IBM SPSS Modeler 16 Python Handbuch für Scripterstellung und Automatisierung

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen" auf Seite 265 gelesen werden.

Produktinformation

Diese Ausgabe bezieht sich auf Version 16, Release 0, Modifikation 0 von IBM SPSS Modeler und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM SPSS Modeler 16 Python Scripting and Automation Guide,
herausgegeben von International Business Machines Corporation, USA

Herausgegeben von:
TSC Germany
Kst. 2877
Oktober 2013

Inhaltsverzeichnis

Kapitel 1. Scripting

Scripting - Überblick

Scripting in IBM® SPSS Modeler ist ein leistungsstarkes Tool, mit dem Prozesse in der Benutzerschnittstelle automatisiert werden. Scripts können dieselben Arten von Aktionen durchführen, die Sie mit einer Maus oder einer Tastatur durchführen. So können Sie Aufgaben automatisieren, die bei einer manuellen Durchführung sehr viele Wiederholungen verlangen oder sehr viel Zeit beanspruchen.

Scripts können zu folgenden Zwecken verwendet werden:

- Eine bestimmte Reihenfolge für Knotenausführungen in einem Stream erzwingen und Knoten bedingt ausführen, in Abhängigkeit davon, ob die Bedingungen für die Ausführung erfüllt wurden.
- Schleifen erstellen, um Knoten im Stream wiederholt auszuführen.
- Eine automatische Abfolge von Aktionen festlegen, für die normalerweise Benutzeraktivitäten erforderlich sind. So können Sie beispielsweise ein Modell erstellen und dieses anschließend testen.
- Komplexe Prozesse einrichten, für die häufige Interventionen des Benutzers notwendig sind, wie dies beispielsweise bei Kreuzvalidierungen der Fall ist, bei denen ein Modell wiederholt generiert und getestet werden muss.
- Prozesse einrichten, mit denen Streams bearbeitet werden. Sie können zum Beispiel einen Modelltrainings-Stream ausführen und automatisch den entsprechenden Modelltest-Stream erstellen.

In diesem Kapitel finden Sie allgemeine Beschreibungen und Beispiele für Scripts auf der Streamebene, Standalone-Scripts und Scripts innerhalb von Superknoten auf der IBM SPSS Modeler-Benutzerschnittstelle. Weitere Informationen zu Scriptsprache, Syntax und Befehlen finden Sie in den nachfolgenden Kapiteln. [1]

Hinweis: Sie können keine Scripts importieren und ausführen, die in IBM SPSS Statistics innerhalb von IBM SPSS Modeler erstellt wurden.

Scripttypen

IBM SPSS Modeler verwendet drei Scripttypen:

- **Stream-Scripts** werden als Streameigenschaft gespeichert und daher zusammen mit einem bestimmten Stream gespeichert und geladen. Beispielsweise können Sie ein Stream-Script schreiben, das das Trainieren und Anwenden eines Modellnuggets automatisiert. Außerdem können Sie angeben, dass bei jeder Ausführung eines bestimmten Streams statt des Inhalts des Streamerstellungsbereichs das Script ausgeführt werden soll.
- **Standalone-Scripts** sind mit keinem bestimmten Stream verknüpft und werden in externen Textdateien gespeichert. Mit einem Standalone-Script können beispielsweise mehrere Streams gemeinsam bearbeitet werden.
- **Superknotenscripts** werden als Streameigenschaft von Superknoten gespeichert. Superknotenscripts stehen nur in Endsuperknoten zur Verfügung. Mit Superknotenscripts kann die Ausführungssequenz der Superknoteninhalte gesteuert werden. Bei Superknoten, bei denen es sich nicht um Endknoten handelt (also Quellen- oder Prozessknoten), können Sie Eigenschaften für den Superknoten bzw. die Knoten, die er enthält, direkt im Stream-Script definieren.

1. Die traditionelle IBM SPSS Modeler-Scriptsprache ist noch zur Verwendung in Verbindung mit IBM SPSS Modeler 16 verfügbar. Weitere Informationen finden Sie im *IBM SPSS Modeler 16 Handbuch für Scripterstellung und Automatisierung*. Anleitungen zur Zuordnung Ihrer vorhandenen traditionellen IBM SPSS Modeler-Scripts zu Python-Scripts finden Sie in Anhang B, „Migration von traditionellem Scripting zu Python-Scripting", auf Seite 255.

Stream-Scripts

Mit Scripts können in einem bestimmten Stream enthaltene Operationen angepasst und zusammen mit dem Stream gespeichert werden. Stream-Scripts können verwendet werden, um eine bestimmte Ausführungsreihenfolge der in einem Stream enthaltenen Endknoten vorzugeben. Die Bearbeitung des mit dem aktuellen Stream gespeicherten Scripts erfolgt im Dialogfeld "Script" des Streams.

So können Sie im Dialogfeld "Streameigenschaften" auf die Registerkarte für das Stream-Script zugreifen:

1. Wählen Sie im Menü "Extras" folgende Optionsfolge aus:
   Streameigenschaften > **Ausführung**
2. Klicken Sie auf die Registerkarte **Ausführung,** um mit den Scripts des aktuellen Streams zu arbeiten.
3. Wählen Sie den Ausführungsmodus aus: **Standard (optionales Script)**.

Mit den Symbolleistensymbolen oben im Dialogfeld "script" des Streams können Sie folgende Operationen durchführen:

- Inhalte eines bereits vorhandenen Standalone-Scripts in das Fenster importieren.
- Script als Textdatei speichern.
- Script drucken.
- Standardscript anhängen.
- Script bearbeiten (Rückgängig machen, Ausschneiden, Kopieren, Einfügen und andere gängige Editierfunktionen).
- Gesamtes aktuelles Script ausführen.
- Ausgewählte Zeilen eines Scripts ausführen.
- Script während der Ausführung stoppen. (Dieses Symbol ist nur während einer Scriptausführung aktiviert.)
- Syntax des Scripts überprüfen und etwaige Fehler zur Untersuchung im unteren Fensterbereich des Dialogfelds anzeigen.

Außerdem können Sie angeben, ob dieses Script bei Ausführung des Streams ausgeführt werden soll oder nicht. Mit der Option **Dieses Script ausführen** legen Sie fest, dass das Script bei jedem Ausführen des Streams gestartet und die im Script angegebene Ausführungsreihenfolge verwendet werden soll. Diese Einstellung führt zu einer Automatisierung auf Streamebene und sorgt für eine schnellere Modellbildung. In der Standardeinstellung wird das Script allerdings während der Streamausführung ignoriert. Auch wenn Sie die Option **Dieses Script ignorieren** auswählen, können Sie das Script stets direkt über dieses Dialogfeld ausführen.

Sie können auch den Scripting-Typ von Python-Scripting in traditionelles Scripting ändern.

Der Scripteditor umfasst die folgenden Funktionen zur Unterstützung von Script-Authoring:

- Syntaxhervorhebung; Schlüsselwörter, Literalwerte (wie Zeichenfolgen und Zahlen) und Kommentare werden hervorgehoben.
- Zeilennummerierung.
- Blockabgleich; wenn der Cursor an den Anfang eines Programmblocks gesetzt wird, wird der entsprechende Endblock ebenfalls hervorgehoben.
- Vorschlag für Auto-Vervollständigen.

Die von der Syntaxhervorhebung verwendeten Farben und Textstile können mit den Anzeigevorgaben von IBM SPSS Modeler angepasst werden. Sie können die Anzeigevorgaben aufrufen, indem Sie **Extras** > **Optionen** > **Benutzeroptionen** auswählen und auf die Registerkarte **Syntax** klicken.

Eine Liste vorgeschlagener Syntaxvervollständigungen kann aufgerufen werden, indem Sie im Kontextmenü **Automatisch vorschlagen** auswählen oder Strg+Leertaste drücken. Mit den Cursortasten können

Sie sich in der Liste nach oben und unten bewegen. Zum Einfügen des ausgewählten Texts drücken Sie dann die Eingabetaste. Drücken Sie Esc, um die automatische Vervollständigung zu verlassen, ohne den vorhandenen Text zu ändern.

Die Registerkarte **Debug** zeigt Debugnachrichten an und kann verwendet werden, um den Scriptstatus auszuwerten, sobald das Script ausgeführt wurde. Die Registerkarte **Debug** besteht aus einem schreibgeschützten Textbereich und einem einzeiligen Eingabetextfeld. Der Textbereich zeigt Text an, der an eine Standardausgabe gesendet wird, z. B. über den Python-Befehl `print`, oder die Standard-Fehlerausgabe der Scripts, z. B. über den Fehlernachrichtentext. Das Eingabetextfeld übernimmt die Eingabe des Benutzers. Diese Eingabe wird dann im Kontext des Scripts ausgewertet, das zuletzt im Dialog ausgeführt wurde (so genannter *Scriptingkontext*). Der Textbereich enthält den Befehl und die resultierende Ausgabe, sodass der Benutzer einen Trace der Befehle sehen kann. Das Eingabetextfeld enthält immer die Eingabeaufforderung (>>> bei Python-Scripting).

In den folgenden Fällen wird ein neuer Scriptingkontext erstellt:

- Ein Script wird über die Schaltfläche “Dieses Script ausführen” oder die Schaltfläche “Ausgewählte Zeilen ausführen” ausgeführt.
- Die Scriptsprache wird geändert.

Wenn ein neuer Scriptingkontext erstellt wird, wird der Inhalt des Textbereichs gelöscht.

Anmerkung: Durch die Ausführung eines Streams außerhalb des Scriptbereichs wird der Scriptkontext des Scriptbereichs nicht geändert. Die Werte von Variablen, die als Teil dieser Ausführung erzeugt werden, sind im Scriptdialog nicht sichtbar.

Standalone-Scripts

Im Dialogfeld "Standalone-Script" wird ein Script erstellt oder bearbeitet, das als Textdatei gespeichert wird. Es zeigt den Namen der Datei und bietet Optionen zum Laden, Speichern, Importieren und Ausführen von Scripts.

So öffnen Sie das Dialogfeld für Standalone-Scripts:

Wählen Sie im Hauptmenü Folgendes:

Extras > **Standalone-Script**

Für Standalone-Scripts stehen dieselbe Symbolleiste und dieselben Optionen zur Überprüfung der Script-Syntax zur Verfügung wie für Stream-Scripts. Weitere Informationen finden Sie im Thema „Stream-Scripts” auf Seite 2.

Superknotenscripts

Mithilfe der Scriptsprache von IBM SPSS Modeler können Sie Scripts innerhalb jedes beliebigen Endsuperknotens erstellen und speichern. Diese Scripts stehen ausschließlich für Endsuperknoten zur Verfügung und werden häufig beim Erstellen von Vorlagenstreams oder zum Erzwingen einer bestimmten Ausführungsreihenfolge für die Superknoteninhalte verwendet. Mit Superknotenscripts können Sie außerdem mehrere Scripts in einem Stream ausführen.

Beispiel: Angenommen, Sie müssen die Ausführungsreihenfolge für einen komplexen Stream angeben und Ihr Superknoten enthält mehrere Knoten, darunter einen Globalwerteknoten, der ausgeführt werden muss, bevor ein neues Feld, das in einem Plotknoten verwendet wird, abgeleitet wird. In diesem Fall können Sie ein Superknotenscript erstellen, das zuerst den Globalwerteknoten ausführt. Die durch diesen Knoten berechneten Werte, wie Durchschnitt oder Standardabweichung, können anschließend bei der Ausführung des Plotknotens verwendet werden.

Innerhalb eines Superknotenscripts können Sie Knoteneigenschaften auf dieselbe Weise angeben wie bei anderen Scripts. Alternativ können Sie die Eigenschaften für einen beliebigen Superknoten oder den darin gekapselten Knoten direkt über ein Stream-Script ändern und definieren. Weitere Informationen finden Sie in Kapitel 19, „Superknoteneigenschaften", auf Seite 249. Diese Methode funktioniert für Quellen- und Prozess-Superknoten ebenso wie für Endsuperknoten.

Hinweis: Da nur Endsuperknoten ihre eigenen Scripts ausführen können, steht die Registerkarte "Scripts" des Dialogfelds "Superknoten" nur für Endsuperknoten zur Verfügung.

So öffnen Sie das Dialogfeld "Superknotenscript" im Haupterstellungsbereich:

Wählen Sie einen Endsuperknoten im Streamerstellungsbereich aus und wählen Sie folgende Option im Menü "Superknoten":

Superknotenscript...

So öffnen Sie das Dialogfeld "Superknotenscript" im vergrößerten Superknoten-Erstellungsbereich:

Klicken Sie mit der rechten Maustaste auf den Superknoten-Erstellungsbereich und wählen Sie aus dem Kontextmenü folgende Optionen:

Superknotenscript...

Verwendung von Schleifen und bedingte Ausführung in Streams

Ab Version 16.0 ermöglicht SPSS Modeler es Ihnen, einige grundlegende Scripts in einem Stream zu erstellen, indem Sie Werte in verschiedenen Dialogfeldern auswählen, statt die Anweisungen direkt in der Scriptsprache schreiben zu müssen. Die beiden Haupttypen von Scripts, die Sie auf diese Weise erstellen können, sind einfache Schleifen sowie eine Möglichkeit zum Ausführen von Knoten, wenn eine Bedingung erfüllt ist.

Sie können Regeln zur Verwendung von Schleifen und für die bedingte Ausführung in einem Stream kombinieren. Sie haben z. B. Daten über den Kfz-Verkauf von Herstellern weltweit Sie könnten eine Schleife definieren, um die Daten in einem Stream zu verarbeiten, Details nach dem Herstellerland ermitteln und die Daten in unterschiedlichen Diagrammen ausgeben mit Details wie Umsatzvolumen nach Modell, Emissionsstufen nach Hersteller und Motorgröße usw. Wenn Sie nur Daten europäischer Hersteller analysieren wollen, könnten Sie auch Bedingungen zur Schleife hinzufügen, die verhindern, dass Diagramme für amerikanische und asiatische Hersteller erstellt werden.

Anmerkung: Da sowohl die Verwendung von Schleifen als auch die bedingte Ausführung auf Hintergrundscripts basieren, werden sie nur auf einen gesamten Stream bei dessen Ausführung angewendet.

- **Verwendung von Schleifen**. Sie können mit Schleifen sich wiederholende Tasks automatisieren. Dabei könnte z. B. eine bestimmte Anzahl von Knoten zu einem Stream hinzugefügt und jeweils ein Knotenparameter einzeln geändert werden. Alternativ könnten Sie die wiederholte Ausführung eines Streams oder einer Verzweigung für eine bestimmte Anzahl von Malen steuern (siehe folgende Beispiele):
  - Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie die Quelle bei jeder Ausführung.
  - Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie bei jeder Ausführung den Wert einer Variablen.
  - Führen Sie den Stream eine bestimmte Anzahl Male aus und geben Sie bei jeder Ausführung ein Extrafeld ein.
  - Erstellen Sie ein Modell eine bestimmte Anzahl Male und ändern Sie bei jeder Erstellung eine Modelleinstellung.

- **Bedingte Ausführung**. Damit können Sie steuern, wie Endknoten in Abhängigkeit von vordefinierten Bedingungen ausgeführt werden (siehe folgende Beispiele):
  - Steuern Sie, ob ein Knoten ausgeführt wird, in Abhängigkeit davon, ob ein bestimmter Wert "true" oder "false" ist.
  - Definieren Sie, ob Schleifen parallel oder sequenziell ausgeführt werden.

Sowohl die Verwendung von Schleifen als auch die bedingte Ausführung werden auf der Registerkarte **Ausführung** im Dialogfeld **Streameigenschaften** definiert. Alle Knoten, die in bedingten oder Schleifenanforderungen verwendet werden, werden mit einem zusätzlichen Symbol im Streamerstellungsbereich angezeigt, um darauf hinzuweisen, dass sie Teil einer Schleifen- oder einer bedingten Ausführung sind.

Sie können auf drei Arten auf die Registerkarte **Ausführung** zugreifen:

- Über die Menüs oben im Hauptdialogfeld:
  1. Wählen Sie im Menü "Extras" folgende Optionsfolge aus:

     Streameigenschaften > **Ausführung**
  2. Klicken Sie auf die Registerkarte **Ausführung**, um mit den Scripts des aktuellen Streams zu arbeiten.
- Aus einem Stream:
  1. Klicken Sie mit der rechten Maustaste auf einen Knoten und wählen Sie **Verwendung von Schleifen/bedingte Ausführung** aus.
  2. Wählen Sie die entsprechende Option aus dem Untermenü aus.
- Klicken Sie in der Grafiksymbolleiste oben im Hauptdialogfeld auf das Symbol für die Streameigenschaften.

Wenn Sie zum ersten Mal Details zur Schleifen- oder zur bedingten Ausführung eingeben, wählen Sie auf der Registerkarte **Ausführung** den Ausführungsmodus **Verwendung von Schleifen/bedingte Ausführung** aus und wählen Sie dann die Unterregisterkarte **Bedingt** oder **Verwendung von Schleifen** aus.

Verwendung von Schleifen in Streams

Mit Schleifen können Sie sich wiederholende Tasks in Streams automatisieren (siehe folgende Beispiele):

- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie die Quelle bei jeder Ausführung.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie bei jeder Ausführung den Wert einer Variablen.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und geben Sie bei jeder Ausführung ein Extrafeld ein.
- Erstellen Sie ein Modell eine bestimmte Anzahl Male und ändern Sie bei jeder Erstellung eine Modelleinstellung.

Sie definieren die zu erfüllenden Bedingungen auf der Unterregisterkarte **Verwendung von Schleifen** der Registerkarte **Ausführung** des Streams. Um die Unterregisterkarte anzuzeigen, wählen Sie den Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aus.

Alle definierten Anforderungen für die Verwendung von Schleifen werden bei der Ausführung des Streams wirksam, wenn der Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aktiviert wurde. Optional können Sie den Script-Code für Ihre Schleifenanforderungen generieren und ihn in den Scripteditor einfügen, indem Sie unten rechts auf der Unterregisterkarte **Verwendung von Schleifen** auf **Einfügen...** klicken. Die Anzeige der Hauptregisterkarte **Ausführung** ändert sich und der Ausführungsmodus **Standard (optionales Script)** wird mit dem Script im oberen Teil der Registerkarte angezeigt. Dies bedeutet, dass Sie eine Schleifenstruktur mithilfe der verschiedenen Optionen im Dialogfeld **Verwendung von Schleifen** definieren können, bevor Sie ein Script generieren, das Sie im Scripteditor

weiter anpassen können. Wenn Sie auf **Einfügen...** klicken, werden auch alle Bedingungen für die bedingte Ausführung, die Sie definiert haben, im generierten Script angezeigt.

So definieren Sie eine Schleife:

1. Erstellen Sie einen Iterationsschlüssel, um die Hauptschleifenstuktur zu definieren, die in einem Stream ausgeführt werden soll. Weitere Informationen finden Sie in Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams.
2. Definieren Sie bei Bedarf eine oder mehrere Iterationsvariablen. Weitere Informationen finden Sie in Erstellen einer Iterationsvariablen zur Verwendung von Schleifen in Stream.
3. Die Iterationen und die Variablen, die Sie erstellt haben, werden im Hauptteil der Unterregisterkarte angezeigt. Standardmäßig werden Iterationen in der aufgeführten Reihenfolge ausgeführt. Um eine Iteration in der Liste nach oben oder unten zu verschieben, wählen Sie sie durch Anklicken aus und ändern Sie die Reihenfolge mit dem Auf- oder Abwärtspfeil in der rechten Spalte der Unterregisterkarte.

Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams

Sie verwenden einen Iterationsschlüssel, um die Hauptschleifenstuktur zu definieren, die in einem Stream ausgeführt werden soll. Bei der Analyse von Kfz-Verkäufen könnten Sie z. B. einen Streamparameter namens *Herstellerland* erstellen und diesen als Iterationsschlüssel verwenden. Beim Ausführen des Streams wird dieser Schlüssel während jeder Iteration auf die einzelnen Länderwerte in Ihren Daten gesetzt. Verwenden Sie das Dialogfeld **Iterationsschlüssel definieren**, um den Schlüssel zu definieren.

Zum Öffnen des Dialogfelds wählen Sie entweder die Schaltfläche **Iterationsschlüssel...** unten links auf der Unterregisterkarte **Verwendung von Schleifen** aus oder klicken Sie mit der rechten Maustaste auf einen beliebigen Knoten im Stream und wählen Sie entweder **Verwendung von Schleifen/Bedingte Ausführung** > **Iterationsschlüssel definieren (Felder)** oder **Verwendung von Schleifen/Bedingte Ausführung** > **Iterationsschlüssel definieren (Werte)** aus. Wenn Sie das Dialogfeld vom Stream aus öffnen, sind einige der Felder wie der Knotenname möglicherweise bereits ausgefüllt.

Füllen Sie die folgenden Felder aus, um einen Iterationsschlüssel zu definieren:

Iteration nach. Sie können eine der folgenden Optionen auswählen:

- **Streamparameter - Felder**. Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert eines vorhandenen Streamparameters nacheinander auf jedes angegebene Feld setzt.
- **Streamparameter - Werte**. Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert eines vorhandenen Streamparameters nacheinander auf jeden angegebenen Wert setzt.
- **Knoteneigenschaft - Felder**. Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert einer Knoteneigenschaft nacheinander auf jedes angegebene Feld setzt.
- **Knoteneigenschaft - Werte**. Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert einer Knoteneigenschaft nacheinander auf jeden angegebenen Wert setzt.

Festzulegen. Wählen Sie das Element aus, dessen Wert bei jeder Schleifenausführung festgelegt werden soll. Sie können eine der folgenden Optionen auswählen:

- **Parameter**. Nur verfügbar, wenn Sie **Streamparameter - Felder** oder **Streamparameter - Werte** auswählen. Wählen Sie den erforderlichen Parameter aus der Liste der verfügbaren Parameter aus.
- **Knoten**. Nur verfügbar, wenn Sie **Knoteneigenschaft - Felder** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie den Knoten aus, für die Sie eine Schleife definieren wollen. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs, Ausgabe-, Export- oder Modellanwendungsknoten.

- **Eigenschaft**. Nur verfügbar, wenn Sie **Knoteneigenschaft - Felder** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie die Eigenschaft des Knotens aus der Liste aus.

Zu verwendende Felder. Nur verfügbar, wenn Sie **Streamparameter - Felder** oder **Knoteneigenschaft - Felder** auswählen. Wählen Sie die Felder in einem Knoten aus, um die Iterationswerte anzugeben. Sie können eine der folgenden Optionen auswählen:

- **Knoten**. Nur verfügbar, wenn Sie **Streamparameter - Felder** auswählen. Wählen Sie den Knoten mit den Details aus, für die Sie eine Schleife konfigurieren möchten. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs, Ausgabe-, Export- oder Modellanwendungsknoten.
- **Feldliste**. Klicken Sie auf die Listenschaltfläche in der rechten Spalte, um das Dialogfeld **Felder auswählen** anzuzeigen, in dem Sie die Felder im Knoten zur Angabe der Iterationsdaten auswählen können. Weitere Informationen finden Sie in „Auswählen von Feldern für Iterationen" auf Seite 8.

Zu verwendende Werte. Nur verfügbar, wenn Sie **Streamparameter - Werte** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie die Werte im ausgewählten Feld aus, die als Iterationswerte verwendet werden sollen. Sie können eine der folgenden Optionen auswählen:

- **Knoten**. Nur verfügbar, wenn Sie **Streamparameter - Werte** auswählen. Wählen Sie den Knoten mit den Details aus, für die Sie eine Schleife konfigurieren möchten. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs, Ausgabe-, Export- oder Modellanwendungsknoten.
- **Feldliste**. Wählen Sie das Feld im Knoten aus, das die Iterationsdaten angeben soll.
- **Werteliste**. Klicken Sie auf die Listenschaltfläche in der rechten Spalte, um das Dialogfeld **Werte auswählen** anzuzeigen, in dem Sie die Werte im Feld zur Angabe der Iterationsdaten auswählen können.

Erstellen einer Iterationsvariablen zur Verwendung von Schleifen in Streams

Sie können Iterationsvariablen verwenden, um die Werte von Streamparametern oder Eigenschaften ausgewählter Knoten in einem Stream bei jeder Schleifenausführung zu ändern. Wenn Ihre Streamschleife z. B. Kfz-Verkaufsdaten analysiert und *Herstellerland* als Iterationsschlüssel verwendet, können Sie eine Diagrammausgabe erhalten, die den Umsatz nach Modell anzeigt, und eine andere Diagrammausgabegabe, die Abgasemissionen anzeigt. In diesen Fällen könnten Sie Iterationsvariablen erstellen, die neue Titel für die entsprechenden Diagramme erstellen, z. B. *Schwedische Fahrzeugemissionen* und *Japanische Kfz-Verkäufe nach Modell*. Verwenden Sie das Dialogfeld **Iterationsvariable definieren**, um die erforderlichen Variablen zu definieren.

Zum Öffnen des Dialogfelds wählen Sie entweder die Schaltfläche **Iterationsvariable...** unten links auf der Unterregisterkarte **Verwendung von Schleifen** aus oder klicken Sie mit der rechten Maustaste auf einen beliebigen Knoten im Stream und wählen Sie **Verwendung von Schleifen/Bedingte Ausführung** > **Iterationsvariable definieren** aus.

Füllen Sie die folgenden Felder aus, um eine Iterationsvariable zu definieren:

Ändern. Wählen Sie den Typ von Attribut aus, den Sie ändern wollen. Sie können zwischen **Streamparameter** oder **Knoteneigenschaft** wählen.

- Wenn Sie **Streamparameter** auswählen, wählen Sie den erforderlichen Parameter aus und definieren Sie dann mit einer der folgenden Optionen (sofern für Ihren Stream verfügbar), auf welchen Wert dieser Parameter bei jeder Schleifeniteration gesetzt werden soll:

- **Globale Variable**. Wählen Sie die globale Variable aus, auf die der Streamparameter gesetzt werden soll.
- **Tabellenausgabezelle**. Um einen Streamparameter auf den Wert in einer Tabellenausgabezelle zu setzen, wählen Sie die Tabelle aus der Liste aus und geben Sie die zu verwendende Zeile und Spalte ein.
- **Manuell eingeben**. Wählen Sie diese Option aus, wenn Sie manuell einen Wert eingeben wollen, den dieser Parameter in den einzelnen Iterationen annehmen soll. Wenn Sie zur Unterregisterkarte **Verwendung von Schleifen** zurückkehren, wird eine neue Spalte erstellt, in die Sie den erforderlichen Text eingeben können.

• Wenn Sie **Knoteneigenschaft** auswählen, wählen Sie den erforderlichen Knoten und eine seiner Eigenschaften aus und legen Sie dann den Wert fest, der für diese Eigenschaft verwendet werden soll. Definieren Sie den neuen Eigenschaftswert mit einer der folgenden Optionen:
  - **Allein**. Der Eigenschaftswert verwendet den Iterationsschlüsselwert. Weitere Informationen finden Sie in „Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams" auf Seite 6.
  - **Als Präfix für Stamm**. Verwendet den Iterationsschlüsselwert als Präfix für den Wert, den Sie in das Feld **Stamm** eingeben.
  - **Als Suffix für Stamm**. Verwendet den Iterationsschlüsselwert als Suffix für den Wert, den Sie in das Feld **Stamm** eingeben.

  Wenn Sie die Präfix- oder die Suffix-Option ausgewählt haben, werden Sie aufgefordert, den zusätzlichen Text in das Feld **Stamm** einzugeben. Wenn Ihr Iterationsschlüssel z. B. *Herstellerland* lautet und Sie die Option **Als Präfix für Stamm** auswählen, könnten Sie in dieses Feld - *Verkauf nach Modell* eingeben.

Auswählen von Feldern für Iterationen

Beim Erstellen von Iterationen können Sie über das Dialogfeld **Felder auswählen** ein oder mehrere Felder auswählen.

Sortieren nach. Sie können verfügbare Felder für die Anzeige sortieren, indem Sie eine der folgenden Optionen auswählen:

- **Natürlich**. Zeigt die Felder in der Reihenfolge an, in der im aktuellen Datenstream an den aktuellen Knoten übergeben wurden.
- **Name**. Verwendet eine alphabetische Reihenfolge zum Sortieren der Felder für die Anzeige.
- **Typ**. Zeigt Felder sortiert nach ihrem Messniveau an. Diese Option ist hilfreich bei der Auswahl von Feldern mit einem bestimmten Messniveau.

Wählen Sie Felder einzeln aus der Liste aus oder wählen Sie bei gedrückter Umschalttaste oder bei gedrückter Steuertaste mehrere Felder aus. Sie können auch die Schaltflächen unter der Liste verwenden, um Gruppen von Feldern basierend auf deren Messniveau auszuwählen oder um alle Felder in der Tabelle aus- oder abzuwählen.

Die zur Auswahl verfügbaren Felder werden gefiltert, sodass nur die Felder angezeigt werden, die für den verwendeten Streamparameter oder die verwendete Knoteneigenschaft geeignet sind. Wenn Sie z. B. einen Streamparameter mit dem Speichertyp "Zeichenfolge" verwenden, werden nur Felder mit diesem Speichertyp angezeigt.

Bedingte Ausführung in Streams

Mit bedingter Ausführung können Sie steuern, wie Endknoten ausgeführt werden, und zwar in Abhängigkeit davon, ob die Streaminhalte den definierten Bedingungen entsprechen. Beispiele:

- Steuern Sie, ob ein Knoten ausgeführt wird, in Abhängigkeit davon, ob ein bestimmter Wert "true" oder "false" ist.
- Definieren Sie, ob Schleifen parallel oder sequenziell ausgeführt werden.

Sie definieren die zu erfüllenden Bedingungen auf der Unterregisterkarte **Bedingt** der Registerkarte **Ausführung** des Streams. Um die Unterregisterkarte anzuzeigen, wählen Sie den Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aus.

Alle definierten Anforderungen für die bedingte Ausführung werden bei der Ausführung des Streams wirksam, wenn der Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aktiviert wurde. Optional können Sie den Script-Code für die Anforderungen Ihrer bedingten Ausführung generieren und ihn in den Scripteditor einfügen, indem Sie unten rechts auf der Unterregisterkarte **Bedingt** auf **Einfügen...** klicken. Die Anzeige der Hauptregisterkarte **Ausführung** ändert sich und der Ausführungsmodus **Standard (optionales Script)** wird mit dem Script im oberen Teil der Registerkarte angezeigt. Dies bedeutet, dass Sie Bedingungen mithilfe der verschiedenen Optionen im Dialogfeld **Verwendung von Schleifen** definieren können, bevor Sie ein Script generieren, das Sie im Scripteditor weiter anpassen können. Wenn Sie auf **Einfügen...** klicken, werden auch alle von Ihnen definierten Schleifenanforderungen im generierten Script angezeigt.

So definieren Sie eine Bedingung:

1. Klicken Sie in der rechten Spalte der Unterregisterkarte **Bedingt** auf die Schaltfläche **Ausführungsanweisung hinzufügen**  , um das Dialogfeld für bedingte Ausführungsanweisungen zu öffnen. In diesem Dialogfeld geben Sie die Bedingung an, die erfüllt sein muss, damit der Knoten ausgeführt wird.
2. Geben Sie im Dialogfeld für bedingte Ausführungsanweisungen Folgendes an:
   a. **Knoten**. Wählen Sie den Knoten aus, für die Sie eine bedingte Ausführung konfigurieren wollen. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur Knoten einer der folgenden Kategorien angezeigt werden: Export-, Diagramm-, Modellierungs- oder Ausgabeknoten.
   b. **Bedingung basierend auf**. Geben Sie die Bedingung an, die erfüllt sein muss, damit der Knoten ausgeführt wird. Sie können unter vier Optionen wählen: **Streamparameter**, **Globale Variable**, **Tabellenausgabezelle** oder **Immer wahr**. Welche Details Sie in der unteren Hälfte des Dialogfelds angeben können, wird von der ausgewählten Bedingung gesteuert.
      - **Streamparameter**. Wählen Sie den Parameter aus der Liste der verfügbaren Parameter aus und wählen Sie dann den Operator für diesen Parameter aus. Beispielsweise kann der Operator **Größer als**, **Gleich**, **Kleiner als**, **Zwischen** usw. sein. Dann geben Sie je nach Operator den Wert oder Minimal- oder Maximalwerte ein.
      - **Globale Variable**. Wählen Sie die Variable aus der Liste der verfügbaren Variablen aus. Beispielsweise kann die Variable **Mittelwert**, **Summe**, **Minimum**, **Maximum**oder **Standardabweichung** sein. Wählen Sie dann den Operator und die erforderlichen Werte aus.
      - **Tabellenausgabezelle**. Wählen Sie den Tabellenknoten aus der Liste der verfügbaren Elemente aus und wählen Sie dann die Zeile oder Spalte in der Tabelle aus. Wählen Sie dann den Operator und die erforderlichen Werte aus.
      - **Immer wahr**. Wählen Sie diese Option aus, wenn der Knoten immer ausgeführt werden muss. Wenn Sie diese Option auswählen, müssen keine weiteren Parameter mehr ausgewählt werden.
3. Wiederholen Sie Schritt 1 und 2 so oft, bis Sie alle erforderlichen Bedingungen definiert haben. Der ausgewählte Knoten und die Bedingung, die erfüllt werden muss, damit der Knoten ausgeführt wird, werden im Hauptteil der Unterregisterkarte in den Spalten **Ausführungsknoten** bzw. **Wenn diese Bedingung wahr ist** angezeigt.
4. Standardmäßig werden Knoten und Bedingungen in der aufgeführten Reihenfolge ausgeführt. Um einen Knoten und eine Bedingung in der Liste nach oben oder unten zu verschieben, wählen Sie diese durch Anklicken aus und ändern Sie die Reihenfolge mit dem Auf- oder Abwärtspfeil in der rechten Spalte der Unterregisterkarte.

Darüber hinaus können Sie die folgenden Optionen unten in der Unterregisterkarte **Bedingt** festlegen:

- **Alle der Reihenfolge nach auswerten**. Wählen Sie diese Option aus, um jede Bedingung in der auf der Unterregisterkarte aufgeführten Reihenfolge auszuwerten. Die Knoten, für die die Bedingungen erfüllt sind, werden ausgeführt, sobald alle Bedingungen ausgewertet wurden.
- **Jeweils nur einen ausführen**. Nur verfügbar, wenn **Alle der Reihenfolge nach auswerten** ausgewählt wurde. Wenn Sie diese Option auswählen und eine Bedingung als "True" ausgewertet wird, wird der Knoten ausgeführt, dem diese Bedingung zugeordnet ist, bevor die nächste Bedingung ausgewertet wird.
- **Auswerten bis zum ersten Treffer**. Wenn Sie diese Option auswählen, wird nur der erste Knoten ausgeführt, für den die angegebenen Bedingungen als "True" ausgewertet werden.

Ausführen und Unterbrechen von Scripts

Es gibt mehrere Möglichkeiten zur Ausführung von Scripts. Beispielsweise führt die Schaltfläche "Dieses Script ausführen" im Dialogfeld für das Stream-Script oder Standalone-Script das vollständige Script aus:

Abbildung 1. Schaltfläche "Dieses Script ausführen"

Die Schaltfläche "Ausgewählte Zeilen ausführen" führt eine einzelne Zeile oder einen Block benachbarter Zeilen aus, die Sie im Script ausgewählt haben:

Abbildung 2. Schaltfläche "Ausgewählte Zeilen ausführen"

Zum Ausführen von Scripts stehen folgende Methoden zur Auswahl:

- Klicken Sie im Dialogfeld für ein Stream-Script oder ein Standalone-Script auf die Schaltfläche "Dieses Script ausführen" oder "Ausgewählte Zeilen ausführen".
- Ausführen eines Streams mit **Dieses Script ausführen** als Standardausführungsmethode.
- Verwenden Sie das Flag `-execute` beim Start im interaktiven Modus. Weitere Informationen finden Sie im Thema „Verwenden von Befehlszeilenargumenten” auf Seite 53.

Hinweis: Ein Superknotenscript wird bei der Ausführung des Superknotens ausgeführt, sofern Sie **Dieses Script ausführen** im Superknotenscript-Dialogfeld ausgewählt haben.

Unterbrechen der Scriptausführung

Während der Scriptausführung ist im Dialogfeld "Stream-Script" die rote Stoppschaltfläche aktiviert. Mit dieser Schaltfläche können Sie die Ausführung des Scripts und aller aktuellen Streams abbrechen.

Suchen und ersetzen

Das Dialogfeld "Suchen/Ersetzen" ist in Situationen verfügbar, in denen Sie Script- oder Ausdruckstext bearbeiten, z. B. im Script-Editor und bei der Definition von Vorlagen im Berichtsknoten. Während der Bearbeitung von Text in einem dieser Bereiche können Sie durch Drücken von `Strg+F` das Dialogfeld aufrufen. Achten Sie dabei darauf, dass sich der Cursor über einem Textbereich befindet. So können Sie beispielsweise bei der Arbeit in einem Füllerknoten das Dialogfeld aus einem der Textbereiche auf der Registerkarte "Einstellungen" aufrufen oder über das Textfeld im Expression Builder.

1. Achten Sie darauf, dass sich der Cursor in einem Textfeld befindet, und drücken Sie Strg+F, um das Dialogfeld "Suchen/Ersetzen" aufzurufen.

2. Geben Sie den Text ein, nach dem Sie suchen möchten, oder treffen Sie eine Auswahl aus der Drop-down-Liste der kürzlich durchsuchten Elemente.
3. Geben Sie, falls erforderlich, den Ersatztext ein.
4. Klicken Sie auf **Weitersuchen**, um die Suche zu starten.
5. Klicken Sie auf **Ersetzen**, um die aktuelle Auswahl zu ersetzen, oder **Alle ersetzen**, um alle bzw. die ausgewählten Instanzen zu ersetzen.
6. Das Dialogfeld wird nach jedem Vorgang geschlossen. Drücken Sie in einem Textbereich die Taste F3, um den letzten Suchvorgang zu wiederholen, bzw. drücken Sie Strg+F, um erneut auf das Dialogfeld zuzugreifen.

Suchoptionen

Groß-/Kleinschreibung beachten. Gibt an, ob beim Suchvorgang die Groß- und Kleinschreibung berücksichtigt wird; beispielsweise, ob *myvar* als identisch mit *myVar* betrachtet wird. Ersetzungstext wird unabhängig von dieser Einstellung stets genau so eingefügt, wie er eingegeben wurde.

Nur ganze Wörter. Gibt an, ob beim Suchvorgang auch Wortteile gefunden werden. Wenn diese Option ausgewählt ist, werden bei einer Suche nach *Tag* Terme wie *Tagesordnung* oder *Tag-und-Nacht-Gleiche* nicht als Treffer ausgegeben.

Reguläre Ausdrücke. Gibt an, ob die Syntax für reguläre Ausdrücke verwendet werden soll (siehe nächsten Abschnitt). Bei Auswahl dieser Option wird die Option **Nur ganze Wörter** inaktiviert und ihr Wert ignoriert.

Nur ausgewählter Text. Legt den Suchumfang bei Verwendung der Option **Alle ersetzen** fest.

Syntax für reguläre Ausdrücke

Mithilfe von regulären Ausdrücken können Sie nach Sonderzeichen (z. B. Tabulatoren oder Zeilenumbrüche), Zeichenklassen bzw. -bereichen, wie *a* bis *d*, beliebige Ziffer oder Nichtziffer, und nach Begrenzungen, wie beispielsweise Zeilenanfang bzw. Zeilenende, suchen. Ein Muster mit regulärem Ausdruck beschreibt die Struktur der Zeichenfolge, die der Ausdruck in einer Eingabezeichenfolge zu finden versucht. Die folgenden Arten von Konstrukten mit regulärem Ausdruck werden unterstützt.

Tabelle 1. Zeichenübereinstimmungen

Zeichen	**Übereinstimmung/Bedeutung**
x	Das Zeichen x
\\	Umgekehrter Schrägstrich
\0n	Das Zeichen mit dem Oktalwert 0n (0 <= n <= 7)
\0nn	Das Zeichen mit dem Oktalwert 0nn (0 <= n <= 7)
\0mnn	Das Zeichen mit dem Oktalwert 0mnn (0 <= m <= 3; 0 <= n <= 7)
\xhh	Das Zeichen mit dem Hexadezimalwert 0xhh
\uhhhh	Das Zeichen mit dem Hexadezimalwert 0xhhhh
\t	Das Tabulatorzeichen ('\u0009')
\n	Das Zeilenvorschubzeichen ('\u000A')
\r	Das Rücklaufzeichen ('\u000D')
\f	Das Formularvorschubzeichen ('\u000C')
\a	Das Alarmzeichen ('\u0007')
\e	Das Escapezeichen ('\u001B')
\cx	Das Steuerzeichen, das x entspricht

Tabelle 2. Übereinstimmende Zeichenklassen

Zeichenklassen	Übereinstimmung/Bedeutung
[abc]	a, b oder c (einfache Klasse)
[^abc]	Beliebiges Zeichen mit Ausnahme von a, b oder c (Differenzmenge)
[a-zA-Z]	a bis einschließlich z bzw. A bis einschließlich Z (Bereich)
[a-d[m-p]]	a bis d bzw. m bis P (Vereinigungsmenge) Alternative Angabemöglichkeit: [a-dm-p]
[a-z&&[def]]	a bis z und d, e bzw. f (Schnittmenge)
[a-z&&[^bc]]	a bis z mit Ausnahme von b und c (Differenzmenge) Alternative Angabemöglichkeit: [ad-z]
[a-z&&[^m-p]]	a bis z mit Ausnahme von m bis p (Differenzmenge) Alternative Angabemöglichkeit: [a-lq-z]

Tabelle 3. Vordefinierte Zeichenklassen

Vordefinierte Zeichenklassen	Übereinstimmung/Bedeutung
.	Beliebiges Zeichen (evtl. Übereinstimmung mit Zeilenabschlusszeichen)
\d	Beliebige Ziffer: [0-9]
\D	Nichtziffer: [^0-9]
\s	Ein Leerzeichen: [\t\n\x0B\f\r]
\S	Ein Nichtleerzeichen: [^\s]
\w	Ein Zeichen: [a-zA-Z_0-9]
\W	Ein Nichtzeichen: [^\w]

Tabelle 4. Übereinstimmungen mit Begrenzungszeichen

Zeichen(folgen) für Begrenzungszeichen	Übereinstimmung/Bedeutung
^	Zeilenanfang
$	Zeilenende
\b	Wortgrenze
\B	Nichtwortgrenze
\A	Beginn der Eingabe
\Z	Ende der Eingabe mit Ausnahme des letzten Abschlusszeichens, sofern vorhanden
\z	Ende der Eingabe

Weitere Informationen zur Verwendung regulärer Ausdrücke und einige Beispiele finden Sie in http://www.ibm.com/developerworks/java/tutorials/j-introtojava2/section9.html.

Beispiele

Der folgende Code sucht nach drei Ziffern am Anfang einer Zeichenfolge und gleicht diese ab:

```
^[0-9]{3}
```

Der folgende Code sucht nach drei Ziffern am Ende einer Zeichenfolge und gleicht diese ab:

```
[0-9]{3}$
```

Kapitel 2. Scriptsprache

Scriptsprache - Überblick

Mit den Scriptfunktionen für IBM SPSS Modeler können Sie Scripts erstellen, die mit der SPSS Modeler-Benutzerschnittstelle arbeiten, Ausgabeobjekte manipulieren und Befehlssyntax ausführen. Sie können Scripts direkt aus SPSS Modeler ausführen.

Scripts in IBM SPSS Modeler werden in der Scriptsprache Python geschrieben. Die von IBM SPSS Modeler verwendete Java-basierte Implementierung von Python wird als Jython bezeichnet. Die Scriptsprache besteht aus folgenden Elementen:

- Format für die Referenzierung von Knoten, Streams, Projekten, Ausgaben und anderen IBM SPSS Modeler-Objekten
- Set mit Scriptanweisungen bzw. Befehlen, die zur Bearbeitung dieser Objekte verwendet werden können.
- Script-Ausdruckssprache für die Festlegung der Werte von Variablen, Parametern und anderen Objekten.
- Unterstützung für Kommentare, Fortsetzungen und Blöcke mit Literaltext.

In den folgenden Abschnitten werden die Python-Scriptsprache, die Jython-Implementierung von Python und die grundlegende Syntax für das erste Arbeiten mit Scripting in IBM SPSS Modeler beschrieben. Informationen zu speziellen Eigenschaften und Befehlen finden Sie in den nachfolgenden Abschnitten.

Python und Jython

Jython ist eine Implementierung der Python-Scriptsprache, die in Java geschrieben ist und in die Java-Plattform integriert ist. Python ist eine leistungsfähige objektorientierte Scriptsprache. Jython ist nützlich, da es die Produktivitätsfunktionen einer ausgereiften Scriptsprache bereitstellt und anders als Python in einer beliebigen Umgebung ausgeführt werden kann, die Java Virtual Machine (JVM) unterstützt. Dies bedeutet, dass die Java-Bibliotheken in JVM beim Schreiben von Programmen zur Verfügung stehen. Mit Jython können Sie sich diesen Unterschied zunutze machen und die Syntax und die meisten Funktionen der Python-Sprache verwenden.

Als Scriptsprache ist Python (und deren Jython-Implementierung) einfach zu erlernen und effizient zu codieren und sie hat die minimal erforderliche Struktur zum Erstellen eines lauffähigen Programms. Code kann zeilenweise interaktiv eingegeben werden. Python ist eine interpretierte Scriptsprache; es gibt keinen Vorkompilierungsschritt wie in Java. Python-Programme sind einfache Textdateien, die bei ihrer Eingabe (nach der Analyse auf Syntaxfehler) interpretiert werden. Einfache Ausdrücke wie z. B. definierte Werte und auch komplexere Aktionen wie Funktionsdefinitionen werden unverzüglich ausgeführt und sind sofort einsatzbereit. Änderungen am Code können schnell getestet werden. Die Scriptinterpretation hat jedoch einige Nachteile. So ist beispielsweise die Verwendung einer nicht definierten Variablen kein Compilerfehler und wird deshalb erst bei der Ausführung der Anweisung erkannt, in der die Variable verwendet wird. In diesem Fall kann das Programm bearbeitet und ausgeführt werden, um den Fehler zu beheben.

Python betrachtet alles, d. h. alle Daten und den gesamten Code, als Objekt. Sie können diese Objekte daher mit Codezeilen manipulieren. Einige Typen wie Zahlen und Zeichenfolgen werden praktischerweise als Werte und nicht als Objekte betrachtet; dies wird von Python unterstützt. Es gibt einen unterstützten Nullwert. Dieser Nullwert hat den reservierten Namen `None`.

Eine umfassendere Einführung in Python- und Jython-Scripting sowie einige Beispielscripts finden Sie in www.ibm.com/developerworks/java/tutorials/j-jython1 und www.ibm.com/developerworks/java/tutorials/j-jython2.

Python-Scripting

Dieses Handbuch zur Python-Scriptsprache bietet eine Einführung in die Komponenten, die am häufigsten beim Scripting in IBM SPSS Modeler verwendet werden, einschließlich der Konzepte und Programmiergrundlagen. Es liefert Ihnen ausreichend Kenntnisse, um mit der Entwicklung Ihrer eigenen Python-Scripts zu beginnen, die in IBM SPSS Modeler verwendet werden sollen.

Operationen

Die Zuordnung erfolgt mittels eines Gleichheitszeichens (=). Wenn Sie z. B. den Wert "3" einer Variablen namens "x" zuweisen wollen, würden Sie die folgende Anweisung verwenden:

```
x = 3
```

Das Gleichheitszeichen wird auch für die Zuordnung von Zeichenfolgedaten zu einer Variablen verwendet. Wenn Sie z. B. den Wert "ein Zeichenfolgewert" einer Variablen namens "y" zuweisen wollen, würden Sie die folgende Anweisung verwenden:

```
y = "ein Zeichenfolgewert"
```

In der folgenden Tabelle werden einige der gängigen Vergleichs- und numerischen Operationen sowie deren Beschreibungen aufgelistet.

Tabelle 5. Allgemeine Vergleichsoperationen und numerische Operationen

Operation	Beschreibung
`x < y`	Ist x kleiner als y?
`x > y`	Ist x größer als y?
`x <= y`	Ist x kleiner-gleich y?
`x >= y`	Ist x größer-gleich y?
`x == y`	Ist x gleich y?
`x != y`	Ist x ungleich y?
`x <> y`	Ist x ungleich y?
`x + y`	y zu x addieren
`x - y`	y von x substrahieren
`x * y`	x mit y multiplizieren
`x / y`	x durch y dividieren
`x ** y`	x hoch y.

Listen

Listen sind Folgen von Elementen. Eine Liste kann eine beliebige Anzahl von Elementen enthalten und die Elemente der Liste können einen beliebigen Objekttyp haben. Listen kann man sich auch als Arrays vorstellen. Die Anzahl der Elemente in einer Liste kann sich durch Hinzufügen, Entfernen oder Ersetzen von Elementen verringern oder erhöhen.

Beispiele

`[]` Eine beliebige Liste ohne Inhalt.

`[1]`	Eine Liste mit einem einzigen Element: einer ganzen Zahl.
`["Mike", 10, "Don", 20]`	Eine Liste mit vier Elementen: zwei Zeichenfolgeelemente und zwei ganzzahlige Elemente.
`[[],[7],[8,9]]`	Eine aus Listen bestehende Liste. Jede Unterliste ist entweder eine Liste ohne Inhalt oder eine Liste mit ganzzahligen Elementen.
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	Eine Liste mit ganzen Zahlen. Dieses Beispiel veranschaulicht die Verwendung von Variablen und Ausdrücken.

Sie können eine Liste einer Variablen zuordnen. Beispiel:

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

Sie können dann auf bestimmte Elemente der Liste zugreifen. Beispiel:

```
mylist[0]
```

Dies resultiert in der folgenden Ausgabe:

```
one
```

Die Zahl in eckigen Klammern ([]) wird als *Index* bezeichnet und verweist auf ein bestimmtes Element der Liste. Die Elemente einer Liste werden mit 0 beginnend indexiert.

Sie können auch einen Bereich von Elementen einer Liste auswählen. Dies wird als *Slicing* bezeichnet. `x[1:3]` beispielsweise wählt das zweite und das dritte Element von `x` aus. Der Index für das Bereichsende muss um 1 größer sein als der Index des letzten auszuwählenden Elements.

Zeichenfolgen

Eine *Zeichenfolge* ist eine unveränderliche Folge von Zeichen, die als Wert behandelt wird. Zeichenfolgen unterstützen alle Funktionen und Operationen mit unveränderlichen Folgen, die in einer neuen Zeichenfolge resultieren. Beispiel: `"abcdef"[1:4]` wird als `"bcd"` ausgegeben.

In Python werden Zeichen als Zeichenfolgen der Länge 1 dargestellt.

Zeichenfolgeliterale werden durch die Verwendung von ein- oder dreifachen Anführungszeichen definiert. Zeichenfolgen, die mit einfachen Anführungszeichen definiert sind, können nicht mehrere Zeilen umfassen, Zeichenfolgen in dreifachen Anführungszeichen dagegen schon. Eine Zeichenfolge kann in einfache Anführungszeichen (') oder doppelte Anführungszeichen (") eingeschlossen werden. Ein Hervorhebungszeichen kann das andere Hervorhebungszeichen ohne Escape-Zeichen enthalten oder das Hervorhebungszeichen wird durch ein Escapezeichen entwertet, d. h., ihm wird der umgekehrte Schrägstrich (\) vorangestellt.

