label	*stringa*	
strong_label	*stringa*	
weak_below_probability	*number*	Quando correlation_mode è impostato su Probabilità, specifica il valore di interruzione per correlazioni deboli. Questo valore deve essere compreso 0 e 1, ad esempio 0.90.
strong_above_probability	*number*	Valore di interruzione per correlazioni forti.
weak_below_absolute	*number*	Quando correlation_mode è impostato su Absolute, specifica il valore di interruzione per correlazioni deboli. Questo valore deve essere compreso 0 e 1, ad esempio 0.90.
strong_above_absolute	*number*	Valore di interruzione per correlazioni forti.

Proprietà statisticsoutputnode

Il nodo Output Statistics consente di chiamare una procedura IBM SPSS Statistics per analizzare i dati di IBM SPSS Modeler. È disponibile una vasta gamma di procedure analitiche di IBM SPSS Statistics. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Le proprietà di questo nodo sono descritte in "Proprietà statisticsoutputnode" a pagina 348.

proprietà tablenode

Il nodo Tabella visualizza i dati in formato tabella, che è inoltre possibile scrivere su un file. Questa funzione è utile tutte le volte che si desidera controllare i valori dei dati o esportarli in un formato di facile lettura.

Esempio

```
node = stream.create("table", "My node")
node.setPropertyValue("highlight_expr", "Age > 30")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("transpose_data", True)
node.setPropertyValue("full_filename", "C:/output/table_output.htm")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabella 240. proprietà tablenode.

Proprietà **tablenode**	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Se si tratta di output su disco, di dati o HTML, rappresenta il nome del file di output.
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su true, specifica il nome da utilizzare.
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output.
transpose_data	*indicatore*	Traspone i dati prima dell'esportazione in modo che le righe rappresentino i campi e le colonne rappresentino i record.
paginate_output	*indicatore*	Quando output_format è HTML, l'output viene separato in pagine.
lines_per_page	*number*	Se utilizzato con paginate_output, specifica le righe per pagina di output.
highlight_expr	*stringa*	
output	*stringa*	Proprietà di sola lettura che restituisce un riferimento all'ultima tabella creata dal nodo.
value_labels	*[[Value LabelString] [Value LabelString] ...]*	Utilizzata per specificare etichette per coppie di valori.
display_places	*numero intero*	Imposta il numero di decimali del campo per la visualizzazione (valida solo per campi con archiviazione di tipo Reale). Se viene specificato il valore -1, verrà utilizzata l'impostazione di default del flusso.
export_places	*numero intero*	Imposta il numero di decimali del campo per l'esportazione (valida solo per campi con archiviazione di tipo Reale). Se viene specificato il valore -1, verrà utilizzata l'impostazione di default del flusso.
decimal_separator	DEFAULT PERIOD COMMA	Imposta il separatore decimale per il campo (valido solo per campi con archiviazione di tipo Reale).

Tabella 240. proprietà tablenode (Continua).

Proprietà tablenode	Tipo di dati	Descrizione proprietà
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ss ST AAAA	Imposta il formato di data per il campo (valida solo per campi con archiviazione di tipoDATE o TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM." "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Imposta il formato di ora per il campo (valida solo per campi con archiviazione di tipo TIME o TIMESTAMP).
column_width	*numero intero*	Imposta la larghezza delle colonne per il campo. Se viene specificato il valore -1, la larghezza delle colonne verrà impostata su Auto.
justify	AUTO CENTER LEFT RIGHT	Imposta la giustificazione delle colonne per il campo.

proprietà transformnode

Il nodo Trasformazioni consente di selezionare e visualizzare in anteprima i risultati di trasformazioni prima di applicarli ai campi selezionati.

Esempio

```
node = stream.create("transform", "My node")
node.setPropertyValue("fields", ["AGE", "INCOME"])
node.setPropertyValue("formula", "Select")
node.setPropertyValue("formula_log_n", True)
node.setPropertyValue("formula_log_n_offset", 1)
```

Tabella 241. proprietà transformnode

Proprietà transformnode	Tipo di dati	Descrizione proprietà
fields	[*campo1... campon*]	I campi da utilizzare nella trasformazione.
formula	All Select	Indica se devono essere calcolate tutte le trasformazioni o solo quelle selezionate.
formula_inverse	*indicatore*	Indica se deve essere utilizzata la trasformazione inversa.
formula_inverse_offset	*numero*	Indica l'offset tra i dati da utilizzare per la formula. Se non è specificata dall'utente, è impostata su 0 per default.
formula_log_n	*indicatore*	Indica se deve essere utilizzata la trasformazione log $_n$.
formula_log_n_offset	*numero*	
formula_log_10	*indicatore*	Indica se deve essere utilizzata la trasformazione $\log_{10}$.
formula_log_10_offset	*numero*	
formula_exponential	*indicatore*	Indica se deve essere utilizzata la trasformazione esponenziale (e^x).
formula_square_root	*indicatore*	Indica se deve essere utilizzata la trasformazione radice quadrata.
use_output_name	*indicatore*	Specifica se viene utilizzato un nome di output personalizzato.
output_name	*stringa*	Se use_output_name è impostata su vero, specifica il nome da utilizzare.
output_mode	Screen File	Utilizzata per specificare la posizione di destinazione per l'output generato dal nodo Output.
output_format	HTML (*.html*) Output (*.cou*)	Utilizzata per specificare il tipo di output.
paginate_output	*indicatore*	Quando output_format è HTML, l'output viene separato in pagine.

Tabella 241. proprietà transformnode (Continua)

Proprietà **transformnode**	Tipo di dati	Descrizione proprietà
`lines_per_page`	*number*	Se utilizzato con `paginate_output`, specifica le righe per pagina di output.
`full_filename`	*stringa*	Indica il nome di file da utilizzare per l'output su file.

Capitolo 17. Proprietà dei nodi di esportazione

Proprietà comuni dei nodi di esportazione

Le seguenti proprietà sono valide per tutti i nodi di esportazione:

Tabella 242. Proprietà comuni dei nodi di esportazione

Proprietà	Valori	Descrizione proprietà
publish_path	*stringa*	Specificare il nome di base da utilizzare per i file immagine e dei parametri pubblicati.
publish_metadata	*indicatore*	Specifica se viene generato un file di metadati che descrive gli input e gli output dell'immagine e dei rispettivi modelli di dati.
publish_use_parameters	*indicatore*	Specifica se i parametri del flusso sono contenuti nel file *.par.
publish_parameters	*elenco di stringhe*	Specifica i parametri da includere.
execute_mode	export_data publish	Specifica se il nodo viene eseguito senza pubblicare il flusso o se il flusso viene pubblicato automaticamente quando si esegue il nodo.

Proprietà asexport

L'esportazione di Analytic Server consente di eseguire un flusso su HDFS (Hadoop Distributed File System).

Esempio

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false","","","",""])
```

Tabella 243. Proprietà asexport.

Proprietà asexport	Tipo di dati	Descrizione proprietà
data_source	*stringa*	Il nome dell'origine dati.
export_mode	*stringa*	Specifica se **accodare** i dati esportati all' origine dati esistente o **sovrascrivere** i dati esportati all'origine dati esistente.
use_default_as	*booleano*	Se impostata su True, utilizza la connessione Analytic Server predefinita configurata nel file options.cfg del server. Se impostata su False, utilizza la connessione di questo nodo.

Tabella 243. Proprietà asexport (Continua).

Proprietà asexport	Tipo di dati	Descrizione proprietà
connection	["string","string","string", "string","string","string","string", "string","string","string", "string" ,"string"]	Una proprietà elenco che contiene i dettagli della connessione di Analytic Server. Il formato è: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] dove: is_secure_connect: indica se viene utilizzata la connessione protetta e può essere true o false. use-kerberos-auth: indica se viene utilizzata l'autenticazione kerberos e può essere true o false. enable-kerberos-debug: indica se viene utilizzata la modalità di debug dell'autenticazione kerberos; può essere true o false.

Proprietà del nodo di esportazione Cognos

Il nodo di esportazione IBM Cognos esporta i dati in un formato che può essere letto dai database Cognos.

Per questo nodo è necessario definire una connessione Cognos e una connessione ODBC.

Connessione Cognos

Le proprietà della connessione Cognos sono le seguenti.

Tabella 244. proprietà cognosexportnode

proprietà cognosexportnode	Tipo di dati	Descrizione proprietà
cognos_connection	*["stringa","flag","stringa","stringa","stringa"]*	Una proprietà elenco contenente i dettagli di connessione per il server Cognos. Il formato è: ["Cognos_server_URL", login_mode, "namespace", "username", "password"] dove: Cognos_server_URL è l'URL del server Cognos contenente la sorgente. login_mode indica se viene utilizzato un accesso anonimo e può essere true o false; se impostato su true, i seguenti campi devono essere impostati su "". namespace specifica il provider di protezione per l'autenticazione utilizzato per accedere al server. username e password sono i dati utilizzati per accedere al server Cognos. Invece di login_mode, sono disponibili le seguenti modalità: • anonymousMode. Ad esempio: ['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"] • credentialMode. Ad esempio: ['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]
		• storedCredentialMode. Ad esempio: ['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"] Dove stored_credential_name rappresenta il nome delle credenziali Cognos nel repository.
nome_package_cognos	*stringa*	Il percorso e il nome del package Cognos in cui esportare i dati, per esempio: /Public Folders/MyPackage
cognos_datasource	*stringa*	
cognos_export_mode	Publish ExportFile	
cognos_filename	*stringa*	

Connessione ODBC

Le proprietà della connessione ODBC sono identiche a quelle riportate per databaseexportnode nella sezione che segue, con la differenza che la proprietà datasource non è valida.

proprietà databaseexportnode

Il nodo di esportazione del database scrive dati in una sorgente dati relazionale compatibile con OBDC. Per scrivere in una sorgente dati ODBC, è necessario utilizzare una sorgente dati esistente e disporre dell'autorizzazione in scrittura per tale sorgente.

Esempio

```
'''
Assumes a datasource named "MyDatasource" has been configured
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tab
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialog
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialog
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP INDEX <index-name>
 ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id", "region"]])
```

Tabella 245. proprietà databaseexportnode

proprietà databaseexportnode	Tipo di dati	Descrizione proprietà
datasource	*stringa*	
username	*stringa*	
password	*stringa*	
epassword	*stringa*	Questo slot è di sola lettura durante l'esecuzione. Per generare una password codificata, utilizzare lo strumento Password disponibile dal menu Strumenti. Per ulteriori informazioni, consultare l'argomento “Creazione di una password codificata” a pagina 52.
table_name	*stringa*	

Tabella 245. proprietà databaseexportnode (Continua)

proprietà databaseexportnode	Tipo di dati	Descrizione proprietà
write_mode	Create Append Merge	
map	*stringa*	Mappa il nome campo di un flusso al nome di una colonna di database (valido solo se write_mode è Merge). In caso di unione è necessario che tutti i campi siano mappati per poter essere esportati. I nomi di campi che non esistono nel database vengono aggiunti come nuove colonne.
key_fields	*elenco*	Specifica il campo del flusso utilizzato come chiave; la proprietà map mostra a che cosa corrisponde nel database.
join	Database Add	
drop_existing_table	*indicatore*	
delete_existing_rows	*indicatore*	
default_string_size	*numero intero*	
type		Proprietà strutturata utilizzata per impostare il tipo di schema.
generate_import	*indicatore*	
use_custom_create_table_command	*indicatore*	Utilizzare lo slot *custom_create_table* per modificare il comando SQL standard CREATE TABLE.
custom_create_table_command	*stringa*	Specifica un comando stringa da utilizzare al posto del comando SQL standard CREATE TABLE.
use_batch	*indicatore*	Le seguenti proprietà sono opzioni avanzate per il caricamento di massa di database. Un valore vero (true) per Use_batch disattiva riga per riga i commit al database.
batch_size	*number*	Specificare il numero di record da inviare al database prima del commit nella memoria.
bulk_loading	Off ODBC External	Specifica il tipo di caricamento di massa. Di seguito vengono elencate opzioni aggiuntive per ODBC ed External.
not_logged	*indicatore*	
odbc_binding	Row Column	Specificare l'associazione in base a righe o colonne per il caricamento di massa tramite ODBC.

Tabella 245. proprietà databaseexportnode (Continua)

proprietà databaseexportnode	Tipo di dati	Descrizione proprietà
loader_delimit_mode	Tab Space Other	Specificare il tipo di delimitatore per il caricamento di massa tramite un programma esterno. Selezionare Other insieme alla proprietà loader_other_delimiter per specificare i delimitatori, quali la virgola (,).
loader_other_delimiter	*stringa*	
specify_data_file	*indicatore*	Un flag vero (true) attiva la proprietà data_file illustrata di seguito, che consente di specificare il nome e il percorso del file in cui scrivere durante il caricamento di massa nel database.
data_file	*stringa*	
specify_loader_program	*indicatore*	Un flag vero (true) attiva la proprietà loader_program illustrata di seguito, che consente di specificare il nome e la posizione di un programma o di uno script di caricamento esterno.
loader_program	*stringa*	
gen_logfile	*indicatore*	Un flag vero (true) attiva la proprietà logfile_name illustrata di seguito, che consente di specificare il nome di un file sul server per generare un registro errori.
logfile_name	*stringa*	
check_table_size	*indicatore*	Un flag vero (true) consente il controllo della tabella che assicura che l'incremento nelle dimensioni della tabella di database corrisponda al numero di righe esportate da IBM SPSS Modeler.
loader_options	*stringa*	Specificare argomenti aggiuntivi, quali -comment e -specialdir, al programma di caricamento.
export_db_primarykey	*indicatore*	Specifica se un determinato campo è una chiave primaria.
use_custom_create_index_command	*indicatore*	Se true, (vero), attiva l'SQL personalizzato per tutti gli indici.
custom_create_index_command	*stringa*	Specifica il comando SQL utilizzato per creare gli indici quando è attivato l'SQL personalizzato. (Questo valore può essere sovrascritto per indici specifici come indicato di seguito).
indexes.INDEXNAME.fields		Crea l'indice specificato, se necessario, ed elenca i nomi dei campi da includere in tale indice.

Tabella 245. proprietà databaseexportnode (Continua)

proprietà databaseexportnode	Tipo di dati	Descrizione proprietà
INDEXNAME "use_custom_create_index_command"	*indicatore*	Utilizzato per attivare o disattivare l'SQL personalizzato per un indice specifico. Consultare gli esempi dopo la tabella riportata di seguito.
INDEXNAME "custom_create_index_command"	*stringa*	Specifica l'SQL personalizzato utilizzato per l'indice specificato. Consultare gli esempi dopo la tabella riportata di seguito.
indexes.INDEXNAME.remove	*indicatore*	Se True, rimuove l'indice specificato dall'insieme di indici.
table_space	*stringa*	Specifica lo spazio di tabella che verrà creato.
use_partition	*indicatore*	Indica che verrà utilizzato il campo di hash distribuito.
partition_field	*stringa*	Specifica il contenuto del campo di hash distribuito.