Beispiele

```
"Dies ist eine Zeichenfolge"
'Dies ist auch eine Zeichenfolge'
"Es ist eine Zeichenfolge"
'Dieses Handbuch heißt "Handbuch für Python-Scripting und -Automatisierung".'
"Dies ist ein durch Escapezeichen entwertetes Anführungszeichen (\") in einer Zeichenfolge in Anführungszeichen"
```

Mehrere durch Leerzeichen voneinander getrennte Zeichenfolgen werden vom Python-Parser automatisch verkettet. Dies vereinfacht die Eingabe langer Zeichenfolgen und das Mischen unterschiedlicher Anführungszeichen in einer einzigen Zeichenfolge. Beispiel:

```
"Diese Zeichenfolge verwendet ' und " 'diese Zeichenfolge verwendet "."
```

Dies resultiert in der folgenden Ausgabe:

```
Diese Zeichenfolge verwendet ' und diese Zeichenfolge verwendet ".
```

Zeichenfolgen unterstützen mehrere nützliche Methoden. Einige dieser Methoden werden in der folgenden Tabelle genannt.

Tabelle 6. Zeichenfolgemethoden

Methode	Verwendung
`s.capitalize()`	Anfangsbuchstabe von `s` wird großgeschrieben.
`s.count(ss {,start {,end}})`	Zählt die Vorkommen von `ss` in `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Testet, ob `s` mit `str` beginnt. Testet, ob `s` mit `str` endet.
`s.expandtabs({size})`	Ersetzt Tabstopps durch Leerzeichen. Standardgröße (`size`) ist 8.
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Sucht den ersten Index von `str` in `s`; wird er nicht gefunden, lautet das Ergebnis -1. `rfind` sucht von rechts nach links.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Sucht den ersten Index von `str` in `s`; wenn er nicht gefunden wird, wird `ValueError` ausgelöst. `rindex` sucht von rechts nach links.
`s.isalnum`	Testet, ob die Zeichenfolge alphanumerisch ist.
`s.isalpha`	Testet, ob die Zeichenfolge alphabetisch ist.
`s.isnum`	Testet, ob die Zeichenfolge numerisch ist.
`s.isupper`	Testet, ob die Zeichenfolge ganz in Großbuchstaben geschrieben ist.
`s.islower`	Testet, ob die Zeichenfolge ganz in Kleinbuchstaben geschrieben ist.
`s.isspace`	Testet, ob die Zeichenfolge nur aus Leerzeichen besteht.
`s.istitle`	Testet, ob die Zeichenfolge eine Folge von alphanumerischen Zeichen mit Großschreibung des ersten Buchstabens ist.

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Konvertiert alles in Kleinbuchstaben.
	Konvertiert alles in Großbuchstaben.
	Invertiert die Groß-/Kleinschreibung.
	Konvertiert alles in Schreibung mit großem Anfangsbuchstaben.

`s.join(seq)`	Verknüpft die Zeichenfolgen in `seq` mit `s` als Trennzeichen.
`s.splitlines({keep})`	Teilt `s` in Zeilen auf. Wenn 'keep' `true` ist, werden die neuen Zeilen beibehalten.
`s.split({sep {, max}})`	Teilt `s` mithilfe von `sep` (Standardeinstellung von `sep` ist ein Leerzeichen) bis zu `max` Male in "Wörter" auf.

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Richtet die Zeichenfolge in einem Feld der Breite `width` links aus.
	Richtet die Zeichenfolge in einem Feld der Breite `width` rechts aus.
	Zentriert die Zeichenfolge in einem Feld der Breite `width`.
	Füllt mit 0 auf.

Tabelle 6. Zeichenfolgemethoden (Forts.)

Methode	Verwendung

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Entfernt führende Leerzeichen.
	Entfernt folgende Leerzeichen.
	Entfernt führende und folgende Leerzeichen.

`s.translate(str {,delc})`	Setzt `s` mithilfe einer Tabelle um, nachdem Zeichen in `delc` entfernt wurden. `str` sollte eine Zeichenfolge der Länge `== 256` sein.
`s.replace(old, new {, max})`	Ersetzt alle oder maximal `max` Vorkommen der Zeichenfolge `old` durch die Zeichenfolge `new`.

Anmerkungen

Anmerkungen sind Kommentare, die durch das Rautenzeichen (Hashzeichen) (#) eingeleitet werden. Der Text, der in derselben Zeile auf die Raute folgt, wird als Teil der Anmerkung betrachtet und ignoriert. Eine Anmerkung kann in einer beliebigen Spalte beginnen. Das folgende Beispiel veranschaulicht die Verwendung von Anmerkungen:

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

Anweisungssyntax

Die Anweisungssyntax für Python ist sehr einfach. Im Allgemeinen ist jede Quellcodezeile eine einzelne Anweisung. Außer bei `expression`- und `assignment`-Anweisungen wird jede Anweisung durch ein Schlüsselwort wie `if` oder `for` eingeleitet. Leerzeilen oder Anmerkungszeilen können an beliebiger Stelle zwischen Anweisungen im Code eingefügt werden. Wenn mehrere Anweisungen in einer Zeile stehen, müssen sie durch ein Semikolon (`;`) voneinander getrennt werden.

Sehr lange Anweisungen können in weiteren Zeilen fortgesetzt werden. In diesem Fall muss die Anweisung, die in der nächsten Zeile fortgesetzt werden soll, mit einem umgekehrten Schrägstrich (\) enden. Beispiel:

```
x = "Eine laaaaaaaaaaaaaaaaaaaange Zeichenfolge" + \
     "noch eine laaaaaaaaaaaaaaaaaaaange Zeichenfolge"
```

Wenn eine Struktur in runde Klammern (`()`), eckige Klammern (`[]`) oder geschweifte Klammern (`{}`) eingeschlossen ist, kann die Anweisung ohne umgekehrten Schrägstrich nach einem Komma in der nächsten Zeile fortgesetzt werden. Beispiel:

```
x = (1, 2, 3, "hallo",
      "auf Wiedersehen", 4, 5, 6)
```

IDs

IDs werden verwendet, um Variablen, Funktionen, Klassen und Schlüsselwörter zu bezeichnen. IDs können eine beliebige Länge haben. Sie müssen jedoch entweder mit einem Groß- oder Kleinbuchstaben oder mit einem Unterstrich (_) beginnen. Namen, die mit einem Unterstrich beginnen, sind im Allgemeinen für interne oder nicht öffentliche Namen reserviert. Nach dem ersten Zeichen kann die ID eine beliebige Anzahl von Buchstaben, Ziffern und Unterstrichen in beliebiger Kombination enthalten.

Es gibt in Jython einige reservierte Wörter, die nicht zur Benennung von Variablen, Funktionen oder Klassen verwendet werden können. Sie sind in die folgenden Kategorien aufgeteilt:

- **Einführungszeichen für Anweisungen:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` und `while`
- **Einführungszeichen für Parameter:** `as`, `import` und `in`
- **Operatoren:** `and`, `in`, `is`, `lambda`, `not` und `or`

Bei inkorrekter Schlüsselwortverwendung tritt in der Regel ein Syntaxfehler auf.

Codeblöcke

Codeblöcke sind Gruppen von Anweisungen, die an Stellen verwendet werden, an denen einzelne Anweisungen erwartet werden. Codeblöcke können auf jede der folgenden Anweisungen folgen: `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` und `class`. Diese Anweisungen führen den Codeblock mit dem Doppelpunkt (`:`) ein. Beispiel:

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

Zur Begrenzung von Codeblöcken wird Einrückung verwendet (anstelle der geschweiften Klammern, die in Java verwendet werden). Alle Zeilen in einem Block müssen an dieselbe Position eingerückt werden. Eine Änderung der Einrückung bedeutet das Ende eines Codeblocks. Gewöhnlich wird pro Ebene um vier Leerschritte eingerückt. Zur Einrückung der Zeilen empfiehlt es sich, Leerzeichen anstelle von Tabstopps zu verwenden. Leerzeichen und Tabstopps dürfen nicht gemischt werden. Die Zeilen im äußersten Block eines Moduls, müssen in Spalte 1 beginnen, da sonst ein Syntaxfehler auftritt.

Die Anweisungen, aus denen ein Codeblock besteht (und die auf einen Doppelpunkt folgen), können auch in einer einzigen Zeile durch Semikolons getrennt angeordnet werden. Beispiel:

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

Übergeben von Argumenten an ein Script

Die Übergabe von Argumenten an ein Script ist nützlich, da dadurch ein Script wiederholt ohne Änderung verwendet werden kann. Die Argumente, die in der Befehlszeile übergeben werden, werden als Werte in der Liste `sys.argv` übergeben. Die Anzahl der übergebenen Werte kann über den Befehl `len(sys.argv)` abgerufen werden. Beispiel:

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

In diesem Beispiel importiert der Befehl `import` die gesamte Klasse `sys`, sodass die für diese Klasse vorhandenen Methoden wie `argv` verwendet werden können.

Das Script in diesem Beispiel kann über die folgende Befehlszeile aufgerufen werden:

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

Daraus resultiert die folgende Ausgabe:

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

Beispiele

Das Schlüsselwort `print` gibt die ihm direkt nachfolgenden Argumente aus. Wenn auf die Anweisung ein Komma folgt, enthält die Ausgabe keinen Zeilenumbruch. Beispiel:

```
print "Dies veranschaulicht die Verwendung eines",
print " Kommas am Ende einer Druckanweisung."
```

Dies resultiert in der folgenden Ausgabe:

```
Dies veranschaulicht die Verwendung eines Kommas am Ende einer Druckanweisung.
```

Die Anweisung for wird zum Durchlaufen eines Codeblocks verwendet. Beispiel:

```
mylist1 = ["eins", "zwei", "drei"]
for lv in mylist1:
    print lv
    continue
```

In diesem Beispiel werden drei Zeichenfolgen der Liste mylist1 zugeordnet. Die Elemente der Liste werden dann ausgegeben, wobei jeweils ein Element in jeder Zeile steht. Dies resultiert in der folgenden Ausgabe:

```
eins
zwei
drei
```

In diesem Beispiel nimmt der Iterator lv nacheinander den Wert jedes Elements in der Liste mylist1 an, während die For-Schleife den Codeblock für jedes Element implementiert. Ein Iterator kann eine gültige Kennung beliebiger Länge sein.

Die Anweisung if ist eine bedingte Anweisung. Sie wertet die Bedingung aus und gibt je nach Ergebnis der Bewertung entweder "true" oder "false" zurück. Beispiel:

```
mylist1 = ["eins", "zwei", "drei"]
for lv in mylist1:
    if lv == "zwei"
        print "Der Wert von ist zwei ", lv
    else
        print "Der Wert von lv ist nicht two, sondern ", lv
    continue
```

In diesem Beispiel wird der Wert des Iterators lv ausgewertet. Wenn lv den Wert zwei hat, wird eine andere Zeichenfolge zurückgegeben als in den Fällen, in denen lv nicht den Wert zwei hat. Dies resultiert in der folgenden Ausgabe:

```
Der Wert von lv nicht nicht two, sondern eins
Der Wert von lv ist zwei
Der Wert von lv ist nicht zwei, sondern drei
```

Mathematische Methoden

Aus dem Modul math können Sie auf nützliche mathematische Methoden zugreifen. Einige dieser Methoden werden in der folgenden Tabelle genannt. Sofern nichts anderes angegeben ist, werden alle Werte als Gleitkommazahlen zurückgegeben.

Tabelle 7. Mathematische Methoden

Methode	Verwendung
math.ceil(x)	Gibt die obere Grenze (ceiling) von x als Gleitkommazahl zurück. Dies ist die kleinste ganze Zahl, die größer-gleich x ist.
math.copysign(x, y)	Gibt x mit dem Vorzeichen von y zurück. copysign(1, -0.0) gibt -1 zurück.
math.fabs(x)	Gibt den absoluten Wert von x zurück.
math.factorial(x)	Gibt x-Fakultät zurück. Wenn x eine negative Zahl oder keine ganze Zahl ist, tritt ein ValueError auf.
math.floor(x)	Gibt die Untergrenze (floor) von x als Gleitkommazahl zurück. Dies ist die größte ganze Zahl, die kleiner-gleich x ist.

Tabelle 7. Mathematische Methoden (Forts.)

Methode	Verwendung
`math.frexp(x)`	Gibt die Mantisse (`m`) und den Exponenten (`e`) von `x` als Paar `(m, e)` zurück. `m` ist eine Gleitkommazahl und `e` ist eine ganze Zahl, sodass sich genau `x == m * 2**e` ergibt. Wenn `x` Null ist, wird `(0.0, 0)` zurückgegeben, sonst wird `0.5 <= abs(m) < 1` zurückgegeben.
`math.fsum(iterable)`	Gibt eine genaue Gleitkommasumme von Werten in `iterable` zurück.
`math.isinf(x)`	Prüft, ob die Gleitkommazahl `x` positiv oder negativ unendlich ist.
`math.isnan(x)`	Prüft, ob die Gleitkommazahl `x` eine Nichtzahl (`NaN` -not a number) ist.
`math.ldexp(x, i)`	Gibt `x * (2**i)` zurück. Dies ist im Wesentlichen die Umkehrfunktion von `frexp`.
`math.modf(x)`	Gibt die ganzzahligen und die Bruchteile von `x` zurück. Beide Ergebnisse übernehmen das Vorzeichen von `x` und sind Gleitkommazahlen.
`math.trunc(x)`	Gibt den `reellen` Wert `x` zurück, der auf eine ganze Zahl abgeschnitten wurde.
`math.exp(x)`	Gibt `e**x` zurück.
`math.log(x[, base])`	Gibt den Logarithmus von `x` zur Basis `base` zurück. Ohne Angabe von `base` wird der natürliche Logarithmus von `x` zurückgegeben.
`math.log1p(x)`	Gibt den natürlichen Logarithmus von `1+x` (`base e`) zurück.
`math.log10(x)`	Gibt den Logarithmus von `x` zur Basis 10 zurück.
`math.pow(x, y)`	Gibt `x` hoch `y` zurück. `pow(1.0, x)` und `pow(x, 0.0)` geben immer 1 zurück, selbst wenn `x` Null oder eine Nichtzahl ist.
`math.sqrt(x)`	Gibt die Quadratwurzel von `x` zurück.

Zusätzlich zu den mathematischen Funktionen gibt es einige nützliche trigonometrische Methoden. Diese Methoden werden in der folgenden Tabelle dargestellt.

Tabelle 8. Trigonometrische Methoden

Methode	Verwendung
`math.acos(x)`	Gibt den Arkuskosinus von `x` in Radianten zurück.
`math.asin(x)`	Gibt den Arkussinus von `x` in Radianten zurück.
`math.atan(x)`	Gibt den Arkustangens von `x` in Radianten zurück.
`math.atan2(y, x)`	Gibt `atan(y / x)` in Radianten zurück.
`math.cos(x)`	Gibt den Kosinus von `x` in Radianten zurück.
`math.hypot(x, y)`	Gibt die euklidische Norm `sqrt(x*x + y*y)` zurück. Dies ist die Länge des Vektors vom Ursprung zum Punkt `(x, y)`.
`math.sin(x)`	Gibt den Sinus von `x` in Radianten zurück.
`math.tan(x)`	Gibt den Tangens von `x` in Radianten zurück.
`math.degrees(x)`	Konvertiert den Winkel `x` von Radianten in Grad.

Tabelle 8. Trigonometrische Methoden (Forts.)

Methode	Verwendung
`math.radians(x)`	Konvertiert den Winkel `x` von Grad in Radianten.
`math.acosh(x)`	Umkehrfunktion des Hyperbelkosinus von `x` zurückgeben
`math.asinh(x)`	Umkehrfunktion des Hyperbelsinus von `x` zurückgeben
`math.atanh(x)`	Umkehrfunktion des Hyperbeltangens von `x` zurückgeben
`math.cosh(x)`	Hyperbelkosinus von `x` zurückgeben
`math.sinh(x)`	Hyperbelsinus von `x` zurückgeben
`math.tanh(x)`	Hyperbeltangens von `x` zurückgeben

Es gibt auch zwei mathematische Konstanten. Der Wert von `math.pi` ist die mathematische Konstante Pi. Der Wert von `math.e` ist die mathematische Konstante e.

Verwendung von Nicht-ASCII-Zeichen

Damit Nicht-ASCII-Zeichen verwendet werden können, ist bei Python eine explizite Codierung und Decodierung von Zeichenfolgen in Unicode erforderlich. In IBM SPSS Modeler wird davon ausgegangen, dass Python-Scripts in UTF-8 codiert sind, einer Standard-Unicode-Codierung, die Nicht-ASCII-Zeichen unterstützt. Das folgende Script wird kompiliert, da der Python-Compiler von SPSS Modeler auf UTF-8 gesetzt wurde.

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

Der resultierende Knoten hat jedoch eine falsche Beschriftung.

Abbildung 3. Falsch dargestellte Knotenbeschriftung mit Nicht-ASCII

Die Beschriftung ist falsch, da das Zeichenfolgeliteral von Python in eine ASCII-Zeichenfolge konvertiert wurde.

Python erlaubt die Angabe von Unicode-Zeichenfolgeliteralen, indem das Zeichenpräfix `u` vor dem Zeichenfolgeliteral hinzugefügt wird:

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

Hiermit wird eine Unicode-Zeichenfolge erstellt und die Beschriftung wird korrekt dargestellt.

Abbildung 4. Korrekt dargestellte Knotenbeschriftung mit Nicht-ASCII

Die Verwendung von Python und Unicode ist ein umfassendes Thema, das über den Rahmen dieses Dokuments hinausgeht. Es gibt viele Handbücher und Onlineressourcen, die sich ausführlich mit diesem Thema befassen.

Objektorientierte Programmierung

Objektorientierte Programmierung basiert auf der Erstellung eines Modells des Zielproblems in Ihren Programmen. Objektorientierte Programmierung verringert Programmierfehler und fördert die Wiederverwendung von Code. Python ist eine objektorientierte Sprache. In Python definierte Objekte haben die folgenden Funktionen:

- **Identität.** Jedes Objekt muss eindeutig sein und diese Bedingung muss testbar sein. Zu diesem Zweck gibt es die Tests `is` und `is not`.
- **Status.** Jedes Objekt muss den Status speichern können. Zu diesem Zweck gibt es Attribute wie Felder und Instanzvariablen.
- **Verhalten.** Jedes Objekt muss seinen Status manipulieren können. Zu diesem Zweck gibt es Methoden.

Python umfasst die folgenden Funktionen zur Unterstützung objektorientierter Programmierung:

- **Klassenbasierte Objekterstellung.** Klassen sind Vorlagen für die Erstellung von Objekten. Objekte sind Datenstrukturen mit zugehörigem Verhalten.
- **Vererbung mit Polymorphie.** Python unterstützt sowohl Einfach- als auch Mehrfachvererbung. Alle Python-Instanzdefinitionsmethoden sind polymorph und können von Unterklassen außer Kraft gesetzt werden.
- **Kapselung mit Ausblenden von Daten.** Python erlaubt das Ausblenden von Attributen. Wenn Attribute ausgeblendet sind, kann von außerhalb der Klasse nur über Methoden der Klasse auf sie zugegriffen werden. Klassen implementieren Methoden, um die Daten zu ändern.

Definieren einer Klasse

In einer Python-Klasse können sowohl Variablen als auch Methoden definiert werden. Anders als in Java können Sie in Python eine beliebige Anzahl öffentlicher Klassen pro Quellendatei (oder *Modul*) definieren. Ein Modul in Python kann daher als Entsprechung eines Pakets in Java betrachtet werden.

In Python werden Klassen mit der Anweisung `class` definiert. Die Anweisung `class` hat das folgende Format:

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

ODER

```
class name (superclasses):
    Zuweisung
    .
    .
    Funktion
    .
    .
```

Beim Definieren einer Klasse haben Sie die Möglichkeit, null oder mehr *Zuordnungsanweisungen* anzugeben. Diese erstellen Klassenattribute, die von allen Instanzen der Klasse gemeinsam genutzt werden. Sie können auch null oder mehr *Funktionsdefinitionen* angeben. Diese Funktionsdefinitionen erstellen Methoden. Die Superklassenliste ist optional.

Der Klassenname sollte in seinem Bereich, d. h. in einem Modul, einer Funktion oder einer Klasse, eindeutig sein. Sie können mehrere Variablen zum Verweis auf dieselbe Klasse definieren.

Erstellen einer Klasseninstanz

Klassen werden zur Aufnahme von (gemeinsam genutzten) Klassenattributen oder zum Erstellen von Klasseninstanzen verwendet. Wenn Sie eine Instanz einer Klasse erstellen wollen, rufen Sie die Klasse so auf, als wäre sie eine Funktion. Beispiel einer Klasse:

```
class MyClass:
    pass
```

Hier wird die Anweisung `pass` verwendet, da eine Anweisung zum Abschließen der Klasse erforderlich ist. Vom Programm aus ist jedoch keine Aktion erforderlich.

Die folgende Anweisung erstellt eine Instanz der Klasse `MyClass`:

```
x = MyClass()
```

Hinzufügen von Attributen zu einer Klasseninstanz

Anders als in Java können Clients in Python Attribute einer Instanz einer Klasse hinzufügen. Nur diese eine Instanz wird geändert. Wenn Sie z. B. Attribute einer Instanz `x` hinzufügen wollen, legen Sie neue Werte für diese Instanz fest:

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

Definieren von Klassenattributen und Methoden

Jede Variable, die in einer Klasse gebunden ist, ist ein *Klassenattribut*. Jede in einer Klasse definierte Funktion ist eine *Methode*. Methoden erhalten als erstes Argument eine Instanz der Klasse, die normalerweise `self` genannt wird. Sie könnten z. B. den folgenden Code eingeben, um einige Klassenattribute und Methoden zu definieren:

```
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
```

In einer Klasse sollten Sie alle Verweise auf Klassenattribute mit dem Klassennamen qualifizieren, z. B. `MyClass.attr1`. Alle Verweise auf Instanzattribute sollten mit der Variablen `self` qualifiziert werden, z. B. `self.text`. Außerhalb der Klasse sollte Sie alle Verweise auf Klassenattribute mit dem Klassennamen

(z. B. `MyClass.attr1`) oder mit einer Instanz der Klasse qualifizieren (z. B. `x.attr1`, wobei `x` eine Instanz der Klasse ist). Außerhalb der Klasse sollten alle Verweise auf Instanzvariablen mit einer Instanz der Klasse qualifiziert werden, z. B. `x.text`.

Ausgeblendete Variablen

Daten können durch das Erstellen *nicht öffentlicher* Variablen ausgeblendet werden. Auf nicht öffentliche Variablen kann nur von der Klasse selbst zugegriffen werden. Wenn Sie Namen der Form `__xxx` oder `__xxx_yyy` deklarieren, d. h. mit zwei vorausgehenden Unterstrichen, fügt der Python-Parser dem deklarierten Namen automatisch den Klassennamen hinzu und erstellt so ausgeblendete Variablen. Beispiel:

```
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
```

Anders als in Java müssen in Python alle Verweise auf Instanzvariablen mit `self` qualifiziert werden; es gibt keine implizierte Verwendung von `this`.

Vererbung

Die Fähigkeit zur Vererbung von Klassen ist elementar für die objektorientierte Programmierung. Python unterstützt sowohl Einfach- als auch Mehrfachvererbung. *Einfachvererbung* bedeutet, dass es nur eine Superklasse geben kann. *Mehrfachvererbung* bedeutet, dass es mehrere Superklasse geben kann.

Die Vererbung wird durch Unterklassenbildung anderer Klassen implementiert. Eine beliebige Anzahl von Python-Klassen können Superklassen sein. In der Jython-Implementierung von Python kann die Vererbung direkt oder indirekt nur von einer Java-Klasse erfolgen. Es muss keine Superklasse angegeben werden.

Jedes Attribut oder jede Methode in einer Superklasse ist auch in jeder Unterklasse enthalten und kann von der Klasse selbst oder von jedem beliebigen Client verwendet werden, sofern das Attribut oder die Methode nicht ausgeblendet ist. Jede Instanz einer Unterklasse kann überall verwendet werden, wo auch eine Instanz einer Superklasse verwendet werden kann; dies ist ein Beispiel für *Polymorphie*. Durch diese Funktionen wird die Wiederverwendung ermöglicht und die Erweiterung erleichtert.

Beispiel

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

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

Kapitel 3. Scripting in IBM SPSS Modeler

Scripttypen

In IBM SPSS Modeler gibt es drei Typen von Scripts:

- *Stream-Scripts* werden verwendet, um die Ausführung eines einzelnen Streams zu steuern. Sie werden im Stream gespeichert.
- *Superknotenscripts* werden verwendet, um das Verhalten von Superknoten zu steuern.
- *Standalone- oder Sitzungsscripts* können verwendet werden, um die Ausführung über eine Reihe unterschiedlicher Streams zu koordinieren.

Es stehen verschiedene Methoden zur Verfügung, die in Scripts in IBM SPSS Modeler verwendet werden können. Mit diesen können Sie auf eine Vielzahl von SPSS Modeler-Funktionen zugreifen. Diese Methoden werden auch in Kapitel 4, „Scripting-API", auf Seite 37 zur Erstellung erweiterter Funktionen verwendet.

Streams, Superknotenstreams und Diagramme

Meistens bedeutet der Begriff *Stream* dasselbe, unabhängig davon, ob es sich um einen Stream handelt, der aus einer Datei geladen oder der in einem Superknoten verwendet wird. Im Allgemeinen ist damit eine Sammlung von Knoten gemeint, die miteinander verbunden sind und ausgeführt werden können. Beim Scripting werden jedoch nicht alle Operationen an allen Stellen unterstützt. Ein Scriptautor sollte sich deshalb bewusst sein, welche Streamvariante er verwendet.

Streams

Ein Stream ist der Hauptdokumenttyp von IBM SPSS Modeler. Er kann gespeichert, geladen, bearbeitet und ausgeführt werden. Streams können auch Parameter, globale Werte, ein Script und weitere zugehörige Informationen haben.

Superknotenstreams

Ein *Superknotenstream* ist der Typ von Stream, der in einem Superknoten verwendet wird. Wie ein normaler Stream enthält er Knoten, die miteinander verbunden sind. Superknotenstreams unterscheiden sich durch eine Reihe von Punkten von einem normalen Stream:

- Parameter und Scripts sind dem Superknoten zugeordnet, dem der Superknotenstream gehört, und nicht dem Superknotenstream selbst.
- Superknotenstreams haben je nach Typ des Superknotens zusätzliche Ein- und Ausgabe-Verbindungsknoten. Diese Verbindungsknoten werden verwendet, um Informationen in den und aus dem Superknotenstream zu leiten. Sie werden automatisch bei der Erstellung des Superknotens erstellt.

Diagramme

Der Begriff *Diagramm* deckt die Funktionen ab, die sowohl von normalen Streams als auch von Superknotenstreams unterstützt werden, z. B. das Hinzufügen und Entfernen von Knoten und das Ändern von Verbindungen zwischen den Knoten.

Ausführen eines Streams

Das folgende Beispiel führt alle ausführbaren Knoten im Stream aus. Es ist der einfachste Typ von Stream-Script:

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

Das folgende Beispiel führt ebenfalls alle ausführbaren Knoten im Stream aus:

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

In diesem Beispiel wird der Stream in einer Variablen namens `stream` gespeichert. Das Speichern des Streams in einer Variablen ist hilfreich, da typischerweise ein Script verwendet wird, um entweder den Stream oder die Knoten in einem Stream zu ändern. Durch die Erstellung einer Variablen, die die Streamergebnisse speichert, ergibt sich ein knapperes Script.

Scriptingkontext

Das Modul `modeler.script` stellt den Kontext bereit, in dem ein Script ausgeführt wird. Das Modul wird zur Laufzeit automatisch in ein SPSS Modeler-Script importiert. Das Modul definiert vier Funktionen, die ein Script mit Zugriff auf seine Ausführungsumgebung bereitstellen:

- Die Funktion `session()` gibt die Sitzung für das Script zurück. Die Sitzung definiert Informationen wie die Ländereinstellung und das SPSS Modeler-Back-End (entweder ein lokaler Prozess oder eine vernetzte SPSS Modeler Server-Instanz) für die Ausführung von Streams.
- Die Funktion `stream()` kann in Verbindung mit Stream- und Superknotenscripts verwendet werden. Diese Funktion gibt den Stream zurück, dem das ausgeführte Streamscript oder Superknotenscript gehört.
- Die Funktion `diagram()` kann in Verbindung mit Superknotenscripts verwendet werden. Diese Funktion gibt das Diagramm im Superknoten zurück. Bei anderen Scripttypen hat diese Funktion dieselbe Rückgabe wie die Funktion `stream()`.
- Die Funktion `supernode()` kann in Verbindung mit Superknotenscripts verwendet werden. Diese Funktion gibt den Superknoten zurück, dem das ausgeführte Script gehört.

Die vier Funktionen und ihre Ausgaben sind in der folgenden Tabelle zusammengefasst.

Tabelle 9. Zusammenfassung der `modeler.script`-Funktionen

Scripttyp	`session()`	`stream()`	`diagram()`	`supernode()`
Standalone	Gibt eine Sitzung zurück	Gibt den aktuellen verwalteten Stream zum Zeitpunkt des Scriptaufrufs zurück (z. B. den Stream, der über die Stapelmodusoption `-stream` übergeben wird), bzw. `None`.	Wie bei `stream()`	Nicht zutreffend
Stream	Gibt eine Sitzung zurück	Gibt einen Stream zurück	Wie bei `stream()`	Nicht zutreffend
Superknoten	Gibt eine Sitzung zurück	Gibt einen Stream zurück	Gibt einen Superknotenstream zurück	Gibt einen Superknoten zurück

Das Modul `modeler.script` definiert auch eine Möglichkeit zum Beenden des Scripts mit einem Beendigungscode. Die Funktion `exit(exit-code)` stoppt die Ausführung des Scripts und gibt den angegebenen ganzzahligen Beendigungscode zurück.

Eine der für einen Stream definierten Methoden ist `runAll(List)`. Diese Methode führt alle ausführbaren Knoten aus. Alle Modelle oder Ausgaben, die durch die Ausführung der Knoten generiert werden, werden der angegebenen Liste hinzugefügt.

Üblicherweise generiert eine Streamausführung Ausgaben wie Modelle, Grafiken oder andere Ausgaben. Zur Erfassung dieser Ausgabe kann ein Script eine Variable angeben, die in eine Liste initialisiert wird. Beispiel:

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

Wenn die Ausführung abgeschlossen ist, kann von der Ergebnisliste auf die von der Ausführung generierten Objekte zugegriffen werden.

Referenzieren vorhandener Knoten

Ein Stream ist oft mit einigen Parametern vordefiniert, die geändert werden müssen, bevor der Stream ausgeführt werden kann. Die Änderung dieser Parameter umfasst die folgenden Tasks:

1. Suchen der Knoten im entsprechenden Stream.
2. Ändern der Knoten- und/oder Streameinstellungen.

Suchen von Knoten

Streams bieten eine Reihe von Möglichkeiten zum Suchen eines vorhandenen Knotens. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 10. Methoden zum Lokalisieren eines vorhandenen Knotens

Methode	Rückgabetyp	Beschreibung
`s.findAll(type, label)`	Sammlung	Gibt eine Liste aller Knoten mit dem angegebenen Typ und der angegebenen Beschriftung zurück. Entweder der Typ oder die Beschriftung kann `None` sein. In diesem Fall wird der jeweils andere Parameter verwendet.
`s.findAll(filter, recursive)`	Sammlung	Gibt eine Sammlung aller Knoten zurück, die vom angegebenen Filter akzeptiert werden. Wenn das rekursive Flag `True` lautet, werden auch Superknoten im angegebenen Stream gesucht.
`s.findByID(id)`	Knoten	Gibt den Knoten mit der angegebenen ID zurück bzw. `None`, wenn kein derartiger Knoten vorhanden ist. Die Suche ist auf den aktuellen Stream eingeschränkt.

Tabelle 10. Methoden zum Lokalisieren eines vorhandenen Knotens (Forts.)

Methode	Rückgabetyp	Beschreibung
`s.findByType(type, label)`	Knoten	Gibt den Knoten mit dem angegebenen Typ und/oder der angegebenen Beschriftung zurück. Entweder der Typ oder der Name kann `None` sein. In diesem Fall wird der jeweils andere Parameter verwendet. Wenn sich für mehreren Knoten eine Übereinstimmung ergibt, wird ein beliebiger ausgewählt und zurückgegeben. Wenn sich für keinen Knoten eine Übereinstimmung ergibt, lautet der Rückgabewert `None`.
`s.findDownstream(fromNodes)`	Sammlung	Sucht in der angegebenen Liste von Knoten und gibt das Set von Knoten an, die den angegebenen Knoten nachfolgen. Die zurückgegebene Liste enthält die ursprünglich bereitgestellten Knoten.
`s.findUpstream(fromNodes)`	Sammlung	Sucht in der angegebenen Liste von Knoten und gibt das Set von Knoten an, die den angegebenen Knoten vorausgehen. Die zurückgegebene Liste enthält die ursprünglich bereitgestellten Knoten.

Wenn z. B. ein Stream einen einzigen Filterknoten enthält, auf den das Script zugreifen muss, kann der Filterknoten mithilfe des folgenden Scripts gefunden werden:

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

Wenn die ID des Knotens (siehe Registerkarte "Anmerkungen" im Knotendialogfeld) bekannt ist, kann anhand der ID nach dem Knoten gesucht werden. Beispiel:

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

Festlegen von Eigenschaften

Knoten, Streams, Modelle und Ausgaben haben Eigenschaften, die abgerufen und in den meisten Fällen auch festgelegt werden können. Eigenschaften werden üblicherweise verwendet, um das Verhalten oder Aussehen des Objekts zu ändern. Die verfügbaren Methoden für den Zugriff und das Festlegen von Objekteigenschaften sind in der folgenden Tabelle zusammengefasst.

Tabelle 11. Methoden zum Zugreifen auf Objekteigenschaften und zum Festlegen dieser Eigenschaften

Methode	Rückgabetyp	Beschreibung
`p.getPropertyValue(propertyName)`	Objekt	Gibt den Wert der angegebenen Eigenschaft zurück, bzw. `None`, wenn keine solche Eigenschaft vorhanden ist.
`p.setPropertyValue(propertyName, value)`	Nicht zutreffend	Definiert den Wert der angegebenen Eigenschaft.

Tabelle 11. Methoden zum Zugreifen auf Objekteigenschaften und zum Festlegen dieser Eigenschaften (Forts.)

Methode	Rückgabetyp	Beschreibung
`p.setPropertyValues(properties)`	Nicht zutreffend	Definiert die Werte der angegebenen Eigenschaften. Jeder Eintrag in der Eigenschaftszuordnung besteht aus einem Schlüssel, der den Eigenschaftsnamen und den Wert darstellt, der der entsprechenden Eigenschaft zugewiesen werden sollte.
`p.getKeyedPropertyValue( propertyName, keyName)`	Objekt	Gibt den Wert der angegebenen Eigenschaft und des zugehörigen Schlüssels zurück, bzw. `None`, wenn keine solche Eigenschaft oder kein solcher Schlüssel vorhanden ist.
`p.setKeyedPropertyValue( propertyName, keyName, value)`	Nicht zutreffend	Definiert den Wert der angegebenen Eigenschaft und des Schlüssels.

Wenn Sie z. B. den Wert eines Knotens des Typs 'Variable Datei' am Anfang eines Streams festlegen wollen, können Sie das folgende Script verwenden:

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

Alternativ könnten Sie ein Feld aus einem Filterknoten filtern. In diesem Fall wird der Wert auch in den Feldnamen eingegeben. Beispiel:

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

Erstellen von Knoten und Ändern von Streams

In einigen Situationen wollen Sie möglicherweise vorhandenen Streams neue Knoten hinzufügen. Die Hinzufügung von Knoten zu vorhandenen Streams umfasst die folgenden Tasks:

1. Erstellen der Knoten.
2. Aktivieren von Links für die Knoten in der vorhandenen Streamfolge.

Erstellen von Knoten

Streams bieten eine Reihe von Möglichkeiten zum Erstellen von Knoten. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 12. Methoden zum Erstellen von Knoten

Methode	Rückgabetyp	Beschreibung
`s.create(nodeType, name)`	Knoten	Erstellt einen Knoten des angegebenen Typs und fügt ihn dem angegebenen Stream hinzu.
`s.createAt(nodeType, name, x, y)`	Knoten	Erstellt einen Knoten des angegebenen Typs und fügt ihn dem angegebenen Stream an der angegebenen Position hinzu. Bei $x < 0$ oder $y < 0$ ist die Position nicht festgelegt.

Tabelle 12. Methoden zum Erstellen von Knoten (Forts.)

Methode	Rückgabetyp	Beschreibung
`s.createModelApplier(modelOutput, name)`	Knoten	Erstellt einen Modellanwendungsknoten, der vom angegebenen Modellausgabeobjekt abgeleitet wird.

Sie können das folgende Script verwenden, um z. B. einen neuen Typknoten in einem Stream zu erstellen:

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

Aktivieren und Aufheben von Links für Knoten

Wenn in einem Stream ein neuer Knoten erstellt wird, muss er in der Folge von Knoten verbunden werden, bevor er verwendet werden kann. Streams bieten eine Reihe von Möglichkeiten zum Aktivieren und Aufheben von Links für Knoten. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 13. Methoden zum Aktivieren und Aufheben von Links für Knoten

Methode	Rückgabetyp	Beschreibung
`s.link(source, target)`	Nicht zutreffend	Erstellt einen neuen Link zwischen dem Quellen- und dem Zielknoten.
`s.link(source, targets)`	Nicht zutreffend	Erstellt neue Links zwischen dem Quellenknoten und jedem Zielknoten in der angegebenen Liste.
`s.linkBetween(inserted, source, target)`	Nicht zutreffend	Verbindet einen Knoten zwischen zwei anderen Knoteninstanzen (dem Quellen- und dem Zielknoten) und legt die Position des eingefügten Knotens zwischen diesen beiden fest. Jeder direkte Link zwischen dem Quellen- und dem Zielknoten wird zuerst entfernt.
`s.linkPath(path)`	Nicht zutreffend	Erstellt einen neuen Pfad zwischen Knoteninstanzen. Der erste Knoten wird mit dem zweiten verbunden, der zweite wird mit dem dritten verbunden usw.
`s.unlink(source, target)`	Nicht zutreffend	Entfernt jeden direkten Link zwischen dem Quellen- und dem Zielknoten.
`s.unlink(source, targets)`	Nicht zutreffend	Entfernt alle direkten Links zwischen dem Quellenknoten und jedem Objekt in der Zielliste.
`s.unlinkPath(path)`	Nicht zutreffend	Entfernt jeden Pfad zwischen Knoteninstanzen.
`s.disconnect(node)`	Nicht zutreffend	Entfernt alle Links zwischen dem angegebenen Knoten und allen anderen Knoten im angegebenen Stream.

Tabelle 13. Methoden zum Aktivieren und Aufheben von Links für Knoten (Forts.)

Methode	Rückgabetyp	Beschreibung
`s.isValidLink(source, target)`	*boolesch*	Gibt `True` zurück, wenn die Erstellung eines Links zwischen dem angegebenen Quellen- und Zielknoten zulässig wäre. Diese Methode prüft, dass beide Objekte zum angegebenen Stream gehören, dass der Quellenknoten einen Link bereitstellen kann, dass der Zielknoten einen Link empfangen kann und dass die Erstellung eines solchen Links keinen Zirkelbezug im Stream verursacht.

Das folgende Beispielscript führt die folgenden fünf Tasks durch:

1. Es erstellt einen Eingabeknoten von Typ 'Variable Datei', einen Filterknoten und einen Tabellenausgabeknoten.
2. Es verbindet die Knoten miteinander.
3. Es legt den Dateinamen im Eingabeknoten von Typ 'Variable Datei' fest.
4. Es filtert das Feld "Drug" aus der Ergebnisausgabe.
5. Es führt den Tabellenknoten aus.

```
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)
```

Importieren, Ersetzen und Löschen von Knoten

Neben dem Erstellen und Verbinden von Knoten ist es oft auch erforderlich, Knoten zu ersetzen oder aus dem Stream zu löschen. Die verfügbaren Methoden zum Importieren, Ersetzen und Löschen von Knoten sind in der folgenden Tabelle zusammengefasst.

Tabelle 14. Methoden zum Importieren, Ersetzen oder Löschen von Knoten

Methode	Rückgabetyp	Beschreibung
`s.replace(originalNode, replacementNode, discardOriginal)`	Nicht zutreffend	Ersetzt den angegebenen Knoten im angegebenen Stream. Sowohl der Originalknoten als auch der Ersetzungsknoten müssen dem angegebenen Stream gehören.

Tabelle 14. Methoden zum Importieren, Ersetzen oder Löschen von Knoten (Forts.)

Methode	Rückgabetyp	Beschreibung
`s.insert(source, nodes, newIDs)`	Liste	Fügt Kopien der Knoten in der angegebenen Liste ein. Es wird angenommen, dass alle Knoten in der angegebenen Liste im angegebenen Stream enthalten sind. Das Flag `newIDs` gibt an, ob für jeden Knoten eine neue ID generiert werden soll oder ob die vorhandene ID kopiert und verwendet werden soll. Es wird angenommen, dass alle Knoten in einem Stream eine eindeutige ID haben. Dieses Flag muss daher auf `True` gesetzt werden, wenn der Quellenstream dem angegebenen Stream entspricht. Die Methode gibt die Liste neu eingefügter Knoten zurück, wobei die Reihenfolge der Knoten nicht definiert ist (d. h. die Reihenfolge entspricht nicht unbedingt der Reihenfolge der Knoten in der Eingabeliste).
`s.delete(node)`	Nicht zutreffend	Löscht den angegebenen Knoten aus dem angegebenen Stream. Der Knoten muss dem angegebenen Stream gehören.
`s.deleteAll(nodes)`	Nicht zutreffend	Löscht alle angegebenen Knoten aus dem angegebenen Stream. Alle Knoten in der Sammlung müssen dem angegebenen Stream gehören.
`s.clear()`	Nicht zutreffend	Löscht alle Knoten aus dem angegebenen Stream.

Traversieren durch Knoten in einem Stream

Eine allgemeine Voraussetzung ist die Ermittlung von Knoten, die einem bestimmten Knoten vorangehen oder nachfolgen. Der Stream bietet eine Reihe von Methoden, die verwendet werden können, um diese Knoten zu ermitteln. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 15. Methoden zum Angeben vorausgehender und nachfolgender Streams

Methode	Rückgabetyp	Beschreibung
`s.iterator()`	Iterator	Gibt einen Iterator über die Knotenobjekte zurück, die im angegebenen Stream enthalten sind. Wenn der Stream zwischen Aufrufen der Funktion `next()` geändert wird, kann das Verhalten des Iterators undefiniert sein.
`s.predecessorAt(node, index)`	Knoten	Gibt den angegebenen unmittelbaren Vorgänger des angegebenen Knotens zurück, bzw. `None`, wenn der Index außerhalb des gültigen Bereichs liegt.
`s.predecessorCount(node)`	*Ganzz*	Gibt die Anzahl der unmittelbaren Vorgänger des angegebenen Knotens zurück.

Tabelle 15. Methoden zum Angeben vorausgehender und nachfolgender Streams (Forts.)

Methode	Rückgabetyp	Beschreibung
`s.predecessors(node)`	Liste	Gibt die unmittelbaren Vorgänger des angegebenen Knotens zurück.
`s.successorAt(node, index)`	Knoten	Gibt den angegebenen unmittelbaren Nachfolger des angegebenen Knotens zurück, bzw. `None`, wenn der Index außerhalb des gültigen Bereichs liegt.
`s.successorCount(node)`	*Ganzz*	Gibt die Anzahl der unmittelbaren Nachfolger des angegebenen Knotens zurück.
`s.successors(node)`	Liste	Gibt die unmittelbaren Nachfolger des angegebenen Knotens zurück.

Abrufen von Informationen zu Knoten

Knoten fallen in eine Reihe unterschiedlicher Kategorien wie Datenimport- und -exportknoten, Modellerstellungsknoten und andere Typen von Knoten. Jeder Knoten stellt eine Reihe von Methoden bereit, mit denen Informationen zum Knoten abgerufen werden können.

Die Methoden, mit denen die ID, der Name und die Beschriftung eines Knotens abgerufen werden können, sind in der folgenden Tabelle zusammengefasst.

Tabelle 16. Methoden zum Abrufen von ID, Name und Beschriftung eines Knotens

Methode	Rückgabetyp	Beschreibung
`n.getLabel()`	*Zeichenfolge*	Gibt die Anzeigebeschriftung des angegebenen Knotens zurück. Die Beschriftung entspricht nur dann dem Wert der Eigenschaft `custom_name`, wenn diese Eigenschaft eine nicht leere Zeichenfolge ist und die Eigenschaft `use_custom_name` nicht definiert ist; andernfalls entspricht die Beschriftung dem Wert von `getName()`.
`n.setLabel(label)`	Nicht zutreffend	Legt die Anzeigebeschriftung des angegebenen Knotens fest. Wenn die neue Beschriftung eine nicht leere Zeichenfolge ist, wird sie der Eigenschaft `custom_name` zugewiesen und die Eigenschaft `use_custom_name` wird auf `False` gesetzt, damit die angegebene Beschriftung Vorrang hat; andernfalls wird der Eigenschaft `custom_name` eine leere Zeichenfolge zugewiesen und die Eigenschaft `use_custom_name` wird auf `True` gesetzt.
`n.getName()`	*Zeichenfolge*	Gibt den Namen des angegebenen Knotens zurück.

Tabelle 16. Methoden zum Abrufen von ID, Name und Beschriftung eines Knotens (Forts.)

Methode	Rückgabetyp	Beschreibung
`n.getID()`	*Zeichenfolge*	Gibt die ID des angegebenen Knotens zurück. Bei jeder Erstellung eines neuen Knotens wird eine neue ID erzeugt. Die ID wird im Knoten als persistent definiert, wenn dieser als Teil eines Streams gespeichert wird, damit beim Öffnen des Streams die Knoten-IDs beibehalten werden. Wenn jedoch ein gespeicherter Knoten in einen Stream eingefügt wird, gilt der eingefügte Knoten als neues Objekt und ihm wird eine neue ID zugewiesen.

Die Methoden, mit denen weitere Informationen zu einem Knoten abgerufen werden können, sind in der folgenden Tabelle zusammengefasst.

Tabelle 17. Methoden zum Abrufen von Informationen zu einem Knoten

Methode	Rückgabetyp	Beschreibung
`n.getTypeName()`	*Zeichenfolge*	Gibt den Scriptnamen dieses Knotens zurück. Dies ist derselbe Name, der zum Erstellen einer neuen Instanz dieses Knoten verwendet werden könnte.
`n.isInitial()`	*boolesch*	Gibt `True` zurück, wenn dies ein *Ursprungsknoten* ist, d. h. ein Knoten, der am Anfang eines Streams auftritt.
`n.isInline()`	*boolesch*	Gibt `True` zurück, wenn dies ein *Inline-Knoten* ist, d. h. ein Knoten, der in der Mitte eines Streams auftritt.
`n.isTerminal()`	*boolesch*	Gibt `True` zurück, wenn dies ein *Endknoten* ist, d. h. ein Knoten, der am Ende eines Streams auftritt.
`n.getXPosition()`	*Ganzz*	Gibt den Offset der X-Position des Knotens im Stream an.
`n.getYPosition()`	*Ganzz*	Gibt den Offset der Y-Position des Knotens im Stream an.
`n.setXYPosition(x, y)`	Nicht zutreffend	Legt die Position des Knotens im Stream fest.
`n.setPositionBetween(source, target)`	Nicht zutreffend	Legt die Position des Knotens im Stream so fest, dass er zwischen den angegebenen Knoten angeordnet ist.
`n.isCacheEnabled()`	*boolesch*	Gibt `True` zurück, wenn der Cache aktiviert ist; andernfalls wird `False` zurückgegeben.
`n.setCacheEnabled(val)`	Nicht zutreffend	Aktiviert oder inaktiviert den Cache für dieses Objekt. Wenn der Cache voll ist und das Caching inaktiviert wird, wird der Cache geleert.

Tabelle 17. Methoden zum Abrufen von Informationen zu einem Knoten (Forts.)

Methode	Rückgabetyp	Beschreibung
`n.isCacheFull()`	*boolesch*	Gibt `True` zurück, wenn der Cache voll ist; andernfalls wird `False` zurückgegeben.
`n.flushCache()`	Nicht zutreffend	Leert den Cache dieses Knotens. Hat keine Auswirkung, wenn der Cache nicht aktiviert oder nicht voll ist.

Kapitel 4. Scripting-API

Einführung in die Scripting-API

Die Scripting-API bietet Zugriff auf eine Vielzahl von SPSS Modeler-Funktionen. Alle bisher beschriebenen Methoden sind Teil der API und können ohne weitere Importe implizit im Script aufgerufen werden. Wenn Sie jedoch auf die API-Klassen verweisen wollen, müssen Sie die API mit der folgenden Anweisung explizit importieren:

```
import modeler.api
```

Diese Importanweisung ist für viele Beispiele der Scripting-API erforderlich.

Beispiel: Mit einem benutzerdefinierten Filter nach Knoten suchen

Im Abschnitt „Suchen von Knoten" auf Seite 27 ist ein Beispiel für die Suche nach einem Knoten in einem Stream enthalten, wobei der Typname des Knotens als Suchkriterium verwendet wird. In einigen Situationen ist eine allgemeinere Suche erforderlich. Diese kann mit der Klasse `NodeFilter` und der Streammethode `findAll()` implementiert werden. Diese Art von Suche umfasst die folgenden beiden Schritte:

1. Erstellen einer neuen Klasse, die `NodeFilter` erweitert und eine benutzerdefinierte Version der Methode `accept()` implementiert.
2. Aufrufen der Streammethode `findAll()` mit einer Instanz dieser neuen Klasse. Dadurch werden alle Knoten zurückgegeben, die die in der Methode `accept()` definierten Kriterien erfüllen.

Im folgenden Beispiel wird gezeigt, wie in einem Stream nach Knoten mit aktiviertem Knotencache gesucht werden kann. Die Liste der zurückgegebenen Knoten könnte verwendet werden, um die Caches dieser Knoten zu leeren oder zu inaktivieren.

```
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)
```

Metadaten: Informationen über Daten

Da Knoten in einem Stream miteinander verbunden sind, sind Informationen über die in jedem Knoten verfügbaren Spalten oder Felder verfügbar. In der Modeler-Benutzerschnittstelle können Sie so z. B. auswählen, nach welchen Felder sortiert oder kumuliert werden soll. Diese Informationen werden als Datenmodell bezeichnet.

Scripts können auch auf das Datenmodell zugreifen, indem sie die Felder betrachten, die in einen Knoten eintreten oder aus einem Knoten austreten. Bei einigen Knoten sind das Eingabe- und das Ausgabedatenmodell gleich. Beispielsweise ordnet ein Knoten "sort" die Datensätze einfach um, ändert jedoch nicht das Datenmodell. Einige, wie der Knoten "derive", können neue Felder hinzufüge n. Andere, wie der Knoten "filter", können Felder umbenennen oder entfernen.

Im folgenden Beispiel nimmt das Script den IBM SPSS Modeler-Standardstream `druglearn.str` und erstellt für jedes Feld ein Modell, wobei eines der Eingabefelder gelöscht wird. Die Vorgehensweise sieht dabei folgendermaßen aus:

1. Zugreifen auf das Ausgabedatenmodell aus dem Typknoten
2. Durchlaufen jedes Felds im Ausgabedatenmodell in einer Schleife

3. Ändern des Knotens "filter" für jedes Eingabefeld
4. Ändern des Namens des zu erstellenden Modells
5. Ausführen des Modellerstellungsknotens

Anmerkung: Bevor Sie das Script im Stream `druglean.str` ausführen, müssen Sie die Scriptsprache auf Python setzen. (Der Stream wurde in einer Vorgängerversion von IBM SPSS Modeler erstellt; die Scriptsprache des Streams ist also auf "Legacy" eingestellt.)

```
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([])
```

Das Datenmodellobjekt bietet eine Reihe von Methoden für den Zugriff auf Informationen über die Felder oder Spalten im Datenmodell. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 18. Methoden des Datenmodellobjekts für den Zugriff auf Informationen über Felder oder Spalten

Methode	Rückgabetyp	Beschreibung
`d.getColumnCount()`	*Ganzz*	Gibt die Anzahl der Spalten im Datenmodell zurück.
`d.columnIterator()`	Iterator	Gibt einen Iterator zurück, der seinerseits jede Spalte in der "natürlichen" Einfügereihenfolge zurückgibt. Der Iterator gibt Spalteninstanzen zurück.
`d.nameIterator()`	Iterator	Gibt einen Iterator zurück, der seinerseits den Namen jeder Spalte in der "natürlichen" Einfügereihenfolge zurückgibt.
`d.contains(name)`	*boolesch*	Gibt `True` zurück, wenn eine Spalte mit dem angegebenen Namen in diesem Datenmodell vorhanden ist; andernfalls wird `False` zurückgegeben.
`d.getColumn(name)`	Spalte	Gibt die Spalte mit dem angegebenen Namen zurück.

Tabelle 18. Methoden des Datenmodellobjekts für den Zugriff auf Informationen über Felder oder Spalten (Forts.)

Methode	Rückgabetyp	Beschreibung
`d.getColumnGroup(name)`	Spaltengruppe	Gibt die angegebene Spaltengruppe zurück, bzw. `None`, wenn keine solche Spaltengruppe vorhanden ist.
`d.getColumnGroupCount()`	*Ganzz*	Gibt die Anzahl der Spaltengruppen in diesem Datenmodell zurück.
`d.columnGroupIterator()`	Iterator	Gibt einen Iterator zurück, der nacheinander jede Spaltengruppe zurückgibt.
`d.toArray()`	Spalte[]	Gibt das Datenmodell als Array von Spalten zurück. Die Spalten sind in ihrer "natürlichen" Einfügereihenfolge geordnet.

Jedes Feld (Spaltenobjekt) umfasst eine Reihe von Methoden für den Zugriff auf Informationen über die Spalte. Die folgende Tabelle enthält eine Auswahl davon.

Tabelle 19. Spaltenobjektmethoden für den Zugriff auf Informationen über die Spalte

Methode	Rückgabetyp	Beschreibung
`c.getColumnName()`	*Zeichenfolge*	Gibt den Namen der Spalte zurück.
`c.getColumnLabel()`	*Zeichenfolge*	Gibt die Beschriftung der Spalte zurück bzw. eine leere Zeichenfolge, wenn der Spalte keine Beschriftung zugeordnet ist.
`c.getMeasureType()`	Maßtyp	Gibt den Maßtyp für die Spalte zurück.
`c.getStorageType()`	Speichertyp	Gibt den Speichertyp für die Spalte zurück.
`c.isMeasureDiscrete()`	*boolesch*	Gibt `True` zurück, wenn die Spalte diskret ist. Spalten, die entweder ein Set oder ein Flag sind, werden als diskret betrachtet.
`c.isModelOutputColumn()`	*boolesch*	Gibt `True` zurück, wenn die Spalte eine Modellausgabespalte ist.
`c.isStorageDatetime()`	*boolesch*	Gibt `True` zurück, wenn der Speicher der Spalte ein Uhrzeit-, ein Datum- oder ein Zeitmarkenwert ist.
`c.isStorageNumeric()`	*boolesch*	Gibt `True` zurück, wenn der Speicher der Spalte eine ganze oder eine reelle Zahl ist.
`c.isValidValue(value)`	*boolesch*	Gibt `True` zurück, wenn der angegebene Wert für diesen Speicher gültig ist, und gibt `valid` zurück, wenn die gültigen Spaltenwerte bekannt sind.
`c.getModelingRole()`	Modellierungsrolle	Gibt die Modellierungsrolle für die Spalte zurück.
`c.getSetValues()`	Objekt[]	Gibt ein Array gültiger Werte für die Spalte zurück, bzw. `None`, wenn die Werte nicht bekannt sind oder wenn die Spalte kein Set ist.

Tabelle 19. Spaltenobjektmethoden für den Zugriff auf Informationen über die Spalte (Forts.)

Methode	Rückgabetyp	Beschreibung
`c.getValueLabel(value)`	*Zeichenfolge*	Gibt die Beschriftung für den Wert in der Spalte zurück bzw. eine leere Zeichenfolge, wenn dem Wert keine Beschriftung zugeordnet ist.
`c.getFalseFlag()`	Objekt	Gibt den Indikatorwert "false" für die Spalte zurück, bzw. `None`, wenn der Wert nicht bekannt ist oder wenn die Spalte kein Flag ist.
`c.getTrueFlag()`	Objekt	Gibt den Indikatorwert "true" für die Spalte zurück, bzw. `None`, wenn der Wert nicht bekannt ist oder wenn die Spalte kein Flag ist.
`c.getLowerBound()`	Objekt	Gibt den unteren Grenzwert für die Werte in der Spalte zurück, bzw. `None`, wenn der Wert nicht bekannt ist oder wenn die Spalte nicht fortlaufend ist.
`c.getUpperBound()`	Objekt	Gibt den oberen Grenzwert für die Werte in der Spalte zurück, bzw. `None`, wenn der Wert nicht bekannt ist oder wenn die Spalte nicht fortlaufend ist.

Für die meisten der Methoden, die auf Informationen über eine Spalte zugreifen, sind entsprechende Methoden im Datenmodellobjekt selbst definiert. Die folgenden beiden Anweisungen sind beispielsweise funktional entsprechend:

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

Zugriff auf generierte Objekte

Bei der Ausführung eines Streams werden in der Regel zusätzliche Ausgabeobjekte erzeugt. Diese zusätzlichen Objekte könnten ein neues Modell oder eine Ausgabe sein, die Informationen bereitstellt, die in nachfolgenden Ausführungen verwendet werden.

Im Beispiel unten wird der Stream `druglearn.str` erneut als Ausgangspunkt für den Stream verwendet. In diesem Beispiel werden alle Knoten im Stream ausgeführt und die Ergebnisse werden in einer Liste gespeichert. Das Script durchläuft die Ergebnisse dann in einer Schleife und alle Modellausgaben aus der Ausführung werden als IBM SPSS Modeler-Modelldatei (`.gm`) gespeichert und das Modell wird im PMML-Format exportiert.

```
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)
```

Die Klasse der ausführbaren Komponente bietet eine praktische Möglichkeit zum Ausführen verschiedener allgemeiner Tasks. Die in dieser Klasse verfügbaren Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 20. Methode der Klasse der ausführbaren Komponente zum Ausführen allgemeiner Tasks

Methode	Rückgabetyp	Beschreibung
`t.createStream(Name, autoVerbindung, autoVerwaltung)`	Stream	Erstellt einen neuen Stream und gibt ihn zurück. Für Code, der Streams nicht öffentlich erstellen muss, ohne sie dem Benutzer sichtbar zu machen, sollte das Flag `autoManage` auf `False` gesetzt werden.
`t.exportDocumentToFile( Dokumentausgabe, Dateiname, Dateiformat)`	Nicht zutreffend	Exportiert die Streambeschreibung unter Verwendung des angegebenen Dateiformats in eine Datei.
`t.exportModelToFile(modelOutput, filename, fileFormat)`	Nicht zutreffend	Exportiert das Modell unter Verwendung des angegebenen Dateiformats in eine Datei.
`t.exportStreamToFile(stream, filename, fileFormat)`	Nicht zutreffend	Exportiert den Stream unter Verwendung des angegebenen Dateiformats in eine Datei.
`t.insertNodeFromFile(filename, diagram)`	Knoten	Liest einen Knoten aus der angegebenen Datei, gibt ihn zurück und fügt ihn in das angegebene Diagramm ein. Damit können sowohl Knoten- als auch Superknotenobjekte gelesen werden.
`t.openDocumentFromFile(filename, autoManage)`	Dokumentausgabe	Liest ein Dokument aus der angegebenen Datei und gibt es zurück.
`t.openModelFromFile(filename, autoManage)`	Modellausgabe	Liest ein Modell aus der angegebenen Datei und gibt es zurück.
`t.openStreamFromFile(filename, autoManage)`	Stream	Liest einen Stream aus der angegebenen Datei und gibt ihn zurück.
`t.saveDocumentToFile( documentOutput, filename)`	Nicht zutreffend	Speichert das Dokument an der angegebenen Dateiposition.
`t.saveModelToFile(modelOutput, filename)`	Nicht zutreffend	Speichert das Modell an der angegebenen Dateiposition.
`t.saveStreamToFile(stream, filename)`	Nicht zutreffend	Speichert den Stream an der angegebenen Dateiposition.

Fehlerbehandlung

Die Python-Sprache bietet Fehlerbehandlung über den Codeblock `try...except`. Dieser kann in Scripts verwendet werden, um Ausnahmebedingungen abzufangen und um Probleme zu behandeln, die sonst zur Beendigung des Scripts führen würden.

Im Beispielscript unten wird versucht, ein Modell aus IBM SPSS Collaboration and Deployment Services Repository abzurufen. Diese Operation kann dazu führen, dass eine Ausnahmebedingung ausgelöst wird. Beispielsweise könnten die Berechtigungsnachweise für die Repository-Anmeldung nicht korrekt eingerichtet sein oder der Repository-Pfad ist falsch. Im Script kann dies dazu führen, dass eine `ModelerException`-Ausnahmebedingung ausgelöst wird (alle von IBM SPSS Modeler generierten Ausnahmebedingungen sind von `modeler.api.ModelerException` abgeleitet).

```
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()
```

Anmerkung: Einige Scriptoperation können dazu führen, dass Java-Standardausnahmebedingungen ausgelöst werden. Diese sind nicht von `ModelerException` abgeleitet. Zum Abfangen aller Java-Ausnahmebedingungen kann ein zusätzlicher except-Block verwendet werden. Beispiel:

```
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()
```

Stream-, Sitzungs- und Superknotenparameter

Parameter sind eine praktische Möglichkeit, um Werte zur Laufzeit zu übergeben, statt sie direkt im Script fest zu codieren. Parameter und ihre Werte werden auf dieselbe Weise definiert wie für Streams, d. h. als Einträge in der Parametertabelle eines Streams oder Superknotens oder als Parameter in der Befehlszeile. Die Stream- und Superknotenklassen implementieren eine Gruppe von Funktionen, die vom ParameterProvider-Objekt definiert werden (siehe folgende Tabelle). Die Sitzung stellt einen Aufruf `getParameters()` bereit, der ein Objekt zurückgibt, das diese Funktionen definiert.

Tabelle 21. Vom ParameterProvider-Objekt definierte Funktionen

Methode	Rückgabetyp	Beschreibung
`p.parameterIterator()`	Iterator	Gibt einen Iterator von Parameternamen für dieses Objekt zurück.

Tabelle 21. Vom ParameterProvider-Objekt definierte Funktionen (Forts.)

Methode	Rückgabetyp	Beschreibung
`p.getParameterDefinition( parameterName)`	ParameterDefinition	Gibt die Parameterdefinition für den Parameter mit dem angegebenen Namen zurück bzw.`None`, wenn kein solcher Parameter in diesem Provider existiert. Das Ergebnis kann eine Momentaufnahme der Definition zum Zeitpunkt des Aufrufs der Methode sein und spiegelt nicht unbedingt nachfolgende Änderungen am Parameter durch den Provider wider.
`p.getParameterLabel(parameterName)`	*Zeichenfolge*	Gibt die Beschriftung des angegebenen Parameters zurück bzw. `None`, wenn kein solcher Parameter vorhanden ist.
`p.setParameterLabel(parameterName, label)`	Nicht zutreffend	Definiert die Beschriftung des angegebenen Parameters.
`p.getParameterStorage( parameterName)`	ParameterStorage	Gibt den Speicher des angegebenen Parameters zurück bzw. `None`, wenn kein solcher Parameter vorhanden ist.
`p.setParameterStorage( parameterName, storage)`	Nicht zutreffend	Definiert den Speicher des angegebenen Parameters.
`p.getParameterType(parameterName)`	ParameterType	Gibt den Typ des angegebenen Parameters zurück bzw. `None`, wenn kein solcher Parameter vorhanden ist.
`p.setParameterType(parameterName, type)`	Nicht zutreffend	Definiert den Typ des angegebenen Parameters.
`p.getParameterValue(parameterName)`	Objekt	Gibt den Wert des angegebenen Parameters zurück bzw. `None`, wenn kein solcher Parameter vorhanden ist.
`p.setParameterValue(parameterName, value)`	Nicht zutreffend	Definiert den Wert des angegebenen Parameters.

Im folgenden Beispiel aggregiert das Script einige Telekommunikationsdaten, um die Region mit den niedrigsten durchschnittlichen Einnahmen zu ermitteln. Dann wird ein Streamparameter mit dieser Region definiert. Dieser Streamparameter wird dann in einem Auswahlknoten verwendet, um diese Region aus den Daten auszuschließen, bevor mit den restlichen Daten ein Abwanderungsmodell erstellt wird.

Dieses Beispiel ist konstruiert, da das Script den Auswahlknoten selbst generiert und daher den korrekten Wert direkt in den Ausdruck für den Auswahlknoten generiert haben könnte. Streams sind jedoch in der Regel vordefiniert, weshalb die Festlegung von Parametern auf diese Weise ein nützliches Beispiel darstellt.

Der erste Teil des Beispielscripts erstellt den Streamparameter, der die Region mit den niedrigsten durchschnittlichen Einnahmen enthält. Das Script erstellt auch die Knoten in der Aggregationsverzweigung und in der Modellerstellungsverzweigung und verbindet diese miteinander.

```
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)
```

Das Beispielscript erzeugt den folgenden Stream.

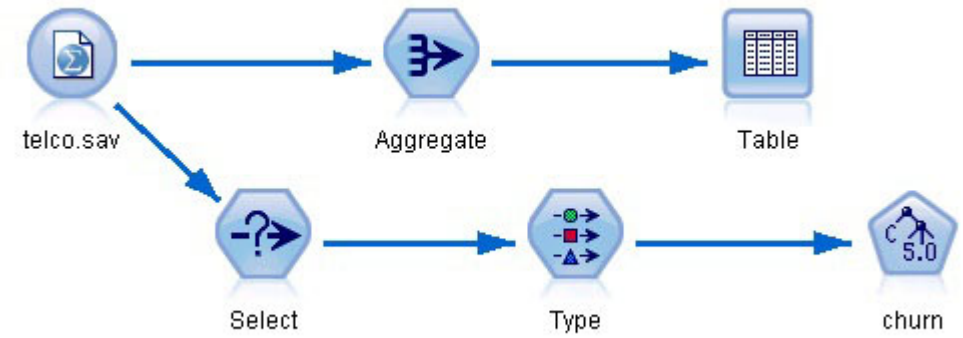


Abbildung 5. Stream, der aus dem Beispielscript resultiert

Der folgende Teil des Beispielscripts führt den Tabellenknoten am Ende der Aggregationsverzweigung aus.

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

Der folgende Teil des Beispielscripts greift auf die Tabellenausgabe zu, die durch die Ausführung des Tabellenknotens generiert wurde. Das Script durchläuft dann die Zeilen in der Tabelle und sucht nach den niedrigsten Durchschnittseinnahmen.

```
# 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
```

Der folgende Teil des Scripts verwendet die Region mit den niedrigsten Durchschnittseinnahmen zur Definition des zuvor erstellten Streamparameters "LowestRegion". Das Script führt dann das Modellerstellungsprogramm aus, wobei die angegebene Region aus den Trainingsdaten ausgeschlossen wird.

```
# 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([])
```

Das vollständige Beispielscript ist im Folgenden aufgeführt.

```
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([])
```

Globale Werte

Globale Werte werden zum Berechnen verschiedener Auswertungsstatistiken für angegebene Felder verwendet. Diese Auswertungswerte sind überall im Stream zugänglich. Globale Werte sind den Streamparametern darin ähnlich, dass über den Stream anhand des Namens auf sie zugegriffen wird. Sie unterscheiden sich von Streamparametern darin, dass die zugehörigen Werte automatisch bei der Ausführung eines Globalwerteknotens aktualisiert werden und nicht vom Scripting oder von der Befehlszeile zugewiesen werden. Der Zugriff auf die globalen Werte für einen Stream erfolgt über den Aufruf der Methode `getGlobalValues()` des Streams.

Das GlobalValues-Objekt definiert die Funktionen, die in der folgenden Tabelle aufgelistet werden.

Tabelle 22. Vom GlobalValues-Objekt definierte Funktionen

Methode	Rückgabetyp	Beschreibung
`g.fieldNameIterator()`	Iterator	Gibt einen Iterator für jeden Feldnamen mit mindestens einem globalen Wert zurück.
`g.getValue(type, fieldName)`	Objekt	Gibt den globalen Wert für den angegebenen Typ und Feldnamen zurück bzw. `None`, wenn kein Wert gefunden werden kann. Als zurückgegebener Wert wird im Allgemeinen eine Zahl erwartet. Einige künftige Funktionen könnten jedoch auch andere Wertetypen zurückgeben.
`g.getValues(fieldName)`	Zuordnung	Gibt eine Zuordnung zurück, die die bekannten Einträge für den angegebenen Feldnamen enthält, bzw. `None`, wenn es keine vorhandenen Einträge für das Feld gibt.

GlobalValues.Type definiert den Typ der verfügbaren Auswertungsstatistiken. Die folgenden Auswertungsstatistikdaten sind verfügbar:

- MAX: Maximalwert des Felds.
- MEAN: Mittelwert des Felds.
- MIN: Minimalwert des Felds.
- STDDEV: Standardabweichung des Felds.
- SUM: Summe der Werte im Feld.

Das folgende Script z. B. greift auf den Mittelwert des Felds "income" zu, das von einem Globalwerteknoten berechnet wird:

```
import modeler.api

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

Arbeiten mit mehreren Streams: Standalone-Scripts

Zum Arbeiten mit mehreren Streams muss ein Standalone-Script verwendet werden. Das Standalone-Script kann in der Benutzerschnittstelle von IBM SPSS Modeler bearbeitet und ausgeführt oder als Befehlszeilenparameter im Stapelmodus übergeben werden.

Das folgende Standalone-Script öffnet zwei Streams. Einer dieser Streams erstellt ein Modell, der zweite plottet die Verteilung der vorhergesagten Werte.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/16/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()
```

Kapitel 5. Tipps zur Scripterstellung

In diesem Abschnitt erhalten Sie einen Überblick über Tipps und Verfahren für die Verwendung von Scripts, wie beispielsweise die Änderung der Streamausführung und die Verwendung von verschlüsselten Kennwörtern in Scripts.

Ändern der Streamausführung

Wenn ein Stream ausgeführt wird, werden die Terminal-Knoten in einer für die Standardsituation optimierten Reihenfolge ausgeführt. In bestimmten Fällen kann eine andere Ausführungsreihenfolge wünschenswert sein. Um die Ausführungsreihenfolge eines Streams zu ändern, führen Sie im Dialogfeld "Streameigenschaften" auf der Registerkarte "Ausführung" folgende Schritte aus:

1. Starten Sie mit einem leeren Script.
2. Klicken Sie in der Symbolleiste auf die Schaltfläche **Standardscript anhängen**, um ein Standard-Stream-Script hinzuzufügen.
3. Bringen Sie die im Standard-Stream-Script enthaltenen Anweisungen in die für die Ausführung gewünschte Reihenfolge.

Arbeiten mit Modellen

Wenn die automatische Modellersetzung in IBM SPSS Modeler aktiviert ist und ein Modellerstellungsknoten über die IBM SPSS Modeler-Benutzerschnittstelle ausgeführt wird, wird ein vorhandenes mit dem Modellerstellungsknoten verbundenes Modellnugget durch das neue Modellnugget ersetzt. Wenn der Modellerstellungsknoten über ein Script ausgeführt wird, wird das vorhandene verbundene Modellnugget nicht ersetzt. Wenn das vorhandene Modellnugget ersetzt werden soll, müssen Sie die Ersetzung des Nuggets explizit in Ihrem Script angeben.

Erstellen eines verschlüsselten Kennworts

In bestimmten Fällen müssen Sie möglicherweise ein Kennwort in ein Script aufnehmen, beispielsweise um auf eine kennwortgeschützte Datenquelle zuzugreifen. Verschlüsselte Kennwörter können in folgenden Elementen verwendet werden:

- Knoteneigenschaften für Datenbankquellenknoten und Ausgabeknoten
- Befehlszeilenargumente für die Anmeldung beim Server
- Die Datenbankverbindungseigenschaften, die in einer *.par*-Datei (die über die Registerkarte "Veröffentlichen" eines Exportknotens generierte Parameterdatei) gespeichert sind.

Über die Benutzerschnittstelle steht ein Tool zur Verfügung, mit dem Sie verschlüsselte Kennwörter auf der Grundlage des Blowfish-Algorithmus erstellen können. (Weitere Informationen finden Sie unter *http://www.schneier.com/blowfish.html*.) Nach der Verschlüsselung können Sie das Kennwort in Scriptdateien und Befehlszeilenargumente kopieren und dort speichern. Die Knoteneigenschaft `epassword`, die für `database` und `databaseexport` verwendet wird, speichert das verschlüsselte Kennwort.

1. Um ein verschlüsseltes Kennwort zu erstellen, wählen Sie im Menü "Extras" folgende Befehlsfolge aus:

   Kennwort verschlüsseln...
2. Geben Sie ein Kennwort im Textfeld "Kennwort" ein.
3. Klicken Sie auf **Verschlüsseln**, um eine Zufallsverschlüsselung des Kennworts zu generieren.
4. Klicken Sie auf die Schaltfläche "Kopieren", um das verschlüsselte Kennwort in die Zwischenablage zu kopieren.
5. Fügen Sie das Kennwort in das gewünschte Script bzw. den gewünschten Parameter ein.

Scriptprüfung

Die Syntax aller Scripttypen können Sie sehr schnell prüfen, indem Sie in der Symbolleiste des Dialogfelds "Standalone-Script" auf die rote Prüfschaltfläche klicken.

Abbildung 6. Symbolleistenschaltflächen für Stream-Scripts

Die Scriptprüfung informiert Sie über alle in Ihrem Code enthaltenen Fehler und macht Verbesserungsvorschläge. Um die den Fehler enthaltende Zeile anzuzeigen, klicken Sie auf das in der unteren Hälfte des Dialogfelds angezeigte Feedback. Der Fehler wird dann rot hervorgehoben.

Scripts in der Befehlszeile

Mit Scripts können Sie Vorgänge ausführen, die normalerweise über die Benutzerschnittstelle durchgeführt werden. Geben Sie in der Befehlszeile beim Start von IBM SPSS Modeler einfach einen Standalone-Stream an und führen Sie ihn aus.

Beispiel:

```
client -script scores.py -execute
```

Das Flag `-script` lädt das angegebene Script, während das Flag `-execute` alle im Script enthaltenen Befehle ausführt.

Angeben von Dateipfaden

Bei der Angabe von Dateipfaden zu Verzeichnissen und Dateien können Sie entweder einen normalen Schrägstrich (`/`) oder einen doppelten umgekehrten Schrägstrich (`\\`) als Verzeichnistrennzeichen verwenden. Beispiel:

```
c:/demos/druglearn.str
```

oder

```
c:\\demos\\druglearn.str
```

Kompatibilität zu früheren Versionen

In früheren IBM SPSS Modeler-Versionen erstellte traditionelle Scripts laufen in der aktuellen Version normalerweise unverändert. Damit die Scripts funktionieren, muss **Traditionell** auf der Registerkarte für das Stream-Script im Dialogfeld **Streameigenschaften** oder im Dialogfeld **Standalone-Script** ausgewählt werden. Modellnuggets können nun automatisch in den Stream aufgenommen werden (das ist die Standardeinstellung) und ein vorhandenes Nugget dieses Typs im Stream ersetzen oder ergänzen. Ob dies tatsächlich geschieht, hängt von den Einstellungen der Optionen **Modell zu Stream hinzufügen** und **Bisheriges Modell ersetzen** ab (**Extras > Optionen > Benutzeroptionen > Benachrichtigungen**). Es kann beispielsweise erforderlich sein, ein Script aus einer früheren Version zu modifizieren, bei der die Nugget-Ersetzung durch Löschen des vorhandenen Nuggets und Einsetzen des neuen erfolgt.

In der aktuellen Version erstellte Scripts funktionieren eventuell nicht in früheren Versionen.

In der aktuellen Version erstellte Python-Scripts funktionieren nicht in früheren Versionen.

Wenn ein in einer älteren Version erstelltes Script einen Befehl verwendet, der mittlerweile ersetzt wurde (oder nicht mehr verwendet wird), wird die alte Form weiterhin unterstützt, es wird jedoch eine Warnnachricht angezeigt. Beispielsweise wurde das alte Schlüsselwort `generated` durch `model` und `clear generated` wurde durch `clear generated palette` ersetzt. Scripts, die die alten Formen verwenden, werden weiterhin ausgeführt, es wird jedoch eine Warnnachricht angezeigt.

Kapitel 6. Befehlszeilenargumente

Aufrufen der Software

Sie können die Befehlszeile Ihres Betriebssystems wie folgt verwenden, um IBM SPSS Modeler zu starten:

1. Öffnen Sie auf einem Computer, auf dem IBM SPSS Modeler installiert ist, ein DOS- oder Befehlszeilenfenster.
2. Um die IBM SPSS Modeler-Schnittstelle im interaktiven Modus zu starten, geben Sie den Befehl `modelerclient` gefolgt von den erforderlichen Argumenten ein. Beispiel:

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

Mithilfe der verfügbaren Argumente (Flags) können Sie eine Verbindung zu einem Server herstellen, Streams laden, Scripts ausführen oder je nach Bedarf weitere Parameter angeben.

Verwenden von Befehlszeilenargumenten

Sie können Befehlszeilenargumente (auch als **Flags** bezeichnet) an den ursprünglichen Befehl `modelerclient` anhängen, um die Vorgehensweise beim Aufrufen von IBM SPSS Modeler zu ändern.

Es sind mehrere Typen von Befehlszeilenargumenten verfügbar, die später in diesem Abschnitt beschrieben werden.

Tabelle 23. Typen von Befehlszeilenargumenten.

Argumenttyp	**Wo beschrieben**
Systemargumente	Weitere Informationen finden Sie im Thema „Systemargumente" auf Seite 54.
Parameterargumente	Weitere Informationen finden Sie im Thema „Parameterargumente" auf Seite 55.
Argumente zum Herstellen einer Serververbindung	Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer Serververbindung" auf Seite 56.
Argumente zum Herstellen einer Verbindung mit IBM SPSS Collaboration and Deployment Services Repository	Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung" auf Seite 57.

Beispielsweise können Sie mit den Flags `-server`, `-stream` und `-execute` wie folgt eine Verbindung zu einem Server herstellen und dann einen Stream laden und ausführen:

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

Beachten Sie: Bei der Ausführung unter einer lokalen Clientinstallation sind die Argumente für die Serververbindung nicht erforderlich.

Parameterwerte, die Leerzeichen enthalten können, können in doppelte Anführungszeichen eingeschlossen werden, z. B.:

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

Sie können auf diese Weise auch IBM SPSS Modeler-Scripts mithilfe des Flags `-script` ausführen.

Fehlersuche bei Befehlszeilenargumenten

Um die Fehlersuche in einer Befehlszeile durchzuführen, starten Sie IBM SPSS Modeler mithilfe des Befehls `modelerclient` mit den gewünschten Argumenten. Dadurch können Sie prüfen, ob die Befehle erwartungsgemäß ausgeführt werden. Außerdem können Sie die Werte jedes Parameters bestätigen, der von der Befehlszeile in das Dialogfeld "Sitzungsparameter" (Menü "Extras", "Sitzungsparameter festlegen") übergeben wird.

Systemargumente

In der nachstehenden Tabelle werden die Systemargumente beschrieben, die für das Aufrufen der Benutzerschnittstelle über die Befehlszeile zur Verfügung stehen.

Tabelle 24. Systemargumente

Argument	Verhalten/Beschreibung
`@` <Befehlsdatei>	Das Symbol @, gefolgt von einem Dateinamen, bezeichnet eine Liste von Befehlen. Wenn der Befehl `modelerclient` auf ein Argument mit dem Symbol @ trifft, werden die Befehle in dieser Datei so abgearbeitet, als hätten Sie diese Befehle direkt in der Befehlszeile eingegeben. Weitere Informationen finden Sie im Thema „Kombinieren mehrerer Argumente” auf Seite 57.
`-directory` <Verzeichnis>	Bestimmt das Standardarbeitsverzeichnis. Im lokalen Modus wird dieses Verzeichnis sowohl für Daten als auch für die Ausgabe herangezogen. Beispiel: `-directory c:/` oder `-directory c:\\`
`-server_directory` <Verzeichnis>	Bestimmt das Serverstandardverzeichnis für Daten. Das Arbeitsverzeichnis, das mithilfe des Flag `-directory` angegeben wird, wird für die Ausgabe genutzt.
`-execute`	Nach dem Starten: Alle Streams, Statusangaben oder Scripts ausführen, die beim Starten geladen waren. Wird ein Script zusätzlich zu einem Stream oder einem Status geladen, wird nur das Script ausgeführt.
`-stream` <Stream>	Beim Starten: Angegebenen Stream laden. Sie können mehrere Streams angeben; der zuletzt genannte Stream wird dabei als aktueller Stream festgelegt.
`-script` <Script>	Beim Starten: Angegebenes Standalone-Script laden. Sie können dieses Script zusätzlich zu einem Stream oder einem Status angeben (siehe unten); beim Starten kann jedoch nur ein einziges Script geladen werden. Wenn das Suffix der Scriptdatei `.py` ist, wird die Datei als Python-Script behandelt; andernfalls wird sie als traditionelles Script behandelt.
`-model` <Modell>	Beim Starten: Angegebenes generiertes Modell (Datei im Format *.gm*) laden.
`-state` <Status>	Beim Starten: Angegebenen gespeicherten Status laden.
`-project` <Projekt>	Angegebenes Projekt laden. Beim Starten kann nur ein einziges Projekt geladen werden.
`-output` <Ausgabe>	Beim Starten: Gespeichertes Ausgabeobjekt (Formatdatei *.cou*) laden.
`-help`	Liste der Befehlszeilenargumente abrufen. Wenn diese Option angegeben ist, werden alle anderen Argumente ignoriert und der Hilfebildschirm wird geöffnet.
`-P` <Name>=<Wert>	Bestimmt einen Startparameter. Kann auch zum Festlegen von Knoteneigenschaften (Slotparameter) herangezogen werden.
`-scriptlang` <python | legacy>	Dieses Argument kann verwendet werden, um die Scriptsprache im Zusammenhang mit der Option `-script` unabhängig vom Suffix der Scriptdatei anzugeben. Beispiel `client -scriptlang python -script scores.txt -execute` Dieser Befehl führt die angegebene Scriptdatei mit Python aus, obwohl das Dateisuffix nicht `.py` ist.

Hinweis: Die Standardverzeichnisse können auch in der Benutzerschnittstelle festgelegt werden. Wählen Sie hierzu im Menü "Datei" die Option **Arbeitsverzeichnis festlegen** bzw. **Server-Verzeichnis festlegen**.

Laden mehrerer Dateien

Über die Befehlszeile können Sie beim Start mehrere Streams, Status und Ausgaben laden, indem Sie für jedes geladene Objekt das relevante Argument wiederholen. Sollen beispielsweise zwei Streams mit den Bezeichnungen *report.str* und *train.str* geladen werden, geben Sie den folgenden Befehl ein:

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

Laden von Objekten aus IBM SPSS Collaboration and Deployment Services Repository

Da Sie bestimmte Objekte aus einer Datei oder aus IBM SPSS Collaboration and Deployment Services Repository laden können (sofern lizenziert), gibt das Dateinamenspräfix `spsscr:` und optional `file:` (für Objekte auf Datenträgern) IBM SPSS Modeler an, wo nach dem Objekt gesucht werden soll. Das Präfix funktioniert mit folgenden Flags:

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

Das Präfix wurde zur Erstellung eines URI verwendet, der den Speicherort des Objekts angibt. Beispiel: `-stream "spsscr:///folder_1/scoring_stream.str"`. Für die Anwesenheit des Präfixes `spsscr:` ist es erforderlich, dass in demselben Befehl eine gültige Verbindung zu IBM SPSS Collaboration and Deployment Services Repository angegeben wurde. Der vollständige Befehl sieht also etwa wie folgt aus:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

Beachten Sie: In der Befehlszeile *müssen* Sie einen URI verwenden. Das einfachere `REPOSITORY-PFAD` wird nicht unterstützt. (Es funktioniert nur innerhalb von Scripts.)

Parameterargumente

Bei der Ausführung von IBM SPSS Modeler über die Befehlszeile können Parameter als Flags herangezogen werden. In Befehlszeilenargumenten wird das Flag `-P` verwendet, um einen Parameter der Form `-P` <*Name*>=<*Wert*> zu kennzeichnen.

Die folgenden Parameter stehen zur Auswahl:

- **Einfache Parameter**
- **Slotparameter** (auch als **Knoteneigenschaften** bezeichnet). Mit diesen Parametern werden die Einstellungen für die Knoten im Stream bearbeitet. Weitere Informationen finden Sie im Thema „Überblick über Knoteneigenschaften” auf Seite 60.
- **Befehlszeilenparameter** dienen zum Ändern der Vorgehensweise beim Aufrufen von IBM SPSS Modeler.

Geben Sie beispielsweise die Benutzernamen und Kennwörter für Datenquellen in Form von Befehlszeilen-Flags an:

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

Das Format stimmt mit dem Parameter `datasource` der Eigenschaft des Knotens `database` überein. Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "database"” auf Seite 68.

Argumente zum Herstellen einer Serververbindung

Das Flag -server besagt, dass IBM SPSS Modeler eine Verbindung zu einem öffentlichen Server aufbauen soll. Mit den Flags -hostname, -use_ssl, -port, -username, -password und -domain legen Sie fest, auf welche Weise IBM SPSS Modeler diese Verbindung zum öffentlichen Server herstellen soll. Wenn kein Argument vom Typ -server angegeben wurde, wird der Standardserver bzw. der lokale Server verwendet.

Beispiele

So stellen Sie eine Verbindung mit einem öffentlichen Server her:

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

So stellen Sie eine Verbindung mit einem Server-Cluster her:

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

Beachten Sie, dass zum Herstellen einer Verbindung mit einem Server-Cluster der Coordinator of Processes über IBM SPSS Collaboration and Deployment Services erforderlich ist. Das Argument -cluster muss also in Verbindung mit den Optionen für eine Repository-Verbindung (spsscr_*) verwendet werden. Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung" auf Seite 57.

Tabelle 25. Argumente zum Herstellen einer Serververbindung.

Argument	Verhalten/Beschreibung
-server	Startet IBM SPSS Modeler im Servermodus. Hierzu wird eine Verbindung zu einem öffentlichen Server mit den Flags -hostname, -port, -username, -password und -domain hergestellt.
-hostname <Name>	Hostname des Server-Computers. Nur im Servermodus verfügbar.
-use_ssl	Gibt an, dass die Verbindung SSL (Secure Socket Layer) verwenden sollte. Dieses Flag ist optional, bei der Standardeinstellung wird SSL *nicht* verwendet.
-port <Nummer>	Portnummer des angegebenen Servers. Nur im Servermodus verfügbar.
-cluster <Name>	Gibt eine Verbindung zu einem Server-Cluster und nicht zu einem benannten Server an; dieses Argument ist eine Alternative zu den Argumenten hostname, port und use_ssl. Bei dem Namen handelt es sich um den Clusternamen oder um einen eindeutigen URI, der den Cluster im IBM SPSS Collaboration and Deployment Services Repository identifiziert. Der Server-Cluster wird von Coordinator of Processes über IBM SPSS Collaboration and Deployment Services verwaltet. Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung" auf Seite 57.
-username <Name>	Benutzername, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar.
-password <Kennwort>	Kennwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. *Hinweis*: Falls das Argument -password nicht verwendet wird, werden Sie aufgefordert, ein Kennwort einzugeben.
-epassword <codierte Kennwortzeichenfolge>	Codiertes Kennwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. *Hinweis*: Ein codiertes Kennwort kann in IBM SPSS Modeler mit den Befehlen im Menü "Extras" erzeugt werden.
-domain <Name>	Domäne, mit der die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar.
-P <Name>=<Wert>	Bestimmt einen Startparameter. Kann auch zum Festlegen von Knoteneigenschaften (Slotparameter) herangezogen werden.

Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung

Hinweis: Für den Zugriff auf ein IBM SPSS Collaboration and Deployment Services-Repository ist eine separate Lizenz erforderlich. Weitere Informationen finden Sie unter http://www.ibm.com/software/analytics/spss/products/deployment/cds/.

Wenn Sie Objekte aus IBM SPSS Collaboration and Deployment Services mithilfe der Befehlszeile speichern oder abrufen möchten, müssen Sie eine gültige Verbindung zum IBM SPSS Collaboration and Deployment Services Repository angeben. Beispiel:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

In der folgenden Tabelle sind die Argumente aufgeführt, die zum Einrichten der Verbindung verwendet werden können.

Tabelle 26. Argumente zum Herstellen einer Verbindung mit IBM SPSS Collaboration and Deployment Services Repository

Argument	Verhalten/Beschreibung
`-spsscr_hostname` <Hostname oder IP-Adresse>	Der Hostname bzw. die IP-Adresse des Servers, auf dem IBM SPSS Collaboration and Deployment Services Repository installiert ist.
`-spsscr_port` <Nummer>	Die Nummer des Ports, an dem IBM SPSS Collaboration and Deployment Services Repository Verbindungen akzeptiert (üblicherweise standardmäßig 8080).
`-spsscr_use_ssl`	Gibt an, dass die Verbindung SSL (Secure Socket Layer) verwenden sollte. Dieses Flag ist optional, bei der Standardeinstellung wird SSL *nicht* verwendet.
`-spsscr_username` <Name>	Benutzername, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
`-spsscr_password` <Kennwort>	Kennwort, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
`-spsscr_epassword` <codiertes Kennwort>	Codiertes Kennwort, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
`-spsscr_domain` <Name>	Domäne, mit der die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt. Dieses Flag ist optional. Verwenden Sie es nur, wenn Sie sich nicht mithilfe von LDAP oder Active Directory anmelden.

Kombinieren mehrerer Argumente

Sie können mehrere Argumente in einer einzigen Befehlsdatei kombinieren, die mit dem Symbol @, gefolgt vom Dateinamen, beim Aufrufen angegeben wird. Auf diese Weise können Sie das Aufrufen über die Befehlszeile verkürzen und die im Betriebssystem geltenden Einschränkungen für die Befehlslänge umgehen. Beim nachstehenden Startbefehl werden beispielsweise die Argumente verwendet, die in der durch `<Befehlsdateiname>` referenzierten Datei angegeben sind.

```
modelerclient @<Befehlsdateiname>
```

Schließen Sie den Dateinamen und den Pfad in Anführungszeichen ein, falls Leerzeichen erforderlich sind, beispielsweise:

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

Die Befehlsdatei kann alle Argumente umfassen, die zuvor beim Starten einzeln angegeben wurden, und zwar mit jeweils einem Argument pro Zeile. Beispiel:

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

Beim Schreiben und Referenzieren von Befehlsdateien sind die folgenden Einschränkungen zu beachten:

- Geben Sie nur je einen Befehl pro Zeile ein.
- Betten Sie kein `@Befehlsdatei`-Argument in eine Befehlsdatei ein.

Kapitel 7. Eigenschaftsreferenz

Überblick über die Eigenschaften

Für Knoten, Streams, Superknoten und Projekte können Sie eine Reihe von verschiedenen Eigenschaften festlegen. Einige Eigenschaften wie Name, Anmerkung und QuickInfo gelten für alle Knoten, während andere sich nur auf bestimmte Knotentypen beziehen. Wieder andere Eigenschaften beziehen sich auf Streamoperationen auf hoher Ebene wie Zwischenspeichern oder das Verhalten von Superknoten. Der Zugriff auf Eigenschaften erfolgt über die Standardbenutzerschnittstelle (z. B. beim Öffnen eines Dialogfelds zum Bearbeiten von Optionen für einen Knoten). Eigenschaften können auf vielfältige Weise verwendet werden.

- Eigenschaften lassen sich mit Scripts ändern, wie in diesem Abschnitt beschrieben. Weitere Informationen finden Sie in „Festlegen von Eigenschaften" auf Seite 28.
- Knoteneigenschaften können in Superknotenparametern verwendet werden.
- Knoteneigenschaften können auch als Teil einer Befehlszeilenoption (mit dem Flag `-P`) beim Starten von IBM SPSS Modeler verwendet werden.

Im Zusammenhang mit Scripts in IBM SPSS Modeler werden Knoten- und Streameigenschaften häufig als **Slotparameter** bezeichnet. In diesem Handbuch werden sie als Knoten- oder Streameigenschaften beschrieben.

Weitere Informationen zur Scriptsprache finden Sie in Kapitel 2, „Scriptsprache", auf Seite 13.

Abkürzungen

Für die Syntax der Knoteneigenschaften werden Standardabkürzungen verwendet. Sich mit den Abkürzungen vertraut zu machen kann beim Erstellen von Scripts sehr hilfreich sein.

Tabelle 27. In der Syntax verwendete Standardabkürzungen.

Abkürzung	Bedeutung
abs	Absoluter Wert
len	Länge
min	Minimum
max	Maximum
correl	Korrelation
covar	Kovarianz
num	Zahl oder numerisch
pct	Prozent oder Prozentsatz
transp	Transparenz
xval	Kreuzvalidierung
var	Varianz oder Variable (in Quellenknoten)

Beispiele für Knoten- und Streameigenschaften

Mit IBM SPSS Modeler können Knoten- und Streameigenschaften auf vielfältige Weise verwendet werden. Meistens werden sie in einem Script, entweder einem **Standalone-Script** zur Automatisierung mehrerer Streams oder Operationen oder einem **Stream-Script** zur Automatisierung von Prozessen innerhalb eines einzelnen Streams, verwendet. Superknotenparameter können ebenfalls innerhalb des Superknotens anhand der Knoteneigenschaften angegeben werden. Auf niedrigster Ebene können Eigenschaften auch als

Befehlszeilenoption zum Starten von IBM SPSS Modeler verwendet werden. Mit dem Argument `-p` als Teil des Befehlszeilenaufrufs können Sie eine Streameigenschaft verwenden, um eine Einstellung im Stream zu ändern.

Weitere Scriptbeispiele finden Sie in „Stream-, Sitzungs- und Superknotenparameter" auf Seite 42 und „Systemargumente" auf Seite 54.

Überblick über Knoteneigenschaften

Jeder Knotentyp besitzt eine eigene Gruppe zulässiger Eigenschaften und jede Eigenschaft besitzt einen Typ. Dabei kann es sich um einen allgemeinen Typ einer Zahl, eines Flags, oder einer Zeichenfolgen handeln. In diesem Fall wird für die Einstellungen der Eigenschaft der richtige Typ erzwungen. Wenn sie nicht erzwungen werden können, wird ein Fehler ausgegeben. Alternativ kann die Eigenschaftsreferenz den Bereich zulässiger Werte, wie `Discard`, `PairAndDiscard` und `IncludeAsText`, angeben. In diesem Fall tritt bei Verwendung eines anderen Werts ein Fehler auf. Flageigenschaften sollten anhand der Werte `True` und `False` gelesen bzw. festgelegt werden. Die Referenztabellen in diesem Handbuch weisen strukturierte Eigenschaften als solche in der Spalte *Eigenschaftsbeschreibung* aus und geben ihr Verwendungsformat an.

Allgemeine Knoteneigenschaften

Zahlreiche Eigenschaften beziehen sich in IBM SPSS Modeler auf alle Knoten (einschließlich Superknoten).

Tabelle 28. Allgemeine Knoteneigenschaften.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
`use_custom_name`	*boolesch*	
`name`	*Zeichenfolge*	Schreibgeschützte Eigenschaft zum Lesen des Namens (entweder automatisch oder benutzerdefiniert) für einen Knoten im Erstellungsbereich.
`custom_name`	*Zeichenfolge*	Legt einen benutzerdefinierten Namen für den Knoten fest.
`tooltip`	*Zeichenfolge*	
`annotation`	*Zeichenfolge*	
`keywords`	*Zeichenfolge*	Strukturierter Slot, der eine Liste der mit dem Objekt verknüpften Schlüsselwörter angibt.
`cache_enabled`	*boolesch*	

`node_type`	`source_supernode`
	`process_supernode`
	`terminal_supernode`
	alle Knotennamen, wie für Scripts angegeben
		Schreibgeschützte Eigenschaft, die den Bezug zu einem Knoten nach Typ herstellt. Statt auf den Knoten nur mit dem Namen zu verweisen, wie `real_income`, können Sie beispielsweise auch den Typ, wie `userinput` oder `filter`, angeben.

Superknotenspezifische Eigenschaften werden wie alle anderen Knoten separat erörtert. Weitere Informationen finden Sie in Kapitel 19, „Superknoteneigenschaften", auf Seite 249.

Kapitel 8. Streameigenschaften

Verschiedene Streameigenschaften können durch Scripts gesteuert werden.

Das Script kann mit der Funktion `stream()` im Modul `modeler.script` auf den aktuellen Stream zugreifen. Beispiel:

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

Um Streameigenschaften zu referenzieren, müssen Sie eine bestimmte Streamvariable verwenden, die durch das Zeichen ^ vor dem Stream gekennzeichnet ist.

Die Knoteneigenschaft dient zur Referenzierung der Knoten im aktuellen Stream.

In der folgenden Tabelle werden die Streameigenschaften beschrieben.

Tabelle 29. Streameigenschaften.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
execute_method	Normal Script	
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	
date_baseline	*Zahl*	
date_2digit_baseline	*Zahl*	

Tabelle 29. Streameigenschaften (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
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	*boolesch*	
import_datetime_as_string	*boolesch*	
decimal_places	*Zahl*	
decimal_symbol	Default Period Comma	
angles_in_radians	*boolesch*	
use_max_set_size	*boolesch*	
max_set_size	*Zahl*	
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	*boolesch*	Dient zur automatischen Aktualisierung von Quellenknoten nach Ausführung des Streams.
script	*Zeichenfolge*	
script_language	Python Legacy	Legt die Scriptsprache für das Stream-Script fest.
annotation	*Zeichenfolge*	
encoding	SystemDefault "UTF-8"	
stream_rewriting	*boolesch*	
stream_rewriting_maximise_sql	*boolesch*	
stream_rewriting_optimise_clem_execution	*boolesch*	
stream_rewriting_optimise_syntax_execution	*boolesch*	
enable_parallelism	*boolesch*	
sql_generation	*boolesch*	
database_caching	*boolesch*	
sql_logging	*boolesch*	
sql_generation_logging	*boolesch*	
sql_log_native	*boolesch*	
sql_log_prettyprint	*boolesch*	

Tabelle 29. Streameigenschaften (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
record_count_suppress_input	*boolesch*	
record_count_feedback_interval	*ganze Zahl*	
use_stream_auto_create_node_ settings	*boolesch*	Bei "true" werden die für den Stream spezifischen Einstellungen verwendet. Andernfalls werden die Vorgaben verwendet.
create_model_applier_for_new_ models	*boolesch*	Bei "true" wird ein neuer Modellanwender hinzugefügt, wenn ein Modellerstellungsprogramm ein neues Modell erstellt und es keine aktiven Update-Links hat.
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Definiert den Typ von Link, wenn ein Modellanwendungsknoten automatisch hinzugefügt wird.
create_source_node_from_builders	*boolesch*	Bei "true" wird ein neuer Quellenknoten hinzugefügt, wenn ein Quellenerstellungsprogramm eine neue Quellenausgabe erstellt und sie keine aktiven Update-Links hat.
create_source_node_update_links	createEnabled createDisabled doNotCreate	Definiert den Typ von Link, wenn ein Quellenknoten automatisch hinzugefügt wird.

Kapitel 9. Eigenschaften von Quellenknoten

Allgemeine Eigenschaften von Quellenknoten

Eigenschaften, die alle Quellenknoten besitzen, sind unten aufgelistet. Die darauf folgenden Themen enthalten Informationen über spezifische Knoten.

Tabelle 30. Allgemeine Eigenschaften von Quellenknoten.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Verschlüsselte Eigenschaft für Feldrollen.
		Hinweis: Die Werte `In` und `Out` werden nicht mehr verwendet.
		In zukünftigen Versionen fällt möglicherweise die Unterstützung dafür weg.

`type`	`Range`
	`Flag`
	`Set`
	`Typeless`
	`Discrete`
	`Ordered Set`
	`Default`
		Feldtyp. Wenn diese Eigenschaft auf *Default* (Standard) gesetzt wird, werden alle Eigenschafteneinstellungen vom Typ`values` gelöscht, und wenn `value_mode` den Wert *Default* (Angeben) besitzt, wird er auf *Read* (Lesen) zurückgesetzt. Wenn `value_mode` bereits auf *Pass* (Übergeben) oder *Read* (Lesen) gesetzt ist, hat das Einstellen von `type` keinerlei Auswirkung.

`storage`	`Unknown`
	`String`
	`Integer`
	`Real`
	`Zeit`
	`Datum`
	`Timestamp`
		Schreibgeschützte verschlüsselte Eigenschaft für Feldspeichertyp.

`check`	`None`
	`Nullify`
	`Coerce`
	`Discard`
	`Warn`
	`Abort`
		Verschlüsselte Eigenschaft für das Überprüfen von Feldtyp und Bereich.

`values`	[*Wert Wert*]	Bei einem stetigen Feld (Bereich) ist der erste Wert das Minimum und der letzte das Maximum. Geben Sie für nominale Felder (Setfelder) alle Werte an. Bei Flagfeldern steht der erste Wert für *falsch* und der letzte für *wahr*. Bei automatischer Festlegung dieser Eigenschaft wird die Eigenschaft `value_mode` auf *Specify* (Angeben) festgelegt.

`value_mode`	`Read`
	`Pass`
	`Read+`
	`Current`
	`Specify`
		Bestimmt, wie Werte beim nächsten Datendurchlauf für ein Feld festgelegt werden.
		Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf *Angeben* festlegen können.
		Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft `Werte` fest.

Tabelle 30. Allgemeine Eigenschaften von Quellenknoten (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
default_value_mode	Read Pass	Gibt die Standardmethode für das Festlegen von Werten für alle Felder an. Diese Einstellung kann mithilfe der Eigenschaft value_mode für bestimmte Felder überschrieben werden.
extend_values	*boolesch*	Gilt, wenn value_mode auf *Read* (Lesen) gesetzt ist. Setzen Sie den Wert auf *T*, um neu gelesene Werte zu bereits für das Feld vorhandenen Werten hinzuzufügen. Setzen Sie den Wert auf *F*, um vorhandene Werte zu verwerfen und sie durch neu gelesene Werte zu ersetzen.
value_labels	*Zeichenfolge*	Wird zur Angabe einer Wertbeschriftung verwendet. Beachten Sie, dass Werte zuerst angegeben werden müssen.
enable_missing	*boolesch*	Bei Festlegung auf *T* wird die Verfolgung von fehlenden Werten für das Feld aktiviert.
missing_values	[*Wert Wert ...*]	Gibt Datenwerte an, die fehlende Daten kennzeichnen.
range_missing	*boolesch*	Wenn diese Eigenschaft auf *T* gesetzt ist, wird angegeben, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist.
missing_lower	*Zeichenfolge*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an.
missing_upper	*Zeichenfolge*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an.
null_missing	*boolesch*	Wenn diese Eigenschaft auf *T* gesetzt ist, werden Nullen (undefinierte Werte, die in der Software als $null$ angezeigt werden) als fehlende Werte betrachtet.
whitespace_missing	*boolesch*	Wenn diese Eigenschaft auf *T* festgelegt ist, werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet.
Beschreibung	*Zeichenfolge*	Dient zur Angabe einer Feldbeschriftung oder Beschreibung.
default_include	*boolesch*	Verschlüsselte Eigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden:
include	*boolesch*	Verschlüsselte Eigenschaft, mit der bestimmt wird, ob einzelne Felder aufgenommen oder gefiltert werden:
new_name	*Zeichenfolge*	

Eigenschaften des Knotens "asimport"

Die Analytic Server-Quelle ermöglicht Ihnen die Ausführung eines Streams unter HDFS (Hadoop Distributed File System).