Nota: Per alcuni database, è possibile specificare che le tabelle del database vengano create per l'esportazione con la compressione (ad esempio, l'equivalente di `CREATE TABLE MYTABLE (...) COMPRESS YES;` in SQL). Le proprietà `use_compression` e `compression_mode` vengono fornite per supportare questa funzione, come segue.

Tabella 246. Proprietà databaseexportnode utilizzando le funzioni di compressione

proprietà databaseexportnode	Tipo di dati	Descrizione proprietà
use_compression	*Booleano*	Se impostata su True, questa opzione utilizza la compressione nella creazione delle tabelle per l'esportazione.
compression_mode	Row Page	Imposta il livello di compressione per i database SQL Server.
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	Imposta il livello di compressione per i database Oracle. Si noti che i valori OLTP, Query_High, Query_Low, Archive_High e Archive_Low richiedono come minimo Oracle 11gR2.

Di seguito è riportato un esempio che mostra il modo in cui modificare il comando `CREATE INDEX` per un indice specifico:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["custom_create_index_command",
 "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

In alternativa, questa operazione può essere eseguita mediante una tabella hash:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id", "region"],
 "use_custom_create_index_command":True, "custom_create_index_command":"CREATE INDEX <index-name> ON
 <table-name> <(index-columns)>"])
```

Proprietà datacollectionexportnode

Il nodo Esportazione di Data Collection esegue l'output di dati nel formato utilizzato dal software di ricerche di mercato Data Collection. Un Data Library Data Collection è necessario che sia installato per utilizzare questo nodo.

Esempio

```
stream = modeler.script.stream()
datacollectionexportnode = stream.createAt("datacollectionexport", "Data Collection", 200, 200)
datacollectionexportnode.setPropertyValue("metadata_file", "c:\\museums.mdd")
datacollectionexportnode.setPropertyValue("merge_metadata", "Overwrite")
datacollectionexportnode.setPropertyValue("casedata_file", "c:\\museumdata.sav")
datacollectionexportnode.setPropertyValue("generate_import", True)
datacollectionexportnode.setPropertyValue("enable_system_variables", True)
```

Tabella 247. proprietà datacollectionexportnode

Proprietà **datacollectionexportnode**	Tipo di dati	Descrizione proprietà
metadata_file	*stringa*	Nome del file di metadati da esportare.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*indicatore*	Specifica se il file *.mdd* esportato deve includere le variabili di sistema di Data Collection.
casedata_file	*stringa*	Il nome del file *.sav* in cui vengono esportati i dati del caso.
generate_import	*indicatore*	

Proprietà excelexportnode

Il nodo Esportazione da Excel esegue l'output di dati in formato file Microsoft Excel .xlsx. Se lo si desidera, è possibile scegliere di avviare Excel automaticamente e aprire il file esportato quando si esegue il nodo.

Esempio

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xlsx")
excelexportnode.setPropertyValue("excel_file_type", "Excel2007")
excelexportnode.setPropertyValue("inc_field_names", True)
excelexportnode.setPropertyValue("inc_labels_as_cell_notes", False)
excelexportnode.setPropertyValue("launch_application", True)
excelexportnode.setPropertyValue("generate_import", True)
```

Tabella 248. proprietà excelexportnode

proprietà excelexportnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	
excel_file_type	Excel2007	

Tabella 248. proprietà excelexportnode (Continua)

proprietà excelexportnode	Tipo di dati	Descrizione proprietà
export_mode	Create Append	
inc_field_names	*indicatore*	Specifica se i nomi dei campi devono essere inclusi nella prima riga del foglio di lavoro.
start_cell	*stringa*	Specifica la cella di partenza per l'esportazione.
worksheet_name	*stringa*	Nome del foglio di lavoro da scrivere.
launch_application	*indicatore*	Specifica se sul file risultante deve essere richiamato Excel. Si noti che il percorso per avviare Excel deve essere specificato nella finestra di dialogo Applicazioni di supporto (menu Strumenti, Applicazioni di supporto).
generate_import	*indicatore*	Specifica se deve essere generato un nodo Importazione da Excel che legga il file di dati esportato.

Proprietà extensionexportnode

Con il nodo Esportazione estensione, è possibile eseguire script R o Python for Spark per esportare i dati.

Esempio Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_export", "extension_export")
node.setPropertyValue("syntax_type", "Python")

python_script = """import spss.pyspark.runtime
from pyspark.sql import SQLContext
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
print df.dtypes[:]
_newDF = df.select("Age","Drug")
print _newDF.dtypes[:]

df.select("Age", "Drug").write.save("c:/data/ageAndDrug.json", format="json")
"""

node.setPropertyValue("python_syntax", python_script)
```

Esempio R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

Tabella 249. Proprietà extensionexportnode

Proprietà extensionexportnode	Tipo di dati	Descrizione proprietà
syntax_type	*R* *Python*	Specifica quale script viene eseguito, R o Python (R è il valore predefinito).
r_syntax	*stringa*	La sintassi di script R da eseguire.
python_syntax	*stringa*	La sintassi di script Python da eseguire.
convert_flags	StringsAndDoubles LogicalValues	Opzione per la conversione dei campi indicatore.
convert_missing	*indicatore*	Opzione per convertire i valore mancanti nel valore R NA.
convert_datetime	*indicatore*	L'opzione per convertire le variabili con formati di date o data/ora in formati data/ora R.
convert_datetime_class	POSIXct POSIXlt	Le opzioni per specificare in quale formato vengono convertite le variabili con formati data o data/ora.

Proprietà jsonexportnode

Il nodo di esportazione JSON emette i dati di output in formato JSON. Consultare Nodo di esportazione JSON per ulteriori informazioni.

Tabella 250. proprietà jsonexportnode

proprietà jsonexportnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Il nome del file completo compreso il percorso.
string_format	*records* *values*	Specificare il formato della stringa JSON. L'impostazione predefinita è records.
generate_import	*indicatore*	Specifica se deve essere generato un nodo di importazione JSON che leggerà il file di dati esportato. Il valore di default è False.

Proprietà outputfilenode

Il nodo di esportazione File flat restituisce dati in un file di testo delimitato. È utile per esportare i dati che possono essere letti da altri software di analisi o fogli di calcolo.

Esempio

```
stream = modeler.script.stream()
outputfile = stream.createAt("outputfile", "File Output", 200, 200)
outputfile.setPropertyValue("full_filename", "c:/output/flatfile_output.txt")
outputfile.setPropertyValue("write_mode", "Append")
outputfile.setPropertyValue("inc_field_names", False)
outputfile.setPropertyValue("use_newline_after_records", False)
outputfile.setPropertyValue("delimit_mode", "Tab")
outputfile.setPropertyValue("other_delimiter", ",")
outputfile.setPropertyValue("quote_mode", "Double")
outputfile.setPropertyValue("other_quote", "*")
outputfile.setPropertyValue("decimal_symbol", "Period")
outputfile.setPropertyValue("generate_import", True)
```

Tabella 251. proprietà outputfilenode

Proprietà **outputfilenode**	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Nome del file di output.
write_mode	Overwrite Append	
inc_field_names	*indicatore*	
use_newline_after_records	*indicatore*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*car*	
quote_mode	None Single Double Other	
other_quote	*indicatore*	
generate_import	*indicatore*	
encoding	StreamDefault SystemDefault "UTF-8"	

Proprietà sasexportnode

Il nodo Esporta SAS restituisce nel formato SAS i dati che devono essere letti in SAS o in un pacchetto software compatibile con SAS. Sono disponibili tre formati di file SAS: SAS per Windows/OS2, SAS per UNIX o SAS Versione 7/8.

Esempio

```
stream = modeler.script.stream()
sasexportnode = stream.createAt("sasexport", "SAS Export", 200, 200)
sasexportnode.setPropertyValue("full_filename", "c:/output/SAS_output.sas7bdat")
sasexportnode.setPropertyValue("format", "SAS8")
sasexportnode.setPropertyValue("export_names", "NamesAndLabels")
sasexportnode.setPropertyValue("generate_import", True)
```

Tabella 252. proprietà sasexportnode

proprietà **sasexportnode**	Tipo di dati	Descrizione proprietà
format	Windows UNIX SAS7 SAS8	Campi delle etichette delle proprietà Variant.
full_filename	*stringa*	
export_names	NamesAndLabels NamesAsLabels	Utilizzata per mappare nomi di campi IBM SPSS Modeler a nomi di variabili IBM SPSS Statistics o SAS durante l'esportazione.
generate_import	*indicatore*	

Proprietà statisticsexportnode

Il nodo Esporta Statistics restituisce i dati in formato IBM SPSS Statistics *.sav* o *.zsav*. È possibile leggere i file .sav o .zsav utilizzando IBM SPSS Statistics Base ed altri prodotti. Questo formato viene inoltre utilizzato per i file cache di IBM SPSS Modeler.

Le proprietà di questo nodo sono descritte in "Proprietà statisticsexportnode" a pagina 349.

Proprietà del nodo tm1odataexport

Il nodo di esportazione IBM Cognos TM1 esporta i dati in un formato che può essere letto dai database Cognos TM1.

Tabella 253. Proprietà del nodo tm1odataexport

Proprietà del nodo **tm1odataexport**	Tipo di dati	Descrizione proprietà
admin_host	*stringa*	L'URL per il nome host dell'API REST.
server_name	*stringa*	Il nome del server TM1 selezionato da admin_host.
credential_type	inputCredential o storedCredential	Utilizzata per indicare il tipo di credenziale.
input_credential	*elenco*	Quando credential_type è inputCredential; specificare il dominio, il nome utente e la password.

Tabella 253. Proprietà del nodo tm1odataexport (Continua)

Proprietà del nodo tm1odataexport	Tipo di dati	Descrizione proprietà
stored_credential_name	*stringa*	Quando credential_type è storedCredential; specificare il nome della credenziale sul server C&DS.
selected_cube	*campo*	Il nome del cubo in cui si stanno esportando i dati. Ad esempio: TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")
spss_field_to_tm1_element_mapping	*elenco*	L'elemento tm1 a cui eseguire il mapping deve essere parte della dimensione colonna per la vista cubo selezionata. Il formato è: [[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]] Sono presenti 2 elenchi per descrivere le informazioni di mapping. Il mapping di un elemento foglia ad una dimensione corrisponde all'esempio 2 riportato di seguito: Esempio 1: il primo elenco: ([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) è utilizzato per le informazioni di associazione della dimensione TM1. Ogni elenco di 3 valori indica le informazioni sul mapping della dimensione. Il terzo valore booleano viene utilizzato per indicare se seleziona un elemento di una dimensione. Ad esempio: "[Field_1, Dimension_1, False]" significa che Field_1 è associato a Dimension_1; "[Element_1, Dimension_2, True]" significa che Element_1 è selezionato per Dimension_2. Esempio 2: il secondo elenco: ([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) viene utilizzato per le informazioni sulla mappa dell'elemento dimensione misure TM1. Ciascun elenco di 3 valori indica informazioni di associazione dell'elemento della misura. Il terzo valore booleano viene utilizzato per indicare che è necessario creare un nuovo elemento. "[Field_2, ExistMeasureElement, False]" significa che Field_2 è associato a ExistMeasureElement; "[Field_3, NewMeasureElement, True]" significa che NewMeasureElement deve essere la dimensione misura scelta in selected_measure e che Field_3 è associato a tale elemento.
selected_measure	*stringa*	Specifica la dimensione della misura. Esempio: setPropertyValue("selected_measure", "Measures")

Proprietà del nodo tm1export (obsoleto)

Il nodo di esportazione IBM Cognos TM1 esporta i dati in un formato che può essere letto dai database Cognos TM1.

Nota: Questo nodo è stato dichiarato obsoleto in Modeler 18.0. Il nome dello script del nodo sostitutivo è *tm1odataexport*.

Tabella 254. Proprietà del nodo tm1export.

Proprietà del nodo tm1export	Tipo di dati	Descrizione proprietà
pm_host	*stringa*	**Nota:** Solo per le versioni 16.0 e 17.0 Il nome host. Ad esempio: `TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*["campo","campo", ... ,"campo"]*	**Nota:** Solo per le versioni 16.0 e 17.0 Una proprietà elenco che contiene i dettagli di connessione per il server TM1. Il formato è: `[ "TM1_Server_Name", "tm1_ username", "tm1_password"]` Ad esempio: `TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])`
selected_cube	*campo*	Il nome del cubo in cui si stanno esportando i dati. Ad esempio: `TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")`

Tabella 254. Proprietà del nodo tm1export (Continua).

Proprietà del nodo **tm1export**	Tipo di dati	Descrizione proprietà
`spssfield_tm1element_mapping`	*elenco*	L'elemento tm1 a cui eseguire il mapping deve essere parte della dimensione colonna per la vista cubo selezionata. Il formato è: `[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]` Sono presenti 2 elenchi per descrivere le informazioni di mapping. Il mapping di un elemento foglia ad una dimensione corrisponde all'esempio 2 riportato di seguito: Esempio 1: il primo elenco: `([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...])` è utilizzato per le informazioni di associazione della dimensione TM1. Ogni elenco di 3 valori indica le informazioni sul mapping della dimensione. Il terzo valore booleano viene utilizzato per indicare se seleziona un elemento di una dimensione. Ad esempio: `"[Field_1, Dimension_1, False]"` significa che `Field_1` è associato a `Dimension_1`; `"[Element_1, Dimension_2, True]"` significa che `Element_1` è selezionato per `Dimension_2`. Esempio 2: il secondo elenco: `([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...])` viene utilizzato per le informazioni sulla mappa dell'elemento dimensione misure TM1. Ciascun elenco di 3 valori indica informazioni di associazione dell'elemento della misura. Il terzo valore booleano viene utilizzato per indicare che è necessario creare un nuovo elemento. `"[Field_2, ExistMeasureElement, False]"` significa che `Field_2` è associato a `ExistMeasureElement`; `"[Field_3, NewMeasureElement, True]"` significa che `NewMeasureElement` deve essere la dimensione misura scelta in `selected_measure` e che `Field_3` è associato a tale elemento.
`selected_measure`	*stringa*	Specifica la dimensione della misura. Esempio: `setPropertyValue("selected_measure", "Measures")`

Proprietà xmlexportnode

Il nodo Esporta XML restituisce i dati in un file in formato XML. Se lo si desidera, è possibile creare un nodo origine XML per leggere nuovamente i dati esportati nello stream.

Esempio