Tabelle 31. Eigenschaften des Knotens "asimport".

Eigenschaften des Knotens asimport	Datentyp	Eigenschaftsbeschreibung
data_source	*Zeichenfolge*	Der Name der Datenquelle.
host	*Zeichenfolge*	Der Name des Analytic Server-Hosts.

Tabelle 31. Eigenschaften des Knotens "asimport" (Forts.).

Eigenschaften des Knotens **asimport**	Datentyp	Eigenschaftsbeschreibung
`Port`	*ganze Zahl*	Der Port, an dem Analytic Server empfangsbereit ist.
`tenant`	*Zeichenfolge*	In einer Multi-Tenant-Umgebung der Name des Nutzers (Tenants), zu dem Sie gehören. In einer Single-Tenant-Umgebung ist dies standardmäßig **`ibm`**.
`set_credentials`	*boolesch*	Wenn die Benutzerauthentifizierung für Analytic Server und den SPSS Modeler-Server identisch ist, setzen Sie diesen Parameter auf **`false`**. Geben Sie andernfalls **`true`** an.
`user_name`	*Zeichenfolge*	Der Benutzername für die Anmeldung bei Analytic Server. Nur erforderlich, wenn `set_credentials` auf "true" gesetzt ist.
`password`	*Zeichenfolge*	Das Kennwort für die Anmeldung bei Analytic Server. Nur erforderlich, wenn `set_credentials` auf "true" gesetzt ist.

Eigenschaften des Knotens "cognosimport"

Der IBM Cognos BI-Quellenknoten importiert Daten aus Cognos BI-Datenbanken.

Tabelle 32. Eigenschaften des Knotens "cognosimport".

Eigenschaften des Knotens **cognosimport**	Datentyp	Eigenschaftsbeschreibung
`mode`	`Data` `Report`	Gibt an, ob Cognos BI-Daten (Standard) oder Berichte importiert werden sollen.
`cognos_connection`	*{"Zeichenfolge",boolean, "Zeichenfolge", "Zeichenfolge", "Zeichenfolge"}*	Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: {"*Cognos-Server-URL*", *Anmeldemodus*, "*Namespace*", "*Benutzername*", "*Kennwort*"} Dabei gilt: *Cognos-Server-URL* ist die URL des Cognos-Servers, der die Quelle enthält. *Anmeldemodus* gibt an, ob eine anonyme Anmeldung verwendet wird, und ist entweder `true` oder `false`; bei Angabe von `true` sollten die folgenden Felder auf `""` gesetzt werden. *Namespace* gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. *Benutzername* und *Kennwort* sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.

Tabelle 32. Eigenschaften des Knotens "cognosimport" (Forts.).

Eigenschaften des Knotens `cognosimport`	**Datentyp**	**Eigenschaftsbeschreibung**
`cognos_package_name`	*Zeichenfolge*	Pfad und Name des Cognos-Pakets, von dem Sie Datenobjekte importieren, z. B.: `/Public Folders/GOSALES` Hinweis: Nur normale Schrägstriche sind gültig.
`cognos_items`	*{"Feld","Feld", ... ,"Feld"}*	Der Name eines oder mehrerer Datenobjekte, die importiert werden sollen. Das Format von *Feld* ist [*Namespace*].[*Abfragesubjekt*].[*Abfrageelement*]
`cognos_filters`	*Feld*	Der Name eines oder mehrerer Filter, die vor dem Datenimport angewendet werden sollen.
`cognos_data_parameters`	*Liste*	Werte für Eingabeaufforderungsparameter für Daten. Paare aus Name und Wert sind in geschweifte Klammern eingeschlossen und mehrere Paare werden durch Kommas getrennt. Die gesamte Zeichenfolge steht in eckigen Klammern. Format: [{"*Param1*", "*Wert*"},...,{"*ParamN*", "*Wert*"}]
`cognos_report_directory`	*Feld*	Der Cognos-Pfad eines Ordners bzw. Pakets, aus dem Berichte importiert werden sollen. Beispiel: `/Public Folders/GOSALES` Hinweis: Nur normale Schrägstriche sind gültig.
`cognos_report_name`	*Feld*	Der Pfad und Name innerhalb des Speicherorts eines zu importierenden Berichts.
`cognos_report_parameters`	*Liste*	Werte für Berichtsparameter. Paare aus Name und Wert sind in geschweifte Klammern eingeschlossen und mehrere Paare werden durch Kommas getrennt. Die gesamte Zeichenfolge steht in eckigen Klammern. Format: [{"*Param1*", "*Wert*"},...,{"*ParamN*", "*Wert*"}]

Eigenschaften des Knotens "database"

Mit dem Datenbankknoten lassen sich Daten aus einer Reihe von anderen Paketen importieren, die ODBC (Open Database Connectivity) verwenden, darunter u. a. Microsoft SQL Server, DB2 und Oracle.

Tabelle 33. Eigenschaften des Knotens "database".

Eigenschaften des Knotens `database`	**Datentyp**	**Eigenschaftsbeschreibung**
`mode`	`Tabelle` `Query`	Legen Sie *Table* (Tabelle) fest, um die Verbindung zu einer Datenbanktabelle über Dialogfeldsteuerelemente herzustellen, oder *Query* (Abfrage), um die ausgewählte Datenbank mit SQL abzufragen.
`datasource`	*Zeichenfolge*	Datenbankname (siehe auch Hinweis unten).
`username`	*Zeichenfolge*	Datenbankverbindungsdetails (siehe auch Hinweis unten).

Tabelle 33. Eigenschaften des Knotens "database" (Forts.).

Eigenschaften des Knotens database	Datentyp	Eigenschaftsbeschreibung
password	*Zeichenfolge*	
epassword	*Zeichenfolge*	Gibt ein codiertes Kennwort an, als Alternative zum festen Codieren eines Kennworts in einem Sktipt. Weitere Informationen finden Sie im Thema „Erstellen eines verschlüsselten Kennworts" auf Seite 49. Diese Eigenschaft ist während der Ausführung schreibgeschützt.
tablename	*Zeichenfolge*	Name der Tabelle, auf die Sie zugreifen wollen.
strip_spaces	None Left Right Both	Optionen zum Verwerfen von führenden und nachfolgenden Leerzeichen in Zeichenfolgen.
use_quotes	AsNeeded Always Never	Geben Sie an, ob die Tabellen- und Spaltennamen in Anführungszeichen eingeschlossen sind, wenn Abfragen an die Datenbank gesendet werden (wenn sie z. B. Leerzeichen oder Satzzeichen enthalten).
query	*Zeichenfolge*	Gibt den SQL-Code für die Abfrage an, die Sie senden wollen.

Hinweis: Wenn der Datenbankname (in der Eigenschaft datasource) Leerzeichen enthält, verwenden Sie anstatt der individuellen Eigenschaften für Datenquelle, Benutzername und Kennwort eine einzige Datenquelleneigenschaft in folgendem Format:

Tabelle 34. Datenquellenspezifische Eigenschaften des Knotens "database".

Eigenschaften des Knotens database	Datentyp	Eigenschaftsbeschreibung
datasource	*Zeichenfolge*	Format: [Datenbankname,Benutzername,Kennwort[,true | false]] Der letzte Parameter ist für die Verwendung bei verschlüsselten Kennwörtern gedacht. Wenn er auf true gesetzt ist, wird das Kennwort vor der Nutzung verschlüsselt.

Verwenden Sie dieses Format auch für Änderungen der Datenquelle; wenn Sie allerdings nur den Benutzernamen oder das Kennwort ändern möchten, können Sie die Eigenschaften Benutzername oder Kennwort verwenden.

Eigenschaften des Knotens "datacollectionimport"

Der IBM SPSS Data Collection-Datenimportknoten importiert Umfragedaten auf der Grundlage des von den IBM SPSS Data Collection-Marktforschungsprodukten verwendeten IBM Data Model. Um diesen Knoten verwenden zu können, muss die IBM SPSS Data Collection Data Library installiert sein.

Abbildung 7.
Dimensionsdatenimportknoten

Tabelle 35. Eigenschaften des Knotens "datacollectionimport".

Eigenschaften des Knotens `datacollectionimport`	**Datentyp**	**Eigenschaftsbeschreibung**

`metadata_name`	*Zeichenfolge*	Der Name des MDSC. Der spezielle Wert `DimensionsMDD` gibt an, dass das standardmäßige IBM SPSS Data Collection-Metadatendokument verwendet werden soll. Weitere mögliche Werte:
		`mrADODsc`
		`mrI2dDsc`
		`mrLogDsc`
		`mrQdiDrsDsc`
		`mrQvDsc`
		`mrSampleReportingMDSC`
		`mrSavDsc`
		`mrSCDsc`
		`mrScriptMDSC`
		
		Der spezielle Wert `none` zeigt an, dass es keinen MDSC gibt.

`metadata_file`	*Zeichenfolge*	Name der Datei, in der die Metadaten gespeichert werden.

`casedata_name`	*Zeichenfolge*	Der Name des CDSC. Mögliche Werte:
		`mrADODsc`
		`mrI2dDsc`
		`mrLogDsc`
		`mrPunchDSC`
		`mrQdiDrsDsc`
		`mrQvDsc`
		`mrRdbDsc2`
		`mrSavDsc`
		`mrScDSC`
		`mrXmlDsc`
		
		Der spezielle Wert `none` zeigt an, dass es keinen CDSC gibt.

`casedata_source_type`	`Unknown`
	`File`
	`Folder`
	`UDL`
	`DSN`
		Gibt den Quellentyp des CDSC an.

`casedata_file`	*Zeichenfolge*	Wenn `casedata_source_type` den Wert *File* aufweist, wird hier die Datei angegeben, die die Falldaten enthält.
`casedata_folder`	*Zeichenfolge*	Wenn `casedata_source_type` den Wert *Folder* aufweist, wird hier der Ordner angegeben, der die Falldaten enthält.
`casedata_udl_string`	*Zeichenfolge*	Wenn `casedata_source_type` den Wert *UDL* aufweist, wird hier die OLD-DB-Verbindungszeichenfolge für die Datenquelle angegeben, die die Falldaten enthält.
`casedata_dsn_string`	*Zeichenfolge*	Wenn `casedata_source_type` den Wert *DSN*, aufweist, wird hier die ODBC-Verbindungszeichenfolge für die Datenquelle angegeben.

Tabelle 35. Eigenschaften des Knotens "datacollectionimport" (Forts.).

Eigenschaften des Knotens datacollectionimport	**Datentyp**	**Eigenschaftsbeschreibung**
casedata_project	*Zeichenfolge*	Beim Lesen von Falldaten aus einer IBM SPSS Data Collection-Datenbank, können Sie den Namen des Projekts eingeben. Bei allen anderen Falldatentypen sollte diese Einstellung leer bleiben.
version_import_mode	All Latest Specify	Gibt an, wie mit Versionen umgegangen werden soll.
specific_version	*Zeichenfolge*	Wenn version_import_mode den Wert *Specify* aufweist, wird hier die Version der zu importierenden Falldaten festgelegt.
use_language	*Zeichenfolge*	Gibt an, ob Beschriftungen einer bestimmten Sprache verwendet werden sollen.
language	*Zeichenfolge*	Wenn use_language den Wert "true" aufweist, wird hier der beim Import zu verwendende Sprachcode festgelegt. Bei diesem Sprachcode sollte es sich um einen der in den Falldaten verfügbaren Sprachcodes handeln.
use_context	*Zeichenfolge*	Gibt an, ob ein bestimmter Kontext importiert werden sollte. Kontexte dienen dazu, die den Antworten zugeordnete Beschreibung zu variieren.
context	*Zeichenfolge*	Wenn use_context den Wert "true"aufweist, wird hier der zu importierende Kontext festgelegt. Bei diesem Kontext sollte es sich um einen der in den Falldaten verfügbaren Kontexte handeln.
use_label_type	*Zeichenfolge*	Gibt an, ob ein bestimmter Beschriftungstyp importiert werden sollte.
label_type	*Zeichenfolge*	Wenn use_label_type den Wert "true" aufweist, wird hier der zu importierende Beschriftungstyp festgelegt. Bei diesem Beschriftungstyp sollte es sich um einen der in den Falldaten verfügbaren Beschriftungstypen handeln.
user_id	*Zeichenfolge*	Bei Datenbanken, bei denen eine explizite Anmeldung erforderlich ist, können Sie eine Benutzer-ID und ein Kennwort für den Zugriff auf die Datenquelle angeben.
password	*Zeichenfolge*	
import_system_variables	Common None All	Gibt an, welche Systemvariablen importiert werden sollen.
import_codes_variables	*boolesch*	
import_sourcefile_variables	*boolesch*	
import_multi_response	MultipleFlags Single	

Eigenschaften des Knotens "excelimport"

Der Excel-Importknoten importiert Daten aus einer beliebigen Version von Microsoft Excel. Es ist keine ODBC-Datenquelle erforderlich.

Tabelle 36. Eigenschaften des Knotens "excelimport".

Eigenschaften des Knotens **excelimport**	Datentyp	Eigenschaftsbeschreibung
excel_file_type	Excel2003 Excel2007	
full_filename	*Zeichenfolge*	Der vollständige Dateiname mit Pfad.
use_named_range	*boolesch*	Gibt an, ob ein benannter Bereich verwendet werden soll. Bei "true" wird die Eigenschaft named_range zur Angabe des zu lesenden Bereichs verwendet. Andere Einstellungen für Arbeitsblatt und Datenbereich werden ignoriert.
named_range	*Zeichenfolge*	
worksheet_mode	Index Name	Gibt an, ob das Arbeitsblatt durch Index oder durch Namen definiert wird.
worksheet_index	*ganze Zahl*	Index des zu lesenden Arbeitsblattes, beginnend mit 0 für das erste Arbeitsblatt, 1 für das zweite usw.
worksheet_name	*Zeichenfolge*	Name des zu lesenden Arbeitsblattes.
data_range_mode	FirstNonBlank ExplicitRange	Gibt an, wie der Bereich festgelegt werden sollte.
blank_rows	StopReading ReturnBlankRows	Wenn data_range_mode den Wert *FirstNonBlank* aufweist, wird hier angegeben, wie mit leeren Zeilen umgegangen werden soll.
explicit_range_start	*Zeichenfolge*	Wenn data_range_mode den Wert *ExplicitRange* aufweist, wird hier der Startpunkt des zu lesenden Bereichs angegeben.
explicit_range_end	*Zeichenfolge*	
read_field_names	*boolesch*	Gibt an, ob die erste Zeile im angegebenen Bereich als Feldnamen (Spaltennamen) verwendet werden soll.

Eigenschaften des Knotens "evimport"

Der Enterprise-Ansichtsknoten erstellt eine Verbindung mit einem IBM SPSS Collaboration and Deployment Services Repository, was es Ihnen ermöglicht, Enterprise-Ansichtsdaten in einen Stream einzulesen und ein Modell in ein Szenario zu packen, auf das andere Benutzer über das Repository zugreifen können.

Tabelle 37. Eigenschaften des Knotens "evimport".

Eigenschaften des Knotens `evimport`	Datentyp	Eigenschaftsbeschreibung
`connection`	*Liste*	Strukturierte Eigenschaft - Liste der Parameter, die eine Enterprise-Ansichtsverbindung ergeben.
`tablename`	*Zeichenfolge*	Der Name einer Tabelle in der Anwendungsansicht.

Eigenschaften des Knotens "fixedfile"

Der Knoten des Typs "Datei (fest)" importiert Daten aus Textdateien mit festen Feldern, also aus Dateien, deren Felder nicht begrenzt sind, sondern an derselben Position beginnen und eine feste Länge haben. Maschinell erzeugte Daten oder Legacydaten werden häufig im Format mit festen Feldern gespeichert.

Tabelle 38. Eigenschaften des Knotens "fixedfile".

Eigenschaften des Knotens `fixedfile`	Datentyp	Eigenschaftsbeschreibung
`record_len`	*Zahl*	Legt die Zahl der Zeichen in jedem Datensatz fest.
`line_oriented`	*boolesch*	Überspringt am Ende jedes Datensatzes das Zeilenwechselzeichen.

`decimal_symbol`	`Default`
	`Comma`
	`Period`
		Der Typ des in Ihrer Datenquelle verwendeten Dezimaltrennzeichens.

`skip_header`	*Zahl*	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeilen an. Dies ist nützlich, um Spaltenkopfzeilen zu ignorieren.
`auto_recognize_datetime`	*boolesch*	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quellendaten identifiziert werden.
`lines_to_scan`	*Zahl*	
`fields`	*Liste*	Strukturierte Eigenschaft.
`full_filename`	*Zeichenfolge*	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.

`strip_spaces`	`None`
	`Left`
	`Right`
	`Both`
		Verwirft beim Importieren führende und nachfolgende Leerzeichen in Zeichenfolgen.

`invalid_char_mode`	`Discard` `Replace`	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Codierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
`invalid_char_replacement`	*Zeichenfolge*	
`use_custom_values`	*boolesch*	

Tabelle 38. Eigenschaften des Knotens "fixedfile" (Forts.).

Eigenschaften des Knotens `fixedfile`	Datentyp	Eigenschaftsbeschreibung

`custom_storage`	`Unknown`
	`String`
	`Integer`
	`Real`
	`Zeit`
	`Datum`
	`Timestamp`	

`custom_date_format`	`"TTMMJJ"`
	`"MMTTJJ"`
	`"JJMMTT"`
	`"JJJJMMTT"`
	`"JJJJTTT"`
	`DAY`
	`MONTH`
	`"TT-MM-JJ"`
	`"TT-MM-JJJJ"`
	`"MM-TT-JJ"`
	`"MM-TT-JJJJ"`
	`"TT-MON-JJ"`
	`"TT-MON-JJJJ"`
	`"JJJJ-MM-TT"`
	`"TT.MM.JJ"`
	`"TT.MM.JJJJ"`
	`"MM.TT.JJ"`
	`"MM.TT.JJJJ"`
	`"TT.MON.JJ"`
	`"TT.MON.JJJJ"`
	`"TT/MM/JJ"`
	`"TT/MM/JJJJ"`
	`"MM/TT/JJ"`
	`"MM/TT/JJJJ"`
	`"TT/MON/JJ"`
	`"TT/MON/JJJJ"`
	`MON JJJJ`
	`q Q JJJJ`
	`ww WK JJJJ`
		Diese Eigenschaft ist nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.

`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"`
		Diese Eigenschaft ist nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.

`custom_decimal_symbol`	*Feld*	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.

`encoding`	`StreamDefault`
	`SystemDefault`
	`"UTF-8"`
		Legt die Textcodierungsmethode fest.

Eigenschaften des Knotens "sasimport"

Der SAS-Importknoten importiert SAS-Daten in IBM SPSS Modeler.

Tabelle 39. Eigenschaften des Knotens "sasimport".

Eigenschaften des Knotens sasimport	Datentyp	Eigenschaftsbeschreibung
format	Windows UNIX Transport SAS7 SAS8 SAS9	Format der zu importierenden Datei.
full_filename	*Zeichenfolge*	Der vollständige, von Ihnen eingegebene Dateiname, einschließlich der Pfadangabe.
member_name	*Zeichenfolge*	Geben Sie ein Mitglied an, das aus der oben angegebenen SAS-Transportdatei importiert werden soll.
read_formats	*boolesch*	Liest Datenformate (wie z. B. Variablenbeschriftungen) aus der angegebenen Formatdatei.
full_format_filename	*Zeichenfolge*	
import_names	NamesAndLabels LabelsasNames	Gibt die Methode für die Zuordnung von Variablennamen und -beschriftungen beim Import an.

Eigenschaften des Knotens "simgen"

Der Simulationsgenerierungsknoten bietet eine einfache Möglichkeit zum Generieren simulierter Daten - entweder völlig neu mit benutzerspezifizierten statistischen Verteilungen oder automatisch mit den Verteilungen, die durch die Ausführung eines Simulationsanpassungsknoten an vorhandenen historischen Daten gewonnen wurden. Dies ist hilfreich, wenn Sie das Ergebnis eines Vorhersagemodells bei Unsicherheiten in den Modelleingaben auswerten wollen.

Tabelle 40. Eigenschaften des Knotens "simgen".

Eigenschaften des Knotens simgen	Datentyp	Eigenschaftsbeschreibung
fields	Strukturierte Eigenschaft	
correlations	Strukturierte Eigenschaft	
max_cases	*ganze Zahl*	Minimalwert ist 1.000, Maximalwert ist 2.147.483.647
create_iteration_field	*boolesch*	
iteration_field_name	*Zeichenfolge*	
replicate_results	*boolesch*	
random_seed	*ganze Zahl*	
overwrite_when_refitting	*boolesch*	

Tabelle 40. Eigenschaften des Knotens "simgen" (Forts.).

Eigenschaften des Knotens simgen	Datentyp	Eigenschaftsbeschreibung
parameter_xml	*Zeichenfolge*	Gibt den Parameter Xml als Zeichenfolge zurück.
distribution	Bernoulli Beta Binomial Categorical Exponential Fixed Gamma Lognormal NegativeBinomialFailures NegativeBinomialTrials Normal Poisson Range Triangular Uniform Weibull	
bernoulli_prob	*Zahl*	0 ≤ bernoulli_prob ≤ 1
beta_shape1	*Zahl*	Muss ≥ 0 sein.
beta_shape2	*Zahl*	Muss ≥ 0 sein.
beta_min	*Zahl*	Optional. Muss kleiner als beta_max sein.
beta_max	*Zahl*	Optional. Muss größer als beta_min sein.
binomial_n	*ganze Zahl*	Muss > 0 sein.
binomial_prob	*Zahl*	0 ≤ binomial_prob ≤ 1
binomial_min	*Zahl*	Optional. Muss kleiner als binomial_max sein.
binomial_max	*Zahl*	Optional. Muss größer als binomial_min sein.
exponential_scale	*Zahl*	Muss > 0 sein.
exponential_min	*Zahl*	Optional. Muss kleiner als exponential_max sein.
exponential_max	*Zahl*	Optional. Muss größer als exponential_min sein.
fixed_value	*Zeichenfolge*	
gamma_shape	*Zahl*	Muss ≥ 0 sein.
gamma_scale	*Zahl*	Muss ≥ 0 sein.
gamma_min	*Zahl*	Optional. Muss kleiner als gamma_max sein.
gamma_max	*Zahl*	Optional. Muss größer als gamma_min sein.
lognormal_shape1	*Zahl*	Muss ≥ 0 sein.
lognormal_shape2	*Zahl*	Muss ≥ 0 sein.
lognormal_min	*Zahl*	Optional. Muss kleiner als lognormal_max sein.
lognormal_max	*Zahl*	Optional. Muss größer als lognormal_min sein.

Tabelle 40. Eigenschaften des Knotens "simgen" (Forts.).

Eigenschaften des Knotens simgen	Datentyp	Eigenschaftsbeschreibung
negative_bin_failures_threshold	*Zahl*	Muss ≥ 0 sein.
negative_bin_failures_prob	*Zahl*	0 ≤ negative_bin_failures_prob ≤ 1
negative_bin_failures_min	*Zahl*	Optional. Muss kleiner als negative_bin_failures_max sein.
negative_bin_failures_max	*Zahl*	Optional. Muss größer als negative_bin_failures_min sein.
negative_bin_trials_threshold	*Zahl*	Muss ≥ 0 sein.
negative_bin_trials_prob	*Zahl*	0 ≤ negative_bin_trials_prob ≤ 1
negative_bin_trials_min	*Zahl*	Optional. Muss kleiner als negative_bin_trials_max sein.
negative_bin_trials_max	*Zahl*	Optional. Muss kleiner als negative_bin_trials_min sein.
normal_mean	*Zahl*	
normal_sd	*Zahl*	Muss > 0 sein.
normal_min	*Zahl*	Optional. Muss kleiner als normal_max sein.
normal_max	*Zahl*	Optional. Muss größer als normal_min sein.
poisson_mean	*Zahl*	Muss ≥ 0 sein.
poisson_min	*Zahl*	Optional. Muss kleiner als poisson_max sein.
poisson_max	*Zahl*	Optional. Muss größer als poisson_min sein.
triangular_mode	*Zahl*	triangular_min ≤ triangular_mode ≤ triangular_max
triangular_min	*Zahl*	Muss kleiner als triangular_mode sein.
triangular_max	*Zahl*	Muss größer als triangular_mode sein.
uniform_min	*Zahl*	Muss kleiner als uniform_max sein.
uniform_max	*Zahl*	Muss größer als uniform_min sein.
weibull_rate	*Zahl*	Muss ≥ 0 sein.
weibull_scale	*Zahl*	Muss ≥ 0 sein.
weibull_location	*Zahl*	Muss ≥ 0 sein.
weibull_min	*Zahl*	Optional. Muss kleiner als weibull_max sein.
weibull_max	*Zahl*	Optional. Muss größer als weibull_min sein.

Die Korrelation kann eine beliebige Zahl zwischen +1 und -1 sein. Sie können beliebig viele Korrelationen angeben. Jede nicht angegebene Korrelation wird auf Null gesetzt. Wenn Felder unbekannt sind, sollte der Korrelationswert in der Korrelationsmatrix (oder -tabelle) festgelegt werden. Er wird als roter Text angezeigt. Wenn Felder unbekannt sind, kann der Knoten nicht ausgeführt werden.

Eigenschaften des Knotens "statisticsimport"

Der IBM SPSS Statistics-Dateiknoten liest Daten aus dem Dateiformat *.sav* ein, das von IBM SPSS Statistics verwendet wird, sowie in IBM SPSS Modeler gespeicherte Cache-Dateien, die ebenfalls dasselbe Format verwenden.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften des Knotens "statisticsimport"" auf Seite 245.

Eigenschaften des Knotens "userinput"

Der Benutzereingabeknoten bietet eine einfache Möglichkeit, künstliche Daten zu erstellen. Dazu können entweder neue Daten ohne Vorlage erstellt oder vorhandene Daten geändert werden. Diese Funktion ist nützlich, wenn Sie z. B. ein Testdataset für die Modellierung erstellen möchten.

Tabelle 41. Eigenschaften des Knotens "userinput".

Eigenschaften des Knotens `userinput`	Datentyp	Eigenschaftsbeschreibung
`data`		Die Daten für die einzelnen Felder können eine unterschiedliche Länge aufweisen, müssen jedoch mit dem Speichertyp des Feldes konsistent sein. Durch Festlegen von Werten für ein nicht vorhandenes Feld wird dieses Feld erstellt. Durch Festlegen der Werte für eine leere Zeichenfolge (`" "`) wird das angegebene Feld gelöscht. **Anmerkung:** Die für diese Eigenschaft eingegebenen Werte müssen Zeichenfolgen sein, keine Zahlen. Die Zahlen 1, 2, 3 und 4 müssen beispielsweise als `"1 2 3 4"` eingegeben werden.
`names`		Strukturierter Slot, der eine vom Knoten erstellte Liste der Feldnamen festlegt oder zurückgibt.

`custom_storage`	`Unknown`
	`String`
	`Integer`
	`Real`
	`Zeit`
	`Datum`
	`Timestamp`
		Schlüsselslot, der den Speichertyp für ein Feld festlegt oder zurückgibt.

`data_mode`	`Combined` `Ordered`	Wenn `Combined` angegeben wird, werden Datensätze für jede Kombination aus Set-Werten und Min./Max.-Werten erstellt. Die Anzahl der erstellten Datensätze entspricht dem Produkt der Anzahl der Werte in jedem Feld. Wenn `Ordered` angegeben wird, wird zur Erstellung einer Datenzeile aus jeder Spalte für jeden Datensatz genau ein Wert abgerufen. Die Anzahl der erstellten Datensätze entspricht der höchsten Anzahl von Werten, die einem Feld zugeordnet sind. Alle Felder mit weniger Datenwerten werden mit Nullwerten aufgefüllt.

Tabelle 41. Eigenschaften des Knotens "userinput" (Forts.).

Eigenschaften des Knotens `userinput`	Datentyp	Eigenschaftsbeschreibung
`values`		*Diese Eigenschaft wird zugunsten von* `data` *nicht mehr verwendet und sollte nicht mehr eingesetzt werden.*

Eigenschaften des Knotens "variablefile"

Der Variablendateiknoten liest Daten aus Textdateien mit freien Feldern, also aus Dateien, deren Datensätze eine konstante Anzahl von Feldern, aber eine variable Anzahl von Zeichen enthalten. Dieser Knoten ist außerdem nützlich für Dateien mit fester Länge, Überschriftentext und bestimmten Anmerkungen.

Tabelle 42. Eigenschaften des Knotens "variablefile".

Eigenschaften des Knotens `variablefile`	Datentyp	Eigenschaftsbeschreibung
`skip_header`	*Zahl*	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeichen an.
`num_fields_auto`	*boolesch*	Stellt die Anzahl der Felder in jedem Datensatz automatisch fest. Datensätze müssen mit einem Zeilenwechselzeichen abgeschlossen werden.
`num_fields`	*Zahl*	Legt die Anzahl der Felder in jedem Datensatz manuell fest.
`delimit_space`	*boolesch*	Gibt an, welches Zeichen in der Datei für die Feldbegrenzungen verwendet wird.
`delimit_tab`	*boolesch*	
`delimit_new_line`	*boolesch*	
`delimit_non_printing`	*boolesch*	
`delimit_comma`	*boolesch*	Wenn das Komma sowohl als Feldtrennzeichen als auch als Dezimaltrennzeichen für Streams festgelegt ist, setzen Sie `delimit_other` auf *true* und legen Sie ein Komma als Trennzeichen fest, indem Sie die Eigenschaft `other` verwenden.
`delimit_other`	*boolesch*	Hier können Sie ein benutzerdefiniertes Trennzeichen festlegen, indem Sie die Eigenschaft `other` verwenden.
`other`	*Zeichenfolge*	Gibt an, welches Trennzeichen verwendet wird, wenn `delimit_other` auf *true* gesetzt ist.

`decimal_symbol`	`Default`
	`Comma`
	`Period`
		Legt das in der Datenquelle verwendete Dezimaltrennzeichen fest.

`multi_blank`	*boolesch*	Behandelt mehrere angrenzende leere Trennzeichen als ein einziges Trennzeichen.
`read_field_names`	*boolesch*	Behandelt die erste Zeile der Datendatei als Beschriftungen für die Spalten.

Tabelle 42. Eigenschaften des Knotens "variablefile" (Forts.).

Eigenschaften des Knotens variablefile	**Datentyp**	**Eigenschaftsbeschreibung**
strip_spaces	None Left Right Both	Verwirft beim Importieren führende und nachfolgende Leerzeichen in Zeichenfolgen.
invalid_char_mode	Discard Replace	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Codierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
invalid_char_replacement	*Zeichenfolge*	
break_case_by_newline	*boolesch*	Gibt an, dass der Zeilenbegrenzer das Zeilenvorschubzeichen ist.
lines_to_scan	*Zahl*	Gibt an, wie viele Zeilen nach angegebenen Datentypen durchsucht werden sollen.
auto_recognize_datetime	*boolesch*	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quellendaten identifiziert werden.
quotes_1	Discard PairAndDiscard IncludeAsText	Gibt an, wie einfache Anführungszeichen beim Import behandelt werden.
quotes_2	Discard PairAndDiscard IncludeAsText	Gibt an, wie doppelte Anführungszeichen beim Import behandelt werden.
full_filename	*Zeichenfolge*	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.
use_custom_values	*boolesch*	
custom_storage	Unknown String Integer Real Zeit Datum Timestamp	

Tabelle 42. Eigenschaften des Knotens "variablefile" (Forts.).

Eigenschaften des Knotens variablefile	**Datentyp**	**Eigenschaftsbeschreibung**
custom_date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
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"	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
custom_decimal_symbol	*Feld*	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
encoding	StreamDefault SystemDefault "UTF-8"	Legt die Textcodierungsmethode fest.

Eigenschaften des Knotens "xmlimport"

Der XML-Quellenknoten importiert Daten im XML-Format in den Stream. Sie können eine einzelne Datei oder alle Dateien in einem Verzeichnis importieren. Optional können Sie eine Schemadatei angeben, aus der die XML-Struktur gelesen werden soll.

Tabelle 43. Eigenschaften des Knotens "xmlimport".

Eigenschaften des Knotens `xmlimport`	Datentyp	Eigenschaftsbeschreibung
`read`	`single` `directory`	Liest eine einzige Datendatei (Standard) oder alle XML-Dateien in einem Verzeichnis.
`recurse`	*boolesch*	Legt fest, ob zusätzlich XML-Dateien aus allen Unterverzeichnissen des angegebenen Verzeichnisses gelesen werden sollen.
`full_filename`	*Zeichenfolge*	(erforderlich) Vollständiger Pfad und Dateiname der zu importierenden XML-Datei (falls `read = single`).
`directory_name`	*Zeichenfolge*	(erforderlich) Vollständiger Pfad und Dateiname des Verzeichnisses, in dem sich die zu importierenden XML-Dateien befinden (falls `read = directory`).
`full_schema_filename`	*Zeichenfolge*	Vollständiger Pfad und Dateiname der XSD- oder DTD-Datei, aus der die XML-Struktur gelesen werden soll. Wenn Sie diesen Parameter auslassen, wird die Struktur aus der XML-Quellendatei gelesen.
`records`	*Zeichenfolge*	XPath-Ausdruck (z. B. /author/name), der die Datensatzgrenze definiert. Jedes Mal, wenn dieses Element in der Quellendatei gefunden wird, wird ein neuer Datensatz erstellt.
`mode`	`read` `specify`	Alle Daten lesen (Standard) oder festlegen, welche Objekte gelesen werden sollen.
`fields`		Liste der zu importierenden Objekte (Elemente und Attribute). Jedes Objekt in der Liste ist ein XPath-Ausdruck.

Kapitel 10. Eigenschaften von Datensatzoperationsknoten

Eigenschaften des Knotens "append"

Der Anhangknoten verkettet Gruppen von Datensätzen miteinander. Er ist insbesondere nützlich für die Kombination von Datasets mit ähnlicher Struktur, aber unterschiedlichen Daten.

Tabelle 44. Eigenschaften des Knotens "append".

Eigenschaften des Knotens append	**Datentyp**	**Eigenschaftsbeschreibung**
`match_by`	`Position` `Name`	Datasets können Sie auf der Grundlage der Position der Felder in der Hauptdatenquelle oder auf der Grundlage der Namen von Feldern der Eingabedatasets anhängen.
`match_case`	*boolesch*	Aktiviert für die Übereinstimmung von Feldnamen die Unterscheidung zwischen Groß- und Kleinschreibung.
`include_fields_from`	`Main` `All`	
`create_tag_field`	*boolesch*	
`tag_field_name`	*Zeichenfolge*	

Eigenschaften des Knotens "aggregate"

Der Aggregatknoten ersetzt eine Sequenz von Eingabedatensätzen durch zusammengefasste, aggregierte Ausgabedatensätze.

Tabelle 45. Eigenschaften des Knotens "aggregate".

Eigenschaften des Knotens aggregate	**Datentyp**	**Eigenschaftsbeschreibung**
`keys`	*[Feld Feld ... Feld]*	Listet Felder auf, die als Schlüssel für die Aggregation verwendet werden können. Bei den Schlüsselfeldern `Geschlecht` und `Region` beispielsweise erhält jede eindeutige Kombination von `M` und `W` mit den Regionen `N` und `S` (vier eindeutige Kombinationen) einen aggregierten Datensatz.
`contiguous`	*boolesch*	Wählen Sie diese Option aus, wenn Sie wissen, dass alle Datensätze mit denselben Schlüsselwerten in der Eingabe als zusammenhängende Gruppe vorliegen (z. B. wenn die Eingabe nach Schlüsselfeldern sortiert ist). Dadurch lässt sich eventuell die Leistungsfähigkeit verbessern.

Tabelle 45. Eigenschaften des Knotens "aggregate" (Forts.).

Eigenschaften des Knotens `aggregate`	Datentyp	Eigenschaftsbeschreibung
`aggregates`		Strukturierte Eigenschaft, die die numerischen Felder auflistet, deren Werte aggregiert werden, sowie die ausgewählten Aggregationsmodi.
`extension`	*Zeichenfolge*	Geben Sie ein Präfix oder Suffix für doppelt aggregierte Felder an (Beispiel unten).
`add_as`	`Suffix` `Prefix`	
`inc_record_count`	*boolesch*	Erstellt ein zusätzliches Feld, das angibt, wie viele Eingabedatensätze aus den einzelnen Aggregatdatensätzen aggregiert wurden.
`count_field`	*Zeichenfolge*	Gibt den Namen des Felds für die Datensatzanzahl an.

Eigenschaften des Knotens "balance"

Der Balancierungsknoten korrigiert Unausgewogenheiten in einem Dataset, sodass dieses eine bestimmte Bedingung erfüllt. Die Balancierungsanweisung passt den Anteil der Datensätze, bei denen eine Bedingung wahr ist, um den angegebenen Faktor an.

Tabelle 46. Eigenschaften des Knotens "balance".

Eigenschaften des Knotens `balance`	Datentyp	Eigenschaftsbeschreibung
`directives`		Strukturierte Eigenschaft, die für eine auf der angegebenen Zahl basierenden Gewichtung des Anteils von Feldwerten verwendet wird (siehe Beispiel unten).
`training_data_only`	*boolesch*	Gibt an, dass nur Trainingsdaten balanciert werden sollen. Wenn im Stream kein Partitionsfeld vorhanden ist, wird diese Option ignoriert.

Für die Eigenschaft des Knotens `directives` wird folgendes Format verwendet:

[{ *Zahl Zeichenfolge* } \ { *Zahl Zeichenfolge*} \ ... { *Zahl Zeichenfolge* }].

Hinweis: Wenn Zeichenfolgen (mithilfe von doppelten Anführungszeichen) in den Ausdruck eingebettet werden, muss ihnen das Escapezeichen " \ " vorangestellt werden. Das Zeichen " \ " dient außerdem als Fortsetzungszeichen, mit dem Sie die Argumente übersichtlich aufführen können.

Eigenschaften des Knotens "derive_stb"

Der Knoten "Space-Time-Boxes" leitet Space-Time-Boxes aus den Feldern für den Breitengrad, den Längengrad und die Zeitmarke ab. Sie können auch mehrere Space-Time-Boxes als Aufenthaltsorte angeben.

Tabelle 47. Eigenschaften des Knotens "derive_stb".

Eigenschaften des Knotens derive_stb	**Datentyp**	**Eigenschaftsbeschreibung**
mode	IndividualRecords Hangouts	
latitude_field	*Feld*	
longitude_field	*Feld*	
timestamp_field	*Feld*	
hangout_density	*Dichte*	Eine einfache Dichte. Gültige Dichtewerte siehe densities.
densities	[*Dichte,Dichte,..., Dichte*]	Jede Dichte ist eine Zeichenfolge, z. B. STB_GH8_1DAY. **Anmerkung:** Es gibt Einschränkungen dazu, welche Dichten gültig sind. Für den Geohash-Teil können Werte von GH1 bis GH15 verwendet werden. Für den Zeitteil können die folgenden Werte verwendet werden: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	*Feld*	

Tabelle 47. Eigenschaften des Knotens "derive_stb" (Forts.).

Eigenschaften des Knotens **derive_stb**	Datentyp	Eigenschaftsbeschreibung
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Muss eine Zeichenfolge sein.
min_events	*ganze Zahl*	Der gültige ganzzahlige Minimalwert ist 2.
qualifying_pct	*ganze Zahl*	Muss im Bereich von 1 bis 100 liegen.
add_extension_as	Präfix Suffix	
name_extension	*Zeichenfolge*	

Eigenschaften des Knotens "distinct"

Der Duplikatknoten entfernt doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Datenstream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden.

Tabelle 48. Eigenschaften des Knotens "distinct".

Eigenschaften des Knotens **distinct**	Datentyp	Eigenschaftsbeschreibung
mode	Include Discard	Duplikatknoten entfernen doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Datenstream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden.
grouping_fields	[*Feld Feld Feld*]	Listet die Felder auf, die verwendet werden, um zu bestimmen, ob die Datensätze identisch sind. **Anmerkung:** Diese Eigenschaft wird ab IBM SPSS Modeler 16 nicht mehr unterstützt.
composite_value	Strukturierter Slot	
composite_values	Strukturierter Slot	
inc_record_count	*boolesch*	Erstellt ein zusätzliches Feld, das angibt, wie viele Eingabedatensätze aus den einzelnen Aggregatdatensätzen aggregiert wurden.

Tabelle 48. Eigenschaften des Knotens "distinct" (Forts.).

Eigenschaften des Knotens `distinct`	Datentyp	Eigenschaftsbeschreibung
`count_field`	*Zeichenfolge*	Gibt den Namen des Felds für die Datensatzanzahl an.
`sort_keys`	Strukturierte Eigenschaft.	**Anmerkung:** Diese Eigenschaft wird ab IBM SPSS Modeler 16 nicht mehr unterstützt.
`default_ascending`	*boolesch*	
`low_distinct_key_count`	*boolesch*	Gibt an, dass Sie nur über eine kleine Anzahl an Datensätzen und/oder eine kleine Anzahl an eindeutigen Werten der Schlüsselfelder verfügen.
`keys_pre_sorted`	*boolesch*	Gibt an, dass alle Datensätze mit denselben Schlüsselwerten in der Eingabe zusammengefasst werden.
`disable_sql_generation`	*boolesch*	

Eigenschaften des Knotens "merge"

Der Zusammenführungsknoten erstellt aus mehreren Eingabedatensätzen einen einzelnen Ausgabedatensatz mit einigen oder allen der Eingabefelder. Er wird zum Zusammenführen von Daten aus verschiedenen Quellen verwendet, beispielsweise Daten über Auslandskunden und erworbene demografische Daten.

Tabelle 49. Eigenschaften des Knotens "merge".

Eigenschaften des Knotens `merge`	Datentyp	Eigenschaftsbeschreibung

`method`	`Order`
	`Keys`
	`Condition`
		Geben Sie an, ob die Datensätze in der Reihenfolge zusammengeführt werden sollen, in der sie in den Datendateien aufgeführt sind, ob eines oder mehrere Felder verwendet werden sollen, um Datensätze mit demselben Wert in den Schlüsselfeldern zusammenzuführen, oder ob die Datensätze zusammengeführt werden, wenn eine bestimmte Bedingung erfüllt ist.

`condition`	*Zeichenfolge*	Wenn `method` auf `Condition` gesetzt ist, wird hier die Bedingung für das Einschließen oder Verwerfen von Datensätzen angegeben.
`key_fields`	[*Feld Feld Feld*]	
`common_keys`	*boolesch*	

`join`	`Inner`
	`FullOuter`
	`PartialOuter`
	`Anti`	

`outer_join_tag.n`	*boolesch*	Bei dieser Eigenschaft ist *n* der im Dialogfeld für die Datasetauswahl angezeigte Tagname. Beachten Sie, dass mehrere Tagnamen angegeben werden können, da jede beliebige Zahl von Datasets unvollständige Datensätze beitragen könnte.
`single_large_input`	*boolesch*	Gibt an, ob die Optimierung verwendet werden soll, wenn eine Eingabe vorhanden ist, die im Vergleich mit den anderen Eingaben relativ groß ist.

Tabelle 49. Eigenschaften des Knotens "merge" (Forts.).

Eigenschaften des Knotens `merge`	Datentyp	Eigenschaftsbeschreibung
`single_large_input_tag`	*Zeichenfolge*	Geben Sie den Tagnamen an, der im Dialogfeld "Großes Dataset auswählen" angezeigt wird. Beachten Sie, dass die Verwendung dieser Eigenschaft leicht von der Eigenschaft `outer_join_tag` abweicht (boolesch gegenüber Zeichenfolge), da nur ein einziges Eingabedataset angegeben werden kann.
`use_existing_sort_keys`	*boolesch*	Gibt an, ob die Eingaben bereits nach einem oder mehreren Schlüsselfeldern sortiert sind.
`existing_sort_keys`	[{*Zeichenfolge* `Ascending`} \ {*Zeichenfolge* `Descending`}]	Gibt die bereits sortieren Felder und ihre Sortierrichtung an.

Eigenschaften des Knotens "rfmaggregate"

Mit dem Knoten "RFM-Aggregat" (Recency-, Frequency-, Monetary-Aggregat) können Sie Daten über die früheren Transaktionen von Kunden verwenden, alle nicht benötigten Daten entfernen und alle verbliebenen Transaktionsdaten zu einer einzigen Zeile zusammenfassen, die angibt, wann der betreffende Kunde zuletzt mit Ihnen in Geschäftskontakt stand, wie viele Transaktionen er vorgenommen hat und wie hoch der Gesamtwert dieser Transaktionen ist.

Tabelle 50. Eigenschaften des Knotens "rfmaggregate".

Eigenschaften des Knotens `rfmaggregate`	Datentyp	Eigenschaftsbeschreibung
`relative_to`	`Fixed` `Today`	Dient zur Angabe des Datums, ausgehend von dem die Aktualität der Transaktionen berechnet werden soll.
`reference_date`	*Datum*	Nur verfügbar, wenn unter `relative_to` die Option `Fixed` festgelegt wurde.
`contiguous`	*boolesch*	Wenn die Daten vorsortiert sind, sodass alle Datensätze mit derselben ID zusammen im Datenstream erscheinen, können Sie mit dieser Option die Verarbeitung beschleunigen.
`id_field`	*Feld*	Dient zur Angabe des für die Identifizierung des Kunden und seiner Transaktionen zu verwendenden Felds.
`date_field`	*Feld*	Dient zur Angabe des Datumsfelds, das für die Berechnung der Aktualität verwendet werden soll.
`value_field`	*Feld*	Dient zur Angabe des Felds, das für die Berechnung der Geldwerts verwendet werden soll.
`extension`	*Zeichenfolge*	Geben Sie ein Präfix oder Suffix für doppelt aggregierte Felder an.
`add_as`	`Suffix` `Prefix`	Gibt an, ob die Erweiterung (`extension`) als Suffix oder als Präfix hinzugefügt werden soll.
`discard_low_value_records`	*boolesch*	Ermöglicht die Verwendung der Einstellung `discard_records_below`.

Tabelle 50. Eigenschaften des Knotens "rfmaggregate" (Forts.).

Eigenschaften des Knotens rfmaggregate	**Datentyp**	**Eigenschaftsbeschreibung**
discard_records_below	*Zahl*	Dient zur Angabe eines Mindestwerts für die bei der Berechnung der RFM-Gesamtwerte verwendeten Transaktionsdetails. Die für den Wert geltenden Einheiten beziehen sich auf das ausgewählte Feld value.
only_recent_transactions	*boolesch*	Dient zur Aktivierung der Einstellung specify_transaction_date bzw. transaction_within_last.
specify_transaction_date	*boolesch*	
transaction_date_after	*Datum*	Nur verfügbar, wenn specify_transaction_date ausgewählt wurde. Dient zur Angabe des Transaktionsdatums, nach dem die Datensätze in die Analyse aufgenommen werden sollen.
transaction_within_last	*Zahl*	Nur verfügbar, wenn transaction_within_last ausgewählt wurde. Hier können Sie anhand von Anzahl und Typ der Zeiträume (Tage, Wochen, Monate oder Jahre) angeben, wie weit die in die Analyse aufzunehmenden Datensätze ausgehend von "Aktualität (Recency) berechnen relativ zu" zurückliegen dürfen.
transaction_scale	Days Weeks Months Years	Nur verfügbar, wenn transaction_within_last ausgewählt wurde. Hier können Sie anhand von Anzahl und Typ der Zeiträume (Tage, Wochen, Monate oder Jahre) angeben, wie weit die in die Analyse aufzunehmenden Datensätze ausgehend von "Aktualität (Recency) berechnen relativ zu" zurückliegen dürfen.
save_r2	*boolesch*	Zeigt für jeden Kunden das Datum der zweitaktuellsten Transaktion an.
save_r3	*boolesch*	Nur verfügbar, wenn save_r2 ausgewählt wurde. Zeigt für jeden Kunden das Datum der drittaktuellsten Transaktion an.

Eigenschaften des Knotens "Rprocess"

Mit dem Knoten "R-Prozess" können Sie Daten aus einem IBM(r) SPSS(r) Modeler-Stream beziehen und diese mit ihrem eigenen benutzerdefinierten R-Script ändern. Nachdem die Daten geändert wurden, werden sie an den Stream zurückgegeben.

Tabelle 51. Eigenschaften des Knotens "Rprocess".

Eigenschaften des Knotens Rprocess	**Datentyp**	**Eigenschaftsbeschreibung**
Syntax	*Zeichenfolge*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*boolesch*	

Tabelle 51. Eigenschaften des Knotens "Rprocess" (Forts.).

Eigenschaften des Knotens **Rprocess**	Datentyp	Eigenschaftsbeschreibung
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*boolesch*	

Eigenschaften des Knotens "sample"

Der Stichprobenknoten wählte ein Subset der Datensätze aus. Es wird eine Vielzahl von Stichprobentypen unterstützt, darunter geschichtete, gruppierte (Clusterstichproben) und nichtzufällige (strukturierte) Stichproben. Eine Stichprobenziehung kann nützlich zur Verbesserung der Leistungsfähigkeit und zur Auswahl von verwandten Datensätzen bzw. Transaktionen für die Analyse sein.

Tabelle 52. Eigenschaften des Knotens "sample".

Eigenschaften des Knotens **sample**	Datentyp	Eigenschaftsbeschreibung
method	Simple Complex	
mode	Include Discard	Einschließen oder Verwerfen von Datensätzen, die die angegebene Bedingung erfüllen.
sample_type	First OneInN RandomPct	Gibt die Methode der Stichprobenziehung an.
first_n	*ganze Zahl*	Datensätze bis zum angegebenen Abbruchpunkt werden eingeschlossen oder verworfen.
one_in_n	*Zahl*	Jeden *n*-ten Datensatz einschließen oder verwerfen.
rand_pct	*Zahl*	Geben Sie den Prozentsatz der einzuschließenden oder zu verwerfenden Datensätze an.
use_max_size	*boolesch*	Aktivieren Sie die Verwendung der Einstellung maximum_size.
maximum_size	*ganze Zahl*	Geben Sie die größte Stichprobe an, die in den Datenstream eingeschlossen oder verworfen werden soll. Diese Option ist redundant und wird daher inaktiviert, wenn First und Include angegeben werden.
set_random_seed	*boolesch*	Aktiviert die Verwendung der Einstellung für den Zufallsstartwert.
random_seed	*ganze Zahl*	Geben Sie den Wert an, der als Startwert für den Zufallsgenerator verwendet wird.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	

Tabelle 52. Eigenschaften des Knotens "sample" (Forts.).