```
stream = modeler.script.stream()
xmlexportnode = stream.createAt("xmlexport", "XML Export", 200, 200)
xmlexportnode.setPropertyValue("full_filename", "c:/export/data.xml")
xmlexportnode.setPropertyValue("map", [["/catalog/book/genre", "genre"], ["/catalog/book/title", "title"]])
```

Tabella 255. proprietà xmlexportnode

proprietà xmlexportnode	**Tipo di dati**	**Descrizione proprietà**
full_filename	*stringa*	(obbligatorio) Percorso e nome file completi del file di esportazione XML.
use_xml_schema	*indicatore*	Specifica se utilizzare uno schema XML (file XSD o DTD) per controllare la struttura dei dati esportati.
full_schema_filename	*stringa*	Percorso e nome file completi del file XSD o DTD da utilizzare. Necessario solo se use_xml_schema è impostato su true (vero).
generate_import	*indicatore*	Genera un nodo origine XML che rileggerà il file di dati esportato nel flusso.
records	*stringa*	Espressione XPath che indica i limiti dei record.
map	*stringa*	Mappa i nomi dei campi alla struttura XML.

Capitolo 18. Proprietà dei nodi IBM SPSS Statistics

Proprietà statisticsimportnode

Il nodo File legge i dati dal file in formato *.sav* o *.zsav* utilizzato da IBM SPSS Statistics e dai file della cache salvati in IBM SPSS Modeler, che utilizza lo stesso formato.

Esempio

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import", 200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drug1n.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

Tabella 256. proprietà statisticsimportnode

Proprietà statisticsimportnode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Il nome del file completo compreso il percorso.
password	*stringa*	La password. Il parametro password deve essere impostato prima del parametro file_encrypted.
file_encrypted	*indicatore*	Indica se il file è protetto o meno da password.
import_names	NamesAndLabels LabelsAsNames	Metodo per gestire nomi ed etichette di variabili.
import_data	DataAndLabels LabelsAsData	Metodo per gestire valori ed etichette.
use_field_format_for_storage	*Booleano*	Specifica se utilizzare le informazioni relative al formato dei campi di IBM SPSS Statistics durante le importazioni.

proprietà statisticstransformnode

Il nodo Trasformazioni Statistics esegue una selezione di comandi di sintassi IBM SPSS Statistics rispetto alle sorgenti dati in IBM SPSS Modeler. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Esempio

```
stream = modeler.script.stream()
statisticstransformnode = stream.createAt("statisticstransform", "Transform", 200, 200)
statisticstransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticstransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
statisticstransformnode.setPropertyValue("check_before_saving", True)
```

Tabella 257. proprietà statisticstransformnode

Proprietà statisticstransformnode	Tipo di dati	Descrizione proprietà
syntax	*stringa*	
check_before_saving	*indicatore*	Convalida la sintassi inserita prima di salvare le voci. Visualizza un messaggio di errore se la sintassi non è valida.
default_include	*indicatore*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.
include	*indicatore*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.
new_name	*stringa*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.

proprietà statisticsmodelnode

Il nodo Modello Statistics consente di analizzare e operare con i dati eseguendo le procedure IBM SPSS Statistics che generano PMML. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Esempio

```
stream = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
```

Proprietà statisticsmodelnode	Tipo di dati	Descrizione proprietà
syntax	*stringa*	
default_include	*indicatore*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.
include	*indicatore*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.
new_name	*stringa*	Per ulteriori informazioni, consultare l'argomento “proprietà filternode” a pagina 144.

Proprietà statisticsoutputnode

Il nodo Output Statistics consente di chiamare una procedura IBM SPSS Statistics per analizzare i dati di IBM SPSS Modeler. È disponibile una vasta gamma di procedure analitiche di IBM SPSS Statistics. Questo nodo richiede una copia di IBM SPSS Statistics con regolare licenza.

Esempio

```
stream = modeler.script.stream()
statisticsoutputnode = stream.createAt("statisticsoutput", "Output", 200, 200)
statisticsoutputnode.setPropertyValue("syntax", "SORT CASES BY Age(A) Sex(A) BP(A) Cholesterol(A)")
statisticsoutputnode.setPropertyValue("use_output_name", False)
statisticsoutputnode.setPropertyValue("output_mode", "File")
statisticsoutputnode.setPropertyValue("full_filename", "Cases by Age, Sex and Medical History")
statisticsoutputnode.setPropertyValue("file_type", "HTML")
```

Tabella 258. proprietà statisticsoutputnode

Proprietà **statisticsoutputnode**	Tipo di dati	Descrizione proprietà
mode	Dialog Syntax	Selezionare l'opzione "Finestra di dialogo IBM SPSS Statistics" o Editor di sintassi
syntax	*stringa*	
use_output_name	*indicatore*	
output_name	*stringa*	
output_mode	Screen File	
full_filename	*stringa*	
file_type	HTML SPV SPW	

Proprietà statisticsexportnode

Il nodo Esporta Statistics restituisce i dati in formato IBM SPSS Statistics *.sav* o *.zsav*. È possibile leggere i file `.sav` o `.zsav` utilizzando IBM SPSS Statistics Base ed altri prodotti. Questo formato viene inoltre utilizzato per i file cache di IBM SPSS Modeler.

Esempio