Eigenschaften des Knotens **sample**	Datentyp	Eigenschaftsbeschreibung
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*Zahl*	
fixed_counts	*ganze Zahl*	
variable_proportions	*Feld*	
variable_counts	*Feld*	
use_min_stratum_size	*boolesch*	
minimum_stratum_size	*ganze Zahl*	Diese Option gilt nur, wenn eine komplexe Stichprobe mit Sample units=Proportions gezogen wird.
use_max_stratum_size	*boolesch*	
maximum_stratum_size	*ganze Zahl*	Diese Option gilt nur, wenn eine komplexe Stichprobe mit Sample units=Proportions gezogen wird.
clusters	*Feld*	
stratify_by	*[Feld1 ... FeldN]*	
specify_input_weight	*boolesch*	
input_weight	*Feld*	
new_output_weight	*Zeichenfolge*	
sizes_proportions	[{string *Zeichenfolgewert*}{string *Zeichenfolgewert*}...]	Wenn sample_units=proportions und sample_size_proportions=Custom festgelegt wurden, wird hiermit ein Wert für jede mögliche Kombination der Werte von Schichtungsfeldern angegeben.
default_proportion	*Zahl*	
sizes_counts	[{string *Zeichenfolgewert*}{string *Zeichenfolgewert*}...]	Dient zur Angabe eines Werts für jede mögliche Kombination der Werte von Schichtungsfeldern. Die Verwendung ist ähnlich wie bei sizes_proportions, es wird jedoch statt eines Anteils eine ganze Zahl angegeben.
default_count	*Zahl*	

Eigenschaften des Knotens "select"

Der Auswahlknoten wählt auf der Grundlage einer bestimmten Bedingung ein Subset von Datensätzen aus einem Datenstream aus oder verwirft sie. Sie können beispielsweise die Datensätze auswählen, die zu einer bestimmten Verkaufsregion gehören.

Tabelle 53. Eigenschaften des Knotens "select".

Eigenschaften des Knotens **select**	Datentyp	Eigenschaftsbeschreibung
mode	Include Discard	Definiert, ob die ausgewählten Datensätze eingeschlossen oder verworfen werden sollen.

Tabelle 53. Eigenschaften des Knotens "select" (Forts.).

Eigenschaften des Knotens **select**	Datentyp	Eigenschaftsbeschreibung
`condition`	*Zeichenfolge*	Bedingung für das Einschließen oder Verwerfen von Datensätzen.

Eigenschaften des Knotens "sort"

Der Sortierknoten sortiert Datensätze anhand der Werte eines oder mehrerer Felder in aufsteigender oder absteigender Reihenfolge.

Tabelle 54. Eigenschaften des Knotens "sort".

Eigenschaften des Knotens **sort**	Datentyp	Eigenschaftsbeschreibung
`keys`	[{*Zeichenfolge* `Ascending`} \ {*Zeichenfolge* `Descending`}]	Gibt die Felder an, nach denen sortiert werden soll. Wenn keine Richtung angegeben ist, wird der Standard verwendet.
`default_ascending`	*boolesch*	Gibt die Standardsortierreihenfolge an.
`use_existing_keys`	*boolesch*	Gibt an, ob die Sortierung durch Verwendung der vorherigen Sortierreihenfolge für bereits sortierte Felder optimiert werden soll.
`existing_keys`		Gibt die bereits sortieren Felder und ihre Sortierrichtung an. Verwendet dasselbe Format wie die Eigenschaft `keys`.

Eigenschaften des Knotens "streamingts"

Der Streaming-ZR-Knoten erstellt und bewertet Zeitreihenmodelle in einem Schritt, ohne dass ein Zeitintervallknoten erforderlich ist.

Tabelle 55. Eigenschaften des Knotens "streamingts".

Eigenschaften des Knotens **streamingts**	Datentyp	Eigenschaftsbeschreibung
`custom_fields`	*boolesch*	Bei `custom_fields=false` werden die Einstellungen aus einem vorausgehenden Typknoten verwendet. Bei `custom_fields=true` müssen `targets` und `inputs` angegeben werden.
`targets`	[*Feld1...FeldN*]	
`inputs`	[*Feld1...FeldN*]	

`method`	`ExpertModeler`
	`Exsmooth`
	`Arima`	

`calculate_conf`	*boolesch*	
`conf_limit_pct`	*real*	

Tabelle 55. Eigenschaften des Knotens "streamingts" (Forts.).

Eigenschaften des Knotens streamingts	**Datentyp**	**Eigenschaftsbeschreibung**
use_time_intervals_node	*boolesch*	Bei use_time_intervals_node=true werden die Einstellungen aus einem vorausgehenden Zeitintervallknoten verwendet. Andernfalls müssen interval_offset_position, interval_offset und interval_type angegeben werden.
interval_offset_position	LastObservation LastRecord	LastObservation bezieht sich auf **Letzte gültige Beobachtung**. LastRecord bezieht sich auf **Vom letzten Datensatz rückwärts zählen**.
interval_offset	*Zahl*	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*Felder*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*boolesch*	
detect_outliers	*boolesch*	
expert_outlier_additive	*boolesch*	
expert_outlier_level_shift	*boolesch*	
expert_outlier_innovational	*boolesch*	
expert_outlier_transient	*boolesch*	
expert_outlier_seasonal_additive	*boolesch*	
expert_outlier_local_trend	*boolesch*	
expert_outlier_additive_patch	*boolesch*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_d	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_q	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten

Tabelle 55. Eigenschaften des Knotens "streamingts" (Forts.).

Eigenschaften des Knotens streamingts	**Datentyp**	**Eigenschaftsbeschreibung**
arima_sp	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_sd	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_sq	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_transformation_type	None SquareRoot NaturalLog	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_include_constant	*boolesch*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
tf_arima_p.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_d.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_q.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_sp.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_sd.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_sq.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_delay.*Feldname*	*ganze Zahl*	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arima_transformation_type.*Feldname*	None SquareRoot NaturalLog	
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*boolesch*	
arima_outlier_level_shift	*boolesch*	
arima_outlier_innovational	*boolesch*	
arima_outlier_transient	*boolesch*	
arima_outlier_seasonal_additive	*boolesch*	
arima_outlier_local_trend	*boolesch*	
arima_outlier_additive_patch	*boolesch*	
deployment_force_rebuild	*boolesch*	

Tabelle 55. Eigenschaften des Knotens "streamingts" (Forts.).

Eigenschaften des Knotens `streamingts`	**Datentyp**	**Eigenschaftsbeschreibung**
`deployment_rebuild_mode`	`Count` `Percent`	
`deployment_rebuild_count`	*Zahl*	
`deployment_rebuild_pct`	*Zahl*	
`deployment_rebuild_field`	*<Feld>*	

Kapitel 11. Eigenschaften von Feldoperationsknoten

Eigenschaften des Knotens "anonymize"

Der Anonymisierungsknoten ändert die Art und Weise, wie Feldnamen und -werte weiter unten im Stream dargestellt werden, und verschleiert damit die ursprünglichen Daten. Dies kann sinnvoll sein, wenn andere Benutzer in die Lage versetzt werden sollen, Modelle unter Verwendung vertraulicher Daten wie beispielsweise Kundennamen zu erstellen.

Tabelle 56. Eigenschaften des Knotens "anonymize".

Eigenschaften des Knotens `anonymize`	**Datentyp**	**Eigenschaftsbeschreibung**
`enable_anonymize`	*boolesch*	Bei Festlegung auf T wird die Anonymisierung von Feldwerten aktiviert (entspricht der Auswahl von **Ja** für das betreffende Feld in der Spalte "Werte anonymisieren").
`use_prefix`	*boolesch*	Bei Festlegung auf T wird ein benutzerdefiniertes Präfix verwendet, sofern eines angegeben wurde. Gilt für Felder, die mit der Hash-Methode anonymisiert werden, und entspricht der Auswahl des Optionsfelds **Benutzerdefiniert** im Dialogfeld "Werte ersetzen" für das betreffende Feld.
`prefix`	*Zeichenfolge*	Entspricht der Eingabe eines Präfixes in das Textfeld im Dialogfeld "Werte ersetzen". Das Standardpräfix ist der Standardwert, wenn keine anderen Angaben gemacht wurden.
`transformation`	`Random` `Fixed`	Bestimmt, ob die Transformationsparameter für ein durch die Transformationsmethode anonymisiertes Feld zufällig oder fest sein sollen.
`set_random_seed`	*boolesch*	Bei Festlegung auf T wird der angegebene Startwert für den Zufallsgenerator verwendet (sofern außerdem "transformation" auf "Random" gesetzt ist).
`random_seed`	*ganze Zahl*	Wenn `set_random_seed` auf T gesetzt ist, wird dieser Wert als Startwert für den Zufallsgenerator verwendet.
`scale`	*Zahl*	Wenn "transformation" auf "Fixed" gesetzt ist, wird dieser Wert als Wert für "Skalieren um" verwendet. Der Höchstwert für die Skalierung ist normalerweise 10; er kann jedoch gesenkt werden, um einen Überlauf zu vermeiden.
`translate`	*Zahl*	Wenn "transformation" auf "Fixed" gesetzt ist, wird dieser Wert als Wert für die Verschiebung ("translate") verwendet. Der Höchstwert für die Verschiebung ist normalerweise 10; er kann jedoch gesenkt werden, um einen Überlauf zu vermeiden.

Eigenschaften des Knotens "autodataprep"

Der Knoten "Automated Data Preparation" (ADP) kann Ihre Daten analysieren und Korrekturen identifizieren, problematische oder vermutlich überflüssige Felder ausschließen, wie erforderlich neue Attribute ableiten und die Leistung durch intelligente Prüf- und Stichprobenverfahren verbessern. Sie können den Knoten vollständig automatisiert nutzen, damit er Korrekturen wählen und anwenden kann. Sie können die Änderungen aber auch prüfen, bevor sie durchgeführt werden, und wie gewünscht akzeptieren, ablehnen oder ändern.

Tabelle 57. Eigenschaften des Knotens "autodataprep".

Eigenschaften des Knotens autodataprep	**Datentyp**	**Eigenschaftsbeschreibung**
objective	Balanced Speed Accuracy Custom	
custom_fields	*boolesch*	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
target	*Feld*	Gibt ein einzelnes Zielfeld an.
inputs	[*Feld1 ... FeldN*]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
use_frequency	*boolesch*	
frequency_field	*Feld*	
use_weight	*boolesch*	
weight_field	*Feld*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*boolesch*	Zugriff auf alle Datums- und Zeitfelder kontrollieren
compute_time_until_date	*boolesch*	
reference_date	Today Fixed	
fixed_date	*Datum*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*boolesch*	
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*boolesch*	
extract_month_from_date	*boolesch*	
extract_day_from_date	*boolesch*	
extract_hour_from_time	*boolesch*	
extract_minute_from_time	*boolesch*	

Tabelle 57. Eigenschaften des Knotens "autodataprep" (Forts.).

Eigenschaften des Knotens autodataprep	**Datentyp**	**Eigenschaftsbeschreibung**
extract_second_from_time	*boolesch*	
exclude_low_quality_inputs	*boolesch*	
exclude_too_many_missing	*boolesch*	
maximum_percentage_missing	*Zahl*	
exclude_too_many_categories	*boolesch*	
maximum_number_categories	*Zahl*	
exclude_if_large_category	*boolesch*	
maximum_percentage_category	*Zahl*	
prepare_inputs_and_target	*boolesch*	
adjust_type_inputs	*boolesch*	
adjust_type_target	*boolesch*	
reorder_nominal_inputs	*boolesch*	
reorder_nominal_target	*boolesch*	
replace_outliers_inputs	*boolesch*	
replace_outliers_target	*boolesch*	
replace_missing_continuous_inputs	*boolesch*	
replace_missing_continuous_target	*boolesch*	
replace_missing_nominal_inputs	*boolesch*	
replace_missing_nominal_target	*boolesch*	
replace_missing_ordinal_inputs	*boolesch*	
replace_missing_ordinal_target	*boolesch*	
maximum_values_for_ordinal	*Zahl*	
minimum_values_for_continuous	*Zahl*	
outlier_cutoff_value	*Zahl*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*boolesch*	
rescaling_method	MinMax ZScore	
min_max_minimum	*Zahl*	
min_max_maximum	*Zahl*	
z_score_final_mean	*Zahl*	
z_score_final_sd	*Zahl*	
rescale_continuous_target	*boolesch*	
target_final_mean	*Zahl*	
target_final_sd	*Zahl*	
transform_select_input_fields	*boolesch*	
maximize_association_with_target	*boolesch*	
p_value_for_merging	*Zahl*	
merge_ordinal_features	*boolesch*	

Tabelle 57. Eigenschaften des Knotens "autodataprep" (Forts.).

Eigenschaften des Knotens **autodataprep**	Datentyp	Eigenschaftsbeschreibung
merge_nominal_features	*boolesch*	
minimum_cases_in_category	*Zahl*	
bin_continuous_fields	*boolesch*	
p_value_for_binning	*Zahl*	
perform_feature_selection	*boolesch*	
p_value_for_selection	*Zahl*	
perform_feature_construction	*boolesch*	
transformed_target_name_extension	*Zeichenfolge*	
transformed_inputs_name_extension	*Zeichenfolge*	
constructed_features_root_name	*Zeichenfolge*	
years_duration_ name_extension	*Zeichenfolge*	
months_duration_ name_extension	*Zeichenfolge*	
days_duration_ name_extension	*Zeichenfolge*	
hours_duration_ name_extension	*Zeichenfolge*	
minutes_duration_ name_extension	*Zeichenfolge*	
seconds_duration_ name_extension	*Zeichenfolge*	
year_cyclical_name_extension	*Zeichenfolge*	
month_cyclical_name_extension	*Zeichenfolge*	
day_cyclical_name_extension	*Zeichenfolge*	
hour_cyclical_name_extension	*Zeichenfolge*	
minute_cyclical_name_extension	*Zeichenfolge*	
second_cyclical_name_extension	*Zeichenfolge*	

Eigenschaften des Knotens "binning"

Der Klassierknoten erstellt automatisch neue nominale Felder (Setfelder) auf der Grundlage der Werte eines oder mehrerer bestehender stetiger Felder (numerischer Bereich). Sie können beispielsweise ein stetiges Einkommensfeld in ein neues kategoriales Feld transformieren, das Einkommensgruppen als Abweichungen vom Mittelwert enthält. Nach der Erstellung von Klassen für das neue Feld können Sie einen Ableitungsknoten anhand der Trennwerte generieren.

Tabelle 58. Eigenschaften des Knotens "binning".

Eigenschaften des Knotens **binning**	Datentyp	Eigenschaftsbeschreibung
fields	*[Feld1 Feld2 ... Feldn]*	Stetige Felder (numerischer Bereich) mit ausstehender Transformation. Sie können mehrere Felder gleichzeitig klassieren.
method	FixedWidth EqualCount Rank SDev Optimal	Methode, die zur Ermittlung der Trennwerte für neue Feld-Bins (Kategorien) verwendet wird.

Tabelle 58. Eigenschaften des Knotens "binning" (Forts.).

Eigenschaften des Knotens binning	Datentyp	Eigenschaftsbeschreibung
rcalculate_bins	Always IfNecessary	Gibt an, ob bei jeder Ausführung des Knotens die Klassen neu berechnet und die Daten in die relevante Klasse eingeordnet werden sollen oder ob Daten nur zu bestehenden Klassen und etwaig hinzugefügten neuen Klassen hinzugefügt werden sollen.
fixed_width_name_extension	*Zeichenfolge*	Die Standarderweiterung lautet *_BIN*.
fixed_width_add_as	Suffix Prefix	Gibt an, ob die Erweiterung am Ende (Suffix) oder am Anfang (Präfix) des Feldnamens eingefügt werden soll. Die Standarderweiterung lautet *income_BIN*.
fixed_bin_method	Width Count	
fixed_bin_count	*ganze Zahl*	Gibt eine ganze Zahl an, die zur Bestimmung der Anzahl der Klassen (Kategorien) mit fester Breite für die neuen Felder verwendet wird.
fixed_bin_width	*real*	Wert (ganzzahlig oder reell), der zu Berechnung der Breite der Klasse verwendet wird.
equal_count_name_ extension	*Zeichenfolge*	Die Standarderweiterung lautet *_TILE*.
equal_count_add_as	Suffix Prefix	Gibt eine Erweiterung (Suffix oder Präfix) an, die für die mithilfe von Standard-N-Perzentilen generierten Felder verwendet wird. Die Standarderweiterung ist *_TILE* plus *N*; dabei steht *N* für die Nummer des Perzentils.
tile4	*boolesch*	Generiert vier Quantilklassen, die jeweils 25 % der Fälle enthalten.
tile5	*boolesch*	Generiert fünf Qintilklassen.
tile10	*boolesch*	Generiert 10 Dezilklassen.
tile20	*boolesch*	Generiert 20 Vingtilklassen.
tile100	*boolesch*	Generiert 100 Perzentilklassen.
use_custom_tile	*boolesch*	
custom_tile_name_extension	*Zeichenfolge*	Die Standarderweiterung lautet *_TILEN*.
custom_tile_add_as	Suffix Prefix	
custom_tile	*ganze Zahl*	
equal_count_method	RecordCount ValueSum	Die Methode RecordCount versucht, jeder Klasse eine gleich große Anzahl von Datensätzen zuzuweisen, während ValueSum Datensätze so zuweist, dass die Summe der Werte in jeder Klasse gleich groß ist.
tied_values_method	Next Current Random	Gibt an, in welche Klasse Daten mit gebundenen Werten (Werten mit Gleichstand) eingeordnet werden sollen.
rank_order	Ascending Descending	Diese Eigenschaft beinhaltet Ascending (der niedrigste Wert wird mit "1" gekennzeichnet) oder Descending (der höchste Wert wird mit "1" gekennzeichnet).

Tabelle 58. Eigenschaften des Knotens "binning" (Forts.).

Eigenschaften des Knotens binning	Datentyp	Eigenschaftsbeschreibung
rank_add_as	Suffix Prefix	Mit dieser Option werden Rang, relativer Rang und Prozentsatzrang angewendet.
rank	*boolesch*	
rank_name_extension	*Zeichenfolge*	Die Standarderweiterung lautet *_RANK*.
rank_fractional	*boolesch*	Weist Fällen Ränge zu, wobei der Wert des neuen Felds gleich dem Rang dividiert durch die Summe der Gewichtungen der nicht fehlenden Fälle ist. Relative Ränge fallen in den Bereich 0-1.
rank_fractional_name_ extension	*Zeichenfolge*	Die Standarderweiterung lautet *_F_RANK*.
rank_pct	*boolesch*	Die einzelnen Ränge werden durch die Anzahl der Datensätze mit gültigen Werten dividiert und mit 100 multipliziert. Als Prozentsatz angegebene relative Ränge fallen in den Bereich 1-100.
rank_pct_name_extension	*Zeichenfolge*	Die Standarderweiterung lautet *_P_RANK*.
sdev_name_extension	*Zeichenfolge*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*Zeichenfolge*	Die Standarderweiterung lautet *_OPTIMAL*.
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	*Feld*	Als Supervisorfeld ausgewähltes Feld, mit dem die für die Klassierung ausgewählten Felder in Bezug stehen.
optimal_merge_bins	*boolesch*	Gibt an, dass alle Klassen mit kleinen Fallzahlen zu einer größeren, benachbarten Klasse hinzugefügt werden.
optimal_small_bin_threshold	*ganze Zahl*	
optimal_pre_bin	*boolesch*	Gibt an, dass eine Vorklassierung des Datasets durchgeführt werden soll.
optimal_max_bins	*ganze Zahl*	Gibt eine Obergrenze an, um das Erstellen einer übermäßig großen Anzahl von Feldern zu verhindern.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Eigenschaften des Knotens "derive"

Der Ableitungsknoten ändert Datenwerte oder erstellt neue Felder aus einem oder mehreren bestehenden Feldern. Er erstellt Felder vom Typ "Formel", "Flag", "Nominal", "Status", "Anzahl" und "Bedingt".

Tabelle 59. Eigenschaften des Knotens "derive".

Eigenschaften des Knotens derive	Datentyp	Eigenschaftsbeschreibung
new_name	*Zeichenfolge*	Name des neuen Felds.
mode	Single Multiple	Gibt eines oder mehrere Felder an.
fields	[*Feld Feld Feld*]	Wird nur im Modus "Multiple" (Mehrere) zur Auswahl mehrerer Felder verwendet.
name_extension	*Zeichenfolge*	Gibt die Erweiterung für die neuen Feldnamen an.
add_as	Suffix Prefix	Fügt die Erweiterung als Präfix (am Anfang) oder als Suffix (am Ende) des Feldnamens ein.
result_type	Formula Flag Set State Count Conditional	Die sechs Typen neuer Felder, die Sie erstellen können.
formula_expr	*Zeichenfolge*	Ausdruck zum Berechnen eines neuen Feldwerts in einem Ableitungsknoten.
flag_expr	*Zeichenfolge*	
flag_true	*Zeichenfolge*	
flag_false	*Zeichenfolge*	
set_default	*Zeichenfolge*	
set_value_cond	*Zeichenfolge*	Wird zur Bereitstellung der Bedingung, die einem bestimmten Wert zugeordnet ist, strukturiert.
state_on_val	*Zeichenfolge*	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "On" (Ein) erfüllt ist.
state_off_val	*Zeichenfolge*	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "Off" (Aus) erfüllt ist.
state_on_expression	*Zeichenfolge*	
state_off_expression	*Zeichenfolge*	
state_initial	On Off	Weist jedem Datensatz des neuen Felds einen Anfangswert On (Ein) oder Off (Aus) zu. Dieser Wert kann sich ändern, wenn die einzelnen Bedingungen erfüllt werden.
count_initial_val	*Zeichenfolge*	
count_inc_condition	*Zeichenfolge*	
count_inc_expression	*Zeichenfolge*	
count_reset_condition	*Zeichenfolge*	

Tabelle 59. Eigenschaften des Knotens "derive" (Forts.).

Eigenschaften des Knotens derive	Datentyp	Eigenschaftsbeschreibung
cond_if_cond	*Zeichenfolge*	
cond_then_expr	*Zeichenfolge*	
cond_else_expr	*Zeichenfolge*	

Eigenschaften des Knotens "ensemble"

Der Ensemble-Knoten kombiniert zwei oder mehr Modellnuggets, um genauere Vorhersagen zu erzielen, als aus einem dieser Modelle allein gewonnen werden können.

Tabelle 60. Eigenschaften des Knotens "ensemble".

Eigenschaften des Knotens ensemble	Datentyp	Eigenschaftsbeschreibung
ensemble_target_field	*Feld*	Gibt das Zielfeld für alle im Ensemble verwendeten Modelle an.
filter_individual_model_output	*boolesch*	Gibt an, ob Scoring-Ergebnisse aus einzelnen Modellen unterdrückt werden sollen.
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_voting_tie_selection	Random HighestConfidence	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.
calculate_standard_error	*boolesch*	Wenn das Zielfeld stetig ist, wird standardmäßig eine Standardfehlerberechnung durchgeführt, um den Unterschied zwischen den gemessenen oder geschätzten Werten und den wahren Werten zu berechnen sowie um zu zeigen, wie hoch die Übereinstimmung dieser Schätzungen war.

Eigenschaften des Knotens "filler"

Der Füllerknoten ersetzt Feldwerte und ändert den Speichertyp. Sie können auswählen, dass die Werte auf der Grundlage einer CLEM-Bedingung wie beispielsweise `@BLANK(@FIELD)` ersetzt werden sollen. Alternativ können Sie auswählen, dass alle Leerstellen oder Nullwerte mit einem bestimmten Wert ersetzt werden sollen. Füllerknoten werden häufig zusammen mit einem Typknoten verwendet, um fehlende Werte zu ersetzen.

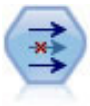
Tabelle 61. Eigenschaften des Knotens "filler".

Eigenschaften des Knotens `filler`	Datentyp	Eigenschaftsbeschreibung
`fields`	[*Feld Feld Feld*]	Felder aus dem Dataset, deren Werte untersucht und ersetzt werden.

`replace_mode`	`Always`
	`Conditional`
	`Blank`
	`Null`
	`BlankAndNull`
		Sie können alle Werte, leere Werte, Nullwerte oder Werte ersetzen, die einer bestimmten Bedingung entsprechen.

`condition`	*Zeichenfolge*	
`replace_with`	*Zeichenfolge*	

Eigenschaften des Knotens "filter"



Der Filterknoten filtert (verwirft) Felder, benennt Felder um und ordnet Felder von einem Quellenknoten einem anderen zu.

Verwenden der Eigenschaft default_include. Beachten Sie, dass die Festlegung des Werts der Eigenschaft `default_include` nicht automatisch zum Ein- oder Ausschluss aller Felder führt; es wird lediglich die Standardvorgehensweise für die ausgewählten Felder festgelegt. Diese Eigenschaft entspricht in ihrer Funktion dem Klicken auf die Schaltfläche **Felder standardmäßig einschließen** im Dialogfeld des Filterknotens.

Tabelle 62. Eigenschaften des Knotens "filter".

Eigenschaften des Knotens `filter`	Datentyp	Eigenschaftsbeschreibung
`default_include`	*boolesch*	Verschlüsselte Eigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden. Beachten Sie, dass die Festlegung dieser Eigenschaft nicht automatisch zum Ein- oder Ausschluss aller Felder führt; es wird lediglich festgelegt, ob die ausgewählten Felder standardmäßig ein- oder ausgeschlossen werden sollen.
`include`	*boolesch*	Verschlüsselte Eigenschaft zum Einbeziehen und Entfernen von Feldern.
`new_name`	*Zeichenfolge*	

Eigenschaften des Knotens "history"

Der Verlaufsknoten erstellt neue Felder mit Daten aus Feldern in vorangegangenen Datensätzen. Verlaufsknoten werden am häufigsten für sequenzielle Daten, beispielsweise Zeitreihendaten, verwendet. Vor der Verwendung eines Verlaufsknotens sollten die Daten mithilfe eines Sortierknotens sortiert werden.

Tabelle 63. Eigenschaften des Knotens "history".

Eigenschaften des Knotens `history`	Datentyp	Eigenschaftsbeschreibung
`fields`	[*Feld Feld Feld*]	Felder, für die Sie einen Verlauf wollen.
`offset`	*Zahl*	Dient zur Angabe des jüngsten Datensatzes (vor dem aktuellen Datensatz), aus dem Verlaufsfeldwerte extrahiert werden sollen.
`span`	*Zahl*	Gibt an, aus wie vielen früheren Datensätzen Werte extrahiert werden sollen.

`unavailable`	`Discard`
	`Leave`
	`Fill`
		Für die Behandlung von Datensätzen, die keine Verlaufswerte besitzen, bezieht sich dies normalerweise auf die ersten Datensätze oben im Dataset, für die es keine vorangegangenen Datensätze gibt, die als Verlauf dienen könnten.

`fill_with`	`Zeichenfolge` `Zahl`	Gibt einen Wert oder eine Zeichenfolge an, die für Datensätze verwendet werden soll, wenn kein Verlaufswert verfügbar ist.

Eigenschaften des Knotens "partition"

Der Partitionsknoten erstellt ein Partitionsfeld, das Daten in getrennte Subsets für die Trainings-, Test- und Validierungsphase der Modellerstellung aufteilt.

Tabelle 64. Eigenschaften des Knotens "partition".

Eigenschaften des Knotens `partition`	Datentyp	Eigenschaftsbeschreibung
`new_name`	*Zeichenfolge*	Der vom Knoten erstellte Name des Partitionsfelds.
`create_validation`	*boolesch*	Gibt an, ob eine Validierungspartition erstellt werden soll.
`training_size`	*ganze Zahl*	Prozentsatz der Datensätze (0-100), die der Trainingspartition zugeordnet werden sollen.
`testing_size`	*ganze Zahl*	Prozentsatz der Datensätze (0-100), die der Testpartition zugeordnet werden sollen.
`validation_size`	*ganze Zahl*	Prozentsatz der Datensätze (0-100), die der Validierungspartition zugeordnet werden sollen. Wird ignoriert, wenn keine Validierungspartition erstellt wird.
`training_label`	*Zeichenfolge*	Beschriftung der Trainingspartition.
`testing_label`	*Zeichenfolge*	Beschriftung der Testpartition.

Tabelle 64. Eigenschaften des Knotens "partition" (Forts.).

Eigenschaften des Knotens `partition`	Datentyp	Eigenschaftsbeschreibung
`validation_label`	*Zeichenfolge*	Beschriftung der Validierungspartition. Wird ignoriert, wenn keine Validierungspartition erstellt wird.

`value_mode`	`System`
	`SystemAndLabel`
	`Label`
		Gibt die Werte an, die für die einzelnen Partitionen in den Daten verwendet werden. Beispiel: Die Trainingsstichprobe kann durch die Systemganzzahl 1, die Beschriftung `Training` bzw. eine Kombination aus beiden durch `1_Training` repräsentiert werden.

`set_random_seed`	*boolesch*	Gibt an, ob ein benutzerdefinierter Startwert für den Zufallsgenerator verwendet werden soll.
`random_seed`	*ganze Zahl*	Ein benutzerdefinierter Startwert für den Zufallsgenerator festlegen. Damit dieser Wert verwendet wird, muss `set_random_seed` auf `True` (wahr) gesetzt sein.
`enable_sql_generation`	*boolesch*	Gibt an, ob SQL-Pushback für die Zuweisung von Datensätzen zu Partitionen verwendet werden soll.
`unique_field`		Gibt das Eingabefeld an, mit dessen Hilfe sichergestellt werden soll, dass Datensätze auf zufällige, aber wiederholbare Weise zu Partitionen zugeordnet werden. Damit dieser Wert verwendet wird, muss `enable_sql_generation` auf `True` (wahr) gesetzt sein.

Eigenschaften des Knotens "reclassify"

Der Umcodierungsknoten transformiert ein Set kategorialer Werte in ein anderes. Die Umcodierung dient zur Reduzierung von Kategorien bzw. Neugruppierung von Daten für die Analyse.

Tabelle 65. Eigenschaften des Knotens "reclassify".

Eigenschaften des Knotens `reclassify`	Datentyp	Eigenschaftsbeschreibung
`mode`	`Single` `Multiple`	`Single` codiert die Kategorien eines einzelnen Felds um. `Multiple` aktiviert Optionen, die die Transformation von mehreren Feldern gleichzeitig erlauben.
`replace_field`	*boolesch*	
`field`	*Zeichenfolge*	Wird nur im Modus "Single" verwendet.
`new_name`	*Zeichenfolge*	Wird nur im Modus "Single" verwendet.
`fields`	*[Feld1 Feld2 ... Feldn]*	Wird nur im Modus "Multiple" verwendet.
`name_extension`	*Zeichenfolge*	Wird nur im Modus "Multiple" verwendet.
`add_as`	`Suffix` `Prefix`	Wird nur im Modus "Multiple" verwendet.
`reclassify`	*Zeichenfolge*	Strukturierte Eigenschaft für Feldwerte.
`use_default`	*boolesch*	Standardwert verwenden.

Tabelle 65. Eigenschaften des Knotens "reclassify" (Forts.).

Eigenschaften des Knotens reclassify	Datentyp	Eigenschaftsbeschreibung
default	*Zeichenfolge*	Standardwert angeben.
pick_list	*[Zeichenfolge ZeichenfolgeZeichenfolge ... Zeichenfolge]*	Ermöglicht einem Benutzer den Import einer Liste bekannter neuer Werte, um die Dropdown-Liste in der Tabelle zu füllen.

Eigenschaften des Knotens "reorder"

Der Knoten "Felder ordnen" definiert die natürliche Reihenfolge, die bei der Anzeige der nachfolgenden Felder verwendet wird. Diese Reihenfolge betrifft die Anzeige von Feldern an unterschiedlichen Stellen, beispielsweise in Tabellen, Listen und in der Feldauswahl. Dieser Vorgang dient beispielsweise dazu, um bei der Arbeit mit umfangreichen Datasets die relevanten Felder deutlicher hervorzuheben.

Tabelle 66. Eigenschaften des Knotens "reorder".

Eigenschaften des Knotens reorder	Datentyp	Eigenschaftsbeschreibung
mode	Custom Auto	Sie können Werte automatisch sortieren oder eine benutzerdefinierte Reihenfolge angeben.
sort_by	Name Type Storage	
ascending	*boolesch*	
start_fields	*[Feld1 Feld2 ... Feldn]*	Nach diesen Feldern werden neue Felder eingefügt.
end_fields	*[Feld1 Feld2 ... Feldn]*	Vor diesen Feldern werden neue Felder eingefügt.

Eigenschaften des Knotens "restructure"

Der Knoten "Umstrukturieren" konvertiert ein nominales Feld oder ein Flagfeld in eine Gruppe von Feldern, die mit den Werten aus einem weiteren Feld ausgefüllt werden können. Beispiel: Aus einem Feld mit dem Namen *Zahlungsart*, mit den Werten *Kreditkarte*, *Bar* und *EC-Karte* werden drei neue Felder erstellt (*Kreditkarte*, *Bar*, *EC-Karte*), die jeweils den Wert der jeweiligen Zahlung enthalten.

Tabelle 67. Eigenschaften des Knotens "restructure".

Eigenschaften des Knotens restructure	Datentyp	Eigenschaftsbeschreibung
fields_from	[*Kategorie Kategorie Kategorie*] all	
include_field_name	*boolesch*	Gibt an, ob der Feldname im umstrukturierten Feldnamen verwendet werden soll.
value_mode	OtherFields Flags	Legt den Modus für die Angabe der Werte für die umstrukturierten Felder fest. Bei OtherFields müssen Sie angeben, welche Felder verwendet werden sollen (siehe unten). Bei Flags sind die Werte numerische Flags.

Tabelle 67. Eigenschaften des Knotens "restructure" (Forts.).

Eigenschaften des Knotens **restructure**	Datentyp	Eigenschaftsbeschreibung
`value_fields`	[*Feld Feld Feld*]	Erforderlich, wenn `value_mode` auf `OtherFields` gesetzt ist. Gibt an, welche Felder als Wertfelder verwendet werden sollen.

Eigenschaften des Knotens "rfmanalysis"

Mit dem Knoten "RFM-Analyse" (Recency-, Frequency-, Monetary-Analyse) können Sie quantitativ ermitteln, welche Kunden wahrscheinlich die besten sind, indem Sie untersuchen, wann sie zuletzt etwas von Ihnen erworben haben (Recency (Aktualität)), wie häufig sie eingekauft haben (Frequency (Häufigkeit)) und wie viel sie für alle Transaktionen zusammengenommen ausgegeben haben (Monetary (Geldwert)).

Tabelle 68. Eigenschaften des Knotens "rfmanalysis".

Eigenschaften des Knotens **rfmanalysis**	Datentyp	Eigenschaftsbeschreibung
`recency`	*Feld*	Gibt das Feld für "Recency" (Aktualität) an. Dabei kann es sich um ein Datum, eine Zeitmarke oder eine einfache Zahl handeln.
`frequency`	*Feld*	Gibt das Feld für "Frequency" (Häufigkeit) an.
`monetary`	*Feld*	Gibt das Feld für "Monetary" (Geldwert) an.
`recency_bins`	*ganze Zahl*	Dient zur Angabe der Anzahl der zu generierenden Aktualitätsklassen.
`recency_weight`	*Zahl*	Dient zur Angabe der Gewichtung für die Aktualitätsdaten. Der Standardwert ist 100.
`frequency_bins`	*ganze Zahl*	Dient zur Angabe der Anzahl der zu generierenden Häufigkeitsklassen.
`frequency_weight`	*Zahl*	Dient zur Angabe der Gewichtung für die Häufigkeitsdaten. Der Standardwert ist 10.
`monetary_bins`	*ganze Zahl*	Dient zur Angabe der Anzahl der zu generierenden Klassen für den Geldwert.
`monetary_weight`	*Zahl*	Dient zur Angabe der Gewichtung für die Geldwertdaten. Der Standardwert lautet 1.
`tied_values_method`	`Next` `Current`	Gibt an, in welche Klasse Daten mit gebundenen Werten (Werten mit Gleichstand) eingeordnet werden sollen.
`recalculate_bins`	`Always` `IfNecessary`	
`add_outliers`	*boolesch*	Nur verfügbar, wenn `recalculate_bins` auf `IfNecessary` gesetzt ist. Wenn diese Einstellung festgelegt wurde, werden Datensätze, die unterhalb der untersten Klasse liegen, zur untersten Klasse hinzugefügt und Datensätze oberhalb der höchsten Klasse werden in die höchste Klasse aufgenommen.

`binned_field`	`Recency`
	`Frequency`
	`Monetary`	

Tabelle 68. Eigenschaften des Knotens "rfmanalysis" (Forts.).

Eigenschaften des Knotens **rfmanalysis**	Datentyp	Eigenschaftsbeschreibung
recency_thresholds	*value value*	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist. Dient zur Angabe der oberen und unteren Schwellenwerte für die Aktualitätsklassen. Der obere Schwellenwert einer Klasse wird als unterer Schwellenwert der nächsten Klasse verwendet. So werden beispielsweise mit [10 30 60] zwei Klassen definiert, wobei für die erste Klasse die Schwellenwerte 10 und 30 gelten und für die zweite Klasse die Schwellenwerte 30 und 60.
frequency_thresholds	*value value*	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.
monetary_thresholds	*value value*	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.

Eigenschaften des Knotens "settoflag"

Der Dichotomknoten leitet mehrere Flagfelder auf der Grundlage der kategorialen Werte ab, die für ein oder mehrere nominale Felder definiert sind.

Tabelle 69. Eigenschaften des Knotens "settoflag".

Eigenschaften des Knotens **settoflag**	Datentyp	Eigenschaftsbeschreibung
fields_from	[*Kategorie Kategorie Kategorie*] all	
true_value	*Zeichenfolge*	Gibt den Wert "Wahr" an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert lautet T.
false_value	*Zeichenfolge*	Gibt den Falsch-Wert an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert lautet F.
use_extension	*boolesch*	Verwenden Sie eine Erweiterung als Suffix oder Präfix für das neue Flagfeld.
extension	*Zeichenfolge*	
add_as	Suffix Prefix	Gibt an, ob die Erweiterung als Suffix oder als Präfix hinzugefügt wird.
aggregate	*boolesch*	Fasst Datensätze anhand von Schlüsselfeldern zu Gruppen zusammen. Alle in einer Gruppe vorhandenen Flagfelder werden aktiviert, wenn einer der Datensätze auf "true" gesetzt wird.
keys	[*Feld Feld Feld*]	Schlüsselfelder.

Eigenschaften des Knotens "statisticstransform"

Der Statistics-Transformationsknoten führt eine Auswahl von IBM SPSS Statistics-Syntaxbefehlen für Datenquellen in IBM SPSS Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften des Knotens "statisticstransform"" auf Seite 245.

Eigenschaften des Knotens "timeintervals"

Der Zeitintervallknoten gibt Intervalle an und erstellt (bei Bedarf) Beschriftungen für die Modellierung von Zeitreihendaten. Wenn die Werte nicht gleichmäßig verteilt sind, kann der Knoten nach Bedarf Werte auffüllen oder aggregieren, um ein gleichmäßiges Intervall zwischen den Datensätzen zu erzeugen.

Tabelle 70. Eigenschaften des Knotens "timeintervals".

Eigenschaften des Knotens `timeintervals`	**Datentyp**	**Eigenschaftsbeschreibung**

`interval_type`	`None`
	`Periods`
	`CyclicPeriods`
	`Years`
	`Quarters`
	`Months`
	`DaysPerWeek`
	`DaysNonPeriodic`
	`HoursPerDay`
	`HoursNonPeriodic`
	`MinutesPerDay`
	`MinutesNonPeriodic`
	`SecondsPerDay`
	`SecondsNonPeriodic`	

`mode`	`Label` `Create`	Gibt an, ob Sie die Datensätze nacheinander beschriftet werden sollen oder ob die Zeitreihe auf der Grundlage eines angegebenen Datums-, Zeitmarken- oder Zeitfelds erstellt werden soll.
`field`	*Feld*	Gibt beim Erstellen der Serie aus den Daten das Feld an, das das Datum bzw. die Uhrzeit für jeden Datensatz anzeigt.
`period_start`	*ganze Zahl*	Gibt das Startintervall für Perioden bzw. zyklische Perioden an.
`cycle_start`	*ganze Zahl*	Startzyklus für zyklische Perioden.
`year_start`	*ganze Zahl*	Bei entsprechenden Intervalltypen das Jahr, in das das erste Intervall fällt.
`quarter_start`	*ganze Zahl*	Bei entsprechenden Intervalltypen das Quartal, in das das erste Intervall fällt.

Tabelle 70. Eigenschaften des Knotens "timeintervals" (Forts.).

Eigenschaften des Knotens timeintervals	**Datentyp**	**Eigenschaftsbeschreibung**
month_start	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	
day_start	*ganze Zahl*	
hour_start	*ganze Zahl*	
minute_start	*ganze Zahl*	
second_start	*ganze Zahl*	
periods_per_cycle	*ganze Zahl*	Bei zyklischen Perioden, die Anzahl innerhalb jedes Zyklus.
fiscal_year_begins	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	Gibt bei vierteljährlichen Intervallen den Monat an, in dem das Geschäftsjahr beginnt.
week_begins_on	Sunday Monday Tuesday Wednesday Donnerstag Friday Saturday Sunday	Gibt bei periodischen Intervallen (Tage pro Woche, Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) den Tag an, an dem die Woche beginnt.
day_begins_hour	*ganze Zahl*	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Stunde an, zu der der Tag beginnt. Kann in Verbindung mit day_begins_minute und day_begins_second verwendet werden, um einen genauen Zeitpunkt anzugeben wie beispielsweise *8:05:01*. Siehe unten stehendes Anwendungsbeispiel.
day_begins_minute	*ganze Zahl*	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Minute an, in der der Tag beginnt (z. B. die *5* in *8:05*).

Tabelle 70. Eigenschaften des Knotens "timeintervals" (Forts.).

Eigenschaften des Knotens timeintervals	**Datentyp**	**Eigenschaftsbeschreibung**
day_begins_second	*ganze Zahl*	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Sekunde an, in der der Tag beginnt (z. B. die *17* in *8:05:17*).
days_per_week	*ganze Zahl*	Gibt bei periodischen Intervallen (Tage pro Woche, Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Anzahl der Tage pro Woche an.
hours_per_day	*ganze Zahl*	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Anzahl der Stunden pro Tag an.
interval_increment	1 Z 3 4 5 6 10 15 20 30	Gibt bei Minuten pro Tag und Sekunden pro Tag die Anzahl der Minuten bzw. Sekunden an, um die der Wert für jeden Datensatz erhöht werden soll.
field_name_extension	*Zeichenfolge*	
field_name_extension_as_prefix	*boolesch*	
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	

Tabelle 70. Eigenschaften des Knotens "timeintervals" (Forts.).

Eigenschaften des Knotens timeintervals	**Datentyp**	**Eigenschaftsbeschreibung**
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	Gibt die Aggregationsmethode für ein Feld an.
pad	Blank MeanOfRecentPoints True False	Gibt die Auffüllmethode für ein Feld an.
agg_mode	All Specify	Gibt an, ob alle Felder mit Standardfunktionen nach Bedarf aggregiert bzw. aufgefüllt werden sollen oder ob die zu verwendenden Felder und Funktionen angegeben werden sollen.
agg_range_default	Mean Sum Mode Min Max	Gibt die beim Aggregieren von stetigen Feldern zu verwendende Standardfunktion an.
agg_set_default	Mode First Last	Gibt die beim Aggregieren von nominalen Feldern zu verwendende Standardfunktion an.
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	Blank MeanOfRecentPoints	Gibt die beim Auffüllen von stetigen Feldern zu verwendende Standardfunktion an.
pad_set_default	Blank MostRecentValue	
pad_flag_default	Blank True False	
max_records_to_create	*ganze Zahl*	Gibt die maximale Anzahl der beim Auffüllen der Reihe zu erstellenden Datensätze an.

Tabelle 70. Eigenschaften des Knotens "timeintervals" (Forts.).

Eigenschaften des Knotens `timeintervals`	Datentyp	Eigenschaftsbeschreibung
`estimation_from_beginning`	*boolesch*	
`estimation_to_end`	*boolesch*	
`estimation_start_offset`	*ganze Zahl*	
`estimation_num_holdouts`	*ganze Zahl*	
`create_future_records`	*boolesch*	
`num_future_records`	*ganze Zahl*	
`create_future_field`	*boolesch*	
`future_field_name`	*Zeichenfolge*	

Eigenschaften des Knotens "transpose"

Der Transponierknoten vertauscht die Daten in Zeilen und Spalten, sodass aus Datensätzen Felder und aus Feldern Datensätze werden.

Tabelle 71. Eigenschaften des Knotens "transpose".

Eigenschaften des Knotens `transpose`	Datentyp	Eigenschaftsbeschreibung
`transposed_names`	`Prefix` `Read`	Neue Felder können automatisch auf der Grundlage eines angegebenen Präfixes generiert oder aus einem bestehenden Feld in den Daten eingelesen werden.
`prefix`	*Zeichenfolge*	
`num_new_fields`	*ganze Zahl*	Bei Verwendung eines Präfixes wird die maximale Anzahl der zu erstellenden Felder angegeben.
`read_from_field`	*Feld*	Felder, aus denen Namen gelesen werden. Es muss sich um ein instanziiertes Feld handeln. Andernfalls tritt bei der Ausführung des Knotens ein Fehler auf.
`max_num_fields`	*ganze Zahl*	Beim Einlesen von Namen aus einem Feld wird eine Obergrenze angegeben, um das Erstellen einer übermäßig großen Anzahl von Feldern zu verhindern.

`transpose_type`	`Numeric`
	`Zeichenfolge`
	`Custom`
		Standardmäßig werden nur stetige Felder (numerischer Bereich) transponiert, Sie können jedoch stattdessen auch ein benutzerdefiniertes Subset numerischer Felder auswählen oder alle Zeichenfolgenfelder transponieren.

`transpose_fields`	[*Feld Feld Feld*]	Gibt die bei Verwendung der Option `Custom` (Angepasst) zu transponierenden Felder an.
`id_field_name`	*Feld*	

Eigenschaften des Knotens "type"

Der Typknoten gibt Feldmetadaten und Eigenschaften an. Sie können beispielsweise ein Messniveau (stetig, nominal, ordinal oder Flag) für die einzelnen Felder angeben, Optionen für den Umgang mit fehlenden Werten und systemdefinierten Nullwerten festlegen, die Rolle eines Felds zu Modellierungszwecken festlegen, Feld- und Wertbeschriftungen angeben oder die Werte für ein Feld angeben.

Beachten Sie: In einigen Fällen müssen Sie den Knoten "type" möglicherweise vollständig instanziieren, damit die anderen Knoten ordnungsgemäß arbeiten, beispielsweise die Eigenschaft `fields from` (Felder aus) des Dichotomknotens. Sie können einfach einen Tabellenknoten anschließen und ausführen, um die Felder zu instanziieren.

Tabelle 72. Eigenschaften des Knotens "type".

Eigenschaften des Knotens type	Datentyp	Eigenschaftsbeschreibung

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Verschlüsselte Eigenschaft für Feldrollen. *Hinweis*: Die Werte `Eingehend` und `Ausgehend` werden nicht mehr verwendet. In zukünftigen Versionen fällt möglicherweise die Unterstützung dafür weg.

`type`	`Range`
	`Flag`
	`Set`
	`Typeless`
	`Discrete`
	`OrderedSet`
	`Default`
		Messniveau des Felds (bisher als "type" bezeichnet). Wenn `type` auf `Default` gesetzt wird, werden alle Parametereinstellungen vom Typ `values` gelöscht, und wenn `value_mode` den Wert `Specify` besitzt, wird er auf `Read` gesetzt. Wird `value_mode` auf `Pass` oder `Read` gesetzt, hat das Einstellen von `type` keinerlei Auswirkung auf `value_mode`. Hinweis: Die intern verwendeten Datentypen unterscheiden sich von den im Typknoten sichtbaren Datentypen. Es ergibt sich folgende Korrespondenz:
		Range -> stetig
		Set - > nominal
		OrderedSet -> ordinal
		Discrete- > kategorial

`storage`	`Unknown`
	`String`
	`Integer`
	`Real`
	`Zeit`
	`Datum`
	`Timestamp`
		Schreibgeschützte verschlüsselte Eigenschaft für Feldspeichertyp.

`check`	`None`
	`Nullify`
	`Coerce`
	`Discard`
	`Warn`
	`Abort`
		Verschlüsselte Eigenschaft für das Überprüfen von Feldtyp und Bereich.

Tabelle 72. Eigenschaften des Knotens "type" (Forts.).

Eigenschaften des Knotens type	Datentyp	Eigenschaftsbeschreibung
values	[*Wert Wert*]	Bei einem stetigen Feld ist der erste Wert das Minimum und der letzte das Maximum. Geben Sie für nominale Felder alle Werte an. Bei Flagfeldern steht der erste Wert für *falsch* und der letzte für *wahr*. Bei automatischer Festlegung dieser Eigenschaft wird die Eigenschaft value_mode auf Specify festgelegt.
value_mode	Read Pass Read+ Current Specify	Bestimmt, wie Werte festgelegt werden. Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf Angeben festlegen können. Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft values fest.
extend_values	*boolesch*	Gilt, wenn value_mode auf Read gesetzt ist. Setzen Sie den Wert auf T, um neu gelesene Werte zu bereits für das Feld vorhandenen Werten hinzuzufügen. Setzen Sie den Wert auf F, um vorhandene Werte zu verwerfen und sie durch neu gelesene Werte zu ersetzen.
enable_missing	*boolesch*	Bei Festlegung auf T wird die Verfolgung von fehlenden Werten für das Feld aktiviert.
missing_values	[*Wert Wert ...*]	Gibt Datenwerte an, die fehlende Daten kennzeichnen.
range_missing	*boolesch*	Gibt an, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist.
missing_lower	*Zeichenfolge*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an.
missing_upper	*Zeichenfolge*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an.
null_missing	*boolesch*	Bei Festlegung auf T werden *Nullen* (undefinierte Werte, die in der Software als $null$ angezeigt werden) als fehlende Werte betrachtet.
whitespace_missing	*boolesch*	Bei Festlegung auf T werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet.
Beschreibung	*Zeichenfolge*	Gibt die Beschreibung für ein Feld an.
value_labels	*[{Wert Beschriftungszeichenfolge} {Wert Beschriftungszeichenfolge} ...]*	Gibt Beschriftungen für Wertpaare an.
display_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld bei der Anzeige fest (gilt nur für Felder mit dem Speichertyp REAL). Der Wert -1 verwendet den Streamstandard.
export_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld beim Exportieren fest (gilt nur für Felder mit dem Speichertyp REAL). Der Wert -1 verwendet den Streamstandard.

Tabelle 72. Eigenschaften des Knotens "type" (Forts.).

Eigenschaften des Knotens type	Datentyp	Eigenschaftsbeschreibung
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REAL).
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Legt das Datumsformat für das Feld fest (gilt nur für Felder mit dem Speichertyp DATE oder 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"	Legt das Zeitformat für das Feld fest (gilt nur für Felder mit dem Speichertyp TIME oder TIMESTAMP).
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Legt das Zahlenanzeigeformat für das Feld fest.
standard_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld für die Anzeige im Standardformat fest. Der Wert -1 verwendet den Streamstandard. Der vorhandene Slot display_places ändert dies zwar auch, wird aber nicht mehr verwendet.
scientific_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld für die Anzeige im wissenschaftlichen Format fest. Der Wert -1 verwendet den Streamstandard.

Tabelle 72. Eigenschaften des Knotens "type" (Forts.).