```
stream = modeler.script.stream()
statisticsexportnode = stream.createAt("statisticsexport", "Export", 200, 200)
statisticsexportnode.setPropertyValue("full_filename", "c:/output/SPSS_Statistics_out.sav")
statisticsexportnode.setPropertyValue("field_names", "Names")
statisticsexportnode.setPropertyValue("launch_application", True)
statisticsexportnode.setPropertyValue("generate_import", True)
```

Tabella 259. Proprietà statisticsexportnode

Proprietà **statisticsexportnode**	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	
file_type	sav zsav	Salva il file in formato *sav* o *zsav*. Ad esempio: `statisticsexportnode.setPropertyValue("file_type","sav")`
encrypt_file	*indicatore*	Indica se il file è protetto o meno da password.
password	*stringa*	La password.

Tabella 259. Proprietà statisticsexportnode (Continua)

Proprietà statisticsexportnode	**Tipo di dati**	**Descrizione proprietà**
launch_application	*indicatore*	
export_names	NamesAndLabels NamesAsLabels	Utilizzata per mappare nomi di campi IBM SPSS Modeler a nomi di variabili IBM SPSS Statistics o SAS durante l'esportazione.
generate_import	*indicatore*	

Capitolo 19. Proprietà del nodo Python

Proprietà gmm

Un modello Gaussian Mixture© è un modello probabilistico che presuppone che tutti i punti dati siano generati da una miscela di un numero finito di distribuzioni gaussiane con parametri non noti. Si può pensare ai modelli a miscela come generalizzazioni di raggruppamenti in cluster di K-medie per incorporare le informazioni relative alla struttura di covarianza dei dati come centri di valori gaussiani latenti. Il nodo Gaussian Mixture in SPSS Modeler espone le funzioni principali e i parametri comunemente utilizzati della libreria Gaussian Mixture. Il nodo è implementato in Python.

Tabella 260. Proprietà gmm

Proprietà gmm	Tipo di dati	Descrizione proprietà
role_use	*booleano*	Specificare False per utilizzare i ruoli predefiniti o True per utilizzare assegnazioni campo personalizzate. Il valore di default è False.
predictors	*campo*	
use_partition	*booleano*	Impostare su True o False per specificare se utilizzare dati partizionati. Il valore di default è False.
covariance_type	*stringa*	Specificare Full, Tied, Diag, o Spherical per impostare il tipo di covarianza.
number_component	*numero intero*	Specificare un numero intero per il numero di componenti Mixture. Il valore minimo è 1. Il valore predefinito è 2.
component_lable	*booleano*	Specificare True per impostare l'etichetta cluster su una stringa o False per impostare l'etichetta cluster su un numero. Il valore di default è False.
label_prefix	*stringa*	Se si utilizza un'etichetta cluster stringa, è possibile specificare un prefisso.
enable_random_seed	*booleano*	Specificare True se si desidera utilizzare un seed casuale. Il valore di default è False.
random_seed	*numero intero*	Se si utilizza un seed casuale, specificare un numero intero da utilizzare per generare i campioni casuali.
tol	*Double*	Specificare la soglia di convergenza Il valore predefinito è 0.000.1.
max_iter	*numero intero*	Specificare il numero massimo di iterazioni da eseguire. Il valore predefinito è 100.
init_params	*stringa*	Impostare il parametro di inizializzazione da utilizzare. Le opzioni sono Kmeans o Random.
warm_start	*booleano*	Specificare True per utilizzare la soluzione dell'ultimo adattamento come inizializzazione per la successiva chiamata di adattamento. Il valore di default è False.

Proprietà hdbscannode

HDBSCAN© (Hierarchical Density-Based Spatial Clustering) utilizza un apprendimento non supervisionato per trovare i cluster, o dense regioni, di un dataset. Il nodo HDBSCAN in SPSS Modeler riporta le funzioni principali ed i parametri comunemente utilizzati di una libreria HDBSCAN. l nodo viene implementato in Python ed è possibile utilizzarlo per raggruppare i dataset in gruppi distinti quando non si è in grado di definire all'inizio le caratteristiche di tali gruppi.

Tabella 261. proprietà hdbscannode

proprietà `hdbscannode`	Tipo di dati	Descrizione proprietà
`inputs`	*campo*	Campi di input per il raggruppamento tramite cluster.
`useHPO`	*booleano*	Specificare `true` o `false` per abilitare o disabilitare l'ottimizzazione degli iperparametri (Hyper-Parameter Optimization - HPO) basata su Rbfopt, che rileva automaticamente la combinazione ottimale di parametri in modo che il modello potrà raggiungere il tasso di errore previsto o più basso sui campioni. Il valore predefinito è `false`.
`min_cluster_size`	*numero intero*	La dimensione minima dei cluster. Specificare un numero intero. Il valore predefinito è `5`.
`min_samples`	*numero intero*	Il numero minimo di campioni in una risorsa per un punto da considerare un punto centrale. Specificare un intero. Se impostato su `0`, viene utilizzato `min_cluster_size`. Il valore predefinito è `0`.
`algorithm`	*stringa*	Specificare l'algoritmo da utilizzare: `best`, `generic`, `prims_kdtree`, `prims_balltree`, `boruvka_kdtree`, o `boruvka_balltree`. L'impostazione di default è `best`.
`metric`	*stringa*	Specificare la metrica da utilizzare nel calcolo della distanza tra le istanze in un array di funzioni: `euclidean`, `cityblock`, `L1`, `L2`, `manhattan`, `braycurtis`, `canberra`, `chebyshev`, `correlation`, `minkowski`, o `sqeuclidean`. L'impostazione di default è `euclidean`.
`useStringLabel`	*booleano*	Specificare `true` per utilizzare un'etichetta del cluster di stringhe oppure `false` per utilizzare un'etichetta cluster di numeri. Il valore predefinito è `false`.
`stringLabelPrefix`	*stringa*	Se il parametro `useStringLabel` è impostato su `true`, specificare un valore per il prefisso dell'etichetta della stringa. Il valore predefinito è `cluster`.
`approx_min_span_tree`	*booleano*	Specificare `true` per accettare una struttura ad albero minima approssimata o `false` se si privilegia la correttezza respetto alla velocità. Il valore predefinito è `true`.
`cluster_selection_method`	*stringa*	Specificare il metodo da utilizzare per la selezione dei cluster dalla struttura ad albero condensata: `eom` o `leaf`. Il valore predefinito è `eom` (algoritmo Excess of Mass).

Tabella 261. proprietà hdbscannode (Continua)

proprietà hdbscannode	Tipo di dati	Descrizione proprietà
allow_single_cluster	*booleano*	Specifcare true se si desidera consentire i risultati di un singolo cluster. Il valore predefinito è false.
p_value	*double*	Specificare il p value da utilizzare se si sta utilizzando minkowski per la metrica. Il valore predefinito è 1.5.
leaf_size	*numero intero*	Se si utilizza un algoritmo di struttura ad albero dello spazio (boruvka_kdtree, o boruvka_balltree), specificare il numero di punti in un nodo foglia della struttura ad albero. IL valore predefinito è 40.
outputValidity	*booleano*	Specificare true o false per controllare se il grafico dell'Indice di validità è incluso nell'output del modello.
outputCondensed	*booleano*	Specificare true o false per controllare se il grafico della Struttura ad albero condensata è incluso nell'output del modello.
outputSingleLinkage	*booleano*	Specificare true o false tper controllare se il grafico della Struttura ad albero collegamento singolo è incluso nell'output del modello.
outputMinSpan	*booleano*	Specificare true o false per controllare se il grafico della Struttura ad albero minima è incluso nell'output del modello.

Proprietà kdemodel

KDE (Kernel Density Estimation)© utilizza gli algoritmi Ball Tree o KD Tree per query efficienti e combina i concetti di apprendimento non supervisionato, ingegneria della funzione e modellazione dei dati. Gli approcci basati su Neighbor come ad esempio KDE rappresentano alcune tra le più popolari e utili tecniche di stima della densità. I nodi di modellazione KDE e di simulazione KDE in SPSS Modeler presentano le funzioni principali e i parametri comunemente utilizzati della libreria KDE. I nodi sono implementati in Python.

Tabella 262. Proprietà kdemodel

Proprietà kdemodel	Tipo di dati	Descrizione proprietà
inputs	*campo*	Campi di input per il raggruppamento tramite cluster.
bandwidth	*double*	Il valore predefinito è 1.
kernel	*stringa*	Il kernel da utilizzare: gaussian, tophat, epanechnikov, exponential, linear, o cosine. L'impostazione predefinita è gaussian.
algorithm	*stringa*	L'algoritmo della struttura ad albero da utilizzare: kd_tree, ball_tree, o auto. Il valore predefinito è auto.

Tabella 262. Proprietà kdemodel (Continua)

Proprietà kdemodel	Tipo di dati	Descrizione proprietà
metric	*stringa*	La metrica da utilizzare quando si calcola la distanza. Per l'algoritmo kd_tree, scegliere tra: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2, o L1. Per l'algoritmo ball_tree, scegliere tra: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath, o Kulsinski. L'impostazione predefinita è Euclidean.
atol	*float*	La tolleranza assoluta del risultato. Una tolleranza superiore generalmente dà luogo ad un'esecuzione più veloce. L'impostazione predefinita è 0.0.
rtol	*float*	La tolleranza relativa desiderata del risultato. Una tolleranza superiore dà luogo ad un'esecuzione più veloce. Il valore predefinito è 1E-8.
breadthFirst	*booleano*	Impostare su True per utilizzare un approccio breadth-first. Impostare su False per utilizzare un approccio depth-first. L'impostazione predefinita è True.
LeafSize	*numero intero*	La dimensione dell'elemento foglia della struttura ad albero sottostante. IL valore predefinito è 40. La modifica di questo valore può impattare significativamente sulle prestazioni.
pValue	*double*	Specificare il valore P da utilizzare se si sta utilizzando Minkowski per la metrica. Il valore predefinito è 1.5.

Proprietà kde

KDE (Kernel Density Estimation)© utilizza gli algoritmi Ball Tree o KD Tree per query efficienti e combina i concetti di apprendimento non supervisionato, ingegneria della funzione e modellazione dei dati. Gli approcci basati su Neighbor come ad esempio KDE rappresentano alcune tra le più popolari e utili tecniche di stima della densità. I nodi di modellazione KDE e di simulazione KDE in SPSS Modeler presentano le funzioni principali e i parametri comunemente utilizzati della libreria KDE. I nodi sono implementati in Python.

Tabella 263. Proprietà kde

Proprietà kde	Tipo di dati	Descrizione proprietà
inputs	*campo*	Campi di input per il raggruppamento tramite cluster.
bandwidth	*double*	Il valore predefinito è 1.
kernel	*stringa*	Il kernel da utilizzare: gaussian o tophat. L'impostazione predefinita è gaussian.
algorithm	*stringa*	L'algoritmo della struttura ad albero da utilizzare: kd_tree, ball_tree, o auto. Il valore predefinito è auto.

Tabella 263. Proprietà kde (Continua)

Proprietà kde	Tipo di dati	Descrizione proprietà
metric	*stringa*	La metrica da utilizzare quando si calcola la distanza. Per l'algoritmo kd_tree scegliere tra: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2, o L1. Per l'algoritmo ball_tree, scegliere tra: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath, o Kulsinski. L'impostazione predefinita è Euclidean.
atol	*float*	La tolleranza assoluta desiderata. Una tolleranza superiore generalmente dà luogo ad un'esecuzione più veloce. L'impostazione predefinita è 0.0.
rtol	*float*	La tolleranza relativa desiderata del risultato. Una tolleranza superiore dà luogo generalmente ad un'esecuzione più veloce. Il valore predefinito è 1E-8.
breadthFirst	*booleano*	Impostare su True per utilizzare un approccio breadth-first. Impostare su False per utilizzare un approccio depth-first. L'impostazione predefinita è True.
LeafSize	*numero intero*	La dimensione dell'elemento foglia della struttura ad albero sottostante. IL valore predefinito è 40. La modifica di questo valore può impattare significativamente sulle prestazioni.
pValue	*double*	Specificare il valore P da utilizzare se si sta utilizzando Minkowski per la metrica. Il valore predefinito è 1.5.

Proprietà gmm

Un modello Gaussian Mixture© è un modello probabilistico che presuppone che tutti i punti dati siano generati da una miscela di un numero finito di distribuzioni gaussiane con parametri non noti. Si può pensare ai modelli a miscela come generalizzazioni di raggruppamenti in cluster di K-medie per incorporare le informazioni relative alla struttura di covarianza dei dati come centri di valori gaussiani latenti. Il nodo Gaussian Mixture in SPSS Modeler espone le funzioni principali e i parametri comunemente utilizzati della libreria Gaussian Mixture. Il nodo è implementato in Python.

Tabella 264. Proprietà gmm

Proprietà gmm	Tipo di dati	Descrizione proprietà
role_use	*booleano*	Specificare False per utilizzare i ruoli predefiniti o True per utilizzare assegnazioni campo personalizzate. Il valore di default è False.
predictors	*campo*	
use_partition	*booleano*	Impostare su True o False per specificare se utilizzare dati partizionati. Il valore di default è False.

Tabella 264. Proprietà gmm (Continua)

Proprietà gmm	Tipo di dati	Descrizione proprietà
covariance_type	*stringa*	Specificare Full, Tied, Diag, o Spherical per impostare il tipo di covarianza.
number_component	*numero intero*	Specificare un numero intero per il numero di componenti Mixture. Il valore minimo è 1. Il valore predefinito è 2.
component_lable	*booleano*	Specificare True per impostare l'etichetta cluster su una stringa o False per impostare l'etichetta cluster su un numero. Il valore di default è False.
label_prefix	*stringa*	Se si utilizza un'etichetta cluster stringa, è possibile specificare un prefisso.
enable_random_seed	*booleano*	Specificare True se si desidera utilizzare un seed casuale. Il valore di default è False.
random_seed	*numero intero*	Se si utilizza un seed casuale, specificare un numero intero da utilizzare per generare i campioni casuali.
tol	*Double*	Specificare la soglia di convergenza Il valore predefinito è 0.000.1.
max_iter	*numero intero*	Specificare il numero massimo di iterazioni da eseguire. Il valore predefinito è 100.
init_params	*stringa*	Impostare il parametro di inizializzazione da utilizzare. Le opzioni sono Kmeans o Random.
warm_start	*booleano*	Specificare True per utilizzare la soluzione dell'ultimo adattamento come inizializzazione per la successiva chiamata di adattamento. Il valore di default è False.

Proprietà di ocsvmnode

Il nodo SVM a una classe utilizza un algoritmo di apprendimento non supervisionato. Il nodo può essere utilizzato per il rilevamento delle novità. Rileverà il limite soft di un insieme dato di esempi, per classificare i nuovi punti come appartenenti o meno all'insieme. Questo nodo SVM a una classe in SPSS Modeler viene implementato in Python e richiede la libreria Python scikit-learn©.

Tabella 265. Proprietà di ocsvmnode

Proprietà di ocsvmnode	Tipo di dati	Descrizione proprietà
role_use	*stringa*	Specificare predefined per utilizzare i ruoli predefiniti oppure custom per utilizzare le assegnazioni di campo personalizzate. Il valore predefinito è predefined.
inputs	*campo*	Elenco dei nomi dei campi per l'input.
splits	*campo*	Elenco dei nomi dei campi per la suddivisione.
use_partition	*Booleano*	Specificare true o false. Il valore predefinito è true. Se questa opzione è impostata su true, durante la creazione del modello verranno utilizzati solo i dati di addestramento.

Tabella 265. Proprietà di ocsvmnode (Continua)

Proprietà di ocsvmnode	Tipo di dati	Descrizione proprietà
mode_type	*stringa*	La modalità. I valori possibile sono simple o expert. Se si specifica simple, tutti i parametri nella scheda Livello avanzato verranno disabilitati.
stopping_criteria	*stringa*	Una stringa di notazione scientifica. I valori possibili sono 1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5 o 1.0E-6. Il valore predefinito è 1.0E-3.
precision	*float*	La precisione di regressione (nu). Limitata ad una frazione degli errori di addestramento e dei vettori di supporto. Specificare un numero maggiore di 0 e minore o uguale a 1.0. Il valore predefinito è 0,1.
kernel	*stringa*	Il tipo di kernel da utilizzare nell'algoritmo. I valori possibili sono linear, poly, rbf, sigmoid o precomputed. Il valore predefinito è rbf.
enable_gamma	*Booleano*	Abilita il parametro gamma. Specificare true o false. Il valore predefinito è true.
gamma	*float*	Questo parametro viene abilitato solo per i kernel rbf, poly e sigmoid. Se il parametro enable_gamma è impostato su false, questo parametro sarà impostato su auto. Se è impostato su true, il valore predefinito è 0,1.
coef0	*float*	Termine indipendente nella funzione kernel. Questo parametro è abilitato solo per il kernel poly ed il kernel sigmoid. Il valore predefinito è 0.0.
degree	*numero intero*	Grado della funzione kernel polinomiale. Questo parametro è abilitato solo per il kernel poly. Specificare qualsiasi numero intero. Il valore predefinito è 3.
shrinking	*Booleano*	Specifica se utilizzare l'opzione euristica di riduzione. Specificare true o false. Il valore di default è false.
enable_cache_size	*Booleano*	Abilita il parametro cache_size. Specificare true o false. Il valore di default è false.
cache_size	*float*	La dimensione della cache del kernel in MB. Il valore predefinito è 200.
enable_random_seed	*Booleano*	Abilita il parametro random_seed. Specificare true o false. Il valore di default è false.
random_seed	*numero intero*	Il seed random da utilizzare durante il mescolamento dei dati per la stima della probabilità. Specificare qualsiasi numero intero.
pc_type	*stringa*	Il tipo di grafico delle coordinate parallele. Le opzioni possibili sono independent o general.
lines_amount	*numero intero*	Il numero massimo di righe da includere nel grafico. Specificare un valore intero compreso tra 1 e 1000.

Tabella 265. Proprietà di ocsvmnode (Continua)

Proprietà di ocsvmnode	Tipo di dati	Descrizione proprietà
lines_fields_custom	*Booleano*	Abilita il parametro lines_fields, che consente di specificare i campi personalizzati da mostrare nell'output del grafico. Se è impostato su false, verranno visualizzati tutti i campi. Se è impostato su true, verranno visualizzati solo i campi specificati con il parametro lines_fields. Per motivi relativi alle prestazioni, verrà visualizzato un massimo di 20 campi.
lines_fields	*campo*	Elenco dei nomi di campo da includere nel grafico come asse verticale.
enable_graphic	*Booleano*	Specificare true o false. Abilita l'output grafico (disabilitare questa opzione se si desidera risparmiare tempo e ridurre la dimensione del file di stream).
enable_hpo	*Booleano*	Specificare true o false per abilitare o disabilitare le opzioni HPO. Se impostato su true, Rbfopt verrà applicato per ricercare automaticamente il "miglior" modello One-Class SVM che raggiunge il valore obiettivo definito dall'utente con il seguente parametro target_objval.
target_objval	*float*	Il valore funzione obiettivo (tasso di errore del modello negli esempi) che si desidera raggiungere (ad esempio il valore ottimale sconosciuto). Impostare questo parametro sul valore appropriato se non si conosce il valore ottimale (ad esempio 0.01).
max_iterations	*numero intero*	Numero massimo di iterazioni per tentare il modello. Il valore predefinito è 1000.
max_evaluations	*numero intero*	Numero massimo di valutazioni della funzione per tentare il modello, in cui l'attenzione è incentrata sulla precisione rispetto alla velocità. Il valore predefinito è 300.

Proprietà rfnode

Il nodo Random Forest utilizza un'implementazione avanzata di un algoritmo Bagging con un modello di struttura ad albero come modello base. Questo nodo di modellazione Random Forest in SPSS Modeler è implementato in Python e richiede la libreria Python scikit-learn©.

Tabella 266. Proprietà rfnode

Proprietà rfnode	Tipo di dati	Descrizione proprietà
role_use	*stringa*	Specificare predefined per utilizzare i ruoli predefiniti oppure custom per utilizzare le assegnazioni di campo personalizzate. Il valore predefinito è predefined.
inputs	*campo*	Elenco dei nomi dei campi per l'input.

Tabella 266. Proprietà rfnode (Continua)

Proprietà rfnode	Tipo di dati	Descrizione proprietà
splits	*campo*	Elenco dei nomi dei campi per la suddivisione.
n_estimators	*numero intero*	Numero di strutture ad albero da creare. Il valore predefinito è 10.
specify_max_depth	*Booleano*	Specifica la profondità massima personalizzata. Se false, i nodi vengono estesi fino a che tutti gli elementi foglia siano puri o fino a quando tutti gli elementi foglia contengano meno di min_samples_split esempi. Il valore predefinito è false.
max_depth	*numero intero*	La profondità massima della struttura ad albero. Il valore predefinito è 10.
min_samples_leaf	*numero intero*	Dimensione minima del nodo foglia. Il valore predefinito è 1.
max_features	*stringa*	Il numero di funzioni da considerare quando si ricerca la miglior suddivisione: • Se auto, quindi max_features=sqrt(n_features) per il classificatore e max_features=sqrt(n_features) per la regressione. • Se sqrt, quindi max_features=sqrt(n_features). • Se log2, quindi max_features=log2 (n_features). Il valore predefinito è auto.
bootstrap	*Booleano*	Utilizzare gli esempi di bootstrap quando si creano le strutture ad albero. Il valore predefinito è true.
oob_score	*Booleano*	Utilizzare gli esempi out-of-bag per valutare la precisione della generalizzazione Il valore di default è false.
extreme	*Booleano*	Utilizzare in estremo le strutture ad albero randomizzate. Il valore predefinito è false.
use_random_seed	*Booleano*	Specificare ciò per ottenere risultati replicati. Il valore predefinito è false.
random_seed	*numero intero*	Il numero casuale generato da utilizzare quando si creano le strutture ad albero. Specificare qualsiasi numero intero.
cache_size	*float*	La dimensione della cache del kernel in MB. Il valore predefinito è 200.
enable_random_seed	*Booleano*	Abilita il parametro random_seed. Specificare true o false. Il valore predefinito è false.
enable_hpo	*Booleano*	Specificare true o false per abilitare o disabilitare le opzioni HPO. Se impostato su true, Rbfopt verrà applicato per determinare automaticamente il "miglior" modello Random Forest che raggiunge il valore obiettivo definito dall'utente con il seguente parametro target_objval.

Tabella 266. Proprietà rfnode (Continua)

Proprietà rfnode	Tipo di dati	Descrizione proprietà
target_objval	*float*	Il valore della funzione obiettivo (tasso di errore del modello negli esempi) che si desidera raggiungere (ad esempio il valore ottimale sconosciuto). Impostare questo parametro sul valore appropriato se non si conosce il valore ottimale (ad esempio, 0.01).
max_iterations	*numero intero*	Il numero massimo di iterazioni per tentare il modello. Il valore predefinito è 1000.
max_evaluations	*numero intero*	Il numero massimo di valutazioni della funzione per tentare il modello, in cui l'attenzione è incentrata sulla precisione piuttosto che sulla velocità. Il valore predefinito è 300.

Proprietà di smotenode

Il nodo SMOTE (Synthetic Minority Over-sampling Technique) fornisce un algoritmo di over-sampling per la gestione dei dataset sbilanciati. Fornisce un metodo avanzato di bilanciamento dei dati. Il processo SMOTE in SPSS Modeler è implementato in Python e richiede la libreria Python imbalanced-learn©.

Tabella 267. Proprietà di smotenode

Proprietà di smotenode	Tipo di dati	Descrizione proprietà