Eigenschaften des Knotens type	Datentyp	Eigenschaftsbeschreibung
currency_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld für die Anzeige im Währungsformat fest. Der Wert -1 verwendet den Streamstandard.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Legt das Symbol für die Zifferngruppierung für das Feld fest.
column_width	*ganze Zahl*	Legt die Spaltenbreite für das Feld fest. Mit dem Wert -1 wird die Spaltenbreite auf Auto (Automatisch) gesetzt.
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest.

Kapitel 12. Eigenschaften von Diagrammknoten

Allgemeine Eigenschaften von Diagrammknoten

In diesem Abschnitt werden die für Diagrammknoten verfügbaren Eigenschaften, einschließlich allgemeiner Eigenschaften sowie knotenspezifischer Eigenschaften, beschrieben.

Tabelle 73. Allgemeine Eigenschaften von Diagrammknoten.

Allgemeine Eigenschaften von Diagrammknoten	**Datentyp**	**Eigenschaftsbeschreibung**
`title`	*Zeichenfolge*	Gibt den Titel an. Beispiel: "Dies ist ein Titel."
`caption`	*Zeichenfolge*	Gibt die Titelzeile an. Beispiel: "Dies ist eine Titelzeile."
`output_mode`	`Screen` `File`	Gibt an, ob die Ausgabe des Diagrammknotens angezeigt oder in eine Datei geschrieben werden soll.

`output_format`	`BMP`
	`JPEG`
	`PNG`
	`HTML`
	`output` (*.cou*)
		Gibt den Ausgabetyp an. Der zulässige Ausgabetyp ist für jeden Knoten unterschiedlich.

`full_filename`	*Zeichenfolge*	Gibt den Zielpfad und den Dateinamen für die vom Diagrammknoten generierten Ausgabe an.
`use_graph_size`	*boolesch*	Gibt an, ob die Größe des Diagramms mithilfe der unten angegebenen Eigenschaften für Breite und Höhe explizit festgelegt wird. Dies betrifft nur Diagramme, die auf dem Bildschirm ausgegeben werden. Nicht für den Verteilungsknoten verfügbar.
`graph_width`	*Zahl*	Wenn `use_graph_size` den Wert `True` aufweist, wird hier die Diagrammhöhe in Pixeln festgelegt.
`graph_height`	*Zahl*	Wenn `use_graph_size` den Wert `True` aufweist, wird hier die Diagrammhöhe in Pixeln festgelegt.

Anmerkungen

Inaktivieren optionaler Felder. Optionale Felder, wie Überlagerungsfelder für Plots, können inaktiviert werden, indem der Eigenschaftswert auf `" "` (leere Zeichenfolge) gesetzt wird.

Angeben von Farben. Die Farben für Titel, Titelzeilen, Hintergründe und Beschriftungen können mit hexadezimalen Zeichenfolgen, die mit einem Hashzeichen (#) beginnen, festgelegt werden.

Die ersten beiden Stellen geben den roten Inhalt an, die mittleren Stellen legen den grünen Inhalt fest und die beiden letzten Stellen definieren den blauen Inhalt. Jede Stelle kann einen Wert im Bereich von 0-9 oder A-F annehmen. Zusammen können diese Werte eine Farbe des RGB-Farbraums (Rot, Grün, Blau) definieren.

Hinweis: Beim Angeben von Farben in Rot, Grün und Blau können Sie den richtigen Farbcode anhand des Field Choosers in der Benutzerschnittstelle festlegen. Bewegen Sie die Maus über die Farbe, um eine QuickInfo mit den gewünschten Informationen anzuzeigen.

Eigenschaften des Knotens "collection"

Der Sammlungsknoten zeigt die Verteilung der Werte für ein numerisches Feld im Verhältnis zu den Werten eines anderen an. (Er erstellt histogrammähnliche Diagramme.) Er eignet sich besonders für die Darstellung einer Variablen oder eines Felds, dessen Werte sich mit der Zeit verändern. Mithilfe eines 3-D-Diagramms können Sie außerdem eine symbolische Achse anlegen, auf der die Verteilungen nach Kategorie aufgetragen sind.

Tabelle 74. Eigenschaften des Knotens "collection".

Eigenschaften des Knotens `collection`	**Datentyp**	**Eigenschaftsbeschreibung**
`over_field`	*Feld*	
`over_label_auto`	*boolesch*	
`over_label`	*Zeichenfolge*	
`collect_field`	*Feld*	
`collect_label_auto`	*boolesch*	
`collect_label`	*Zeichenfolge*	
`three_D`	*boolesch*	
`by_field`	*Feld*	
`by_label_auto`	*boolesch*	
`by_label`	*Zeichenfolge*	

`operation`	`Sum`
	`Mean`
	`Min`
	`Max`
	`SDev`	

`color_field`	*Zeichenfolge*	
`panel_field`	*Zeichenfolge*	
`animation_field`	*Zeichenfolge*	
`range_mode`	`Automatic` `UserDefined`	
`range_min`	*Zahl*	
`range_max`	*Zahl*	
`bins`	`ByNumber` `ByWidth`	
`num_bins`	*Zahl*	
`bin_width`	*Zahl*	
`use_grid`	*boolesch*	
`graph_background`	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
`page_background`	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.

Eigenschaften des Knotens "distribution"

Der Verteilungsknoten zeigt das Auftreten symbolischer (kategorialer) Werte wie beispielsweise Hypothekenart oder Geschlecht. Verteilungsknoten eignen sich insbesondere zum Aufzeigen von Unausgewogenheiten in den Daten, die mithilfe eines Balancierungsknotens vor dem Erstellen eines Modells ausgeglichen werden können.

Tabelle 75. Eigenschaften des Knotens "distribution".

Eigenschaften des Knotens `distribution`	Datentyp	Eigenschaftsbeschreibung
`plot`	`SelectedFields` `Flags`	
`x_field`	*Feld*	
`color_field`	*Feld*	Überlagerungsfeld
`normalize`	*boolesch*	
`sort_mode`	`ByOccurence` `Alphabetic`	
`use_proportional_scale`	*boolesch*	

Eigenschaften des Knotens "evaluation"

Der Evaluierungsknoten erleichtert die Evaluation und den Vergleich von Vorhersagemodellen. Das Evaluierungsdiagramm zeigt, wie gut Modelle bestimmte Ergebnisse vorhersagen. Die Datensätze werden auf der Grundlage des vorhergesagten Werts und des Konfidenzwerts für die Vorhersage sortiert. Die Datensätze werden in gleich große Gruppen (**Quantile**) aufgeteilt. Anschließend wird der Wert des Geschäftskriteriums für jedes Quantil geplottet, vom höchsten Wert bis zum niedrigsten Wert. Mehrere Modelle werden als separate Linien im Plot dargestellt.

Tabelle 76. Eigenschaften des Knotens "evaluation".

Eigenschaften des Knotens `evaluation`	Datentyp	Eigenschaftsbeschreibung

`chart_type`	`Gains`
	`Response`
	`Lift`
	`Profit`
	`ROI`
	`ROC`	

`inc_baseline`	*boolesch*	
`field_detection_method`	`Metadata` `Name`	
`use_fixed_cost`	*boolesch*	
`cost_value`	*Zahl*	
`cost_field`	*Zeichenfolge*	
`use_fixed_revenue`	*boolesch*	
`revenue_value`	*Zahl*	
`revenue_field`	*Zeichenfolge*	
`use_fixed_weight`	*boolesch*	
`weight_value`	*Zahl*	

Tabelle 76. Eigenschaften des Knotens "evaluation" (Forts.).

Eigenschaften des Knotens `evaluation`	**Datentyp**	**Eigenschaftsbeschreibung**
`weight_field`	*Feld*	

`n_tile`	`Quartiles`
	`Quintles`
	`Deciles`
	`Vingtiles`
	`Percentiles`
	`1000-tiles`	

`cumulative`	*Flag*	
`style`	`Line` `Point`	

`point_type`	`Rectangle`
	`Dot`
	`Triangle`
	`Hexagon`
	`Plus`
	`Pentagon`
	`Star`
	`BowTie`
	`HorizontalDash`
	`VerticalDash`
	`IronCross`
	`Factory`
	`House`
	`Cathedral`
	`OnionDome`
	`ConcaveTriangle`
	`OblateGlobe`
	`CatEye`
	`FourSidedPillow`
	`RoundRectangle`
	`Fan`	

`export_data`	*boolesch*	
`data_filename`	*Zeichenfolge*	
`delimiter`	*Zeichenfolge*	
`new_line`	*boolesch*	
`inc_field_names`	*boolesch*	
`inc_best_line`	*boolesch*	
`inc_business_rule`	*boolesch*	
`business_rule_condition`	*Zeichenfolge*	
`plot_score_fields`	*boolesch*	
`score_fields`	*[Feld1 ... FeldN]*	
`target_field`	*Feld*	
`use_hit_condition`	*boolesch*	
`hit_condition`	*Zeichenfolge*	
`use_score_expression`	*boolesch*	
`score_expression`	*Zeichenfolge*	
`caption_auto`	*boolesch*	

Eigenschaften des Knotens "graphboard"

Der Diagrammtafelknoten bietet viele verschiedene Diagrammtypen in einem einzigen Knoten. Bei Verwendung dieses Knotens können Sie die Datenfelder auswählen, die Sie untersuchen möchten, und anschließend eines der für die ausgewählten Daten verfügbaren Diagramme auswählen. Der Knoten filtert automatisch alle Diagrammtypen heraus, die nicht für die Feldauswahl geeignet sind.

Hinweis: Wenn Sie eine Eigenschaft festlegen, die für den Diagrammtyp ungültig ist (z. B. ein `y_field` für ein Histogramm), wird diese Eigenschaft ignoriert.

Tabelle 77. Eigenschaften des Knotens "graphboard".

Eigenschaften des Knotens **graphboard**	Datentyp	Eigenschaftsbeschreibung
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	Identifiziert den Diagrammtyp.
x_field	*Feld*	Legt eine benutzerdefinierte Beschriftung für die *x*-Achse fest. Nur für Beschriftungen verfügbar.
y_field	*Feld*	Legt eine benutzerdefinierte Beschriftung für die *y*-Achse fest. Nur für Beschriftungen verfügbar.
z_field	*Feld*	In einigen 3-D-Diagrammen verwendet.
color_field	*Feld*	In Heat-Maps verwendet.

Tabelle 77. Eigenschaften des Knotens "graphboard" (Forts.).

Eigenschaften des Knotens graphboard	Datentyp	Eigenschaftsbeschreibung
size_field	*Feld*	In Blasendiagrammen verwendet.
categories_field	*Feld*	
values_field	*Feld*	
rows_field	*Feld*	
columns_field	*Feld*	
fields	*Feld*	
start_longitude_field	*Feld*	Bei Pfeilen auf einer Referenzkarte verwendet.
end_longitude_field	*Feld*	
start_latitude_field	*Feld*	
end_latitude_field	*Feld*	
data_key_field	*Feld*	In verschiedenen Karten verwendet.
panelrow_field	*Zeichenfolge*	
panelcol_field	*Zeichenfolge*	
animation_field	*Zeichenfolge*	
longitude_field	*Feld*	Bei Koordinaten in Karten verwendet.
latitude_field	*Feld*	
map_color_field	*Feld*	

Eigenschaften des Knotens "histogram"

Der Histogrammknoten zeigt das Auftreten bestimmter Werte in numerischen Feldern. Damit werden häufig die Daten vor der weiteren Bearbeitung und der Modellerstellung untersucht. Ähnlich wie der Verteilungsknoten kann der Histogrammknoten oft Unausgewogenheiten in den Daten aufdecken.

Tabelle 78. Eigenschaften des Knotens "histogram".

Eigenschaften des Knotens histogram	Datentyp	Eigenschaftsbeschreibung
field	*Feld*	
color_field	*Feld*	
panel_field	*Feld*	
animation_field	*Feld*	
range_mode	Automatic UserDefined	
range_min	*Zahl*	
range_max	*Zahl*	
bins	ByNumber ByWidth	
num_bins	*Zahl*	
bin_width	*Zahl*	
normalize	*boolesch*	

Tabelle 78. Eigenschaften des Knotens "histogram" (Forts.).

Eigenschaften des Knotens **histogram**	Datentyp	Eigenschaftsbeschreibung
separate_bands	*boolesch*	
x_label_auto	*boolesch*	
x_label	*Zeichenfolge*	
y_label_auto	*boolesch*	
y_label	*Zeichenfolge*	
use_grid	*boolesch*	
graph_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
normal_curve	*boolesch*	Gibt an, ob die Normalverteilungskurve in der Ausgabe angezeigt werden soll.

Eigenschaften des Knotens "multiplot"

Ein Multiplot erstellt ein Plot, bei dem mehrere *Y*-Felder über einem einzelnen *X*-Feld dargestellt werden. Die *Y*-Felder werden als farbige Linien geplottet, die jeweils einem Plotknoten mit dem Stil **Linie** und dem X-Modus **Sortieren** entsprechen. Multiplots sind hilfreich, wenn die Fluktuation mehrerer Variablen im Laufe der Zeit untersucht werden soll.

Tabelle 79. Eigenschaften des Knotens "multiplot".

Eigenschaften des Knotens **multiplot**	Datentyp	Eigenschaftsbeschreibung
x_field	*Feld*	
y_fields	[*Feld Feld Feld*]	
panel_field	*Feld*	
animation_field	*Feld*	
normalize	*boolesch*	
use_overlay_expr	*boolesch*	
overlay_expression	*Zeichenfolge*	
records_limit	*Zahl*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*boolesch*	
x_label	*Zeichenfolge*	
y_label_auto	*boolesch*	
y_label	*Zeichenfolge*	
use_grid	*boolesch*	
graph_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.

Eigenschaften des Knotens "plot"

Der Plotknoten zeigt die Beziehung zwischen numerischen Feldern an. Sie können einen Plot mithilfe von Punkten (Streudiagramm) oder mit Linien erstellen.

Tabelle 80. Eigenschaften des Knotens "plot".

Eigenschaften des Knotens `plot`	Datentyp	Eigenschaftsbeschreibung
`x_field`	*Feld*	Legt eine benutzerdefinierte Beschriftung für die *x*-Achse fest. Nur für Beschriftungen verfügbar.
`y_field`	*Feld*	Legt eine benutzerdefinierte Beschriftung für die *y*-Achse fest. Nur für Beschriftungen verfügbar.
`three_D`	*boolesch*	Legt eine benutzerdefinierte Beschriftung für die *y*-Achse fest. Nur für Beschriftungen in 3-D-Diagrammen verfügbar.
`z_field`	*Feld*	
`color_field`	*Feld*	Überlagerungsfeld
`size_field`	*Feld*	
`shape_field`	*Feld*	
`panel_field`	*Feld*	Gibt ein nominales oder Flagfeld an, mit dem je ein separates Diagramm für jede Kategorie angelegt werden soll. Die Diagramme werden gemeinsam in einem Ausgabefenster dargestellt.
`animation_field`	*Feld*	Gibt ein nominales oder Flagfeld an, mit dem die Kategorien für die Datenwerte in Form einer Reihe von Diagrammen dargestellt werden, die nacheinander als Animation angezeigt werden.
`transp_field`	*Feld*	Gibt ein Feld an, mit dem die Kategorien für die Datenwerte in Form von verschiedenen Transparenzebenen für die einzelnen Kategorien dargestellt werden. Für Liniendiagramme nicht verfügbar.

`overlay_type`	`None`
	`Smoother`
	`Function`
		Gibt an, ob eine Überlagerungsfunktion oder ein LOESS-Smoother angezeigt werden soll.

`overlay_expression`	*Zeichenfolge*	Gibt den Ausdruck an, der verwendet werden soll, wenn `overlay_type` auf `Function` gesetzt ist.
`style`	`Point` `Line`	

Tabelle 80. Eigenschaften des Knotens "plot" (Forts.).

Eigenschaften des Knotens **plot**	Datentyp	Eigenschaftsbeschreibung
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*Zahl*	
x_range_max	*Zahl*	
y_range_mode	Automatic UserDefined	
y_range_min	*Zahl*	
y_range_max	*Zahl*	
z_range_mode	Automatic UserDefined	
z_range_min	*Zahl*	
z_range_max	*Zahl*	
Streuen	*boolesch*	
records_limit	*Zahl*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*boolesch*	
x_label	*Zeichenfolge*	
y_label_auto	*boolesch*	
y_label	*Zeichenfolge*	
z_label_auto	*boolesch*	
z_label	*Zeichenfolge*	
use_grid	*boolesch*	
graph_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.

Tabelle 80. Eigenschaften des Knotens "plot" (Forts.).

Eigenschaften des Knotens plot	Datentyp	Eigenschaftsbeschreibung
page_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
use_overlay_expr	*boolesch*	Wird zugunsten von overlay_type nicht mehr verwendet.

Eigenschaften des Knotens "timeplot"

Der Zeitdiagrammknoten zeigt ein oder mehrere Sets mit Zeitreihendaten an. Normalerweise wird zuerst mithilfe eines Zeitintervallknotens ein *TimeLabel*-Feld erstellt, das dann zur Beschriftung der *x*-Achse verwendet wird.

Tabelle 81. Eigenschaften des Knotens "timeplot".

Eigenschaften des Knotens timeplot	Datentyp	Eigenschaftsbeschreibung
plot_series	Series Models	
use_custom_x_field	*boolesch*	
x_field	*Feld*	
y_fields	[*Feld Feld Feld*]	
panel	*boolesch*	
normalize	*boolesch*	
line	*boolesch*	
points	*boolesch*	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	*boolesch*	Sie können nur dann Glättungselemente zum Plot hinzufügen, wenn Sie panel auf True setzen.
use_records_limit	*boolesch*	
records_limit	*ganze Zahl*	
symbol_size	*Zahl*	Gibt eine Symbolgröße an.

Tabelle 81. Eigenschaften des Knotens "timeplot" (Forts.).

Eigenschaften des Knotens **timeplot**	Datentyp	Eigenschaftsbeschreibung
panel_layout	Horizontal Vertical	

Eigenschaften des Knotens "web"

Der Netzdiagrammknoten zeigt die Stärke der Beziehung zwischen den Werten aus mindestens zwei symbolischen (kategorialen) Feldern. Im Diagramm wird die Verbindungsstärke durch unterschiedlich breite Linien angezeigt. Mit Netzdiagrammknoten können Sie beispielsweise die Beziehung zwischen dem Kauf einer Gruppe von Artikeln auf einer e-Commerce-Website untersuchen.

Tabelle 82. Eigenschaften des Knotens "web".

Eigenschaften des Knotens **web**	Datentyp	Eigenschaftsbeschreibung
use_directed_web	*boolesch*	
fields	[*Feld Feld Feld*]	
to_field	*Feld*	
from_fields	[*Feld Feld Feld*]	
true_flags_only	*boolesch*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*boolesch*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*Zahl*	
links_above	*Zahl*	
discard_links_min	*boolesch*	
links_min_records	*Zahl*	
discard_links_max	*boolesch*	
links_max_records	*Zahl*	
weak_below	*Zahl*	
strong_above	*Zahl*	
link_size_continuous	*boolesch*	
web_display	Circular Network Directed Grid	
graph_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
symbol_size	*Zahl*	Gibt eine Symbolgröße an.

Kapitel 13. Eigenschaften von Modellierungsknoten

Allgemeine Eigenschaften von Modellierungsknoten

Folgende Eigenschaften haben einige oder alle Modellierungsknoten gemeinsam. Etwaige Ausnahmen sind in der Dokumentation für die einzelnen Modellierungsknoten angegeben.

Tabelle 83. Allgemeine Eigenschaften von Modellierungsknoten.

Eigenschaft	Werte	Eigenschaftsbeschreibung
`custom_fields`	*boolesch*	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.

`target`
ODER
`targets`
	Feld
	
	ODER
	[*Feld1 ... FeldN*]
		Gibt je nach Modelltyp ein einzelnes Zielfeld oder mehrere Zielfelder an.

`inputs`	[*Feld1 ... FeldN*]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
`partition`	*Feld*	
`use_partitioned_data`	*boolesch*	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
`use_split_data`	*boolesch*	
`splits`	*[Feld1 ... FeldN]*	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an. Nur wirksam, wenn `use_split_data` auf `True` gesetzt ist.
`use_frequency`	*boolesch*	Gewichtungs- und Häufigkeitsfelder werden von bestimmten Modellen je nach Angabe für die einzelnen Modelltypen verwendet.
`frequency_field`	*Feld*	
`use_weight`	*boolesch*	
`weight_field`	*Feld*	
`use_model_name`	*boolesch*	
`model_name`	*Zeichenfolge*	Benutzerdefinierter Name für neues Modell.
`mode`	`Simple` `Expert`	

Eigenschaften des Knotens "anomalydetection"

Der Anomalieerkennungsknoten ermittelt ungewöhnliche Fälle bzw. "Ausreißer", die nicht den Mustern der "normalen" Daten entsprechen. Mit diesem Knoten können Ausreißer ermittelt werden, selbst wenn sie keinem bereits bekannten Muster entsprechen und selbst wenn Sie nicht genau wissen, wonach Sie suchen.

Tabelle 84. Eigenschaften des Knotens "anomalydetection".

Eigenschaften des Knotens `anomalydetection`	**Werte**	**Eigenschaftsbeschreibung**
`inputs`	*[Feld1 ... FeldN]*	Anomalieerkennungsmodelle führen ein Screening von Datensätzen auf der Grundlage der angegebenen Eingabefelder durch. Sie verwenden kein Zielfeld. Gewichtungs- und Häufigkeitsfelder werden ebenfalls nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
`mode`	`Expert` `Simple`	

`anomaly_method`	`IndexLevel`
	`PerRecords`
	`NumRecords`
		Gibt die Methode für die Bestimmung des Trennwerts zur Kennzeichnung von Datensätzen als anomal an.

`index_level`	*Zahl*	Gibt den minimalen Trennwert für die Kennzeichnung von Anomalien an.
`percent_records`	*Zahl*	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage des Prozentsatzes der Datensätze in den Trainingsdaten fest.
`num_records`	*Zahl*	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage der Anzahl der Datensätze in den Trainingsdaten fest.
`num_fields`	*ganze Zahl*	Die Anzahl der für die einzelnen anomalen Datensätze zu meldenden Felder.
`impute_missing_values`	*boolesch*	
`adjustment_coeff`	*Zahl*	Wert, der zum Balancieren der relativen Gewichtung verwendet wird, das den stetigen und den kategorialen Feldern bei der Berechnung der Distanz zugewiesen wird.
`peer_group_num_auto`	*boolesch*	Berechnet automatisch die Anzahl der Peergruppen.
`min_num_peer_groups`	*ganze Zahl*	Gibt an, wie viele Peergruppen mindestens verwendet werden, wenn `peer_group_num_auto` auf `True` gesetzt ist.
`max_num_per_groups`	*ganze Zahl*	Gibt die maximale Anzahl an Peergruppen an.
`num_peer_groups`	*ganze Zahl*	Gibt an, wie viele Peergruppen verwendet werden, wenn `peer_group_num_auto` auf `False` gesetzt ist.
`noise_level`	*Zahl*	Bestimmt, wie Ausreißer bei der Clusterbildung behandelt werden. Geben Sie einen Wert zwischen 0 und 0,5 an.
`noise_ratio`	*Zahl*	Gibt an, welcher Anteil des der Komponente zugeordneten Arbeitsspeichers für die Rausch-Pufferung verwendet werden soll. Geben Sie einen Wert zwischen 0 und 0,5 an.

Eigenschaften des Knotens "apriori"

Der Apriori-Knoten extrahiert ein Regelset aus den Daten und daraus die Regeln mit dem höchsten Informationsgehalt. Apriori bietet fünf verschiedene Methoden zur Auswahl von Regeln und verwendet ein ausgereiftes Indizierungsschema zur effizienten Verarbeitung großer Datasets. Bei großen Problemen ist Apriori in der Regel schneller zu trainieren, es gibt keine willkürliche Begrenzung für die Anzahl der Regeln, die beibehalten werden können, und es können Regeln mit bis zu 32 Vorbedingungen verarbeitet werden. Bei Apriori müssen alle Ein- und Ausgabefelder kategorial sein; dafür bietet es jedoch eine bessere Leistung, da es für diesen Datentyp optimiert ist.

Tabelle 85. Eigenschaften des Knotens "apriori".

Eigenschaften des Knotens apriori	**Werte**	**Eigenschaftsbeschreibung**
consequents	*Feld*	Apriori-Modelle verwenden anstelle von standardmäßigen Ziel- und Eingabefeldern Sukzedenzien bzw. Antezedenzien. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
antecedents	[*Feld1 ... FeldN*]	
min_supp	*Zahl*	
min_conf	*Zahl*	
max_antecedents	*Zahl*	
true_flags	*boolesch*	
optimize	Speed Memory	
use_transactional_data	*boolesch*	
contiguous	*boolesch*	
id_field	*Zeichenfolge*	
content_field	*Zeichenfolge*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*Zahl*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften des Knotens "autoclassifier"

Mit dem Knoten "Autom. Klassifikationsmerkmal" können Sie eine Reihe verschiedener Modelle für binäre Ergebnisse ("Ja" oder "Nein", "Abwanderung" oder "Keine Abwanderung" usw.) erstellen und vergleichen, um den besten Ansatz für die jeweilige Analyse auszuwählen. Es wird eine Reihe von Modellierungsalgorithmen unterstützt, sodass Sie die gewünschten Methoden, die spezifischen Optionen für die jeweilige Methode und die Kriterien zum Vergleich der Ergebnisse auswählen können. Der Knoten generiert eine Gruppe von Modellen, die auf den angegebenen Optionen beruhen, und erstellt anhand der von Ihnen angegebenen Kriterien eine Rangordnung der besten Kandidaten.

Tabelle 86. Eigenschaften des Knotens "autoclassifier".

Eigenschaften des Knotens `autoclassifier`	**Werte**	**Eigenschaftsbeschreibung**
`target`	*Feld*	Für Flagziele verlangt der Knoten "Automatisches Klassifikationsmerkmal" ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem können Gewichtungs- und Häufigkeitsfelder angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.

`ranking_measure`	`Accuracy`
	`Area_under_curve`
	`Profit`
	`Lift`
	`Num_variables`	

`ranking_dataset`	`Training` `Test`	
`number_of_models`	*ganze Zahl*	Anzahl der Modelle, die in das Modellnugget aufgenommen werden sollen. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
`calculate_variable_importance`	*boolesch*	
`enable_accuracy_limit`	*boolesch*	
`accuracy_limit`	*ganze Zahl*	Ganze Zahl zwischen 0 und 100.
`enable_ area_under_curve _limit`	*boolesch*	
`area_under_curve_limit`	*Zahl*	Reelle Zahl zwischen 0,0 und 1,0.
`enable_profit_limit`	*boolesch*	
`profit_limit`	*Zahl*	Ganze Zahl größer als 0.
`enable_lift_limit`	*boolesch*	
`lift_limit`	*Zahl*	Reelle Zahl größer als 1,0.
`enable_number_of_variables_limit`	*boolesch*	
`number_of_variables_limit`	*Zahl*	Ganze Zahl größer als 0.
`use_fixed_cost`	*boolesch*	
`fixed_cost`	*Zahl*	Reelle Zahl größer 0.0.
`variable_cost`	*Feld*	
`use_fixed_revenue`	*boolesch*	
`fixed_revenue`	*Zahl*	Reelle Zahl größer 0.0.

Tabelle 86. Eigenschaften des Knotens "autoclassifier" (Forts.).

Eigenschaften des Knotens autoclassifier	**Werte**	**Eigenschaftsbeschreibung**
variable_revenue	*Feld*	
use_fixed_weight	*boolesch*	
fixed_weight	*Zahl*	Reelle Zahl größer als 0,0
variable_weight	*Feld*	
lift_percentile	*Zahl*	Ganze Zahl zwischen 0 und 100.
enable_model_build_time_limit	*boolesch*	
model_build_time_limit	*Zahl*	Ganze Zahl für die Anzahl der Minuten, die maximal für die Erstellung jedes einzelnen Modells aufgewendet werden.
enable_stop_after_time_limit	*boolesch*	
stop_after_time_limit	*Zahl*	Reelle Zahl für die Anzahl der Stunden, die als Obergrenze für die insgesamt verstrichene Zeit für den Durchlauf eines automatischen Klassifikationsmerkmals verwendet wird.
enable_stop_after_valid_model_ produced	*boolesch*	
use_costs	*boolesch*	
<Algorithmus>	*boolesch*	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
<Algorithmus>.<Eigenschaft>	*Zeichenfolge*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmuseigenschaften”.

Festlegen der Algorithmuseigenschaften

Die Algorithmusnamen für den Knoten "Automatisches Klassifikationsmerkmal" lauten `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` und `knn`.

Die Algorithmusnamen für den Knoten "Auto-Numerisch" lauten `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` und `knn`.

Algorithmusnamen für den Knoten "Autom. Cluster" sind `twostep`, `k-means` und `kohonen`.

Für die Eigenschaftsnamen wird der jeweils für den Algorithmusknoten dokumentierte Standard verwendet.

Algorithmuseigenschaften, die Punkte oder andere Satzzeichen enthalten, müssen in einzelne Anführungsstriche eingebettet sein.

Als Eigenschaft können auch mehrere Werte zugewiesen werden.

Anmerkungen:

- Bei der Angabe der Werte `true` und `false` müssen Kleinbuchstaben verwendet werden (also nicht `False`).

- In Fällen, in denen bestimmte Algorithmusoptionen nicht im Knoten "Automatisches Klassifikationsmerkmal" verfügbar sind oder in denen nur ein einzelner Wert und kein Wertebereich angegeben werden kann, gelten dieselben Einschränkungen bei der Scripterstellung wie beim standardmäßigen Zugriff auf den Knoten.

Eigenschaften des Knotens "autocluster"

Mit dem Knoten "Autom. Cluster" können Sie Clustering-Modelle, die Gruppen und Datensätze mit ähnlichen Merkmalen identifizieren, schätzen und vergleichen. Die Funktionsweise des Knotens gleicht der von anderen Knoten für automatisierte Modellierung, und Sie können in einem einzigen Modellierungsdurchgang mit mehreren Optionskombinationen experimentieren. Modelle können mithilfe grundlegender Messwerte für Filterung und Rangfolge der Nützlichkeit von Clustermodellen verglichen werden, um ein Maß auf der Basis der Wichtigkeit von bestimmten Feldern zu liefern.

Tabelle 87. Eigenschaften des Knotens "autocluster".

Eigenschaften des Knotens autocluster	**Werte**	**Eigenschaftsbeschreibung**
`evaluation`	*Feld*	*Hinweis*: Nur Knoten "Autom. Cluster". Kennzeichnet das Feld, für das ein Wichtigkeitswert berechnet wird. Kann auch verwendet werden, um festzustellen, wie gut das Cluster den Wert dieses Felds differenziert, also wie gut das Modell den Wert für dieses Feld vorhersagen kann.

`ranking_measure`	`Silhouette`
	`Num_clusters`
	`Size_smallest_cluster`
	`Size_largest_cluster`
	`Smallest_to_largest`
	`Importance`	

`ranking_dataset`	`Training` `Test`	
`summary_limit`	*ganze Zahl*	Anzahl der im Bericht aufzuführenden Modelle. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
`enable_silhouette_limit`	*boolesch*	
`silhouette_limit`	*ganze Zahl*	Ganze Zahl zwischen 0 und 100.
`enable_number_less_limit`	*boolesch*	
`number_less_limit`	*Zahl*	Reelle Zahl zwischen 0,0 und 1,0.
`enable_number_greater_limit`	*boolesch*	
`number_greater_limit`	*Zahl*	Ganze Zahl größer als 0.
`enable_smallest_cluster_limit`	*boolesch*	
`smallest_cluster_units`	`Percentage` `Counts`	
`smallest_cluster_limit_percentage`	*Zahl*	
`smallest_cluster_limit_count`	*ganze Zahl*	Ganze Zahl größer als 0.
`enable_largest_cluster_limit`	*boolesch*	
`largest_cluster_units`	`Percentage` `Counts`	
`largest_cluster_limit_percentage`	*Zahl*	

Tabelle 87. Eigenschaften des Knotens "autocluster" (Forts.).

Eigenschaften des Knotens **autocluster**	Werte	Eigenschaftsbeschreibung
largest_cluster_limit_count	*ganze Zahl*	
enable_smallest_largest_limit	*boolesch*	
smallest_largest_limit	*Zahl*	
enable_importance_limit	*boolesch*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*Zahl*	Ganze Zahl zwischen 0 und 100.
importance_limit_less_than	*Zahl*	Ganze Zahl zwischen 0 und 100.
<Algorithmus>	*boolesch*	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
<Algorithmus>.<Eigenschaft>	*Zeichenfolge*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmuseigenschaften" auf Seite 137.

Eigenschaften des Knotens "autonumeric"

Der Knoten "Auto-Numerisch" schätzt und vergleicht mit einer Reihe verschiedener Methoden Modelle für die Ergebnisse stetiger numerischer Bereiche. Der Knoten arbeitet auf dieselbe Weise wie der Knoten "Automatisches Klassifikationsmerkmal": Sie können die zu verwendenden Algorithmen auswählen und in einem Modellierungsdurchlauf mit mehreren Optionskombinationen experimentieren. Folgende Algorithmen werden unterstützt: C&RT-Baum, CHAID, lineare Regression, verallgemeinerte lineare Regression und Support Vector Machines (SVM). Modelle können anhand von Korrelation, relativem Fehler bzw. Anzahl der verwendeten Variablen verglichen werden.

Tabelle 88. Eigenschaften des Knotens "autonumeric".

Eigenschaften des Knotens **autonumeric**	Werte	Eigenschaftsbeschreibung
custom_fields	*boolesch*	Bei "True" (Wahr) werden anstelle der Typknoteneinstellungen Einstellungen aus benutzerdefinierten Feldern verwendet.
target	*Feld*	Für den Knoten "Auto-Numerisch" sind ein einzelnes Ziel und eines oder mehrere Eingabefelder erforderlich. Außerdem können Gewichtungs- und Häufigkeitsfelder angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
inputs	*[Feld1 ... Feld2]*	
partition	*Feld*	
use_frequency	*boolesch*	
frequency_field	*Feld*	
use_weight	*boolesch*	

Tabelle 88. Eigenschaften des Knotens "autonumeric" (Forts.).

Eigenschaften des Knotens `autonumeric`	**Werte**	**Eigenschaftsbeschreibung**
`weight_field`	*Feld*	
`use_partitioned_data`	*boolesch*	Wenn ein Partitionsfeld definiert ist, werden nur die Trainingsdaten für die Modellerstellung verwendet.
`ranking_measure`	`Correlation` `NumberOfFields`	
`ranking_dataset`	`Test` `Training`	
`number_of_models`	*ganze Zahl*	Anzahl der Modelle, die in das Modellnugget aufgenommen werden sollen. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
`calculate_variable_importance`	*boolesch*	
`enable_correlation_limit`	*boolesch*	
`correlation_limit`	*ganze Zahl*	
`enable_number_of_fields_limit`	*boolesch*	
`number_of_fields_limit`	*ganze Zahl*	
`enable_relative_error_limit`	*boolesch*	
`relative_error_limit`	*ganze Zahl*	
`enable_model_build_time_limit`	*boolesch*	
`model_build_time_limit`	*ganze Zahl*	
`enable_stop_after_time_limit`	*boolesch*	
`stop_after_time_limit`	*ganze Zahl*	
`stop_if_valid_model`	*boolesch*	
`<Algorithmus>`	*boolesch*	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
`<Algorithmus>.<Eigenschaft>`	*Zeichenfolge*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmuseigenschaften” auf Seite 137.

Eigenschaften des Knotens "bayesnet"

Mithilfe des Bayes-Netzknotens können Sie ein Wahrscheinlichkeitsmodell erstellen, indem Sie beobachtete und aufgezeichnete Hinweise mit Weltwissen kombinieren, um die Wahrscheinlichkeit ihres Vorkommens zu ermitteln. Der Knoten ist speziell für Netze vom Typ "Tree Augmented Naïve Bayes" (TAN) und "Markov-Decke" gedacht, die in erster Linie zur Klassifizierung verwendet werden.

Tabelle 89. Eigenschaften des Knotens "bayesnet".

Eigenschaften des Knotens bayesnet	Werte	Eigenschaftsbeschreibung
inputs	*[Feld1 ... FeldN]*	Bayes-Netzmodelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Stetige Felder werden automatisch klassiert. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
continue_training_existing_model	*boolesch*	
structure_type	TAN MarkovBlanket	Dient zur Auswahl der beim Erstellen des Bayes-Netzes zu verwendenden Struktur.
use_feature_selection	*boolesch*	
parameter_learning_method	Likelihood Bayes	Gibt die Methode an, die zur Schätzung der Tabellen zur bedingten Wahrscheinlichkeit zwischen Knoten verwendet wird, wenn die Werte der übergeordneten Elemente bekannt sind.
mode	Expert Simple	
missing_values	*boolesch*	
all_probabilities	*boolesch*	
independence	Likelihood Pearson	Gibt die Methode an, die zur Einschätzung verwendet wird, ob paarige Beobachtungen bei zwei Variablen voneinander unabhängig sind.
significance_level	*Zahl*	Gibt den Trennwert für die Bestimmung der Unabhängigkeit an.
maximal_conditioning_set	*Zahl*	Legt die Maximalzahl der für die Unabhängigkeitstests zu verwendenden Konditionierungsvariablen fest.
inputs_always_selected	*[Feld1 ... FeldN]*	Gibt an, welche Felder aus dem Dataset immer beim Erstellen des Bayes-Netzes verwendet werden. *Hinweis*: Das Zielfeld ist immer ausgewählt.
maximum_number_inputs	*Zahl*	Gibt die maximale Anzahl an Eingabefeldern an, die beim Erstellen des Bayes-Netzes verwendet werden sollen.
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "buildr"

Der R-Erstellungsknoten ermöglicht es Ihnen, ein benutzerdefiniertes R-Script einzugeben, um die Modellerstellung und das Modellscoring, die in IBM SPSS Modeler implementiert sind, auszuführen.

Tabelle 90. Eigenschaften von "buildr".

Eigenschaften des Knotens `buildr`	Werte	Eigenschaftsbeschreibung
`build_syntax`	*Zeichenfolge*	R-Scriptsyntax für die Modellerstellung.
`score_syntax`	*Zeichenfolge*	R-Scriptsyntax für das Modellscoring.
`convert_flags`	`StringsAndDoubles` `LogicalValues`	Option zum Konvertieren von Flagfeldern.
`convert_datetime`	*boolesch*	Option zum Konvertieren von Variablen mit Datums- oder Datums-/Zeitformaten in R-Datums-/Zeitformate.
`convert_datetime_class`	`POSIXct` `POSIXlt`	Optionen, die angeben, in welches Format Variablen mit Datums- oder Datums-/Zeitformaten konvertiert werden.
`convert_missing`	*boolesch*	Option zum Konvertieren fehlender Werte in R-Werte `"NA"`.
`output_html`	*boolesch*	Option für die Anzeige von Diagrammrn im R-Modellnugget.
`output_text`	*boolesch*	Option zum Schreiben von R-Konsolentext auf eine Registerkarte des R-Modellnuggets.

Eigenschaften des Knotens "c50"

Der C5.0-Knoten erstellt entweder einen Entscheidungsbaum oder ein Regelset. Das Modell teilt die Stichprobe auf der Basis des Felds auf, das auf der jeweiligen Ebene den maximalen Informationsgewinn liefert. Das Zielfeld muss kategorial sein. Es sind mehrere Aufteilungen in mehr als zwei Untergruppen zulässig.

Tabelle 91. Eigenschaften des Knotens "c50".

Eigenschaften des Knotens `c50`	Werte	Eigenschaftsbeschreibung
`target`	*Feld*	C50-Modelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
`output_type`	`DecisionTree` `RuleSet`	
`group_symbolics`	*boolesch*	
`use_boost`	*boolesch*	
`boost_num_trials`	*Zahl*	
`use_xval`	*boolesch*	

Tabelle 91. Eigenschaften des Knotens "c50" (Forts.).

Eigenschaften des Knotens c50	Werte	Eigenschaftsbeschreibung
xval_num_folds	*Zahl*	
mode	Simple Expert	
favor	Accuracy Generality	Genauigkeit oder Allgemeingültigkeit werden vorselektiert.
expected_noise	*Zahl*	
min_child_records	*Zahl*	
pruning_severity	*Zahl*	
use_costs	*boolesch*	
costs	*strukturiert*	Hierbei handelt es sich um eine strukturierte Eigenschaft.
use_winnowing	*boolesch*	
use_global_pruning	*boolesch*	Standardmäßig auf True gesetzt.
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "carma"

Beim CARMA-Modell wird ein Regelset aus den Daten extrahiert, ohne dass Sie Eingabe- oder Zielfelder angeben müssen. Im Gegensatz zu Apriori bietet der CARMA-Knoten Erstellungseinstellungen für die Regelunterstützung (Unterstützung für Antezedens und Sukzedens) und nicht nur für die Antezedens-Unterstützung. Die erstellten Rgeln können somit für eine größere Palette an Anwendungen verwendet werden, beispielsweise um eine Liste mit Produkten und Dienstleistungen (Antezedenzien) zu finden, deren Nachfolger (Sukzedens) das Element darstellt, das Sie in der Ferienzeit desselben Jahres bewerben möchten.

Tabelle 92. Eigenschaften des Knotens "carma".

Eigenschaften des Knotens carma	Werte	Eigenschaftsbeschreibung
inputs	*[Feld1 ... Feldn]*	CARMA-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
id_field	*Feld*	Das Feld wird als ID-Feld für die Modellerstellung verwendet.
contiguous	*boolesch*	Dient zur Angabe, ob IDs im ID-Feld zusammenhängend sind.
use_transactional_data	*boolesch*	
content_field	*Feld*	

Tabelle 92. Eigenschaften des Knotens "carma" (Forts.).

Eigenschaften des Knotens carma	Werte	Eigenschaftsbeschreibung
min_supp	*Zahl(Prozent)*	Bezieht sich auf die Regelunterstützung und nicht auf die Antezedens-Unterstützung. Der Standardwert ist 20 %.
min_conf	*Zahl(Prozent)*	Der Standardwert ist 20 %.
max_size	*Zahl*	Der Standardwert ist 10.
mode	Simple Expert	Der Standardwert ist Simple.
exclude_multiple	*boolesch*	Schließt Regeln mit mehreren Sukzedenzien aus. Standardmäßig ist dieser Wert False.
use_pruning	*boolesch*	Standardmäßig ist dieser Wert False.
pruning_value	*Zahl*	Der Standardwert ist 500.
vary_support	*boolesch*	
estimated_transactions	*ganze Zahl*	
rules_without_antecedents	*boolesch*	

Eigenschaften des Knotens "cart"

Der Knoten für Klassifizierungs- und Regressions-Bäume (C&RT-Bäume) erstellt einen Entscheidungsbaum, mit dem Sie zukünftige Beobachtungen vorhersagen oder klassifizieren können. Bei dieser Methode wird eine rekursive Partitionierung verwendet, um die Trainingsdatensätze in Segmente aufzuteilen. Dabei wird bei jedem Schritt die Unreinheit verringert und ein Knoten im Baum wird als "rein" betrachtet, wenn 100 % der Fälle in eine bestimmte Kategorie des Zielfelds fallen. Ziel- und Eingabefelder können numerische Bereiche oder kategorial (nominal, ordinal oder Flags) sein. Alle Aufteilungen sind binär (nur zwei Untergruppen).

Tabelle 93. Eigenschaften des Knotens "cart".

Eigenschaften des Knotens cart	Werte	Eigenschaftsbeschreibung
target	*Feld*	Modelle vom Typ "C&R-Baum" verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
continue_training_existing_model	*boolesch*	
objective	Standard Boosting Bagging psm	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*boolesch*	

Tabelle 93. Eigenschaften des Knotens "cart" (Forts.).

Eigenschaften des Knotens cart	Werte	Eigenschaftsbeschreibung
tree_directives	*Zeichenfolge*	Geben Sie Aufbauregeln für die Erweiterung des Baums an. Aufbauregeln können in dreifache Anführungszeichen gesetzt werden, um auf Escapezeichen für neue Zeilen oder Anführungszeichen verzichten zu können. Beachten Sie, dass die Anweisungen auf kleinste Änderungen in den Daten- oder Modellierungsoptionen reagieren und nicht für andere Datasets verallgemeinert werden können.
use_max_depth	Default Custom	
max_depth	*ganze Zahl*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	*boolesch*	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	*boolesch*	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	*Zahl*	Maximale Differenz.
max_surrogates	*Zahl*	Maximale Anzahl Ersatztrenner.
use_percentage	*boolesch*	
min_parent_records_pc	*Zahl*	
min_child_records_pc	*Zahl*	
min_parent_records_abs	*Zahl*	
min_child_records_abs	*Zahl*	
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft.
adjust_priors	*boolesch*	
trails	*Zahl*	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	*boolesch*	Boosting auf sehr große Datasets anwenden.
min_impurity	*Zahl*	
impurity_measure	Gini Twoing Ordered	

Tabelle 93. Eigenschaften des Knotens "cart" (Forts.).

Eigenschaften des Knotens cart	Werte	Eigenschaftsbeschreibung
train_pct	*Zahl*	Set zur Verhinderung übermäßiger Anpassung.
set_random_seed	*boolesch*	Option "Ergebnisse replizieren".
seed	*Zahl*	
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "chaid"

Der CHAID-Knoten erzeugt Entscheidungsbäume unter Verwendung von Chi-Quadrat-Statistiken zur Ermittlung optimaler Aufteilungen. Im Gegensatz zu den Knoten vom Typ "C&RT-Baum" und "QUEST" kann CHAID nicht binäre Bäume generieren, d. h. Bäume mit Aufteilungen mit mehr als zwei Verzweigungen. Ziel- und Eingabefelder können in einem numerischen Bereich (stetig) oder kategorial sein. Exhaustive CHAID ist eine Änderung von CHAID, die noch gründlicher vorgeht, indem sie alle möglichen Aufteilungen untersucht, allerdings mehr Rechenzeit beansprucht.

Tabelle 94. Eigenschaften des Knotens "chaid".

Eigenschaften des Knotens chaid	Werte	Eigenschaftsbeschreibung
target	*Feld*	CHAID-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
continue_training_existing_model	*boolesch*	
objective	Standard Boosting Bagging psm	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*boolesch*	
tree_directives	*Zeichenfolge*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*ganze Zahl*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
use_percentage	*boolesch*	
min_parent_records_pc	*Zahl*	

Tabelle 94. Eigenschaften des Knotens "chaid" (Forts.).

Eigenschaften des Knotens chaid	Werte	Eigenschaftsbeschreibung
min_child_records_pc	*Zahl*	
min_parent_records_abs	*Zahl*	
min_child_records_abs	*Zahl*	
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.
trails	*Zahl*	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	*boolesch*	Boosting auf sehr große Datasets anwenden.
split_alpha	*Zahl*	Signifikanzschwelle für Aufteilung.
merge_alpha	*Zahl*	Signifikanzschwelle für Zusammenführung.
bonferroni_adjustment	*boolesch*	Signifikanzwerte mit der Bonferroni-Methode anpassen.
split_merged_categories	*boolesch*	Erneutes Aufteilen zusammengeführter Kategorien zulassen.
chi_square	Pearson LR	Verwendetes Verfahren für die Berechnung der Chi-Quadrat-Statistik: Pearson oder Likelihood-Quotient
epsilon	*Zahl*	Minimale Änderung in der erwarteten Zellhäufigkeit...
max_iterations	*Zahl*	Maximale Anzahl der Iterationen für Konvergenz.
set_random_seed	*ganze Zahl*	
seed	*Zahl*	
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*ganze Zahl*	

Eigenschaften des Knotens "coxreg"

Der Knoten vom Typ "Cox-Regression" ermöglicht Ihnen auch bei zensierten Datensätzen die Erstellung eines Überlebensmodells für Daten über die Zeit bis zum Eintreten des Ereignisses. Das Modell erstellt eine Überlebensfunktion, die die Wahrscheinlichkeit vorhersagt, dass das untersuchte Ereignis für bestimmte Werte der Eingabevariablen zu einem bestimmten Zeitpunkt (*t*) eingetreten ist.

Tabelle 95. Eigenschaften des Knotens "coxreg".

Eigenschaften des Knotens coxreg	Werte	Eigenschaftsbeschreibung
survival_time	*Feld*	Cox-Regressionsmodelle erfordern ein einzelnes Feld, das die Überlebenszeiten enthält.
target	*Feld*	Cox-Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
method	Enter Stepwise BackwardsStepwise	
groups	*Feld*	
model_type	MainEffects Custom	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Expert Simple	
max_iterations	*Zahl*	
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	*Zahl*	
probability_removal	*Zahl*	
output_display	EachStep LastStep	
ci_enable	*boolesch*	
ci_value	90 95 99	
correlation	*boolesch*	

Tabelle 95. Eigenschaften des Knotens "coxreg" (Forts.).

Eigenschaften des Knotens coxreg	Werte	Eigenschaftsbeschreibung
display_baseline	*boolesch*	
survival	*boolesch*	
hazard	*boolesch*	
log_minus_log	*boolesch*	
one_minus_survival	*boolesch*	
separate_line	*Feld*	
value	*Zahl* oder *Zeichenfolge*	Wenn für ein Feld kein Wert angegeben ist, wird die Standardoption "Mittelwert" für das betreffende Feld verwendet.

Eigenschaften des Knotens "decisionlist"

Der Knoten "Entscheidungsliste" kennzeichnet Untergruppen bzw. Segmente, die eine höhere oder geringere Wahrscheinlichkeit für ein bestimmtes binäres Ergebnis aufweisen als die Gesamtpopulation. Sie könnten beispielsweise nach Kunden suchen, deren Abwanderung unwahrscheinlich ist oder die mit großer Wahrscheinlichkeit positiv auf eine Kampagne reagieren. Sie können Ihr Fachwissen in das Modell integrieren, indem Sie eigene, benutzerdefinierte Segmente hinzufügen und eine Vorschau anzeigen, in der alternative Modelle nebeneinander angezeigt werden, um die Ergebnisse zu vergleichen. Entscheidungslistenmodelle bestehen aus einer Liste von Regeln, bei denen jede Regel eine Bedingung und ein Ergebnis aufweist. Regeln werden in der vorgegebenen Reihenfolge angewendet und die erste Regel, die zutrifft, bestimmt das Ergebnis.

Tabelle 96. Eigenschaften des Knotens "decisionlist".

Eigenschaften des Knotens decisionlist	Werte	Eigenschaftsbeschreibung
target	*Feld*	Entscheidungslistenmodelle verwenden ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	Bezieht sich auf das Finden von Segmenten; dabei entspricht "Up" einer hohen Wahrscheinlichkeit und "Down" einer geringen Wahrscheinlichkeit.
target_value	*Zeichenfolge*	Wenn dieser Wert nicht angegeben wird, nimmt er für Flags den Wert "True" (Wahr) an.
max_rules	*ganze Zahl*	Die maximale Anzahl der Segmente ausschließlich des Rests.
min_group_size	*ganze Zahl*	Mindestsegmentgröße.
min_group_size_pct	*Zahl*	Mindestsegmentgröße als Prozentsatz.

Tabelle 96. Eigenschaften des Knotens "decisionlist" (Forts.).

Eigenschaften des Knotens `decisionlist`	**Werte**	**Eigenschaftsbeschreibung**
`confidence_level`	*Zahl*	Mindestschwellenwert, den ein Eingabefeld aufweist, um die Wahrscheinlichkeit eines Treffers zu verbessern (Lift), damit es zu einer Segmentdefinition hinzugefügt werden kann.
`max_segments_per_rule`	*ganze Zahl*	
`mode`	`Simple` `Expert`	
`bin_method`	`EqualWidth` `EqualCount`	
`bin_count`	*Zahl*	
`max_models_per_cycle`	*ganze Zahl*	Suchbreite für Listen.
`max_rules_per_cycle`	*ganze Zahl*	Suchbreite für Segmentregeln.
`segment_growth`	*Zahl*	
`include_missing`	*boolesch*	
`final_results_only`	*boolesch*	
`reuse_fields`	*boolesch*	Ermöglicht die Wiederverwendung von Attributen (Eingabefelder, die in Regeln vorkommen).
`max_alternatives`	*ganze Zahl*	
`calculate_raw_propensities`	*boolesch*	
`calculate_adjusted_propensities`	*boolesch*	
`adjusted_propensity_partition`	`Test` `Validation`	

Eigenschaften des Knotens "discriminant"

Bei der Diskriminanzanalyse werden strengere Annahmen als bei der logistischen Regression verwendet, sie kann jedoch eine wertvolle Alternative oder Ergänzung zu einer logistischen Regressionsanalyse sein, wenn diese Annahmen erfüllt sind.

Tabelle 97. Eigenschaften des Knotens "discriminant".

Eigenschaften des Knotens `discriminant`	**Werte**	**Eigenschaftsbeschreibung**
`target`	*Feld*	Diskriminanzmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
`method`	`Enter` `Stepwise`	
`mode`	`Simple` `Expert`	

Tabelle 97. Eigenschaften des Knotens "discriminant" (Forts.).

Eigenschaften des Knotens discriminant	**Werte**	**Eigenschaftsbeschreibung**
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*boolesch*	Statistikoptionen im Dialogfeld "Erweiterte Ausgabe".
univariate_anovas	*boolesch*	
box_m	*boolesch*	
within_group_covariance	*boolesch*	
within_groups_correlation	*boolesch*	
separate_groups_covariance	*boolesch*	
total_covariance	*boolesch*	
fishers	*boolesch*	
unstandardized	*boolesch*	
casewise_results	*boolesch*	Klassifizierungsoptionen im Dialogfeld "Erweiterte Ausgabe".
limit_to_first	*Zahl*	Der Standardwert ist 10.
summary_table	*boolesch*	
leave_one_classification	*boolesch*	
combined_groups	*boolesch*	
separate_groups_covariance	*boolesch*	Matrizenoption **Gruppenspezifische Kovarianzmatrix**
territorial_map	*boolesch*	
combined_groups	*boolesch*	Plotoption **Kombinierte Gruppen**.
separate_groups	*boolesch*	Plotoption **Gruppenspezifisch**.
summary_of_steps	*boolesch*	
F_pairwise	*boolesch*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*Zahl*	
criteria	UseValue UseProbability	
F_value_entry	*Zahl*	Der Standardwert ist 3,84.
F_value_removal	*Zahl*	Der Standardwert ist 2,71.
probability_entry	*Zahl*	Der Standardwert ist 0,05.
probability_removal	*Zahl*	Der Standardwert ist 0,10.
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Tabelle 97. Eigenschaften des Knotens "discriminant" (Forts.).

Eigenschaften des Knotens **discriminant**	Werte	Eigenschaftsbeschreibung
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "factor"

Der Faktor/PCA-Knoten bietet leistungsstarke Datenreduktionsverfahren zur Verringerung der Komplexität der Daten. Die Hauptkomponentenanalyse (PCA) findet lineare Kombinationen der Eingabefelder, die die Varianz im gesamten Set der Felder am besten erfassen, wenn die Komponenten orthogonal (senkrecht) zueinander sind. Mit der Faktorenanalyse wird versucht, die zugrunde liegenden Faktoren zu bestimmen, die die Korrelationsmuster innerhalb eines Sets beobachteter Felder erklären. Bei beiden Ansätzen besteht das Ziel darin, eine kleinere Zahl abgeleiteter Felder zu finden, mit denen die Informationen in der ursprünglichen Menge der Felder effektiv zusammengefasst werden können.

Tabelle 98. Eigenschaften des Knotens "factor".

Eigenschaften des Knotens **factor**	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	PCA-/Faktormodelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*Zahl*	
complete_records	*boolesch*	
matrix	Correlation Covariance	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*Zahl*	
max_factor	*Zahl*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	

Tabelle 98. Eigenschaften des Knotens "factor" (Forts.).

Eigenschaften des Knotens factor	Werte	Eigenschaftsbeschreibung
delta	*Zahl*	Wenn Sie DirectOblimin als Rotationsdatentyp auswählen, können Sie einen Wert für delta festlegen. Wenn Sie keinen Wert festlegen, wird für delta der Standardwert verwendet.
kappa	*Zahl*	Wenn Sie Promax als Rotationsdatentyp auswählen, können Sie einen Wert für kappa festlegen. Wenn Sie keinen Wert festlegen, wird für kappa der Standardwert verwendet.
sort_values	*boolesch*	
hide_values	*boolesch*	
hide_below	*Zahl*	

Eigenschaften des Knotens "featureselection"

Der Merkmalauswahlknoten sichtet die Eingabefelder, um auf der Grundlage einer Reihe von Kriterien (z. B. dem Prozentsatz der fehlenden Werte) zu entscheiden, ob diese entfernt werden sollen. Anschließend erstellt er eine Wichtigkeitsrangfolge der verbleibenden Eingaben in Bezug auf ein angegebenes Ziel. Beispiel: Angenommen, Sie haben ein Dataset mit Hunderten potenzieller Eingaben. Welche davon sind voraussichtlich für die Modellierung von medizinischen Behandlungsergebnissen von Bedeutung?

Tabelle 99. Eigenschaften des Knotens "featureselection".

Eigenschaften des Knotens featureselection	Werte	Eigenschaftsbeschreibung
target	*Feld*	Merkmalauswahlmodelle teilen Prädiktoren relativ zum angegebenen Ziel in Ränge ein. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
screen_single_category	*boolesch*	Bei True wird ein Screening der Felder durchgeführt, bei denen zu viele Datensätze (im Verhältnis zur Gesamtzahl der Datensätze) in dieselbe Kategorie fallen.
max_single_category	*Zahl*	Gibt den Schwellenwert an, der verwendet wird, wenn screen_single_category aufTrue gesetzt ist.
screen_missing_values	*boolesch*	Bei True wird ein Screening der Felder durchgeführt, die zu viele fehlende Werte (ausgedrückt als Prozentsatz der Gesamtzahl an Datensätzen) aufweisen.
max_missing_values	*Zahl*	

Tabelle 99. Eigenschaften des Knotens "featureselection" (Forts.).

Eigenschaften des Knotens `featureselection`	**Werte**	**Eigenschaftsbeschreibung**
`screen_num_categories`	*boolesch*	Bei `True` wird ein Screening der Felder durchgeführt, die zu viele Kategorien im Verhältnis zur Gesamtzahl der Datensätze aufweisen.
`max_num_categories`	*Zahl*	
`screen_std_dev`	*boolesch*	Bei `True` wird ein Screening der Felder durchgeführt, deren Standardabweichung kleiner oder gleich dem angegebenen Mindestwert ist.
`min_std_dev`	*Zahl*	
`screen_coeff_of_var`	*boolesch*	Bei `True` wird ein Screening der Felder durchgeführt, deren Varianzkoeffizient kleiner oder gleich dem angegebenen Mindestwert ist.
`min_coeff_of_var`	*Zahl*	

`criteria`	`Pearson`
	`Likelihood`
	`CramersV`
	`Lambda`
		Wenn kategoriale Prädiktoren hinsichtlich eines kategorialen Ziels nach Rängen geordnet werden, wird hier das Maß angegeben, auf dem der Wert für die Wichtigkeit beruht.

`unimportant_below`	*Zahl*	Gibt die *p*-Schwellenwerte an, die verwendet werden, um Variablen als "bedeutsam", "marginal" bzw. "unbedeutend" eingestuft werden. Zulässig sind Werte von 0,0 bis 1,0.
`important_above`	*Zahl*	Zulässig sind Werte von 0,0 bis 1,0.
`unimportant_label`	*Zeichenfolge*	Gibt die Beschriftung für die Rangstufe "unbedeutsam" an.
`marginal_label`	*Zeichenfolge*	
`important_label`	*Zeichenfolge*	

`selection_mode`	`ImportanceLevel`
	`ImportanceValue`
	`TopN`	

`select_important`	*boolesch*	Wenn `selection_mode` auf `ImportanceLevel` gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.
`select_marginal`	*boolesch*	Wenn `selection_mode` auf `ImportanceLevel` gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
`select_unimportant`	*boolesch*	Wenn `selection_mode` auf `ImportanceLevel` gesetzt ist, wird hier angegeben, ob unbedeutende Felder ausgewählt werden sollen.
`importance_value`	*Zahl*	Wenn `selection_mode` auf `ImportanceValue` gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 100.

Tabelle 99. Eigenschaften des Knotens "featureselection" (Forts.).

Eigenschaften des Knotens **featureselection**	Werte	Eigenschaftsbeschreibung
top_n	*ganze Zahl*	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 1000.

Eigenschaften des Knotens "genlin"

Das verallgemeinerte lineare Modell erweitert das allgemeine lineare Modell so, dass die abhängige Variable über eine angegebene Verknüpfungsfunktion in linearem Zusammenhang zu den Faktoren und Kovariaten steht. Außerdem ist es mit diesem Modell möglich, dass die abhängige Variable eine von der Normalverteilung abweichende Verteilung aufweist. Es deckt die Funktionen einer großen Bandbreite an Statistikmodellen ab, darunter lineare Regression, logistische Regression, loglineare Modelle für Häufigkeitsdaten und Überlebensmodelle mit Intervallzensierung.

Tabelle 100. Eigenschaften des Knotens "genlin".

Eigenschaften des Knotens **genlin**	Werte	Eigenschaftsbeschreibung
target	*Feld*	Verallgemeinerte lineare Modelle erfordern ein einzelnes Zielfeld, bei dem es sich um ein nominales oder ein Flagfeld handeln muss, und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
use_weight	*boolesch*	
weight_field	*Feld*	Der Feldtyp ist nur stetig.
target_represents_trials	*boolesch*	
trials_type	Variable FixedValue	
trials_field	*Feld*	Der Feldttyp ist stetig, Flag oder ordinal.
trials_number	*Zahl*	Der Standardwert ist 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*Feld*	Der Feldtyp ist nur stetig.
offset_value	*Zahl*	Muss eine reelle Zahl sein.
base_category	Last First	
include_intercept	*boolesch*	
mode	Simple Expert	

Tabelle 100. Eigenschaften des Knotens "genlin" (Forts.).

Eigenschaften des Knotens genlin	Werte	Eigenschaftsbeschreibung
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Invers normal. NEGBIN: Negativ binomial.
negbin_para_type	Specify Estimate	
negbin_parameter	*Zahl*	Der Standardwert ist 1.Muss eine nicht negative reelle Zahl enthalten.
tweedie_parameter	*Zahl*	
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: Log-Log komplementär. LOGC: Log-Komplement. NEGBIN: Negativ binomial. NLOGLOG: Log-Log negativ. CUMCAUCHIT: Cauchit (kumulativ). CUMCLOGLOG: Log-Log komplementär (kumulativ). CUMLOGIT: Logit (kumulativ). CUMNLOGLOG: Log-Log negativ (kumulativ). CUMPROBIT: Probit (kumulativ).
power	*Zahl*	Der Wert muss eine reelle Zahl ungleich null sein.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*Zahl*	Der Standardwert ist 1; nur positive ganze Zahlen zulässig.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*Zahl*	Der Standardwert ist 1; muss größer als 0 sein.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*Zahl*	Der Standardwert ist 100; nur nicht negative ganze Zahlen.
max_step_halving	*Zahl*	Der Standardwert ist 5; nur positive ganze Zahlen.
check_separation	*boolesch*	
start_iteration	*Zahl*	Der Standardwert ist 20; nur positive ganze Zahlen zulässig.
estimates_change	*boolesch*	

Tabelle 100. Eigenschaften des Knotens "genlin" (Forts.).

Eigenschaften des Knotens genlin	Werte	Eigenschaftsbeschreibung
estimates_change_min	*Zahl*	Der Standardwert ist 1E-006; nur positive Zahlen zulässig.
estimates_change_type	Absolute Relative	
loglikelihood_change	*boolesch*	
loglikelihood_change_min	*Zahl*	Nur positive Zahlen zulässig.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*boolesch*	
hessian_convergence_min	*Zahl*	Nur positive Zahlen zulässig.
hessian_convergence_type	Absolute Relative	
case_summary	*boolesch*	
contrast_matrices	*boolesch*	
descriptive_statistics	*boolesch*	
estimable_functions	*boolesch*	
model_info	*boolesch*	
iteration_history	*boolesch*	
goodness_of_fit	*boolesch*	
print_interval	*Zahl*	Der Standardwert ist 1; muss eine positive ganze Zahl sein.
model_summary	*boolesch*	
lagrange_multiplier	*boolesch*	
parameter_estimates	*boolesch*	
include_exponential	*boolesch*	
covariance_estimates	*boolesch*	
correlation_estimates	*boolesch*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*Zahl*	Der Standardwert ist 0,0001.
confidence_interval	*Zahl*	Der Standardwert ist 95.
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	

Tabelle 100. Eigenschaften des Knotens "genlin" (Forts.).

Eigenschaften des Knotens genlin	Werte	Eigenschaftsbeschreibung
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "glmm"

Verallgemeinerte lineare gemischte Modelle (Generalized Linear Mixed Models; GLMM) erweitern lineare Modelle so, dass das Ziel nicht normalverteilt zu sein braucht und über eine angegebene Verknüpfungsfunktion in einer linearen Beziehung zu den Faktoren und Kovariaten steht und die Beobachtungen korreliert werden können. Verallgemeinerte lineare gemischte Modelle decken eine breite Palette verschiedener Modelle ab, von einfacher linearer Regression bis hin zu komplexen Mehrebenenmodellen für nicht normalverteilte Longitudinaldaten.

Tabelle 101. Eigenschaften des Knotens "glmm".

Eigenschaften des Knotens glmm	Werte	Eigenschaftsbeschreibung
residual_subject_spec	*strukturiert*	Die Wertekombination der angegebenen kategorialen Felder, die Subjekte innerhalb des Datasets eindeutig definieren.
repeated_measures	*strukturiert*	Felder zur Ermittlung von wiederholten Beobachtungen.
residual_group_spec	[*Feld1 ... FeldN*]	Felder, die unabhängige Sätze von Kovarianzparametern für wiederholte Effekte definieren.
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Definiert die Kovarianzstruktur für Residuen.
custom_target	*boolesch*	Gibt an, ob das im vorausgehenden Knoten definierte Ziel (false) oder das im Feld target_field festgelegte benutzerdefinierte Ziel (true) verwendet werden soll.
target_field	*Feld*	Als Ziel zu verwendendes Feld, wenn custom_target auf true gesetzt ist.
use_trials	*boolesch*	Gibt an, ob zusätzliche Felder oder Werte zur Angabe der Anzahl an Tests verwendet werden sollen, wenn es sich bei der Zielantwort um eine Reihe von Ereignissen handelt, die während Tests auftreten. Die Standardeinstellung ist false.

Tabelle 101. Eigenschaften des Knotens "glmm" (Forts.).

Eigenschaften des Knotens glmm	Werte	Eigenschaftsbeschreibung
use_field_or_value	Field Value	Gibt an, ob die Anzahl an Test in einem Feld (Standard) oder als Wert angegeben werden soll.
trials_field	*Feld*	Feld zur Angabe der Anzahl an Tests.
trials_value	*ganze Zahl*	Wert zur Angabe der Anzahl an Tests. Wenn angegeben, ist der Minimalwert 1.
use_custom_target_reference	*boolesch*	Gibt an, ob eine benutzerdefinierte Referenzkategorie für ein kategoriales Ziel verwendet werden soll. Die Standardeinstellung ist false.
target_reference_value	*Zeichenfolge*	Zu verwendende Referenzkategorie, wenn use_custom_target_reference auf true gesetzt ist.
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Allgemeine Modelle für die Verteilung von Werten für das Ziel. Wählen Sie Custom aus, um einen Verteilungstyp aus der von target_distribution bereitgestellten Liste festzulegen.
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson	Verteilung von Werten für das Ziel, wenn dist_link_combination auf Custom gesetzt ist.
link_function_type	IDENTITY LOGC LOG CLOGLOG LOGIT NLOGLOG PROBIT POWER CAUCHIT	Verknüpfungsfunktion zum Herstellen von Beziehungen zwischen Zielwerten und Prädiktoren. Wenn target_distribution auf Binomial gesetzt ist, können Sie jede der aufgelisteten Verknüpfungsfunktionen verwenden. Wenn target_distribution auf Multinomial gesetzt ist, können Sie CLOGLOG, CAUCHIT, LOGIT, NLOGLOG oder PROBIT verwenden. Wenn target_distribution auf einen anderen Wert als Binomial oder Multinomial gesetzt ist, können Sie IDENTITY, LOG oder POWER verwenden.
link_function_param	*Zahl*	Für die Verknüpfungsfunktion zu verwendender Parameterwert. Wird nur verwendet, wenn normal_link_function oder link_function_type auf POWER gesetzt ist.

Tabelle 101. Eigenschaften des Knotens "glmm" (Forts.).

Eigenschaften des Knotens glmm	Werte	Eigenschaftsbeschreibung
use_predefined_inputs	*boolesch*	Gibt an, ob die Felder, die in einer übergeordneten Ebene als Eingabefelder definiert wurden (true), oder die Felder in fixed_effects_list (false) als Felder für feste Effekte verwendet werden sollen. Die Standardeinstellung ist false.
fixed_effects_list	*strukturiert*	Wenn use_predefined_inputs auf false gesetzt ist, werden die Eingabefelder als Felder für feste Effekte verwendet.
use_intercept	*boolesch*	Wenn true gesetzt ist (Standardeinstellung), wird der konstante Term in das Modell einbezogen.
random_effects_list	*strukturiert*	Liste von Feldern, die als zufällige Effekte festgelegt werden.
regression_weight_field	*Feld*	Zur Analysegewichtung zu verwendendes Feld.
use_offset	None offset_value offset_field	Gibt an, wie der Offset festgelegt wird. Lautet der Wert None, wird kein Offset verwendet.
offset_value	*Zahl*	Für den Offset zu verwendender Wert, wenn use_offset auf offset_value gesetzt ist.
offset_field	*Feld*	Für den Offsetwert zu verwendendes Feld, wenn use_offset auf offset_field gesetzt ist.
target_category_order	Ascending Descending Data	Sortierreihenfolge für kategoriale Ziele. Der Wert Data gibt an, dass die Sortierreihenfolge der Daten verwendet wird. Die Standardeinstellung ist Ascending.
inputs_category_order	Ascending Descending Data	Sortierreihenfolge für kategoriale Prädiktoren. Der Wert Data gibt an, dass die Sortierreihenfolge der Daten verwendet wird. Die Standardeinstellung ist Ascending.
max_iterations	*ganze Zahl*	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden. Eine nicht negative ganze Zahl. Der Standardwert ist 100.
confidence_level	*ganze Zahl*	Konfidenzniveau für die Berechnung von Intervallschätzungen der Modellkoeffizienten. Eine nicht negative ganze Zahl. Der Maximalwert ist 100 und der Standardwert ist 95.
degrees_of_freedom_method	Fixed Varied	Gibt an, wie Freiheitsgrade für Signifikanztests berechnet werden.
test_fixed_effects_coeffecients	Model Robust	Methode zur Berechnung der Kovarianzmatrix für Parameterschätzungen.
use_p_converge	*boolesch*	Option für Parameterkonvergenz.
p_converge	*Zahl*	Leerzeichen oder ein beliebiger positiver Wert.

Tabelle 101. Eigenschaften des Knotens "glmm" (Forts.).

Eigenschaften des Knotens glmm	Werte	Eigenschaftsbeschreibung
p_converge_type	Absolute Relative	
use_l_converge	*boolesch*	Option für Log-Likelihood-Konvergenz.
l_converge	*Zahl*	Leerzeichen oder ein beliebiger positiver Wert.
l_converge_type	Absolute Relative	
use_h_converge	*boolesch*	Option für Konvergenz der Hesse-Matrix.
h_converge	*Zahl*	Leerzeichen oder ein beliebiger positiver Wert.
h_converge_type	Absolute Relative	
max_fisher_steps	*ganze Zahl*	
singularity_tolerance	*Zahl*	
use_model_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Name (true) oder ein vom System generierter Name (false) für das Modell verwendet werden soll. Die Standardeinstellung ist false.
model_name	*Zeichenfolge*	Gibt den zu verwendenden Modellnamen an, wenn use_model_name auf true gesetzt ist.
confidence	onProbability onIncrease	Grundlage für die Berechnung des Konfidenzwerts für das Scoring: höchste vorhergesagte Wahrscheinlichkeit oder Differenz zwischen der höchsten und zweithöchsten vorhergesagten Wahrscheinlichkeit.
score_category_probabilities	*boolesch*	Auf true gesetzt, werden vorhergesagte Wahrscheinlichkeiten für kategoriale Ziele generiert. Die Standardeinstellung ist false.
max_categories	*ganze Zahl*	Wenn score_category_probabilities auf true gesetzt ist, wird hier die maximale Anzahl der zu speichernden Kategorien festgelegt.
score_propensity	*boolesch*	Auf true gesetzt, werden Propensity-Scores für Flagzielfelder generiert, die die Wahrscheinlichkeit angeben, mit der das Feld den Wert "true" haben wird.
emeans	*structure*	Für jedes kategoriale Feld aus der Liste mit festen Effekten wird hier angegeben, ob geschätzte Randmittel generiert werden sollen.
covariance_list	*structure*	Für jedes stetige Feld aus der Liste mit festen Effekten wird hier angegeben, ob der Mittelwert oder ein benutzerdefinierter Wert für die Berechnung der geschätzten Randmittel verwendet werden soll.

Tabelle 101. Eigenschaften des Knotens "glmm" (Forts.).

Eigenschaften des Knotens glmm	Werte	Eigenschaftsbeschreibung
mean_scale	Original Transformed	Gibt an, ob geschätzte Randmittel anhand der ursprünglichen Skala des Ziels (Standard) oder anhand der Transformation der Verknüpfungsfunktion berechnet werden sollen.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Zu verwendende Anpassungsmethode bei Hypothesentests mit mehreren Kontrasten.

Eigenschaften des Knotens "kmeans"

Der K-Means-Knoten teilt das Dataset in unterschiedliche Gruppen (oder Cluster) auf. Bei diesem Verfahren wird eine festgelegte Anzahl von Clustern definiert, den Clustern werden iterativ Datensätze zugewiesen und die Clusterzentren werden angepasst, bis eine weitere Verfeinerung keine wesentliche Verbesserung des Modells mehr darstellen würde. Statt zu versuchen, ein Ergebnis vorherzusagen, versucht *K*-Means mithilfe eines als "nicht überwachtes Lernen" bezeichneten Verfahrens Muster im Set der Eingabefelder zu entdecken.

Tabelle 102. Eigenschaften des Knotens "kmeans".

Eigenschaften des Knotens kmeans	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	K-Means-Modelle führen eine Clusteranalyse an einer Menge von Eingabefeldern durch, verwenden jedoch kein Zielfeld. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
num_clusters	*Zahl*	
gen_distance	*boolesch*	
cluster_label	String Zahl	
label_prefix	*Zeichenfolge*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*Zahl*	
tolerance	*Zahl*	
encoding_value	*Zahl*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften des Knotens "knn"

Der Knoten "*k*-Nächste Nachbarn" (KNN) verknüpft einen neuen Fall mit der Kategorie oder dem Wert der *k* Objekte, die ihm im Prädiktorraum am nächsten liegen, wobei *k* eine ganze Zahl ist. Ähnliche Fälle liegen nah beieinander und Fälle mit geringer Ähnlichkeit sind weit voneinander entfernt.

Tabelle 103. Eigenschaften des Knotens "knn".

Eigenschaften des Knotens knn	Werte	Eigenschaftsbeschreibung
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*boolesch*	
use_case_labels	*boolesch*	Kontrollkästchen markieren, um nächste Option zu aktivieren.
case_labels_field	*Feld*	
identify_focal_cases	*boolesch*	Kontrollkästchen markieren, um nächste Option zu aktivieren.
focal_cases_field	*Feld*	
automatic_k_selection	*boolesch*	
fixed_k	*ganze Zahl*	Nur aktiviert, wenn automatic_k_selection auf False eingestellt ist.
minimum_k	*ganze Zahl*	Nur aktiviert, wenn automatic_k_selection auf True eingestellt ist.
maximum_k	*ganze Zahl*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*boolesch*	
range_predictions	Mean Median	
perform_feature_selection	*boolesch*	
forced_entry_inputs	[*Feld1 ... FeldN*]	
stop_on_error_ratio	*boolesch*	
number_to_select	*ganze Zahl*	
minimum_change	*Zahl*	
validation_fold_assign_by_field	*boolesch*	
number_of_folds	*ganze Zahl*	Nur aktiviert, wenn validation_fold_assign_by_field auf False eingestellt ist.
set_random_seed	*boolesch*	
random_seed	*Zahl*	
folds_field	*Feld*	Nur aktiviert, wenn validation_fold_assign_by_field auf True eingestellt ist.

Tabelle 103. Eigenschaften des Knotens "knn" (Forts.).

Eigenschaften des Knotens knn	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
save_distances	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "kohonen"

Der Kohonen-Knoten erstellt eine Art von neuronalem Netz, das verwendet werden kann, um ein Clustering des Datasets in einzelne Gruppen vorzunehmen. Wenn das Netz voll trainiert ist, sollten ähnliche Datensätze auf der Ausgabekarte eng nebeneinander stehen, während Datensätze, die sich unterscheiden, weit voneinander entfernt sein sollten. Die Zahl der von jeder Einheit im Modellnugget erfassten Beobachtungen gibt Aufschluss über die starken Einheiten. Dadurch wird ein Eindruck von der ungefähren Zahl der Cluster vermittelt.

Tabelle 104. Eigenschaften des Knotens "kohonen".

Eigenschaften des Knotens kohonen	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	Kohonen-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
continue	*boolesch*	
show_feedback	*boolesch*	
stop_on	Default Zeit	
time	*Zahl*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
cluster_label	*boolesch*	
mode	Simple Expert	
width	*Zahl*	
length	*Zahl*	
decay_style	Linear Exponential	
phase1_neighborhood	*Zahl*	
phase1_eta	*Zahl*	
phase1_cycles	*Zahl*	
phase2_neighborhood	*Zahl*	
phase2_eta	*Zahl*	

Tabelle 104. Eigenschaften des Knotens "kohonen" (Forts.).

Eigenschaften des Knotens kohonen	Werte	Eigenschaftsbeschreibung
phase2_cycles	*Zahl*	

Eigenschaften des Knotens "linear"

Bei linearen Regressionsmodellen wird ein stetiges Ziel auf der Basis linearer Beziehungen zwischen dem Ziel und einem oder mehreren Prädiktoren vorhergesagt.

Tabelle 105. Eigenschaften des Knotens "linear".

Eigenschaften des Knotens linear	Werte	Eigenschaftsbeschreibung
target	*Feld*	Gibt ein einzelnes Zielfeld an.
inputs	[*Feld1 ... FeldN*]	Im Modell verwendete Prädiktorfelder.
continue_training_existing_model	*boolesch*	
objective	Standard Bagging Boosting psm	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
use_auto_data_preparation	*boolesch*	
confidence_level	*Zahl*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*Zahl*	
probability_removal	*Zahl*	
use_max_effects	*boolesch*	
max_effects	*Zahl*	
use_max_steps	*boolesch*	
max_steps	*Zahl*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*Zahl*	
use_random_seed	*boolesch*	
random_seed	*Zahl*	
use_custom_model_name	*boolesch*	
custom_model_name	*Zeichenfolge*	

Tabelle 105. Eigenschaften des Knotens "linear" (Forts.).

Eigenschaften des Knotens `linear`	Werte	Eigenschaftsbeschreibung
`use_custom_name`	*boolesch*	
`custom_name`	*Zeichenfolge*	
`tooltip`	*Zeichenfolge*	
`keywords`	*Zeichenfolge*	
`annotation`	*Zeichenfolge*	

Eigenschaften des Knotens "logreg"

Die logistische Regression ist ein statistisches Verfahren zur Klassifizierung von Datensätzen auf der Grundlage der Werte von Eingabefeldern. Sie ist analog zur linearen Regression, außer dass statt eines numerischen Bereichs ein kategoriales Zielfeld verwendet wird.

Tabelle 106. Eigenschaften des Knotens "logreg".

Eigenschaften des Knotens `logreg`	Werte	Eigenschaftsbeschreibung
`target`	*Feld*	Logistische Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten” auf Seite 133.
`logistic_procedure`	`Binomial` `Multinomial`	
`include_constant`	*boolesch*	
`mode`	`Simple` `Expert`	

`method`	`Enter`
	`Stepwise`
	`Forwards`
	`Backwards`
	`BackwardsStepwise`	

`binomial_method`	`Enter`
	`Forwards`
	`Backwards`	

`model_type`	`MainEffects`
	`FullFactorial`
	`Custom`
		Wenn als Modelltyp `FullFactorial` festgelegt ist, werden keine Schrittmethoden ausgeführt, auch wenn diese ebenfalls angegeben wurden. Stattdessen wird die Methode `Enter` verwendet.
		
		Wenn der Modelltyp auf `Custom` gesetzt ist, jedoch keine benutzerdefinierten Felder angegeben wurden, wird ein Haupteffektmodell erstellt.

`custom_terms`	[*{BD Geschlecht}{BD}{Alter}*]	
`multinomial_base_category`	*Zeichenfolge*	Gibt an, wie die Referenzkategorie bestimmt wird.

Tabelle 106. Eigenschaften des Knotens "logreg" (Forts.).

Eigenschaften des Knotens logreg	Werte	Eigenschaftsbeschreibung
binomial_categorical_input	*Zeichenfolge*	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Verschlüsselte Eigenschaft für kategoriale Eingaben, die angibt, wie der Kontrast bestimmt wird.
binomial_input_category	First Last	Verschlüsselte Eigenschaft für kategoriale Eingaben, die angibt, wie die Referenzkategorie bestimmt wird.
scale	None UserDefined Pearson Deviance	
scale_value	*Zahl*	
all_probabilities	*boolesch*	
tolerance	1,0E-5 1,0E-6 1,0E-7 1,0E-8 1,0E-9 1,0E-10	
min_terms	*Zahl*	
use_max_terms	*boolesch*	
max_terms	*Zahl*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*Zahl*	
probability_removal	*Zahl*	
binomial_probability_entry	*Zahl*	
binomial_probability_removal	*Zahl*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*Zahl*	
max_steps	*Zahl*	
p_converge	1,0E-4 1,0E-5 1,0E-6 1,0E-7 1,0E-8 0	

Tabelle 106. Eigenschaften des Knotens "logreg" (Forts.).

Eigenschaften des Knotens logreg	Werte	Eigenschaftsbeschreibung
l_converge	1,0E-1 1,0E-2 1,0E-3 1,0E-4 1,0E-5 0	
delta	*Zahl*	
iteration_history	*boolesch*	
history_steps	*Zahl*	
summary	*boolesch*	
likelihood_ratio	*boolesch*	
asymptotic_correlation	*boolesch*	
goodness_fit	*boolesch*	
parameters	*boolesch*	
confidence_interval	*Zahl*	
asymptotic_covariance	*boolesch*	
classification_table	*boolesch*	
stepwise_summary	*boolesch*	
info_criteria	*boolesch*	
monotonicity_measures	*boolesch*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*boolesch*	
binomial_parameters	*boolesch*	
binomial_iteration_history	*boolesch*	
binomial_classification_plots	*boolesch*	
binomial_ci_enable	*boolesch*	
binomial_ci	*Zahl*	
binomial_residual	outliers all	
binomial_residual_enable	*boolesch*	
binomial_outlier_threshold	*Zahl*	
binomial_classification_cutoff	*Zahl*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	

Eigenschaften des Knotens "neuralnet"

Vorsicht: In dieser Version ist eine neuere Fassung des Netzmodellierungsknotens mit erweiterten Funktionen verfügbar, die im nächsten Abschnitt beschrieben wird (*neuralnetwork*). Sie können zwar auch weiterhin Modelle mit der Vorgängerversion erstellen und scoren, doch empfehlen wir die Aktualisierung Ihrer Scripts zur Verwendung der neuen Version. Details der vorherigen Version werden hier aus Referenzgründen aufbewahrt.

Tabelle 107. Eigenschaften des Knotens "neuralnet".

Eigenschaften des Knotens neuralnet	**Werte**	**Eigenschaftsbeschreibung**
targets	[*Feld1 ... FeldN*]	Der Netzknoten erwartet mindestens ein Zielfeld und mindestens ein Eingabefeld. Häufigkeits- und Gewichtungsfelder werden ignoriert. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*boolesch*	
train_pct	*Zahl*	
set_random_seed	*boolesch*	
random_seed	*Zahl*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Zeit	Stoppmodus.
accuracy	*Zahl*	Stoppgenauigkeit.
cycles	*Zahl*	Zu trainierende Zyklen.
time	*Zahl*	Dauer der Trainingsphase (Minuten)
continue	*boolesch*	
show_feedback	*boolesch*	
binary_encode	*boolesch*	
use_last_model	*boolesch*	
gen_logfile	*boolesch*	
logfile_name	*Zeichenfolge*	
alpha	*Zahl*	
initial_eta	*Zahl*	
high_eta	*Zahl*	
low_eta	*Zahl*	
eta_decay_cycles	*Zahl*	

Tabelle 107. Eigenschaften des Knotens "neuralnet" (Forts.).

Eigenschaften des Knotens neuralnet	**Werte**	**Eigenschaftsbeschreibung**
hid_layers	One Two Three	
hl_units_one	*Zahl*	
hl_units_two	*Zahl*	
hl_units_three	*Zahl*	
persistence	*Zahl*	
m_topologies	*Zeichenfolge*	
m_non_pyramids	*boolesch*	
m_persistence	*Zahl*	
p_hid_layers	One Two Three	
p_hl_units_one	*Zahl*	
p_hl_units_two	*Zahl*	
p_hl_units_three	*Zahl*	
p_persistence	*Zahl*	
p_hid_rate	*Zahl*	
p_hid_pers	*Zahl*	
p_inp_rate	*Zahl*	
p_inp_pers	*Zahl*	
p_overall_pers	*Zahl*	
r_persistence	*Zahl*	
r_num_clusters	*Zahl*	
r_eta_auto	*boolesch*	
r_alpha	*Zahl*	
r_eta	*Zahl*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
calculate_variable_importance	*boolesch*	Hinweis: Die in früheren Versionen verwendete Eigenschaft sensitivity_analysis wird zugunsten dieser Eigenschaft nicht mehr verwendet. Die alte Eigenschaft wird weiterhin unterstützt, es wird jedoch calculate_variable_importance empfohlen.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "neuralnetwork"

Der Netzknoten verwendet ein vereinfachtes Modell der Art und Weise, wie ein menschliches Gehirn Informationen verarbeitet. Es funktioniert, indem eine große Anzahl miteinander verbundener einfacher Verarbeitungseinheiten simuliert wird, die abstrakten Versionen von Neuronen ähnlich sind. Neuronale Netze sind leistungsstarke Mehrzweckschätzer, für deren Training und Anwendung nur sehr geringe statistische oder mathematische Kenntnisse erforderlich sind.

Tabelle 108. Eigenschaften des Knotens "neuralnetwork".

Eigenschaften des Knotens `neuralnetwork`	**Werte**	**Eigenschaftsbeschreibung**
`targets`	[*Feld1 ... FeldN*]	Gibt die Zielfelder an.
`inputs`	[*Feld1 ... FeldN*]	Im Modell verwendete Prädiktorfelder.
`splits`	*[Feld1 ... FeldN*	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an.
`use_partition`	*boolesch*	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
`continue`	*boolesch*	Training des bestehenden Modells fortsetzen.

`objective`	`Standard`
	`Bagging`
	`Boosting`
	`psm`
		`PSM` wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.

`method`	`MultilayerPerceptron` `RadialBasisFunction`	
`use_custom_layers`	*boolesch*	
`first_layer_units`	*Zahl*	
`second_layer_units`	*Zahl*	
`use_max_time`	*boolesch*	
`max_time`	*Zahl*	
`use_max_cycles`	*boolesch*	
`max_cycles`	*Zahl*	
`use_min_accuracy`	*boolesch*	
`min_accuracy`	*Zahl*	

`combining_rule_categorical`	`Voting`
	`HighestProbability`
	`HighestMeanProbability`	

`combining_rule_continuous`	`Mean` `Median`	
`component_models_n`	*Zahl*	
`overfit_prevention_pct`	*Zahl*	
`use_random_seed`	*boolesch*	
`random_seed`	*Zahl*	

Tabelle 108. Eigenschaften des Knotens "neuralnetwork" (Forts.).

Eigenschaften des Knotens neuralnetwork	Werte	Eigenschaftsbeschreibung
missing_values	listwiseDeletion missingValueImputation	
use_custom_model_name	*boolesch*	
custom_model_name	*Zeichenfolge*	
confidence	onProbability onIncrease	
score_category_probabilities	*boolesch*	
max_categories	*Zahl*	
score_propensity	*boolesch*	
use_custom_name	*boolesch*	
custom_name	*Zeichenfolge*	
tooltip	*Zeichenfolge*	
keywords	*Zeichenfolge*	
annotation	*Zeichenfolge*	

Eigenschaften des Knotens "quest"

Der QUEST-Knoten bietet eine binäre Klassifizierungsmethode zum Erstellen von Entscheidungsbäumen, die dazu dient, die für große C&R-Baumanalysen erforderliche Verarbeitungszeit zu verkürzen. Gleichzeitig soll die in den Klassifizierungsbaummodellen festgestellte Tendenz verringert werden, die darin besteht, dass Eingaben bevorzugt werden, die mehr Aufteilungen erlauben. Eingabefelder können stetig (numerische Bereiche) sein, das Zielfeld muss aber kategorial sein. Alle Aufteilungen sind binär.

Tabelle 109. Eigenschaften des Knotens "quest".

Eigenschaften des Knotens quest	Werte	Eigenschaftsbeschreibung
target	*Feld*	QUEST-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
continue_training_existing_model	*boolesch*	
objective	Standard Boosting Bagging psm	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*boolesch*	
tree_directives	*Zeichenfolge*	
use_max_depth	Default Custom	

Tabelle 109. Eigenschaften des Knotens "quest" (Forts.).

Eigenschaften des Knotens quest	Werte	Eigenschaftsbeschreibung
max_depth	*ganze Zahl*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	*boolesch*	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	*boolesch*	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	*Zahl*	Maximale Differenz.
max_surrogates	*Zahl*	Maximale Anzahl Ersatztrenner.
use_percentage	*boolesch*	
min_parent_records_pc	*Zahl*	
min_child_records_pc	*Zahl*	
min_parent_records_abs	*Zahl*	
min_child_records_abs	*Zahl*	
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft.
adjust_priors	*boolesch*	
trails	*Zahl*	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	*boolesch*	Boosting auf sehr große Datasets anwenden.
split_alpha	*Zahl*	Signifikanzschwelle für Aufteilung.
train_pct	*Zahl*	Set zur Verhinderung übermäßiger Anpassung.
set_random_seed	*boolesch*	Option "Ergebnisse replizieren".
seed	*Zahl*	
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "regression"

Die lineare Regression ist ein statistisches Verfahren zur Zusammenfassung von Daten und die Erstellung von Vorhersagen durch Anpassung einer geraden Linie oder Fläche, mit der die Diskrepanzen zwischen den vorhergesagten und den tatsächlichen Ausgabewerten minimiert werden.

Hinweis: Der Regressionsknoten wird in einer zukünftigen Version durch den Linearknoten ersetzt. Es wird empfohlen, dass Sie von nun an lineare Modelle für lineare Regression verwenden.

Tabelle 110. Eigenschaften des Knotens "regression".

Eigenschaften des Knotens regression	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
method	Enter Stepwise Backwards Forwards	
include_constant	*boolesch*	
use_weight	*boolesch*	
weight_field	*Feld*	
mode	Simple Expert	
complete_records	*boolesch*	
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	Verwenden Sie für Argumente doppelte Anführungszeichen.
stepping_method	useP useF	useP: F-Wahrscheinlichkeit verwenden useF: F-Wert verwenden
probability_entry	*Zahl*	
probability_removal	*Zahl*	
F_value_entry	*Zahl*	
F_value_removal	*Zahl*	
selection_criteria	*boolesch*	
confidence_interval	*boolesch*	
covariance_matrix	*boolesch*	

Tabelle 110. Eigenschaften des Knotens "regression" (Forts.).

Eigenschaften des Knotens regression	Werte	Eigenschaftsbeschreibung
collinearity_diagnostics	*boolesch*	
regression_coefficients	*boolesch*	
exclude_fields	*boolesch*	
durbin_watson	*boolesch*	
model_fit	*boolesch*	
r_squared_change	*boolesch*	
p_correlations	*boolesch*	
descriptives	*boolesch*	
calculate_variable_importance	*boolesch*	

Eigenschaften des Knotens "sequence"

Der Sequenzknoten erkennt Assoziationsregeln in sequenziellen oder zeitorientierten Daten. Eine Sequenz ist eine Liste mit Elementsets, die in einer vorhersagbaren Reihenfolge auftreten. Beispiel: Ein Kunde, der einen Rasierer und After-Shave-Lotion kauft, kauft möglicherweise beim nächsten Einkauf Rasiercreme. Der Sequenzknoten basiert auf dem CARMA-Assoziationsregelalgorithmus, der eine effiziente bidirektionale Methode zum Suchen von Sequenzen verwendet.

Tabelle 111. Eigenschaften des Knotens "sequence".

Eigenschaften des Knotens sequence	Werte	Eigenschaftsbeschreibung
id_field	*Feld*	Um ein Sequenzmodell zu erstellen, müssen Sie ein ID-Feld, ein optionales Zeitfeld und mindestens ein Inhaltsfeld angeben. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
time_field	*Feld*	
use_time_field	*boolesch*	
content_fields	[*Feld1 ... FeldN*]	
contiguous	*boolesch*	
min_supp	*Zahl*	
min_conf	*Zahl*	
max_size	*Zahl*	
max_predictions	*Zahl*	
mode	Simple Expert	
use_max_duration	*boolesch*	
max_duration	*Zahl*	
use_gaps	*boolesch*	
min_item_gap	*Zahl*	

Tabelle 111. Eigenschaften des Knotens "sequence" (Forts.).

Eigenschaften des Knotens sequence	Werte	Eigenschaftsbeschreibung
max_item_gap	*Zahl*	
use_pruning	*boolesch*	
pruning_value	*Zahl*	
set_mem_sequences	*boolesch*	
mem_sequences	*ganze Zahl*	

Eigenschaften des Knotens "slrm"

Mithilfe des Knotens für das lernfähige Antwortmodell (Self-Learning Response Model, SLRM) können Sie ein Modell erstellen, in dem das Modell anhand eines einzelnen neuen Falls oder einer kleinen Anzahl neuer Fälle neu eingeschätzt werden kann, ohne dass das Modell mit allen Daten neu trainiert werden muss.

Tabelle 112. Eigenschaften des Knotens "slrm".

Eigenschaften des Knotens slrm	Werte	Eigenschaftsbeschreibung
target	*Feld*	Beim Zielfeld muss es sich um ein nominales oder ein Flagfeld handeln. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
target_response	*Feld*	Der Typ muss "Flag" sein.
continue_training_existing_model	*boolesch*	
target_field_values	*boolesch*	Alle verwenden: Alle Werte aus der Quelle verwenden. Angeben: Erforderliche Werte auswählen.
target_field_values_specify	*[Feld1 ... FeldN]*	
include_model_assessment	*boolesch*	
model_assessment_random_seed	*Zahl*	Muss eine reelle Zahl sein.
model_assessment_sample_size	*Zahl*	Muss eine reelle Zahl sein.
model_assessment_iterations	*Zahl*	Anzahl der Iterationen.
display_model_evaluation	*boolesch*	
max_predictions	*Zahl*	
randomization	*Zahl*	
scoring_random_seed	*Zahl*	
sort	Ascending Descending	Gibt an, ob die Angebote mit den höchsten oder die mit den niedrigsten Scores zuerst angezeigt werden.
model_reliability	*boolesch*	
calculate_variable_importance	*boolesch*	

Eigenschaften des Knotens "statisticsmodel"

Mithilfe des Statistics-Modellknotens können Sie Ihre Daten analysieren und bearbeiten, indem Sie IBM SPSS Statistics-Prozeduren ausführen, die PMML erzeugen. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften des Knotens "statisticsmodel"" auf Seite 246.

Eigenschaften des Knotens "svm"

Der Knoten "Support Vector Machine" (SVM) ermöglicht die Klassifizierung von Daten in eine von zwei Gruppen ohne Überanpassung. SVM eignet sich gut für umfangreiche Datasets, beispielsweise solche mit einer großen Anzahl an Eingabefeldern.

Tabelle 113. Eigenschaften des Knotens "svm".

Eigenschaften des Knotens svm	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
stopping_criteria	1,0E-1 1,0E-2 1.0E-3 (Standard) 1,0E-4 1,0E-5 1,0E-6	Bestimmt, wann der Optimierungsalgorithmus gestoppt werden soll.
regularization	*Zahl*	Auch als C-Parameter bekannt.
precision	*Zahl*	Nur verwendet, wenn es sich beim Messniveau des Zielfelds um Continuous (Stetig) handelt.
kernel	RBF(Standard) Polynomial Sigmoid Linear	Typ der für die Transformation verwendeten Kernfunktion.
rbf_gamma	*Zahl*	Nur verwendet, wenn für kernel der Typ RBF gilt.
gamma	*Zahl*	Nur verwendet, wenn für kernel der TypPolynomial oder Sigmoid gilt.
bias	*Zahl*	
degree	*Zahl*	Nur verwendet, wenn für kernel der Typ Polynomial gilt.
calculate_variable_importance	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	
adjusted_propensity_partition	Test Validation	

Eigenschaften des Knotens "timeseries"

Der Zeitreihenknoten berechnet Schätzungen für die exponenielle Glättung sowie univariate und multivariate ARIMA-Modelle (ARIMA steht für Autoregressive Integrated Moving Average (autoregressiver integrierter gleitender Durchschnitt)) für Zeitreihendaten und erstellt Vorhersagen über die zukünftige Leistung. Einem Zeitreihenknoten muss stets ein Zeitintervallknoten vorangehen.

Tabelle 114. Eigenschaften des Knotens "timeseries".

Eigenschaften des Knotens `timeseries`	Werte	Eigenschaftsbeschreibung
`targets`	*Feld*	Der Zeitreihenknoten sagt eines oder mehrere Ziele voraus; optional können dabei ein oder mehrere Eingabefelder als Prädiktoren verwendet werden. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
`continue`	*boolesch*	

`method`	`ExpertModeler`
	`Exsmooth`
	`Arima`
	`Reuse`	

`expert_modeler_method`	*boolesch*	
`consider_seasonal`	*boolesch*	
`detect_outliers`	*boolesch*	
`expert_outlier_additive`	*boolesch*	
`expert_outlier_level_shift`	*boolesch*	
`expert_outlier_innovational`	*boolesch*	
`expert_outlier_level_shift`	*boolesch*	
`expert_outlier_transient`	*boolesch*	
`expert_outlier_seasonal_additive`	*boolesch*	
`expert_outlier_local_trend`	*boolesch*	
`expert_outlier_additive_patch`	*boolesch*	

`exsmooth_model_type`	`Simple`
	`HoltsLinearTrend`
	`BrownsLinearTrend`
	`DampedTrend`
	`SimpleSeasonal`
	`WintersAdditive`
	`WintersMultiplicative`	

`exsmooth_transformation_type`	`None`
	`SquareRoot`
	`NaturalLog`	

`arima_p`	*ganze Zahl*	

Tabelle 114. Eigenschaften des Knotens "timeseries" (Forts.).

Eigenschaften des Knotens timeseries	Werte	Eigenschaftsbeschreibung
arima_d	*ganze Zahl*	
arima_q	*ganze Zahl*	
arima_sp	*ganze Zahl*	
arima_sd	*ganze Zahl*	
arima_sq	*ganze Zahl*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*boolesch*	
tf_arima_p. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_d. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_q. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_sp. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_sd. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_sq. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_delay. *Feldname*	*ganze Zahl*	Für Transferfunktionen.
tf_arima_transformation_type. *Feldname*	None SquareRoot NaturalLog	Für Transferfunktionen.
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*boolesch*	
arima_outlier_level_shift	*boolesch*	
arima_outlier_innovational	*boolesch*	
arima_outlier_transient	*boolesch*	
arima_outlier_seasonal_additive	*boolesch*	
arima_outlier_local_trend	*boolesch*	
arima_outlier_additive_patch	*boolesch*	
conf_limit_pct	*real*	
max_lags	*ganze Zahl*	
events	*Felder*	
scoring_model_only	*boolesch*	Für Modelle mit sehr großen Zahlen (Zehntausende) von Zeitreihen.

Eigenschaften des Knotens "twostep"

Der TwoStep-Knoten verwendet eine aus zwei Schritten bestehende Clusterbildungsmethode. Im ersten Schritt wird ein einzelner Durchlauf durch die Daten vorgenommen, bei dem die Eingangsrohdaten zu einem verwaltbaren Set von Subclustern komprimiert werden. Im zweiten Schritt werden die Subcluster mithilfe einer hierarchischen Methode zur Clusterbildung nach und nach in immer größere Cluster zusammengeführt. TwoStep hat den Vorteil, dass die optimale Anzahl an Clustern für die Trainingsdaten automatisch geschätzt wird. Mit dem Verfahren können gemischte Feldtypen und große Datasets effizient verarbeitet werden.

Tabelle 115. Eigenschaften des Knotens "twostep".

Eigenschaften des Knotens **twostep**	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	TwoStep-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtungs- und Häufigkeitsfelder werden nicht erkannt. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten" auf Seite 133.
standardize	*boolesch*	
exclude_outliers	*boolesch*	
percentage	*Zahl*	
cluster_num_auto	*boolesch*	
min_num_clusters	*Zahl*	
max_num_clusters	*Zahl*	
num_clusters	*Zahl*	
cluster_label	Zeichenfolge Zahl	
label_prefix	*Zeichenfolge*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Kapitel 14. Eigenschaften von Modellnuggetknoten

Modellnuggetknoten besitzen dieselben allgemeinen Eigenschaften wie andere Knoten. Weitere Informationen finden Sie im Thema „Allgemeine Knoteneigenschaften" auf Seite 60.

Eigenschaften des Knotens "applyanomalydetection"

Mithilfe von Modellierungsknoten vom Typ "Anomalieerkennung" kann ein Modellnugget vom Typ "Anomalieerkennung" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyanomalydetection*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "anomalydetection"" auf Seite 133.

Tabelle 116. Eigenschaften des Knotens "applyanomalydetection".

Eigenschaften des Knotens `applyanomalydetection`	**Werte**	**Eigenschaftsbeschreibung**

`anomaly_score_method`	`FlagAndScore`
	`FlagOnly`
	`ScoreOnly`
		Bestimmt, welche Ausgaben für das Scoring erstellt werden.

`num_fields`	*ganze Zahl*	Zu meldende Felder.
`discard_records`	*boolesch*	Gibt an, ob Datensätze aus der Ausgabe verworfen werden sollen oder nicht.
`discard_anomalous_records`	*boolesch*	Gibt an, ob die anomalen oder die *nicht* anomalen Datensätze verworfen werden sollen. Der Standardwert ist `off`, was bedeutet, dass die *nicht* anomalen Datensätze verworfen werden. Bei `on` werden die anomalen Datensätze verworfen. Diese Eigenschaft ist nur aktiviert, wenn die Eigenschaft `discard_records` aktiviert ist.