target_field	*campo*	Il campo obiettivo.
sample_ratio	*stringa*	Abilita un valore di rapporto personalizzato. Le due opzioni sono Auto (sample_ratio_auto) o Imposta rapporto (sample_ratio_manual).
sample_ratio_value	*float*	Il rapporto è il numero dei campioni nella classe di minoranza rispetto al numero di campioni nella classe di maggioranza. Deve essere maggiore di 0 e minore o uguale a 1. Il valore predefinito è auto.
enable_random_seed	*Booleano*	Se impostato su true, la proprietà random_seed verrà abilitata.
random_seed	*numero intero*	Il seed utilizzato dal generatore di numeri random.
k_neighbours	*numero intero*	Il numero di elementi adiacenti più vicini da utilizzare per creare campioni sintetici. Il valore predefinito è 5.
m_neighbours	*numero intero*	Il numero di elementi adiacenti più vicini da utilizzare per determinare se un campione di minoranza è in pericolo. Questa opzione è abilitata solo con i tipi di algoritmo SMOTE borderline1 e borderline2. Il valore predefinito è 10.
algorithm_kind	*stringa*	Il tipo di algoritmo SMOTE: regular, borderline1 o borderline2.

Tabella 267. Proprietà di smotenode (Continua)

Proprietà di smotenode	Tipo di dati	Descrizione proprietà
usepartition	*Booleano*	Se è impostato su true, verranno utilizzati solo i dati di addestramento per la creazione del modello. Il valore predefinito è true.

proprietà dello spectralcluster

L'algoritmo Spectral Clustering © utilizza diversi autovettori per proiettare i dati in uno spazio con meno dimensioni. Quindi un algoritmo di cluster k-means viene applicato nel nuovo spazio per separare i dati in cluster. È ragionevolmente veloce per i piccoli record con molti campi e molto dispendioso dal punto di vista dei costi per i grandi set di dati. Il nodo Spectral Clustering in SPSS Modeler espone le funzioni principali e i parametri comunemente utilizzati della libreria Spectral Clustering. Il nodo è implementato in Python.

Tabella 268. proprietà dello spectralcluster

proprietà di spectralcluster	Tipo di dati	Descrizione proprietà
InputFields	*campo*	Tutti i predittori per il raggruppamento.
nCluster	*numero intero*	La dimensione del sottospazio di proiezione. Specificare un valore di 2 o più. Il valore predefinito è 8.
nInit	*numero intero*	Numero di volte in cui l'algoritmo k-means verrà eseguito con diversi semi centroidi. I risultati finali saranno il miglior risultato di n_init esecuzioni consecutive in termini di inerzia. Specificare un valore di 1 o più. Il valore predefinito è 10.
assignLabels	*stringa*	La strategia da utilizzare per assegnare le etichette nello spazio di incorporamento. Ci sono due modi per assegnare le etichette dopo l'incorporamento laplaciano. K-means può essere applicato ed è una scelta popolare – ma può anche essere sensibile all'inizializzazione. La discretizzazione è un altro approccio, che è meno sensibile all'inizializzazione casuale. Specificare kmeans o discretize. Il valore predefinito è kmeans.
useStringLabel	*booleano*	Indica se utilizzare un'etichetta di stringa per il campo dell'etichetta del cluster. Se impostato su true, il valore di stringa nel parametro stringLabelPrefix verrà utilizzato come prefisso del campo dell'etichetta del cluster. Ad esempio, se il parametro stringLabelPrefix è impostato su Cluster, 'etichetta del cluster nel campo dell'etichetta del cluster sarà Cluster-1, Cluster-2, e così via.
stringLabelPrefix	*stringa*	Specificare il formato per il campo di appartenenza del cluster generato. L'appartenenza al cluster può essere indicata come una stringa con il prefisso dell'etichetta specificato (ad esempio, Cluster-1, Cluster-2, e così via) o come numero. Il valore predefinito è cluster.

Tabella 268. proprietà dello spectralcluster (Continua)

proprietà di spectralcluster	Tipo di dati	Descrizione proprietà
randomSeed	*numero intero*	Specificare un valore per il seme casuale. Oppure utilizzare il valore predefinito di 0, che usa lo stato random globale da numpy.random.
affinity	*stringa*	Affinità è il metodo per valutare la distanza a coppie o l'affinità di serie di campioni. Specificare RBF, Nearest_neighbors, Sigmoid, Polynomial, Linear, Cosine, Laplacian, o Chi2. Il valore predefinito è RBF.
gamma	*double*	Coefficiente del Kernel per kernel RBF, Polynomial, Sigmoid, Laplacian e Chi2. Ignorato per il kernel Nearest_neighbors. L'impostazione predefinita è 1.0.
degree	*double*	Grado del kernel Polynomial. Ignorato da altri kernel. Il valore predefinito è 3.0.
coef0	*double*	Coefficiente zero per i kernel Polynomial and Sigmoid. Ignorato da altri kernel. L'impostazione predefinita è 1.0.
nNeighbors	*numero intero*	Numero di neighbors da utilizzare quando si costruisce la matrice di affinità usando il metodo dei neighbors più vicini. Ignorato per il kernel RBF. Il valore predefinito è 10.
eigenSolver	*stringa*	La strategia di scomposizione degli autovalori da utilizzare. Specificare None, Arpack, Lobpcg o Amg. Il valore predefinito è None. Amg richiede l'installazione di pyamg. Può essere più veloce su problemi molto grandi, sparsi, ma può anche portare a instabilità.
eigenTol	*double*	Criterio di arresto per la eigendecomposizione della matrice Laplacian quando si usa Arpack per eigenSolver.

Proprietà tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) è uno strumento per la visualizzazione dei dati altamente dimensionali. Converte le affinità dei punti dati in probabilità. Questo nodo t-SNE in SPSS Modeler è implementato in Python e richiede la libreria scikit-learn©.

Tabella 269. Proprietà tsnenode

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
mode_type	*stringa*	Specificare la modalità simple o expert.
n_components	*stringa*	Dimensione dello spazio integrato (2D or 3D). Specificare 2 o 3. Il valore predefinito è 2.
method	*stringa*	Specificare barnes_hut o exact. Il valore predefinito è barnes_hut.
init	*stringa*	Inizializzazione dell'integrazione. Specificare random o pca. Il valore predefinito è random.

Tabella 269. Proprietà tsnenode (Continua)

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
target_field	*stringa*	Il nome del campo di destinazione. Sarà una mappa colorata nel grafico di output. Il grafico utilizzerà un colore se viene specificato il campo di destinazione.
perplexity	*float*	La perplessità è correlata al numero di elementi adiacenti più vicini utilizzati in altri algoritmi di apprendimento. Generalmente, i dataset di dimensioni maggiori richiedono una perplessità maggiore. Considerare la selezione di un valore compreso tra 5 e 50. Il valore predefinito è 30.
early_exaggeration	*float*	Questa impostazione controlla quanto i cluster naturali nello spazio originale saranno vicini nello spazio integrato e la quantità di spazio tra di loro. Il valore predefinito è 12.0.
learning_rate	*float*	Il valore predefinito è 200.
n_iter	*numero intero*	Il numero massimo di iterazioni per l'ottimizzazione. Impostare ad almeno 250. Il valore predefinito è 1000.
angle	*float*	La dimensione angolare di un nodo distante misurata da un punto. Specificare un valore nell'intervallo compreso tra 0 e 1. Il valore predefinito è 0.5.
enable_random_seed	*Booleano*	Impostare su true per abilitare il parametro random_seed. Il valore predefinito è false.
random_seed	*numero intero*	Il seed random da utilizzare. Il valore predefinito è None.
n_iter_without_progress	*numero intero*	Numero massimo di iterazioni senza avanzamento. Il valore predefinito è 300.
min_grad_norm	*stringa*	Se la norma del gradiente è inferiore a questa soglia, l'ottimizzazione viene arrestata. Il valore predefinito è 1.0E-7. I valori possibili sono: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*Booleano*	Impostare su true per eseguire t-SNE con diverse perplessità differenti. Il valore predefinito è false.
output_Rename	*Booleano*	Specificare true se si desidera fornire un nome personalizzato o su false per denominare l'output in modo automatico. Il valore predefinito è false.
output_to	*stringa*	Specificare Screen o Output. Il valore predefinito è Screen.

Tabella 269. Proprietà tsnenode (Continua)

Proprietà tsnenode	Tipo di dati	Descrizione proprietà
full_filename	*stringa*	Specificare il nome del file di output.
output_file_type	*stringa*	Formato del file di output. Specificare HTML o Output object. Il valore predefinito è HTML.

Proprietà xgboostlinearnode

XGBoost Linear© rappresenta un'implementazione avanzata di un algoritmo gradient boosting con un modello lineare come modello di base. Gli algoritmi Boosting rilevano in modo interattivo i classificatori deboli e li aggiungono ai classificatori forti finali. Il nodo XGBoost Linear in SPSS Modeler è implementato in Python.

Tabella 270. Proprietà di xgboostlinearnode

Proprietà di xgboostlinearnode	Tipo di dati	Descrizione proprietà
TargetField	*campo*	
InputFields	*campo*	
alpha	*Double*	Il parametro booster linear alpha. Specificare 0 o un valore maggiore. Il valore predefinito è 0.
lambda	*Double*	Il parametro booster linear lambda. Specificare 0 o un valore maggiore. Il valore predefinito è 1.
lambdaBias	*Double*	Il parametro booster linear di distorsione lambda. Specificare qualsiasi numero. Il valore predefinito è 0.
numBoostRound	*numero intero*	Il numero di iterazioni boost per la creazione del modello. Specificare un valore compreso tra 1 e 1000. Il valore predefinito è 10.
objectiveType	*stringa*	Il tipo di obiettivo per l'attività di apprendimento. I valori possibili sono ⌂reg:linear,⌂ reg:logistic,⌂ reg:gamma,⌂ reg:tweedie, count:poisson, ⌂rank:pairwise, binary:logistic o multi. Notare che per gli obiettivi indicatore, è possibile utilizzare solo binary:logistic o multi. Se si utilizza multi, il risultato del punteggio mostrerà i tipi di obiettivo XGBoost multi:softmax e multi:softprob.
random_seed	*numero intero*	Il seed random. Qualsiasi numero compreso tra 0 e 9999999. Il valore predefinito è 0.
useHPO	*Booleano*	Specificare true o false per abilitare o disabilitare le opzioni HPO. Se impostato su true, Rbfopt verrà applicato per ricercare automaticamente il "miglior" modello One-Class SVM che raggiunge il valore obiettivo definito dall'utente con il parametro target_objval.

Proprietà di xgboosttreenode

XGBoost Tree© rappresenta un'implementazione avanzata di un algoritmo gradient boosting con un modello struttura ad albero come modello di base. Gli algoritmi Boosting rilevano in modo interattivo i classificatori deboli e li aggiungono ai classificatori forti finali. XGBoost Tree è molto flessibile e fornisce diversi parametri che possono soddisfare le esigenze della maggior parte degli utenti, quindi il nodo XGBoost Tree in SPSS Modeler riporta le funzioni principali ed i parametri utilizzati più di frequente. Il nodo è implementato in Python.

Tabella 271. Proprietà di xgboosttreenode

Proprietà di xgboosttreenode	Tipo di dati	Descrizione proprietà
TargetField	*campo*	I campi obiettivo.
InputFields	*campo*	I campi di input.
treeMethod	*stringa*	Il metodo della struttura ad albero per la creazione del modello. I valori possibili sono auto, exact o approx. Il valore predefinito è auto.
numBoostRound	*numero intero*	Il numero di iterazioni boost per la creazione del modello. Specificare un valore compreso tra 1 e 1000. Il valore predefinito è 10.
maxDepth	*numero intero*	La profondità massima per l'espansione della struttura ad albero. Specificare il valore 1 o un valore maggiore. Il valore predefinito è 6.
minChildWeight	*Double*	Il peso minimo figlio per l'espansione della struttura ad albero. Specificare il valore 0 o un valore maggiore. Il valore predefinito è 1.
maxDeltaStep	*Double*	Il passo delta massimo per l'espansione della struttura ad albero. Specificare il valore 0 o un valore maggiore. Il valore predefinito è 0.
objectiveType	*stringa*	Il tipo di obiettivo per l'attività di apprendimento. I valori possibili sono ⌂reg:linear,⌂ reg:logistic,⌂ reg:gamma,⌂ reg:tweedie, count:poisson, ⌂rank:pairwise, binary:logistic o multi. Notare che per gli obiettivi indicatore, è possibile utilizzare solo binary:logistic o multi. Se si utilizza multi, il risultato del punteggio mostrerà i tipi di obiettivo XGBoost multi:softmax e multi:softprob.
earlyStopping	*Booleano*	Indica se utilizzare la funzione di arresto anticipato. Il valore di default è False.
earlyStoppingRounds	*numero intero*	Gli errori di convalida devono diminuire ad ogni iterazione di interruzione anticipata per proseguire con l'addestramento. Il valore predefinito è 10.
evaluationDataRatio	*Double*	Rotazione dei dati di input per gli errori di convalida. Il valore predefinito è 0,3.
random_seed	*numero intero*	Il seed random. Qualsiasi numero compreso tra 0 e 9999999. Il valore predefinito è 0.
sampleSize	*Double*	Il campione secondario per il controllo del sovradattamento. Specificare un valore compreso tra 0,1 e 1,0. Il valore predefinito è 0,1.

Tabella 271. Proprietà di xgboosttreenode (Continua)

Proprietà di xgboosttreenode	Tipo di dati	Descrizione proprietà
eta	*Double*	Il valore eta per il controllo del sovradattamento. Specificare un valore compreso tra 0 e 1. Il valore predefinito è 0,3.
gamma	*Double*	Il valore gamma per il controllo del sovradattamento. Specificare 0 o un valore maggiore. Il valore predefinito è 6.
colsSampleRatio	*Double*	Il campione di colonne in base alla struttura ad albero per il controllo del sovradattamento. Specificare un valore compreso tra 0,01 e 1. Il valore predefinito è 1.
colsSampleLevel	*Double*	Il campione di colonne in base al livello per il controllo del sovradattamento. Specificare un valore compreso tra 0,01 e 1. Il valore predefinito è 1.
lambda	*Double*	Il valore lambda per il controllo del sovradattamento. Specificare 0 o un valore maggiore. Il valore predefinito è 1.
alpha	*Double*	Il valore alpha per il controllo del sovradattamento. Specificare 0 o un valore maggiore. Il valore predefinito è 0.
scalePosWeight	*Double*	La ponderazione per la gestione dei dataset sbilanciati. Il valore predefinito è 1.

Capitolo 20. Proprietà nodo Spark

Proprietà isotonicasnode

La regressione isotonica appartiene alla famiglia di algoritmi di regressione. Il nodo Isotonic-AS in SPSS Modeler è implementato in Spark. Per i dettagli relativi agli algoritmi di regressione isotonica, consultare https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html.

Tabella 272. Proprietà proisotonicasnode

Proprietà `isotonicasnode`	Tipo di dati	Descrizione proprietà
`label`	*stringa*	Questa proprietà è una variabile dipendente per cui viene calcolata la regressione isotonica.
`features`	*stringa*	Questa proprietà è una variabile indipendente.
`weightCol`	*stringa*	Il peso rappresenta un numero di misure. Il valore predefinito è 1.
`isotonic`	*Booleano*	Questa proprietà indica se il tipo è `isotonico` o `antitonico`.
`featureIndex`	*numero intero*	Questa proprietà è per l'indice della funzione se `featuresCol` è una colonna vettore. Il valore predefinito è `0`.

Proprietà kmeansasnode

K-Means è uno degli algoritmi di clustering più comunemente usati. Raggruppa i punti dati in un numero predefinito di cluster. Il nodo K-Means-AS in SPSS Modeler è implementato in Spark. Per i dettagli relativi agli algoritmi K-Means, consultare https://spark.apache.org/docs/2.2.0/ml-clustering.html. Si noti che il nodo K-Means-AS esegue automaticamente una codifica one-hot per le variabili categoriali.

Tabella 273. proprietà kmeansasnode

Proprietà`kmeansasnode`	Valori	Descrizione proprietà
`roleUse`	*stringa*	Specificare `predefined` per utilizzare i ruoli predefiniti oppure `custom` per utilizzare le assegnazioni di campo personalizzate. Il valore predefinito è `predefined`.
`autoModel`	*Booleano*	Specificare `true` per utilizzare il nome predefinito (`$S-prediction`) per il nuovo campo di calcolo del punteggio generato, oppure `false` per utilizzare un nome personalizzato. Il valore predefinito è `true`.
`features`	*campo*	Elenco dei nomi dei campi per l'input quando la proprietà `roleUse` è impostata su `custom`.

Tabella 273. proprietà kmeansasnode (Continua)

Proprietàkmeansasnode	Valori	Descrizione proprietà
name	*stringa*	Il nome del nuovo campo di calcolo del punteggio generato quando la proprietà autoModel è impostata su false.
clustersNum	*numero intero*	Il numero di cluster da creare. Il valore predefinito è 5.
initMode	*stringa*	L'algoritmo di inizializzazione. I valori possibili sono k-means|| o random. Il valore predefinito è k-means||.
initSteps	*numero intero*	Il numero di fasi di inizializzazione quando initMode è impostato su k-means||. Il valore predefinito è 2.
advancedSettings	*Booleano*	Specificare true per rendere disponibili le seguenti quattro proprietà. Il valore predefinito è false.
maxIteration	*numero intero*	Numero massimo di iterazioni per l'ottimizzazione. Il valore predefinito è 20.
tolerance	*stringa*	La tolleranza per arrestare le iterazioni. Le impostazioni possibili sono 1.0E-1, 1.0E-2, ..., 1.0E-6. Il valore predefinito è 1.0E-4.
setSeed	*Booleano*	Specificare true per utilizzare un seed random. Il valore predefinito è false.
randomSeed	*numero intero*	Il seed random personalizzato quando la proprietà setSeed è true.

Proprietà xgboostasnode

XGBoost è un'implementazione di un algoritmo boosting di gradiente. Gli algoritmi Boosting rilevano in modo interattivo i classificatori deboli e li aggiungono ai classificatori forti finali. XGBoost è molto flessibile e fornisce molti parametri che possono essere irresistibili per la maggior parte di utenti, così il nodo XGBoost-AS in SPSS Modeler espone le funzioni principali e i parametri comunemente utilizzati. Il nodo XGBoost-AS è implementato in Spark.

Tabella 274. Proprietà xgboostasnode

Proprietà xgboostasnode	Tipo di dati	Descrizione proprietà
target_field	*campo*	Elenco dei nomi campo per l'obiettivo.
input_fields	*campo*	Elenco dei nomi campo per gli input.
nWorkers	*numero intero*	Il numero di worker utilizzati per insegnare il modello XGBoost. Il valore predefinito è 1.
numThreadPerTask	*numero intero*	Il numero di thread utilizzati per worker. Il valore predefinito è 1.
useExternalMemory	*Booleano*	Indica se utilizzare la memoria esterna come cache. Il valore predefinito è **false**.
boosterType	*stringa*	Il tipo booster da utilizzare. Le opzioni disponibili sono gbtree, gblinear o dart. Il valore predefinito è gbtree.

Tabella 274. Proprietà xgboostasnode (Continua)

Proprietà xgboostasnode	Tipo di dati	Descrizione proprietà
numBoostRound	*numero intero*	Il numero di iterazioni per il boosting. Specificare il valore 0 o un valore maggiore. Il valore predefinito è 10.
scalePosWeight	*Double*	Controlla il bilanciamento dei pesi positivi e negativi. Il valore predefinito è 1.
randomseed	*numero intero*	Il seed utilizzato dal generatore di numeri random. Il valore predefinito è **0**.
objectiveType	*stringa*	L'obiettivo di apprendimento. I valori possibili sono ⌂reg:linear,⌂ reg:logistic,⌂ reg:gamma,⌂ reg:tweedie, rank:pairwise, binary:logistic, o multi. Notare che per gli obiettivi indicatore, è possibile utilizzare solo binary:logistic o multi. Se si utilizza multi, il risultato del punteggio mostrerà i tipi di obiettivo XGBoost multi:softmax e multi:softprob. L'impostazione predefinita è reg:linear.
evalMetric	*stringa*	Le metriche di valutazione per i dati di convalida. Verrà assegnata una metrica predefinita in base all'obiettivo. I possibili valori sono rmse, mae, logloss, error, merror, mlogloss, auc, ndcg, map o gamma-deviance. L'impostazione predefinita è rmse.
lambda	*Double*	Il termine regolarizzazione L2 sui pesi. Incrementando questo valore si renderà il modello più conservativo. Specificare 0 o un valore maggiore. Il valore predefinito è 1.
alpha	*Double*	Il termine di regolarizzazione L1 sui pesi. Incrementando questo valore si renderà il modello più conservativo. Specificare 0 o un valore maggiore. Il valore predefinito è 0.
lambdaBias	*Double*	Il termine di regolarizzazione L2 nella distorsione. Se viene utilizzato il tipo di booster gblinear, sarà disponibile questo parametro booster lineare distorsione lambda. Specificare 0 o un valore maggiore. Il valore predefinito è 0.
treeMethod	*stringa*	Se viene utilizzato il tipo di booster gbtree o dart, sono disponibili questo parametro del metodo struttura ad albero per l'espansione della struttura ad albero (e gli altri parametri della struttura ad albero che seguono). Specifica l'algoritmo di costruzione della struttura ad albero XGBoost da utilizzare. Le opzioni disponibili sono auto, exact o approx. L'impostazione predefinita è auto.
maxDepth	*numero intero*	La profondità massima per le strutture ad albero. Specificare un valore di 2 o superiore. Il valore predefinito è 6.
minChildWeight	*Double*	La somma minima dei pesi dell'istanza (hessiana) necessaria in un elemento figlio. Specificare il valore 0 o un valore maggiore. Il valore predefinito è 1.

Tabella 274. Proprietà xgboostasnode (Continua)

Proprietà xgboostasnode	Tipo di dati	Descrizione proprietà
maxDeltaStep	*Double*	Il passo delta massimo da consentire per ciascuna stima del peso della struttura ad albero. Specificare il valore 0 o un valore maggiore. Il valore predefinito è 0.
sampleSize	*Double*	Il campione secondario è il rapporto dell'istanza di addestramento. Specificare un valore compreso tra 0,1 e 1,0. L'impostazione predefinita è 1.0.
eta	*Double*	La riduzione della dimensione del passo utilizzata durante il passo aggiornamento per evitare sovradattamento. Specificare un valore compreso tra 0 e 1. L'impostazione predefinita è 0.3.
gamma	*Double*	La riduzione di perdita minima richiesta per eseguire una successiva partizione sul nodo foglia della struttura ad albero. Specificare 0 o un valore maggiore. Il valore predefinito è 6.
colsSampleRatio	*Double*	Il rapporto campione secondario di colonne quando si costruisce la struttura ad albero. Specificare un valore compreso tra 0,01 e 1. Il valore predefinito è 1.
colsSampleLevel	*Double*	Il rapporto campione secondario di colonne per ogni suddivisione, in ogni livello. Specificare un valore compreso tra 0,01 e 1. Il valore predefinito è 1.
normalizeType	*stringa*	Se viene utilizzato il tipo booster dart, sono disponibili questo parametro dart e i seguenti tre parametri dart. Questo parametro imposta l'algoritmo di normalizzazione. Specificare tree o forest. L'impostazione predefinita è tree.
sampleType	*stringa*	Il tipo di algoritmo di campionamento. Specificare uniform o weighted. L'impostazione predefinita è uniform.
rateDrop	*Double*	Il parametro booster dart del tasso di dropout. Specificare un valore compreso tra 0.0 e 1.0. L'impostazione predefinita è 0.0.
skipDrop	*Double*	Il parametro booster dart per la probabilità di ignorare dropout. Specificare un valore compreso tra 0.0 e 1.0. L'impostazione predefinita è 0.0.

Capitolo 21. Proprietà dei Supernodi

Nelle tabelle seguenti vengono illustrate le proprietà specifiche dei Supernodi. Si noti che le proprietà comuni dei nodi si applicano anche ai Supernodi.

Tabella 275. Proprietà supernodo terminale

Nome proprietà	Tipo di proprietà/Elenco di valori	Descrizione proprietà
execute_method	Script Normal	
script	*stringa*	

Parametri dei Supernodi

Per creare o impostare i parametri dei Supernodi è possibile utilizzare gli script, con il seguente formato generale:

```
mySuperNode.setParameterValue("minvalue", 30)
```

È possibile richiamare il valore del parametro con:

```
value mySuperNode.getParameterValue("minvalue")
```

Ricerca di Supernodi esistenti

È possibile ricercare i Supernodi nei flussi utilizzando la funzione findByType():

```
source_supernode = modeler.script.stream().findByType("source_super", None)
process_supernode = modeler.script.stream().findByType("process_super", None)
terminal_supernode = modeler.script.stream().findByType("terminal_super", None)
```

Impostazione delle proprietà dei nodi incapsulati

È possibile impostare le proprietà per nodi specifici incapsulati all'interno di un Supernodo accedendo al diagramma figlio all'interno del Supernodo. Si supponga per esempio di disporre di un Supernodo origine con un nodo Testo variabile incapsulato per la lettura dei dati. È possibile passare il nome del file da leggere (specificato utilizzando la proprietà `full_filename`) accedendo al diagramma figlio e ricercando il nodo pertinente nel modo riportato di seguito:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Creazione di Supernodi

Se si desidera creare ex-novo un Supernodo ed il relativo contenuto, è possibile procedere in modo simile creando il Supernodo, accedendo al diagramma figlio e creando i nodi desiderati. Inoltre, è necessario verificare che i nodi all'interno del diagramma del Supernodo siano collegati ai nodi connettore di input e/o di output. Ad esempio, se si desidera creare un Supernodo di elaborazione:

```
process_supernode = modeler.script.stream().createAt("process_super", "My SuperNode", 200, 200)
childDiagram = process_supernode.getChildDiagram()
filternode = childDiagram.createAt("filter", "My Filter", 100, 100)
childDiagram.linkFromInputConnector(filternode)
childDiagram.linkToOutputConnector(filternode)
```

Appendice A. Riferimento dei nomi del nodo

Questa sezione fornisce un riferimento per i nomi degli script dei nodi in IBM SPSS Modeler.

Nomi dei nugget del modello

Ai nugget del modello (detti anche modelli generati) può essere fatto riferimento per tipo, come avviene per gli oggetti nodo e output. Le seguenti tabelle elencano i nomi di riferimento degli oggetti modello.

Si noti che questi nomi vengono utilizzati specificamente per fare riferimento ai nugget del modello nella palette Modelli (nell'angolo superiore destro della finestra di IBM SPSS Modeler). Per fare riferimento ai nodi modello aggiunti a un flusso ai fini del calcolo del punteggio, viene utilizzato un insieme diverso di nomi preceduti dal prefisso `apply...`. Per ulteriori informazioni, consultare l'argomento Proprietà dei nodi dei nugget del modello.

Nota: in circostanze normali, si consiglia di fare riferimento ai modelli per nome *e* tipo, in modo da evitare confusione.

Tabella 276. Nomi dei nugget del modello (palette Modelli).

Nome modello	**Modello**
anomalydetection	Anomalia
apriori	Apriori
autoclassifier	Classificatore automatico
autocluster	Cluster automatico
autonumeric	Numerico automatico
bayesnet	Rete bayesiana
c50	C5.0
carma	Carma
cart	C&R Tree
chaid	CHAID
coxreg	regressione di Cox
decisionlist	Elenco di decisioni
discriminante	Discriminante
fattore	PCA/Fattore
featureselection	Selezione delle funzioni
genlin	Regressione lineare generalizzata
glmm	GLMM
kmeans	Medie K
knn	*k*-elemento adiacente più vicino
kohonen	Kohonen
regressione	Lineare
logreg	Regressione logistica
neuralnetwork	Rete neurale
quest	QUEST

Tabella 276. Nomi dei nugget del modello (palette Modelli) (Continua).

Nome modello	Modello
regressione	Regressione lineare
sequence	Sequenza
slrm	Modello di risposta di autoapprendimento
statisticsmodel	modello IBM SPSS Statistics
svm	Support vector machine
timeseries	Serie temporali
twostep	TwoStep

Tabella 277. Nomi dei nugget del modello (palette Modelli in-database).

Nome modello	Modello
db2imcluster	Raggruppamento cluster IBM ISW
db2imlog	Regressione logistica IBM ISW
db2imnb	IBM ISW Naive Bayes
db2imreg	Regressione IBM ISW
db2imtree	Struttura ad albero delle decisioni IBM ISW
msassoc	Regole di associazione MS
msbayes	Naive Bayes MS
mscluster	Raggruppamento cluster MS
mslogistic	Regressione logistica MS
msneuralnetwork	Rete neurale MS
msregression	Regressione lineare MS
mssequencecluster	MS Sequence Clustering
mstimeseries	Serie temporali MS
mstree	Struttura ad albero delle decisioni MS
netezzabayes	Rete di Bayes Netezza
netezzadectree	Struttura ad albero delle decisioni di Netezza
netezzadivcluster	Raggruppamento cluster divisivo Netezza
netezzaglm	Lineare generalizzato Netezza
netezzakmeans	Medie K Netezza
netezzaknn	KNN Netezza
netezzalineregression	Regressione lineare Netezza
netezzanaivebayes	Naive Bayes Netezza
netezzapca	PCA Netezza
netezzaregtree	Struttura ad albero di regressione Netezza
netezzatimeseries	Serie temporali Netezza
oraabn	Bayes adattivo Oracle
oraai	Oracle AI
oradecisiontree	struttura ad albero delle decisioni Oracle
oraglm	GLM Oracle
orakmeans	Oracle *k*-Medie

Tabella 277. Nomi dei nugget del modello (palette Modelli in-database) (Continua).

Nome modello	Modello
oranb	Naive Bayes Oracle
oranmf	NMF Oracle
oraocluster	O-Cluster Oracle
orasvm	SVM Oracle

Per evitare nomi di modelli duplicati

Quando si utilizzano gli script per manipolare i modelli generati, ricordare che la duplicazione dei nomi dei modelli può comportare riferimenti ambigui. Per evitare questo problema, si consiglia di richiedere nomi univoci per i modelli generati durante lo script.

Per impostare le opzioni per i nomi di modelli duplicati:

1. Dai menu, scegliere:
   Strumenti > **Opzioni utente**
2. Fare clic sulla scheda **Notifiche**.
3. Selezionare **Sostituisci modello precedente** per limitare la duplicazione dei nomi per i modelli generati.

Il comportamento dell'esecuzione dello script può variare tra SPSS Modeler e IBM SPSS Collaboration and Deployment Services nel caso in cui i riferimenti ai modelli siano ambigui. SPSS Modeler client contiene l'opzione "Sostituisci modello precedente", che sostituisce automaticamente i modelli con lo stesso nome (ad esempio, nel caso in cui uno script esegua iterazioni sulla base di un ciclo per generare ogni volta un modello diverso). Tuttavia, questa opzione non è disponibile quando il medesimo script viene eseguito in IBM SPSS Collaboration and Deployment Services. Per evitare questa situazione, si può rinominare il modello generato da ogni iterazione, evitando così riferimenti ambigui ai modelli, oppure eliminare il modello corrente (aggiungendo, ad esempio un'istruzione `clear generated palette`) prima della fine del ciclo.

Nomi dei tipi di output

La seguente tabella elenca tutti i tipi di oggetti output e i nodi che li creano. Per un elenco completo dei formati di esportazione disponibili per ciascun tipo di oggetto di output, vedere la descrizione delle proprietà per il nodo che crea il tipo di output, disponibile in Proprietà comuni dei nodi Grafici e Proprietà dei nodi Output.

Tabella 278. Tipi di oggetto di output e i nodi che li creano.

Tipo di oggetto di output	Nodo
analysisoutput	Analisi
collectionoutput	Raccolta
dataauditoutput	Verifica dati
distributionoutput	Distribuzione
evaluationoutput	Valutazione
histogramoutput	Istogramma
matrixoutput	Matrice
meansoutput	Medie
multiplotoutput	Multiplot
plotoutput	Grafico

Tabella 278. Tipi di oggetto di output e i nodi che li creano (Continua).

Tipo di oggetto di output	**Nodo**
qualityoutput	Qualità
reportdocumentoutput	Questo tipo di oggetto non proviene da un nodo, ma si tratta dell'output creato da un report progetto
reportoutput	Report
statisticsprocedureoutput	Output di Statistics
statisticsoutput	Statistiche
tableoutput	Tabella
timeplotoutput	Grafico temporale
weboutput	Web

Appendice B. Migrazione da script legacy a script Python

Panoramica sulla migrazione di script Legacy

Questa sezione fornisce un riepilogo delle differenze tra gli script Python e legacy in IBM SPSS Modeler e fornisce informazioni relative alla migrazione degli script legacy in script Python. In questa sezione, è riportato un elenco di comandi Legacy SPSS Modeler standard e dei comandi Python equivalenti.

Differenze generali

Gli script legacy devono gran parte della loro progettazione agli script di comandi del sistema operativo. Gli script legacy sono orientati alla linea di comando e sebbene ci siano alcune strutture a blocco, per esempio `if...then...else...endif` e `for...endfor`, i rientri generalmente non sono significativi.

Negli script Python, il rientro è significativo e le linee che appartengono allo stesso blocco logico devono avere lo stesso livello di rientro.

Nota: È necessario prestare attenzione quando si copia ed incolla il codice scritto in Python. Una linea che ha un rientro che utilizza le tabulazioni, nell'editor potrebbe sembrare uguale ad un altra che utilizza un rientro con gli spazi. Comunque, loscript Python genererà un errore a causa del fatto che le linee non hanno gli stessi rientri.

Contesto di script

Il contesto di script definisce l'ambiente in cui viene eseguito lo script, per esempio il flusso o il supernodo che esegue lo script. Nello script legacy il contesto è implicito, che significa, per esempio, che si presuppone che tutti i riferimenti del nodo in uno script del flusso siano all'interno del flusso che esegue lo script.

Nello script Python, il contesto dello script viene fornito esplicitamente tramite il modulo `modeler.script`. Per esempio, uno script Python del flusso può accedere al flusso che esegue lo script con il seguente codice:

```
s = modeler.script.stream()
```

Le funzioni correlate del flusso possono quindi essere invocate attraverso l'oggetto restituito.

Comandi o funzioni

Gli script Legacy sono comandi orientati. Questo significa che ogni riga di script in genere viene avviata con il comando da eseguire seguito da parametri, ad esempio:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python utilizza funzioni che vengono di solito invocate attraverso un oggetto (un modulo, classe o oggetto) che definisce la funzione, ad esempio:

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Valori letterali e commenti

Alcuni comandi relativi a commenti e valori letterali comunemente utilizzati in IBM SPSS Modeler dispongono di comandi equivalenti negli script Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 279. Mapping degli script Legacy con gli script Python per valori letterali e commenti.

Script Legacy	**Script Python**
Intero, ad esempio `4`	Stesso
Float, ad esempio `0.003`	Stesso
Stringhe racchiuse tra apici singoli, ad esempio `'Hello'`	Stesso **Nota:** I valori letterali della stringa contenente caratteri non ASCII devono essere preceduti da una `u` per garantire che siano rappresentati come Unicode.
Stringhe racchiuse tra virgolette, ad esempio `"Hello again"`	Stesso **Nota:** I valori letterali della stringa contenente caratteri non ASCII devono essere preceduti da una `u` per garantire che siano rappresentati come Unicode.
Stringhe di lunghezza elevata, ad esempio `"""This is a string that spans multiple lines"""`	Stesso
Elenchi, ad esempio `[1 2 3]`	`[1, 2, 3]`
Riferimento a variabile, ad esempio `set x = 3`	`x = 3`
Continuazione di riga (\), ad esempio `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
Commento di blocco, ad esempio `/* This is a long comment over a line. */`	`""" This is a long comment over a line. """`
Commento di riga, ad esempio `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Operatori

Alcuni comandi dell'operatore comunemente utilizzati in IBM SPSS Modeler dispongono di comandi equivalenti negli script Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 280. Mapping degli script Legacy con gli script Python per gli operatori.

Script Legacy	**Script Python**

`NUM1 + NUM2`
`LIST + ITEM`
`LIST1 + LIST2`
	`NUM1 + NUM2`
	`LIST.append(ITEM)`
	`LIST1.extend(LIST2)`

`NUM1 – NUM2` `LIST - ITEM`	`NUM1 – NUM2` `LIST.remove(ITEM)`
`NUM1 * NUM2`	`NUM1 * NUM2`
`NUM1 / NUM2`	`NUM1 / NUM2`

Tabella 280. Mapping degli script Legacy con gli script Python per gli operatori (Continua).

Script Legacy	Script Python
= ==	==
/= /==	!=
X ** Y	X ** Y
X < Y X <= Y X > Y X >= Y	X < Y X <= Y X > Y X >= Y
X div Y X rem Y X mod Y	X // Y X % Y X % Y
and or not(EXPR)	and or not EXPR

Istruzioni condizionali e cicli

Alcuni comandi condizionali e di ciclo comunemente utilizzati in IBM SPSS Modeler hanno comandi equivalenti nel linguaggio di script di Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 281. Mapping degli script Legacy con gli script Python per istruzioni condizionali e cicli.

Script Legacy	Script Python

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	
	o
	
	VAR = INT1
	while VAR <= INT2:
	...
	VAR += 1

for VAR in LIST ... endfor	for VAR in LIST: ...
for VAR in_fields_to NODE ... endfor	for VAR in NODE.getInputDataModel(): ...
for VAR in_fields_at NODE ... endfor	for VAR in NODE.getOutputDataModel(): ...

if...then
...
elseif...then
...
else
...
endif
	if ...:
	...
	elif ...:
	...
	else:
	...

with TYPE OBJECT ... endwith	Nessun equivalente
var VAR1	La dichiarazione della variabile non è richiesta

Variabili

Nello script legacy, le variabili vengono dichiarate prima di farvi riferimento, per esempio:

```
var mynode
set mynode = create typenode at 96 96
```

Nello script Python, le variabili vengono dichiarate prima di farvi riferimento, per esempio:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

Nello script legacy, i riferimenti alle variabili devono essere esplicitamente rimossi utilizzando l'operatore ^, per esempio:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Come molti linguaggi di script, ciò non è necessario nello script Python, per esempio:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Tipi di nodo, output e modello

Negli script legacy, i diversi tipi di oggetto (nodo, output e modello) hanno tipicamente il tipo accodato al tipo di oggetto. Ad esempio, il nodo Ricava ha il tipo `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

Le API IBM SPSS Modeler in Python non includono il suffisso `node`, per cui il nodo Ricava ha il tipo `derive`, ad esempio:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

La sola differenza nei nomi del tipo negli script legacy e Python è la mancanza del suffisso del tipo.

Nomi proprietà

I nomi delle proprietà sono gli stessi sia nello script legacy che in quello Python. Ad esempio, nel nodo Testo Variabile, la proprietà che definisce la posizione del file è `full_filename` in entrambi gli ambienti di script.

Riferimenti a nodi

Molti script legacy utilizzano una ricerca implicita per trovare ed accedere al nodo da modificare. Per esempio, i comandi seguenti ricercano nel flusso corrento un nodo Tipo con etichetta "Type", quindi impostano la direzione (o ruolo di modellazione) del campo "Age" ad input ed il campo "Drug" come target, che è il valore che deve essere previsto:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

Negli script Python, gli oggetti nodo devono essere individuati esplicitamente prima di richiamare la funzione per impostare il valore della proprietà, per esempio:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Nota: In questo caso, "Target" deve essere tra virgolette.

Gli script Python possono utilizzare in alternativa l'enumerazione `ModelingRole` nel package `modeler.api`.

Sebbene la versione di script Python può essere più dettagliata, si ottengono prestazioni a runtime migliori poiché la ricerca del nodo di solito viene eseguita solo una volta al giorno. Nell'esempio di script legacy, la ricerca di un nodo viene fatta per ogni comando.

È supportata anche la ricerca dei nodi tramite ID (l'ID del nodo è visible nella scheda Annotazioni della finestra di dialogo del nodo). Per esempio, nello script legacy:

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

Il seguente script mostra lo stesso esempio negli script Python:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Ottenimento ed impostazione di proprietà

Gli script Legacy utilizzano il comando `set` per assegnare un valore. Il termine successivo al comando `set` può essere una definizione di proprietà. Il seguente script mostra due possibili formati di script per l'impostazione delle proprietà:

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

Negli script Python, lo stesso risultato si ottiene utilizzando le funzioni `setPropertyValue()` e `setKeyedPropertyValue()`, ad esempio:

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

Negli script legacy, l'accesso ai valori delle proprietà può essere ottenuto utilizzando il comando `get`, ad esempio:

```
var n v
set n = get node :filternode
set v = ^n.name
```

Negli script Python, lo stesso risultato si ottiene utilizzando la funzione `getPropertyValue()`, ad esempio:

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Modifica dei flussi

Negli script legacy, il comando `create` viene utilizzato per creare un nuovo nodo, ad esempio:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

Negli script Python, i flussi hanno vari metodi per la creazione di nodi, ad esempio:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

Negli script legacy, il comando `connect` viene utilizzato per creare collegamenti tra nodi, per esempio:

```
connect ^agg to ^select
```

Negli script Python, il metodo `link` viene utilizzato per creare collegamenti tra nodi, ad esempio:

```
stream.link(agg, select)
```

Negli script legacy, il comando `disconnect` viene utilizzato per rimuovere i collegamenti tra i nodi, ad esempio:

```
disconnect ^agg from ^select
```

Negli script Python, il metodo `unlink` viene utilizzato per rimuovere i collegamenti tra i nodi, ad esempio:

```
stream.unlink(agg, select)
```

Negli script legacy, il comando `position` viene utilizzato per posizionare i nodi sull'area di disegno del flusso o tra altri nodi, ad esempio:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

Negli script Python, lo stesso risultato viene ottenuto utilizzando due metodi separati: `setXYPosition` e `setPositionBetween`. Ad esempio:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Operazioni nodo

Alcuni comandi di operazione nodo che sono comunemente utilizzati in IBM SPSS Modeler hanno un comando equivalente nel linguaggio di script Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 282. Mapping degli script Legacy con gli script Python per le operazioni di nodo.

Script Legacy	Script Python
`create` *nodespec* `at x y`	*stream*`.create(`*type*`, `*name*`)` *stream*`.createAt(`*type*`, `*name*`, x, y)` *stream*`.createBetween(`*type*`, `*name*`, preNode, postNode)` *stream*`.createModelApplier(`*model*`, `*name*`)`
`connect` *fromNode* `to` *toNode*	*stream*`.link(`*fromNode*`, `*toNode*`)`
`delete` *node*	*stream*`.delete(`*node*`)`
`disable` *node*	*stream*`.setEnabled(`*node*`, False)`
`enable` *node*	*stream*`.setEnabled(`*node*`, True)`
`disconnect` *fromNode* `from` *toNode*	*stream*`.unlink(`*fromNode*`, `*toNode*`)` *stream*`.disconnect(`*node*`)`
`duplicate` *node*	*node*`.duplicate()`
`execute` *node*	*stream*`.runSelected(`*nodes*`, `*results*`)` *stream*`.runAll(`*results*`)`
`flush` *node*	*node*`.flushCache()`
`position` *node* `at x y`	*node*`.setXYPosition(x, y)`
`position` *node* `between` *node1* `and` *node2*	*node*`.setPositionBetween(`*node1*`, `*node2*`)`
`rename` *node* `as` *name*	*node*`.setLabel(`*name*`)`

Esecuzione di cicli

Negli script legacy, vi sono due opzioni principali di esecuzione di cicli che sono supportate:

- Esecuzione di cicli *Conteggiati*, dove una variabile indice si sposta tra due limiti interi.
- Esecuzione di cicli in *Sequenza* che ciclano attraverso una sequenza di valori, associando il valore corrente alla variabile dell'esecuzione di cicli.

Il seguente script è un esempio di esecuzione di cicli conteggiato negli script legacy:

```
for i from 1 to 10
 println ^i
endfor
```

Il seguente script è un esempio di esecuzione di cicli in sequenza negli script legacy:

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

Vi sono anche altri tipi di esecuzione di cicli che possono essere utilizzati:

- Iterazione tra i modelli nella tavolozza dei modelli oppure tra gli output nella tavolozza degli output.
- Iterazione tra i campi che entrano o escono da un nodo.

Gli script Python supportano anche diversi tipi di esecuzione di cicli. Il seguente script è un esempio di esecuzione di cicli conteggiati negli script Python:

```
i = 1
while i <= 10:
 print i
 i += 1
```

Il seguente script è un esempio di esecuzione di cicli in sequenza negli script Python:

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

L'esecuzione di cicli in sequenza è molto flessibile e quando viene associata ai metodi API di IBM SPSS Modeler può supportare la maggioranza di casi di utilizzo di script legacy. Il seguente esempio mostra come utilizzare una esecuzione di cicli in sequenza negli script Python per scorrere attraverso i campi che provengono da un nodo:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

esecuzione di flussi

Durante l'esecuzione del flusso, il modello o gli oggetti di output che vengono generati vengono aggiunti ad uno dei gestori dell'oggetto. Negli script legacy, lo script deve individuare gli oggetti creati dal gestore dell'oggetto oppure accedere all'output generato più recentemente dal nodo che ha generato l'output stesso.

Il flusso di esecuzione in Python è diverso, in quel caso ogni oggetto modello o output che sono generati dall'esecuzione vengono restituiti in un elenco che viene inoltrato alla funzione di esecuzione. Questo rende più semplice l'accesso ai risultati di un flusso di esecuzione.

Gli script legacy supportano tre comandi di esecuzione del flusso:

- `execute_all` esegue tutti i nodi terminale eseguibili nel flusso.
- `execute_script` esegue lo script del flusso indipendentemente dall'impostazione di esecuzione dello script.
- `execute` *`node`* esegue il nodo specificato.

Gli script Python supportano un insieme analogo di funzioni:

- *`stream`*`.runAll(`*`results-list`*`)` esegue tutti i nodi terminali eseguibili nel flusso.

- *stream*.runScript(*results-list*) esegue lo script del flusso indipendentemente dall'impostazione dell'esecuzione dello script.
- *stream*.runSelected(*node-array*, *results-list*) esegue l'insieme di nodi specificato nell'ordine in cui sono stati forniti.
- *node*.run(*results-list*) esegue il nodo specificato.

In uno script, l'esecuzione di un flusso può essere interrotta utilizzando il comando exit seguito da un codice intero facoltativo, per esempio:

```
exit 1
```

In uno script Python, lo stesso risultato si ottiene con il seguente script:

```
modeler.script.exit(1)
```

Accesso ad oggetti attraverso il file system ed il repository

Negli script legacy, è possibile aprire un flusso esistente, un modello o un oggetto di output utilizzando il comando open, per esempio:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

Negli script Python, esiste una classe TaskRunner che è accessibile dalla sessione e può essere utilizzata per effettuare una azione simile, per esempio:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Per salvare un oggetto negli script legacy, è possibile utilizzare il comando save, per esempio:

```
save stream s as "c:/my streams/new_modeling.str"
```

L'approccio equivalente negli script Python sarebbe quello di utilizzare la classe TaskRunner, per esempio:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Le operazioni basate su IBM SPSS Collaboration and Deployment Services Repository sono supportate negli script legacy attraverso i comandi retrieve estore, per esempio:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

Negli script Python, la funzionalità equivalente potrebbe essere accessibile tramite l'oggetto Repository associato alla sessione, per esempio:

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

Nota: L'accesso al repository richiede che la sessione sia stata configurata con una connessione al repository valida.

Operazioni di flusso

Alcuni comandi di operazioni di flusso che sono comunemente utilizzati in IBM SPSS Modeler hanno un comando equivalente nel linguaggio di script di Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 283. Mapping degli script Legacy con gli script Python per le operazioni di flusso.

Script Legacy	Script Python
`create stream DEFAULT_FILENAME`	*taskrunner*.createStream(*name*, *autoConnect*, *autoManage*)
`close stream`	*stream*.close()
`clear stream`	*stream*.clear()
`get stream stream`	Nessun equivalente
`load stream path`	Nessun equivalente
`open stream path`	*taskrunner*.openStreamFromFile(*path*, *autoManage*)
`save stream as path`	*taskrunner*.saveStreamToFile(*stream*, *path*)
`retreive stream path`	*repository*.retreiveStream(*path*, *version*, *label*, *autoManage*)
`store stream as path`	*repository*.storeStream(*stream*, *path*, *label*)

Operazioni del modello

Alcuni comandi di operazione di modello che sono comunemente utilizzati in IBM SPSS Modeler hanno un comando equivalente nel linguaggio di script Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 284. Mapping degli script Legacy con gli script Python per le operazioni di modello.

Script Legacy	Script Python
`open model path`	*taskrunner*.openModelFromFile(*path*, *autoManage*)
`save model as path`	*taskrunner*.saveModelToFile(*model*, *path*)
`retrieve model path`	*repository*.retrieveModel(*path*, *version*, *label*, *autoManage*)
`store model as path`	*repository*.storeModel(*model*, *path*, *label*)

Operazioni di output di documento

Alcuni comandi di operazioni di output di documenti che sono comunemente utilizzati in IBM SPSS Modeler hanno un comando equivalente nel linguaggio di script di Python. Ciò può rendere più semplice la conversione degli script Legacy SPSS Modeler esistenti in script Python da utilizzare in IBM SPSS Modeler 17.

Tabella 285. Mapping degli script legacy con gli script Python per le operazioni di output di documenti.

Script Legacy	Script Python
`open output path`	*taskrunner*.openDocumentFromFile(*path*, *autoManage*)
`save output as path`	*taskrunner*.saveDocumentToFile(*output*, *path*)
`retrieve output path`	*repository*.retrieveDocument(*path*, *version*, *label*, *autoManage*)
`store output as path`	*repository*.storeDocument(*output*, *path*, *label*)

Altre differenze tra script legacy e script Python

Gli script Legacy forniscono supporto per la gestione di progetti IBM SPSS Modeler. Attualmente gli script Python non lo supportano.

Gli script Legacy forniscono supporto per il caricamento di oggetti *stato* (combinazioni di flussi e modelli). Gli oggetti Stato sono obsoleti da IBM SPSS Modeler 8.0. Gli script Python non supportano gli oggetti Stato.

Gli script Python offrono le seguenti funzioni aggiuntive che non sono disponibili negli script legacy:

- Definizioni di classe e funzione
- Gestione degli errori
- Supporto più sofisticato di input/output
- Moduli esterni e terze parti

Informazioni particolari

Queste informazioni sono state sviluppate per prodotti e servizi offerti negli Stati Uniti. Questo materiale potrebbe essere disponibile da IBM in altre lingue. Tuttavia, potrebbe essere necessario disporre di una propria copia del prodotto o versione di prodotto in quella lingua per potervi accedere.

IBM può non offrire i prodotti, i servizi o le funzioni presentati in questo documento in altri paesi. Consultare il rappresentante locale IBM per le informazioni sui prodotti e servizi attualmente disponibili nella propria zona. Qualsiasi riferimento ad un prodotto, programma o servizio IBM non implica o intende dichiarare che solo quel prodotto, programma o servizio IBM può essere utilizzato. In sostituzione a quelli forniti da IBM, è possibile utilizzare prodotti, programmi o servizi funzionalmente equivalenti che non comportino violazione dei diritti di proprietà intellettuale o di altri diritti IBM. Tuttavia, è responsabilità dell'utente valutare e verificare il funzionamento di qualsiasi prodotto, programma o servizio non IBM.

IBM può avere applicazioni di brevetti o brevetti in corso relativi all'argomento descritto in questo documento. La consegna del presente documento non conferisce alcuna licenza rispetto a questi brevetti. Chi desiderasse ricevere informazioni relative a licenze può rivolgersi per iscritto a:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Per richieste di licenze relative ad informazioni double-byte (DBCS) contattare il Dipartimento di Proprietà Intellettuale IBM nel proprio paese o inviare richieste per iscritto a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

IBM (INTERNATIONAL BUSINESS MACHINES CORPORATION) FORNISCE LA PRESENTE PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA" SENZA GARANZIE DI ALCUN TIPO, ESPRESSE O IMPLICITE, IVI INCLUSE, A TITOLO DI ESEMPIO, GARANZIE IMPLICITE DI NON VIOLAZIONE, DI COMMERCIABILITÀ E DI IDONEITÀ PER UNO SCOPO PARTICOLARE. Alcune giurisdizioni non escludono le garanzie implicite; di conseguenza la suddetta esclusione potrebbe, in questo caso, non essere applicabile.

Le presenti informazioni possono includere imprecisioni tecniche o errori tipografici. Le modifiche periodiche apportate alle informazioni contenute in questa pubblicazione verranno inserite nelle nuove edizioni della pubblicazione. IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto nel manuale in qualsiasi momento e senza preavviso.

Tutti i riferimenti a siti Web non IBM sono forniti unicamente a scopo di consultazione e non devono essere in alcun modo considerati come complementari a tali siti Web. I matteriali disponibili su tali siti Web non fanno parte del materiale relativo a questo prodotto IBM e l'utilizzo di questi è a discrezione dell'utente.

IBM può utilizzare o distribuire qualsiasi informazione fornita dall'utente nel modo che ritiene più idoneo senza incorrere in alcun obbligo nei confronti dell'utente stesso.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire: (i) lo scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni dovrebbero contattare:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, l'addebito di un canone.

Il programma concesso in licenza descritto nel presente documento e tutto il materiale concesso in licenza disponibile sono forniti da IBM in base ai termini dell'IBM Customer Agreement, dell'IBM International Program License Agreement o di qualsiasi altro accordo equivalente tra le parti.

I dati delle prestazioni e gli esempi client citati vengono presentati solo a scopo illustrativo. I risultati delle prestazioni effettive possono variare in base alle configurazioni specifiche e alle condizioni di funzionamento.

Le informazioni relative a prodotti non IBM sono ottenute dai fornitori di quei prodotti, dagli annunci pubblicati o da altre fonti disponibili al pubblico. IBM non ha testato quei prodotti e non può garantire l'accuratezza delle prestazioni, la compatibilità o qualsiasi altra dichiarazione relativa a prodotti non IBM. Commenti relativi alle prestazioni di prodotti non IBM, dovrebbero essere indirizzati ai fornitori di questi prodotti.

Qualsiasi affermazione relativa agli obiettivi e alla direzione futura di IBM è soggetta a modifica o revoca senza preavviso e concerne esclusivamente gli scopi dell'azienda.

Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali. Per fornire una descrizione il più possibile esaustiva, gli esempi includono nomi di persone, società, marchi e prodotti. Tutti questi nomi sono fittizi e qualsiasi somiglianza a persone o aziende commerciali reali è puramente casuale.

Marchi

IBM, il logo IBM e ibm.com sono marchi o marchi registrati di International Business Machines Corp., registrati in numerose giurisdizioni del mondo. I nomi di altri prodotti e servizi potrebbero essere marchi di IBM o di altre società. Per un elenco aggiornato di marchi IBM, consultare il web nella sezione Copyright and trademark information, all'indirizzo www.ibm.com/legal/copytrade.shtml.

Adobe, il logo Adobe logo, PostScript ed il logo PostScript sono marchi o marchi registrati di Adobe Systems Incorporated negli Stati Uniti e/o in altri paesi.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium e Pentium sono marchi o marchi registrati di Intel Corporation o relative controllate negli Stati Uniti e altri paesi.

Linux è un marchio registrato di Linus Torvalds negli Stati Uniti e/o in altri paesi.

Microsoft, Windows, Windows NT e il logo Windows sono marchi di Microsoft Corporation negli Stati Uniti e/o in altri paesi.

UNIX è un marchio registrato di Open Group negli Stati Uniti e/o in altri paesi.

Java e tutti i marchi e i logo relativi a Java sono marchi commerciali o marchi registrati di Oracle e/o delle sue affiliate.

Termini e condizioni per la documentazione del prodotto

Le autorizzazioni per l'uso delle presenti pubblicazioni sono concesse in conformità con i seguenti termini e condizioni.

Applicabilità

I termini e le condizioni riportati di seguito si aggiungono alle condizioni di utilizzo per il sito Web IBM.

Uso personale

È possibile riprodurre tali pubblicazioni per uso personale e non commerciale nel rispetto di tutte le informazioni relative alla proprietà. Non è possibile distribuire, visualizzare o utilizzare tali pubblicazioni, o una parte di esse, senza l'esplicito consenso di IBM.

Uso commerciale

È possibile riprodurre, distribuire e visualizzare queste pubblicazioni unicamente all'interno del proprio gruppo aziendale a condizione che vengano conservate tutte le indicazioni relative alla proprietà. Non è possibile effettuare lavori derivati di queste pubblicazioni o riprodurre, distribuire o visualizzare queste pubblicazioni o qualsiasi loro parte al di fuori del proprio gruppo aziendale senza chiaro consenso da parte di IBM.

Diritti

Fatto salvo quanto espressamente concesso in questa autorizzazione, non sono concesse altre autorizzazioni, licenze o diritti, espressi o impliciti, relativi alle pubblicazioni o a qualsiasi informazione, dato, software o altra proprietà intellettuale qui contenuta.

IBM si riserva il diritto di ritirare le autorizzazioni qui concesse qualora, a propria discrezione, l'utilizzo di queste Pubblicazioni sia a danno dei propri interessi o, come determinato da IBM, qualora non siano rispettate in modo appropriato le suddette istruzioni.

Non è consentito scaricare, esportare o riesportare queste informazioni se non nei limiti stabiliti dalle leggi e normative applicabili, ivi comprese tutte le leggi e le normative sull'esportazione vigenti negli Stati Uniti.

IBM NON GARANTISCE IL CONTENUTO DI QUESTE PUBBLICAZIONI. ESSE SONO FORNITE "NELLO STATO IN CUI SI TROVANO", SENZA GARANZIE DI ALCUN TIPO, ESPRESSE O IMPLICITE, INCLUSE, A TITOLO ESEMPLIFICATIVO, GARANZIE DI COMMERCIABILITA', IDONEITA' PER UNO SCOPO SPECIFICO E DI NON VIOLAZIONE.

Indice analitico

N

O

P

R

S

V

Stampato in Italia