Eigenschaften des Knotens "applyapriori"

Mithilfe von Apriori-Modellierungsknoten kann ein Modellnugget vom Typ "Apriori" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyapriori*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "apriori"" auf Seite 135.

Tabelle 117. Eigenschaften des Knotens "applyapriori".

Eigenschaften des Knotens `applyapriori`	**Werte**	**Eigenschaftsbeschreibung**
`max_predictions`	*Anzahl (ganze Zahl)*	
`ignore_unmatached`	*boolesch*	
`allow_repeats`	*boolesch*	

`check_basket`	`NoPredictions`
	`Predictions`
	`NoCheck`	

`criterion`	`Confidence`
	`Support`
	`RuleSupport`
	`Lift`
	`Deployability`	

Eigenschaften des Knotens "applyautoclassifier"

Mithilfe von Modellierungsknoten des Typs "Automatisches Klassifikationsmerkmal" kann ein Modellnugget vom Typ "Automatisches Klassifikationsmerkmal" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyautoclassifier*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "autoclassifier"" auf Seite 136.

Tabelle 118. Eigenschaften des Knotens "applyautoclassifier".

Eigenschaften des Knotens **applyautoclassifier**	Werte	Eigenschaftsbeschreibung
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Setfeld ist.
set_voting_tie_selection	Random HighestConfidence	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.

Eigenschaften des Knotens "applyautocluster"

Mithilfe von Modellierungsknoten des Typs "Autom. Cluster" kann ein Modellnugget vom Typ "Autom. Cluster" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyautocluster*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "autocluster"" auf Seite 138.

Eigenschaften des Knotens "applyautonumeric"

Mithilfe von Modellierungsknoten des Typs "Auto-Numerisch" kann ein Modellnugget vom Typ "Auto-Numerisch" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyautonumeric*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "autonumeric"" auf Seite 139.

Tabelle 119. Eigenschaften des Knotens "applyautonumeric".

Eigenschaften des Knotens **applyautonumeric**	Werte	Eigenschaftsbeschreibung
calculate_standard_error	*boolesch*	

Eigenschaften des Knotens "applybayesnet"

Mithilfe von Bayes-Netzmodellierungsknoten kann ein Modellnugget vom Typ "Bayes-Netz" generiert werden. Der Scriptname dieses Modellnuggets lautet *applybayesnet*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "bayesnet"" auf Seite 140.

Tabelle 120. Eigenschaften des Knotens "applybayesnet".

Eigenschaften des Knotens **applybayesnet**	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
raw_propensity	*boolesch*	
adjusted_propensity	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applyc50"

Mithilfe von C5.0-Modellierungsknoten kann ein C5.0-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyc50*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "c50"" auf Seite 142.

Tabelle 121. Eigenschaften des Knotens "applyc50".

Eigenschaften des Knotens **applyc50**	Werte	Eigenschaftsbeschreibung
sql_generate	Never NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen des Regelsets.
calculate_conf	*boolesch*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applycarma"

Mithilfe von CARMA-Modellierungsknoten kann ein CARMA-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applycarma*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "carma"" auf Seite 143.

Eigenschaften des Knotens "applycart"

Mithilfe von Modellierungsknoten vom Typ "C&R-Baum" kann ein Modellnugget vom Typ "C&R-Baum" generiert werden. Der Scriptname dieses Modellnuggets lautet *applycart*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "cart"" auf Seite 144.

Tabelle 122. Eigenschaften des Knotens "applycart".

Eigenschaften des Knotens **applycart**	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen des Regelsets.
calculate_conf	*boolesch*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
display_rule_id	*boolesch*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applychaid"

Mithilfe von CHAID-Modellierungsknoten kann ein CHAID-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applychaid*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "chaid"" auf Seite 146.

Tabelle 123. Eigenschaften des Knotens "applychaid".

Eigenschaften des Knotens **applychaid**	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues	
calculate_conf	*boolesch*	
display_rule_id	*boolesch*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applycoxreg"

Mithilfe von Cox-Modellierungsknoten kann ein Cox-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applycoxreg*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "coxreg"" auf Seite 147.

Tabelle 124. Eigenschaften des Knotens "applycoxreg".

Eigenschaften des Knotens **applycoxreg**	Werte	Eigenschaftsbeschreibung
future_time_as	Intervals Fields	
time_interval	*Zahl*	
num_future_times	*ganze Zahl*	
time_field	*Feld*	
past_survival_time	*Feld*	

Tabelle 124. Eigenschaften des Knotens "applycoxreg" (Forts.).

Eigenschaften des Knotens **applycoxreg**	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
cumulative_hazard	*boolesch*	

Eigenschaften des Knotens "applydecisionlist"

Mithilfe von Entscheidungslistenmodellierungsknoten kann ein Modellnugget vom Typ "Entscheidungsliste" generiert werden. Der Scriptname dieses Modellnuggets lautet *applydecisionlist*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "decisionlist"" auf Seite 149.

Tabelle 125. Eigenschaften des Knotens "applydecisionlist".

Eigenschaften des Knotens **applydecisionlist**	Werte	Eigenschaftsbeschreibung
enable_sql_generation	*boolesch*	Wenn dieser Wert wahr ist, versucht IBM SPSS Modeler, das Entscheidungslistenmodell über einen Pushback-Vorgang an SQL zurückzuführen.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applydiscriminant"

Mithilfe von Diskriminanzmodellierungsknoten kann ein Diskriminanzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applydiscriminant*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "discriminant"" auf Seite 150.

Tabelle 126. Eigenschaften des Knotens "applydiscriminant".

Eigenschaften des Knotens **applydiscriminant**	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applyfactor"

Mithilfe von Faktormodellierungsknoten kann ein Faktormodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyfactor*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "factor"" auf Seite 152.

Eigenschaften des Knotens "applyfeatureselection"

Mithilfe von Modellierungsknoten vom Typ "Merkmalauswahl" kann ein Modellnugget vom Typ "Merkmalauswahl" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyfeatureselection*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "featureselection"" auf Seite 153.

Tabelle 127. Eigenschaften des Knotens "applyfeatureselection".

Eigenschaften des Knotens **applyfeatureselection**	Werte	Eigenschaftsbeschreibung
selected_ranked_fields		Gibt an, welche Felder mit Rangzahlen im Modell-Browser markiert werden.
selected_screened_fields		Gibt an, welche im Screening untersuchten Felder im Modell-Browser markiert werden.

Eigenschaften des Knotens "applygeneralizedlinear"

Mithilfe von Modellierungsknoten vom Typ "Verallgemeinert linear (genlin)" kann ein Modellnugget vom Typ "Verallgemeinert linear" generiert werden. Der Scriptname dieses Modellnuggets lautet *applygeneralizedlinear*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "genlin"" auf Seite 155.

Tabelle 128. Eigenschaften des Knotens "applygeneralizedlinear".

Eigenschaften des Knotens **applygeneralizedlinear**	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applyglmm"

Mithilfe von GLMM-Modellierungsknoten kann ein GLMM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyglmm*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "glmm"" auf Seite 158.

Tabelle 129. Eigenschaften des Knotens "applyglmm".

Eigenschaften des Knotens **applyglmm**	Werte	Eigenschaftsbeschreibung
confidence	onProbability onIncrease	Grundlage für die Berechnung des Konfidenzwerts für das Scoring: höchste vorhergesagte Wahrscheinlichkeit oder Differenz zwischen der höchsten und zweithöchsten vorhergesagten Wahrscheinlichkeit.
score_category_probabilities	*boolesch*	Auf True gesetzt, werden die vorhergesagten Wahrscheinlichkeiten für kategoriale Ziele generiert. Für jede Kategorie wird ein Feld erstellt. Die Standardeinstellung ist False.
max_categories	*ganze Zahl*	Maximal zu erstellende Anzahl an Kategorien, für die Wahrscheinlichkeiten vorhergesagt werden sollen. Wird nur verwendet, wenn score_category_probabilities auf True gesetzt ist.

Tabelle 129. Eigenschaften des Knotens "applyglmm" (Forts.).

Eigenschaften des Knotens **applyglmm**	Werte	Eigenschaftsbeschreibung
score_propensity	*boolesch*	Auf True gesetzt, werden Raw-Propensity-Scores (die die Wahrscheinlichkeit für "True" angeben) für Modelle mit Flagzielen erzeugt. Bei aktiven Partitionen werden außerdem Adjusted-Propensitys-Scores anhand der Testpartition erzeugt. Die Standardeinstellung ist False.

Eigenschaften des Knotens "applykmeans"

Mithilfe von K-Means-Modellierungsknoten kann ein K-Means-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applykmeans*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "kmeans"" auf Seite 162.

Eigenschaften des Knotens "applyknn"

Mithilfe von KNN-Modellierungsknoten kann ein KNN-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyknn*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "knn"" auf Seite 163.

Tabelle 130. Eigenschaften des Knotens "applyknn".

Eigenschaften des Knotens **applyknn**	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
save_distances	*boolesch*	

Eigenschaften des Knotens "applykohonen"

Mithilfe von Kohonen-Modellierungsknoten kann ein Kohonen-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applykohonen*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "kohonen"" auf Seite 164.

Eigenschaften des Knotens "applylinear"

Mithilfe von Cox-Modellierungsknoten kann ein Nugget für ein lineares Modell generiert werden. Der Scriptname dieses Modellnuggets lautet *applylinear*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "linear"" auf Seite 165.

Tabelle 131. Eigenschaften des Knotens "applylinear".

Eigenschaften des Knotens **applylinear**	Werte	Eigenschaftsbeschreibung
use_custom_name	*boolesch*	
custom_name	*Zeichenfolge*	
enable_sql_generation	*boolesch*	

Eigenschaften des Knotens "applylogreg"

Mithilfe von Modellierungsknoten vom Typ "Logistische Regression" kann ein Modellnugget vom Typ "Logistische Regression" generiert werden. Der Scriptname dieses Modellnuggets lautet *applylogreg*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "logreg"" auf Seite 166.

Tabelle 132. Eigenschaften des Knotens "applylogreg".

Eigenschaften des Knotens `applylogreg`	Werte	Eigenschaftsbeschreibung
`calculate_raw_propensities`	*boolesch*	
`calculate_conf`	*boolesch*	
`enable_sql_generation`	*boolesch*	

Eigenschaften des Knotens "applyneuralnet"

Mithilfe von Netzmodellierungsknoten kann ein Netzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyneuralnet*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "neuralnet"" auf Seite 169.

Vorsicht: In dieser Version ist eine neuere Fassung des Netz-Nuggets mit erweiterten Funktionen verfügbar, die im nächsten Abschnitt beschrieben wird (*applyneuralnetwork*). Die Vorgängerversion ist zwar weiterhin verfügbar, wir empfehlen jedoch die Aktualisierung Ihrer Scripts zur Verwendung der neuen Version. Zur Referenz sind hier Details zur Vorgängerversion enthalten, doch wird die Unterstützung dafür in zukünftigen Versionen wegfallen.

Tabelle 133. Eigenschaften des Knotens "applyneuralnet".

Eigenschaften des Knotens `applyneuralnet`	Werte	Eigenschaftsbeschreibung
`calculate_conf`	*boolesch*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
`enable_sql_generation`	*boolesch*	
`nn_score_method`	`Difference` `SoftMax`	
`calculate_raw_propensities`	*boolesch*	
`calculate_adjusted_propensities`	*boolesch*	

Eigenschaften des Knotens "applyneuralnetwork"

Mithilfe von Netzmodellierungsknoten kann ein Netzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyneuralnetwork*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "neuralnetwork"" auf Seite 171.

Tabelle 134. Eigenschaften des Knotens "applyneuralnetwork".

Eigenschaften des Knotens `applyneuralnetwork`	Werte	Eigenschaftsbeschreibung
`use_custom_name`	*boolesch*	
`custom_name`	*Zeichenfolge*	
`confidence`	`onProbability` `onIncrease`	

Tabelle 134. Eigenschaften des Knotens "applyneuralnetwork" (Forts.).

Eigenschaften des Knotens applyneuralnetwork	**Werte**	**Eigenschaftsbeschreibung**
score_category_probabilities	*boolesch*	
max_categories	*Zahl*	
score_propensity	*boolesch*	

Eigenschaften des Knotens "applyquest"

Mithilfe von QUEST-Modellierungsknoten kann ein QUEST-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyquest*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "quest"" auf Seite 172.

Tabelle 135. Eigenschaften des Knotens "applyquest".

Eigenschaften des Knotens applyquest	**Werte**	**Eigenschaftsbeschreibung**
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	*boolesch*	
display_rule_id	*boolesch*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applyregression"

Mithilfe von Modellierungsknoten vom Typ "Lineare Regression" kann ein Modellnugget vom Typ "Lineare Regression" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyregression*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "regression"" auf Seite 174.

Eigenschaften des Knotens "applyr"

Mithilfe von R-Erstellungsknoten kann ein R-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyr*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "buildr"" auf Seite 142.

Tabelle 136. Eigenschaften des Knotens "applyr"

Eigenschaften des Knotens applyr	**Werte**	**Eigenschaftsbeschreibung**
score_syntax	*Zeichenfolge*	R-Scriptsyntax für das Modellscoring.
convert_flags	StringsAndDoubles LogicalValues	Option zum Konvertieren von Flagfeldern.
convert_datetime	*boolesch*	Option zum Konvertieren von Variablen mit Datums- oder Datums-/Zeitformaten in R-Datums-/Zeitformate.

Tabelle 136. Eigenschaften des Knotens "applyr" (Forts.)

Eigenschaften des Knotens applyr	Werte	Eigenschaftsbeschreibung
convert_datetime_class	POSIXct POSIXlt	Optionen, die angeben, in welches Format Variablen mit Datums- oder Datums-/Zeitformaten konvertiert werden.
convert_missing	*boolesch*	Option zum Konvertieren fehlender Werte in R-Werte "NA".

Eigenschaften des Knotens "applyselflearning"

Mithilfe von Modellierungsknoten vom Typ (Self-Learning Response Model (SLRM) kann ein SLRM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyselflearning*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "slrm"" auf Seite 176.

Tabelle 137. Eigenschaften des Knotens "applyselflearning".

Eigenschaften des Knotens applyselflearning	Werte	Eigenschaftsbeschreibung
max_predictions	*Zahl*	
randomization	*Zahl*	
scoring_random_seed	*Zahl*	
sort	ascending descending	Gibt an, ob die Angebote mit den höchsten oder die mit den niedrigsten Scores zuerst angezeigt werden.
model_reliability	*boolesch*	Berücksichtigt die Option für die Reliabilität auf der Registerkarte "Einstellungen".

Eigenschaften des Knotens "applysequence"

Mithilfe von Sequenzmodellierungsknoten kann ein Sequenzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applysequence*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "sequence"" auf Seite 175.

Eigenschaften des Knotens "applysvm"

Mithilfe von SVM-Modellierungsknoten kann ein SVM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applysvm*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "svm"" auf Seite 177.

Tabelle 138. Eigenschaften des Knotens "applysvm".

Eigenschaften des Knotens applysvm	Werte	Eigenschaftsbeschreibung
all_probabilities	*boolesch*	
calculate_raw_propensities	*boolesch*	
calculate_adjusted_propensities	*boolesch*	

Eigenschaften des Knotens "applytimeseries"

Mithilfe von Zeitreihenmodellierungsknoten kann ein Zeitreihenmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applytimeseries*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "timeseries"" auf Seite 178.

Tabelle 139. Eigenschaften des Knotens "applytimeseries".

Eigenschaften des Knotens `applytimeseries`	**Werte**	**Eigenschaftsbeschreibung**
`calculate_conf`	*boolesch*	
`calculate_residuals`	*boolesch*	

Eigenschaften des Knotens "applytwostep"

Mithilfe von TwoStep-Modellierungsknoten kann ein TwoStep-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applytwostep*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie im Thema „Eigenschaften des Knotens "twostep"" auf Seite 179.

Kapitel 15. Eigenschaften von Datenbankmodellierungsknoten

IBM SPSS Modeler unterstützt die Integration mit Data-Mining-Tools und Datenmodellierungstools von Datenbankanbietern, z. B. Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM DB2 InfoSphere Warehouse und IBM Netezza Analytics. Sie können mithilfe von datenbankeigenen Algorithmen Modelle erstellen und scoren, ohne dazu die IBM SPSS Modeler-Anwendung verlassen zu müssen. Datenbankmodelle können außerdem mithilfe von Scripterstellung unter Verwendung der in diesem Abschnitt beschriebenen Eigenschaften erstellt und bearbeitet werden.

Knoteneigenschaften für Microsoft-Modellierung

Eigenschaften von Microsoft-Modellierungsknoten

Allgemeine Eigenschaften

Folgende Eigenschaften haben alle Microsoft-Datenbankmodellierungsknoten gemeinsam.

Tabelle 140. Allgemeine Eigenschaften von Microsoft-Knoten.

Allgemeine Eigenschaften von Microsoft-Knoten	**Werte**	**Eigenschaftsbeschreibung**
`analysis_database_name`	*Zeichenfolge*	Name der Analysis Services-Datenbank.
`analysis_server_name`	*Zeichenfolge*	Name des Analysis Services-Hosts.
`use_transactional_data`	*boolesch*	Gibt an, ob die Eingabedaten in Tabellen- oder Transaktionsformat vorliegen.
`inputs`	*[Feld Feld Feld]*	Eingabefelder für Tabellendaten.
`target`	*Feld*	Vorhergesagtes Feld (gilt nicht für MS-Clustering- oder Sequenz-Clustering-Knoten).
`unique_field`	*Feld*	Schlüsselfeld.
`msas_parameters`	*strukturiert*	Algorithmusparameter. Weitere Informationen finden Sie im Thema „Algorithmusparameter" auf Seite 194.
`with_drillthrough`	*boolesch*	Mit Drillthrough-Option.

MS-Entscheidungsbaum

Für Knoten vom Typ `mstree` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Clustering

Für Knoten vom Typ `mscluster` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Assoziationsregeln

Die folgenden Eigenschaften sind für Knoten des Typs `msassoc` verfügbar:

Tabelle 141. Eigenschaften des Knotens "msassoc".

Eigenschaften des Knotens `msassoc`	**Werte**	**Eigenschaftsbeschreibung**
`id_field`	*Feld*	Identifiziert jede Transaktion in den Daten.

Tabelle 141. Eigenschaften des Knotens "msassoc" (Forts.).

Eigenschaften des Knotens msassoc	Werte	Eigenschaftsbeschreibung
trans_inputs	*[Feld Feld Feld]*	Eingabefelder für Transaktionsdaten.
transactional_target	*Feld*	Prädiktorfeld (Transaktionsdaten).

MS Naive Bayes

Für Knoten vom Typ msbayes sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Lineare Regression

Für Knoten vom Typ msregression sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Neuronales Netz

Für Knoten vom Typ msneuralnetwork sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Logistische Regression

Für Knoten vom Typ mslogistic sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS Time Series

Für Knoten vom Typ mstimeseries sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Sequenzclustering

Die folgenden speziellen Eigenschaften sind für Knoten des Typs mssequencecluster verfügbar:

Tabelle 142. Eigenschaften des Knotens "mssequencecluster".

Eigenschaften des Knotens mssequencecluster	Werte	Eigenschaftsbeschreibung
id_field	*Feld*	Identifiziert jede Transaktion in den Daten.
input_fields	*[Feld Feld Feld]*	Eingabefelder für Transaktionsdaten.
sequence_field	*Feld*	Sequenz-ID.
target_field	*Feld*	Prädiktorfeld (Tabellendaten).

Algorithmusparameter

Jeder Microsoft-Datenbankmodelltyp weist spezifische Parameter auf, die mithilfe der Eigenschaft msas_parameters festgelegt werden können.

Diese Parameter werden vom SQL-Server abgeleitet. Gehen Sie wie folgt vor, um die relevanten Parameter für die einzelnen Knoten anzuzeigen:

1. Platzieren Sie einen Datenbankquellenknoten im Erstellungsbereich.
2. Öffnen Sie den Datenbankquellenknoten.
3. Wählen Sie eine gültige Quelle in der Dropdown-Liste **Datenquelle** aus.

4. Wählen Sie eine gültige Tabelle in der Liste **Tabellenname** aus.
5. Klicken Sie auf **OK**, um den Datenbankquellenknoten zu schließen.
6. Fügen Sie den Microsoft-Datenbankmodellierungsknoten ein, dessen Eigenschaften aufgelistet werden sollen.
7. Öffnen Sie den Datenbankmodellierungsknoten.
8. Wählen Sie die Registerkarte **Experten**.

Die verfügbaren `msas_parameters`-Eigenschaften für diesen Knoten werden angezeigt.

Eigenschaften von Microsoft-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der Microsoft-Datenbankmodellierungsknoten erstellt wurden.

MS-Entscheidungsbaum

Tabelle 143. Eigenschaften des MS-Entscheidungsbaums.

Eigenschaften des Knotens `applymstree`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.
`datasource`	*Zeichenfolge*	Name der SQL Server ODBC-Datenquelle (DSN).
`sql_generate`	*boolesch*	Aktiviert die SQL-Erzeugung.

MS - Lineare Regression

Tabelle 144. MS Lineare Regression - Eigenschaften.

Eigenschaften des Knotens `applymsregression`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.

MS - Neuronales Netz

Tabelle 145. MS Neuronales Netz - Eigenschaften.

Eigenschaften des Knotens `applymsneuralnetwork`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.

MS - Logistische Regression

Tabelle 146. MS Logistische Regression - Eigenschaften.

Eigenschaften des Knotens `applymslogistic`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.

MS Time Series

Tabelle 147. MS Time Series-Eigenschaften.

Eigenschaften des Knotens `applymstimeseries`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.
`start_from`	`new_prediction` `historical_prediction`	Gibt an, ob Zukunfts- oder historische Vorhersagen getroffen werden.
`new_step`	*Zahl*	Definiert die Startzeit für Zukunftsvorhersagen.
`historical_step`	*Zahl*	Definiert die Startzeit für historische Vorhersagen.
`end_step`	*Zahl*	Definiert die Endzeit für Vorhersagen.

MS-Sequenzclustering

Tabelle 148. MS-Sequenz-Clustering-Eigenschaften.

Eigenschaften des Knotens `applymssequencecluster`	**Werte**	**Beschreibung**
`analysis_database_name`	*Zeichenfolge*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
`analysis_server_name`	*Zeichenfolge*	Name des Analyseserver-Hosts.

Knoteneigenschaften für Oracle-Modellierung

Eigenschaften von Oracle-Modellierungsknoten

Folgende Eigenschaften haben alle Oracle-Datenbankmodellierungsknoten gemeinsam.

Tabelle 149. Allgemeine Eigenschaften von Oracle-Knoten.

Allgemeine Eigenschaften von Oracle-Knoten	Werte	Eigenschaftsbeschreibung
target	*Feld*	
inputs	*Liste mit Feldern*	
partition	*Feld*	Feld wird verwendet, um die Daten in getrennte Stichproben für die Trainings-, Test- und Validierungsphase der Modellbildung aufzuteilen.
datasource		
username		
password		
epassword		
use_model_name	*boolesch*	
model_name	*Zeichenfolge*	Benutzerdefinierter Name für neues Modell.
use_partitioned_data	*boolesch*	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
unique_field	*Feld*	
auto_data_prep	*boolesch*	Aktiviert oder inaktiviert die automatische Datenvorbereitungsfunktion von Oracle (nur 11g-Datenbanken).
costs	*strukturiert*	Strukturierte Eigenschaft.
mode	Simple Expert	Wenn diese Einstellung auf Simple (Einfach) gesetzt ist, werden bestimmte Eigenschaften ignoriert (vgl. die Eigenschaften der einzelnen Knoten).
use_prediction_probability	*boolesch*	
prediction_probability	*Zeichenfolge*	
use_prediction_set	*boolesch*	

Oracle Naive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs oranb verfügbar.

Tabelle 150. Eigenschaften des Knotens "oranb".

Eigenschaften des Knotens oranb	Werte	Eigenschaftsbeschreibung
singleton_threshold	*Zahl*	0,0-1,0.*
pairwise_threshold	*Zahl*	0,0-1,0.*
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft.

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Adaptive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs oraabn verfügbar.

Tabelle 151. Eigenschaften des Knotens "oraabn".

Eigenschaften des Knotens oraabn	Werte	Eigenschaftsbeschreibung
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*boolesch*	*
execution_time_limit	*ganze Zahl*	Der Wert muss größer als 0 sein.*
max_naive_bayes_predictors	*ganze Zahl*	Der Wert muss größer als 0 sein.*
max_predictors	*ganze Zahl*	Der Wert muss größer als 0 sein.*
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft.

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Support Vector Machines

Die folgenden Eigenschaften sind für Knoten des Typs orasvm verfügbar.

Tabelle 152. Eigenschaften des Knotens "orasvm".

Eigenschaften des Knotens orasvm	Werte	Eigenschaftsbeschreibung
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*ganze Zahl*	Nur gaußscher Kern. Der Wert muss größer als 0 sein.*
convergence_tolerance	*Zahl*	Der Wert muss größer als 0 sein.*
use_standard_deviation	*boolesch*	Nur gaußscher Kern.*
standard_deviation	*Zahl*	Der Wert muss größer als 0 sein.*
use_epsilon	*boolesch*	Nur Regressionsmodelle.*
epsilon	*Zahl*	Der Wert muss größer als 0 sein.*
use_complexity_factor	*boolesch*	*
complexity_factor	*Zahl*	*
use_outlier_rate	*boolesch*	Nur Ein-Klassen-Variante.*
outlier_rate	*Zahl*	Nur Ein-Klassen-Variante. 0,0-1,0.*

Tabelle 152. Eigenschaften des Knotens "orasvm" (Forts.).

Eigenschaften des Knotens **orasvm**	Werte	Eigenschaftsbeschreibung
weights	Data Equal Custom	
custom_weights	*strukturiert*	Strukturierte Eigenschaft.

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Verallgemeinerte lineare Modelle von Oracle

Die folgenden Eigenschaften sind für Knoten des Typs oraglm verfügbar.

Tabelle 153. Eigenschaften des Knotens "oraglm".

Eigenschaften des Knotens **oraglm**	Werte	Eigenschaftsbeschreibung
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*boolesch*	*
row_weights_field	*Feld*	*
save_row_diagnostics	*boolesch*	*
row_diagnostics_table	*Zeichenfolge*	*
coefficient_confidence	*Zahl*	*
use_reference_category	*boolesch*	*
reference_category	*Zeichenfolge*	*
ridge_regression	Auto Off On	*
parameter_value	*Zahl*	*
vif_for_ridge	*boolesch*	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Decision Tree

Die folgenden Eigenschaften sind für Knoten des Typs

Tabelle 154. Eigenschaften des Knotens "oradecisiontree".

Eigenschaften des Knotens **oradecisiontree**	Werte	Eigenschaftsbeschreibung
use_costs	*boolesch*	
impurity_metric	Entropy Gini	
term_max_depth	*ganze Zahl*	2-20.*
term_minpct_node	*Zahl*	0,0-10,0.*
term_minpct_split	*Zahl*	0,0-20,0.*

Tabelle 154. Eigenschaften des Knotens "oradecisiontree" (Forts.).

Eigenschaften des Knotens `oradecisiontree`	**Werte**	**Eigenschaftsbeschreibung**
`term_minrec_node`	*ganze Zahl*	Der Wert muss größer als 0 sein.*
`term_minrec_split`	*ganze Zahl*	Der Wert muss größer als 0 sein.*
`display_rule_ids`	*boolesch*	*

* Eigenschaft wird ignoriert, wenn `mode` auf `Simple` gesetzt ist.

Oracle O-Cluster

Die folgenden Eigenschaften sind für Knoten des Typs `oraocluster` verfügbar.

Tabelle 155. Eigenschaften des Knotens "oraocluster".

Eigenschaften des Knotens `oraocluster`	**Werte**	**Eigenschaftsbeschreibung**
`max_num_clusters`	*ganze Zahl*	Der Wert muss größer als 0 sein.
`max_buffer`	*ganze Zahl*	Der Wert muss größer als 0 sein.*
`sensitivity`	*Zahl*	0,0-1,0.*

* Eigenschaft wird ignoriert, wenn `mode` auf `Simple` gesetzt ist.

Oracle KMeans

Die folgenden Eigenschaften sind für Knoten des Typs `orakmeans` verfügbar.

Tabelle 156. Eigenschaften des Knotens "orakmeans".

Eigenschaften des Knotens `orakmeans`	**Werte**	**Eigenschaftsbeschreibung**
`num_clusters`	*ganze Zahl*	Der Wert muss größer als 0 sein.

`normalization_method`	`zscore`
	`minmax`
	`none`	

`distance_function`	`Euclidean` `Cosine`	
`iterations`	*ganze Zahl*	0-20.*
`conv_tolerance`	*Zahl*	0,0-0,5.*
`split_criterion`	`Variance` `Size`	Die Standardeinstellung ist "Variance".
`num_bins`	*ganze Zahl*	Der Wert muss größer als 0 sein.*
`block_growth`	*ganze Zahl*	1-5.*
`min_pct_attr_support`	*Zahl*	0,0-1,0.*

* Eigenschaft wird ignoriert, wenn `mode` auf `Simple` gesetzt ist.

Oracle NMF

Die folgenden Eigenschaften sind für Knoten des Typs `oranmf` verfügbar.

Tabelle 157. Eigenschaften des Knotens "oranmf".

Eigenschaften des Knotens **oranmf**	Werte	Eigenschaftsbeschreibung
normalization_method	minmax none	
use_num_features	*boolesch*	*
num_features	*ganze Zahl*	0-1. Der Standardwert wird vom Algorithmus durch Schätzung aus den Daten ermittelt.*
random_seed	*Zahl*	*
num_iterations	*ganze Zahl*	0-500.*
conv_tolerance	*Zahl*	0,0-0,5.*
display_all_features	*boolesch*	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Apriori

Die folgenden Eigenschaften sind für Knoten des Typs oraapriori verfügbar.

Tabelle 158. Eigenschaften des Knotens "oraapriori".

Eigenschaften des Knotens **oraapriori**	Werte	Eigenschaftsbeschreibung
content_field	*Feld*	
id_field	*Feld*	
max_rule_length	*ganze Zahl*	2-20.
min_confidence	*Zahl*	0,0-1,0.
min_support	*Zahl*	0,0-1,0.
use_transactional_data	*boolesch*	

Oracle Minimum Description Length (MDL)

Für Knoten vom Typ oramdl sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Oracle-Eigenschaften am Anfang dieses Abschnitts.

Oracle Attribute Importance (AI)

Die folgenden Eigenschaften sind für Knoten des Typs oraai verfügbar.

Tabelle 159. Eigenschaften des Knotens "oraai".

Eigenschaften des Knotens **oraai**	Werte	Eigenschaftsbeschreibung
custom_fields	*boolesch*	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*boolesch*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.

Tabelle 159. Eigenschaften des Knotens "oraai" (Forts.).

Eigenschaften des Knotens oraai	Werte	Eigenschaftsbeschreibung
important_label	*Zeichenfolge*	Gibt die Beschriftung für die Rangstufe "bedeutsam" an.
select_marginal	*boolesch*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
marginal_label	*Zeichenfolge*	Gibt die Beschriftung für die Rangstufe "marginal" an.
important_above	*Zahl*	0,0-1,0.
select_unimportant	*boolesch*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob unbedeutende Felder ausgewählt werden sollen.
unimportant_label	*Zeichenfolge*	Gibt die Beschriftung für die Rangstufe "unbedeutsam" an.
unimportant_below	*Zahl*	0,0-1,0.
importance_value	*Zahl*	Wenn selection_mode auf ImportanceValue gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 100.
top_n	*Zahl*	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 1000.

Eigenschaften von Oracle-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der Oracle-Modelle erstellt wurden.

Oracle Naive Bayes

Für Knoten vom Typ applyoranb sind keine speziellen Eigenschaften definiert.

Oracle Adaptive Bayes

Für Knoten vom Typ applyoraabn sind keine speziellen Eigenschaften definiert.

Oracle Support Vector Machines

Für Knoten vom Typ applyorasvm sind keine speziellen Eigenschaften definiert.

Oracle Decision Tree

Die folgenden Eigenschaften sind für Knoten des Typs applyoradecisiontree verfügbar.

Tabelle 160. Eigenschaften des Knotens "applyoradecisiontree".

Eigenschaften des Knotens applyoradecisiontree	Werte	Eigenschaftsbeschreibung
use_costs	*boolesch*	
display_rule_ids	*boolesch*	

Oracle O-Cluster

Für Knoten vom Typ applyoraocluster sind keine speziellen Eigenschaften definiert.

Oracle KMeans

Für Knoten vom Typ `applyorakmeans` sind keine speziellen Eigenschaften definiert.

Oracle NMF

Die folgende Eigenschaften sind für Knoten des Typs `applyoranmf` verfügbar:

Tabelle 161. Eigenschaften des Knotens "applyoranmf".

Eigenschaften des Knotens `applyoranmf`	**Werte**	**Eigenschaftsbeschreibung**
`display_all_features`	*boolesch*	

Oracle Apriori

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Oracle MDL

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Knoteneigenschaften für IBM DB2-Modellierung

Eigenschaften von IBM DB2-Modellierungsknoten

Folgende Eigenschaften haben alle IBM InfoSphere Warehouse-Datenbankmodellierungsknoten (ISW-Datenbankmodellierungsknoten) gemeinsam:

Tabelle 162. Allgemeine Eigenschaften von ISW-Knoten.

Allgemeine Eigenschaften von ISW-Knoten	**Werte**	**Eigenschaftsbeschreibung**
`inputs`	*Liste mit Feldern*	
`datasource`		
`username`		
`password`		
`epassword`		
`enable_power_options`	*boolesch*	
`power_options_max_memory`	*ganze Zahl*	Der Wert muss größer als 32 sein.
`power_options_cmdline`	*Zeichenfolge*	
`mining_data_custom_sql`	*Zeichenfolge*	
`logical_data_custom_sql`	*Zeichenfolge*	
`mining_settings_custom_sql`		

ISW-Entscheidungsbaum

Die folgenden Eigenschaften sind für Knoten des Typs `db2imtree` verfügbar.

Tabelle 163. Eigenschaften des Knotens "db2imtree".

Eigenschaften des Knotens `db2imtree`	**Werte**	**Eigenschaftsbeschreibung**
`target`	*Feld*	

Tabelle 163. Eigenschaften des Knotens "db2imtree" (Forts.).

Eigenschaften des Knotens **db2imtree**	Werte	Eigenschaftsbeschreibung
perform_test_run	*boolesch*	
use_max_tree_depth	*boolesch*	
max_tree_depth	*ganze Zahl*	Der Wert muss größer als 0 sein.
use_maximum_purity	*boolesch*	
maximum_purity	*Zahl*	Eine Zahl zwischen 0 und 100.
use_minimum_internal_cases	*boolesch*	
minimum_internal_cases	*ganze Zahl*	Wert größer als 1.
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.

ISW-Assoziation

Die folgenden Eigenschaften sind für Knoten des Typs db2imassoc verfügbar.

Tabelle 164. Eigenschaften des Knotens "db2imassoc".

Eigenschaften des Knotens **db2imassoc**	Werte	Eigenschaftsbeschreibung
use_transactional_data	*boolesch*	
id_field	*Feld*	
content_field	*Feld*	
data_table_layout	basic limited_length	
max_rule_size	*ganze Zahl*	Der Wert muss größer als 2 sein.
min_rule_support	*Zahl*	0-100 %
min_rule_confidence	*Zahl*	0-100 %
use_item_constraints	*boolesch*	
item_constraints_type	Include Exclude	
use_taxonomy	*boolesch*	
taxonomy_table_name	*Zeichenfolge*	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	*Zeichenfolge*	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	*Zeichenfolge*	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	*boolesch*	Steuert, ob in IBM SPSS Modeler gespeicherte Taxonomieinformationen bei der Modellerstellung hochgeladen werden sollen. Die Taxonomietabelle wird verworfen, wenn sie bereits vorhanden ist. Die Taxonomieinformationen werden mit dem Modellknoten gespeichert und können über die Schaltflächen **Kategorien bearbeiten** und **Taxonomie bearbeiten** bearbeitet werden.

ISW-Sequenz

Die folgenden Eigenschaften sind für Knoten des Typs db2imsequence verfügbar.

Tabelle 165. Eigenschaften des Knotens "db2imsequence".

Eigenschaften des Knotens db2imsequence	**Werte**	**Eigenschaftsbeschreibung**
id_field	*Feld*	
group_field	*Feld*	
content_field	*Feld*	
max_rule_size	*ganze Zahl*	Der Wert muss größer als 2 sein.
min_rule_support	*Zahl*	0-100 %
min_rule_confidence	*Zahl*	0-100 %
use_item_constraints	*boolesch*	
item_constraints_type	Include Exclude	
use_taxonomy	*boolesch*	
taxonomy_table_name	*Zeichenfolge*	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	*Zeichenfolge*	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	*Zeichenfolge*	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	*boolesch*	Steuert, ob in IBM SPSS Modeler gespeicherte Taxonomieinformationen bei der Modellerstellung hochgeladen werden sollen. Die Taxonomietabelle wird verworfen, wenn sie bereits vorhanden ist. Die Taxonomieinformationen werden mit dem Modellknoten gespeichert und können über die Schaltflächen **Kategorien bearbeiten** und **Taxonomie bearbeiten** bearbeitet werden.

ISW-Regression

Die folgenden Eigenschaften sind für Knoten des Typs db2imreg verfügbar.

Tabelle 166. Eigenschaften des Knotens "db2imreg".

Eigenschaften des Knotens db2imreg	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	
regression_method	transform linear polynomial rbf	In der nächsten Tabelle finden Sie Eigenschaften, die nur gelten, wenn regression_method auf rbf gesetzt ist.
perform_test_run	*Feld*	
limit_rsquared_value	*boolesch*	
max_rsquared_value	*Zahl*	Der Wert muss zwischen 0,0 und 1,0 liegen.
use_execution_time_limit	*boolesch*	

Tabelle 166. Eigenschaften des Knotens "db2imreg" (Forts.).

Eigenschaften des Knotens db2imreg	**Werte**	**Eigenschaftsbeschreibung**
execution_time_limit_mins	*ganze Zahl*	Der Wert muss größer als 0 sein.
use_max_degree_polynomial	*boolesch*	
max_degree_polynomial	*ganze Zahl*	
use_intercept	*boolesch*	
use_auto_feature_selection_method	*boolesch*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*boolesch*	
min_significance_level	*Zahl*	
use_min_significance_level	*boolesch*	

Die folgenden Eigenschaften gelten nur, wenn regression_method auf rbf gesetzt ist.

Tabelle 167. Eigenschaften von "db2imreg", wenn "regression_method" auf "rbf" gesetzt ist.

Eigenschaften des Knotens db2imreg	Werte	Eigenschaftsbeschreibung
use_output_sample_size	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
output_sample_size	*ganze Zahl*	Die Standardeinstellung ist 2. Minimum ist 1.
use_input_sample_size	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
input_sample_size	*ganze Zahl*	Die Standardeinstellung ist 2. Minimum ist 1.
use_max_num_centers	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
max_num_centers	*ganze Zahl*	Die Standardeinstellung ist 20. Minimum ist 1.
use_min_region_size	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
min_region_size	*ganze Zahl*	Die Standardeinstellung ist 15. Minimum ist 1.
use_max_data_passes	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
max_data_passes	*ganze Zahl*	Die Standardeinstellung ist 5. Minimum ist 2.
use_min_data_passes	*boolesch*	Wenn wahr, den Wert automatisch auf Standard setzen.
min_data_passes	*ganze Zahl*	Die Standardeinstellung ist 5. Minimum ist 2.

ISW-Clustering

Die folgenden Eigenschaften sind für Knoten des Typs db2imcluster verfügbar.

Tabelle 168. Eigenschaften des Knotens "db2imcluster".

Eigenschaften des Knotens db2imcluster	**Werte**	**Eigenschaftsbeschreibung**
cluster_method	demographic kohonen birch	
kohonen_num_rows	*ganze Zahl*	
kohonen_num_columns	*ganze Zahl*	
kohonen_passes	*ganze Zahl*	
use_num_passes_limit	*boolesch*	
use_num_clusters_limit	*boolesch*	
max_num_clusters	*ganze Zahl*	Wert größer als 1.
birch_dist_measure	log_likelihood euclidean	Die Standardeinstellung ist log_likelihood.
birch_num_cfleaves	*ganze Zahl*	Die Standardeinstellung ist 1000.
birch_num_refine_passes	*ganze Zahl*	Die Standardeinstellung lautet 3; Minimum ist 1.
use_execution_time_limit	*boolesch*	
execution_time_limit_mins	*ganze Zahl*	Der Wert muss größer als 0 sein.
min_data_percentage	*Zahl*	0-100 %
use_similarity_threshold	*boolesch*	
similarity_threshold	*Zahl*	Der Wert muss zwischen 0,0 und 1,0 liegen.

ISW Naive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs db2imnbs verfügbar.

Tabelle 169. Eigenschaften des Knotens "db2imnb".

Eigenschaften des Knotens db2imnb	**Werte**	**Eigenschaftsbeschreibung**
perform_test_run	*boolesch*	
probability_threshold	*Zahl*	Die Standardeinstellung ist 0,001. Minimalwert ist 0; Maximalwert ist 1.000
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.

ISW Logistische Regression

Die folgenden Eigenschaften sind für Knoten des Typs db2imlog verfügbar.

Tabelle 170. Eigenschaften des Knotens "db2imlog".

Eigenschaften des Knotens db2imlog	**Werte**	**Eigenschaftsbeschreibung**
perform_test_run	*boolesch*	
use_costs	*boolesch*	
costs	*strukturiert*	Strukturierte Eigenschaft.

ISW-Zeitreihen

Hinweis: der Eingabefeldparameter wird für diesen Knoten nicht verwendet. Wenn der Eingabefeldparameter in dem Script gefunden wird, erscheint eine Warnung, dass der Knoten als eingehende Felder *time* und *targets*, aber keine Eingabefelder hat.

Die folgenden Eigenschaften sind für Knoten des Typs `db2imtimeseries` verfügbar.

Tabelle 171. Eigenschaften des Knotens "db2imtimeseries".

Eigenschaften des Knotens `db2imtimeseries`	**Werte**	**Eigenschaftsbeschreibung**
`time`	*Feld*	"Integer", "time" oder "date" sind zulässig.
`targets`	*Liste mit Feldern*	

`forecasting_algorithm`	`arima`
	`exponential_smoothing`
	`seasonal_trend_decomposition`	

`forecasting_end_time`	`auto`
	`ganze Zahl`
	`Datum`
	`time`	

`use_records_all`	*boolesch*	Wenn falsch, müssen `use_records_start` und `use_records_end` festgelegt werden.
`use_records_start`	*ganze Zahl / Zeit / Datum*	Abhängig vom Zeitfeldtyp
`use_records_end`	*ganze Zahl / Zeit / Datum*	Abhängig vom Zeitfeldtyp

`interpolation_method`	`none`
	`linear`
	`exponential_splines`
	`cubic_splines`	

Eigenschaften von IBM DB2-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der IBM DB2 ISW-Modelle erstellt wurden.

ISW-Entscheidungsbaum

Für Knoten vom Typ `applydb2imtree` sind keine speziellen Eigenschaften definiert.

ISW-Assoziation

Dieses Modellnugget kann nicht in Scripts verwendet werden.

ISW-Sequenz

Dieses Modellnugget kann nicht in Scripts verwendet werden.

ISW-Regression

Für Knoten vom Typ `applydb2imreg` sind keine speziellen Eigenschaften definiert.

ISW-Clustering

Für Knoten vom Typ applydb2imcluster sind keine speziellen Eigenschaften definiert.

ISW Naive Bayes

Für Knoten vom Typ applydb2imnb sind keine speziellen Eigenschaften definiert.

ISW Logistische Regression

Für Knoten vom Typ applydb2imlog sind keine speziellen Eigenschaften definiert.

ISW-Zeitreihen

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Knoteneigenschaften für IBM Netezza Analytics-Modellierung

Eigenschaften von Netezza-Modellierungsknoten

Folgende Eigenschaften haben alle IBM Netezza-Datenbankmodellierungsknoten gemeinsam.

Tabelle 172. Allgemeine Eigenschaften von Netezza-Knoten.

Allgemeine Eigenschaften von Netezza-Knoten	**Werte**	**Eigenschaftsbeschreibung**
custom_fields	*boolesch*	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
inputs	*[Feld1 ... FeldN]*	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
target	*Feld*	Zielfeld (stetig oder kategorial).
record_id	*Feld*	Das als eindeutige ID für einen Datensatz zu verwendende Feld.
use_upstream_connection	*boolesch*	Falls "True" (Standard), die in einem vorausgehenden Knoten angegebenen Verbindungsdetails. Wird bei Angabe von move_data_to_connection nicht verwendet.
move_data_connection	*boolesch*	Falls "True", wird der Wert in die durch connection angegebene Datenbank verschoben. Wird bei Angabe von use_upstream_connection nicht verwendet.
connection	*strukturiert*	Die Verbindungszeichenfolge für die Netezza-Datenbank, in der das Modell gespeichert ist. Strukturierte Eigenschaft im Format: ['odbc' '<DSN>' '<Benutzername>' '<KW>' '<K>' '<Verbattribs>' {true|false}] Dabei gilt: <DSN> ist der Datenquellenname <Benutzername> und <KW> sind der Benutzername und das Kennwort für die Datenbank <K> ist der Katalogname <Verbattribs> sind die Verbindungsattribute true | false gibt an, ob das Kennwort erforderlich ist.
table_name	*Zeichenfolge*	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.

Tabelle 172. Allgemeine Eigenschaften von Netezza-Knoten (Forts.).

Allgemeine Eigenschaften von Netezza-Knoten	Werte	Eigenschaftsbeschreibung
use_model_name	*boolesch*	Falls "True", wird der von model_name angegebene Name als Name des Modells verwendet. Andernfalls wird der Modellname vom System erstellt.
model_name	*Zeichenfolge*	Benutzerdefinierter Name für neues Modell.
include_input_fields	*boolesch*	Falls "True", werden alle Eingabefelder nach unten weitergegeben. Andernfalls werden nur record_id und vom Modell erstellte Felder weitergegeben.

Netezza-Entscheidungsbaum

Die folgenden Eigenschaften sind für Knoten des Typs netezzadectree verfügbar.

Tabelle 173. Eigenschaften des Knotens "netezzadectree".

Eigenschaften des Knotens netezzadectree	Werte	Eigenschaftsbeschreibung
impurity_measure	Entropy Gini	Das Maß der Unreinheit, das verwendet wird, um die beste Position für eine Baumteilung zu ermitteln.
max_tree_depth	*ganze Zahl*	Maximale Anzahl der Ebenen, auf die der Baum erweitert werden kann. Der Standardwert ist 63 (größter zulässiger Wert).
min_improvement_splits	*Zahl*	Mindestverbesserung in Unreinheit, damit eine Aufteilung stattfinden kann. Die Standardeinstellung ist 0.01.
min_instances_split	*ganze Zahl*	Mindestanzahl der nicht aufgeteilten Datensätze, die verbleiben müssen, bevor eine Aufteilung stattfinden kann. Der Standardwert ist 2 (kleinster zulässiger Wert).
weights	*strukturiert*	Relative Gewichtungen für Klassen. Strukturierte Eigenschaft. Standardgewichtung ist für alle Klassen 1.
pruning_measure	Acc wAcc	Die Standardeinstellung ist Acc (Genauigkeit). Bei der alternativen Einstellung wAcc (gewichtete Genauigkeit) werden Klassengewichtungen in die Reduzierung/Beschneidung mit einbezogen.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	In der Standardeinstellung wird allTrainingData zur Schätzung der Modellgenauigkeit verwendet. Verwenden Sie partitionTrainingData, um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdataset aus einer angegebenen Datenbanktabelle zu verwenden.

Tabelle 173. Eigenschaften des Knotens "netezzadectree" (Forts.).

Eigenschaften des Knotens netezzadectree	**Werte**	**Eigenschaftsbeschreibung**
perc_training_data	*Zahl*	Wenn prune_tree_options auf partitionTrainingData gesetzt ist, wird der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	*ganze Zahl*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wenn prune_tree_options auf partitionTrainingData gesetzt ist. Die Standardeinstellung ist 1.
pruning_table	*Zeichenfolge*	Tabellenname eines separaten Datasets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	*boolesch*	Wenn "True"; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.

Netezza-K-Means

Die folgenden Eigenschaften sind für Knoten des Typs netezzakmeans verfügbar.

Tabelle 174. Eigenschaften des Knotens "netezzakmeans".

Eigenschaften des Knotens netezzakmeans	**Werte**	**Eigenschaftsbeschreibung**
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
num_clusters	*ganze Zahl*	Anzahl der zu erstellenden Cluster; Standardwert ist 3.
max_iterations	*ganze Zahl*	Anzahl der Algorithmusiterationen, nach der das Modelltrainig beendet werden soll; Standardwert ist 5.
rand_seed	*ganze Zahl*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert; Standardwert ist 12345.

Netezza-Bayes-Netz

Die folgenden Eigenschaften sind für Knoten des Typs netezzabayes verfügbar.

Tabelle 175. Eigenschaften des Knotens "netezzabayes".

Eigenschaften des Knotens netezzabayes	**Werte**	**Eigenschaftsbeschreibung**
base_index	*ganze Zahl*	Die numerische Kennung, die zur internen Verwaltung dem ersten Eingabefeld zugewiesen wird; Standardwert ist 777.
sample_size	*ganze Zahl*	Umfang der zu ziehenden Stichprobe, wenn die Anzahl der Attribute sehr groß ist; Standardwert ist 10.000.

Tabelle 175. Eigenschaften des Knotens "netezzabayes" (Forts.).

Eigenschaften des Knotens netezzabayes	**Werte**	**Eigenschaftsbeschreibung**
`display_additional_information`	*boolesch*	Wenn "True", werden weitere Fortschrittsinformationen in einem Nachrichtendialogfeld angezeigt.

`type_of_prediction`	`best`
	`neighbors`
	`nn-neighbors`
		Typ des zu verwendenden Vorhersagealgorithmus: best (Nachbar mit höchster Korrelation), neighbors (gewichtete Vorhersage von Nachbarn) oder nn-neighbors (Nicht-NULL-Nachbarn).

Netezza - Naive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs `netezzanaivebayes` verfügbar.

Tabelle 176. Eigenschaften des Knotens "netezzanaivebayes".

Eigenschaften des Knotens netezzanaivebayes	**Werte**	**Eigenschaftsbeschreibung**
`compute_probabilities`	*boolesch*	Wenn "True"; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.
`use_m_estimation`	*boolesch*	Wenn "True", wird das m-Schätzverfahren zur Vermeidung der Wahrscheinlichkeit null während der Schätzung verwendet.

Netezza-KNN

Die folgenden Eigenschaften sind für Knoten des Typs `netezzaknn` verfügbar.

Tabelle 177. Eigenschaften des Knotens "netezzaknn".

Eigenschaften des Knotens netezzaknn	**Werte**	**Eigenschaftsbeschreibung**
`weights`	*strukturiert*	Strukturierte Eigenschaft, die zur Zuweisung von Gewichtungen für die einzelnen Klassen verwendet wird.

`distance_measure`	`Euclidean`
	`Manhattan`
	`Canberra`
	`Maximum`
		Methode zur Messung des Abstands zwischen Datenpunkten.

`num_nearest_neighbors`	*ganze Zahl*	Anzahl der nächsten Nachbarn für einen bestimmten Fall; Standardwert ist 3
`standardize_measurements`	*boolesch*	Wenn "True", werden vor der Berechnung der Abstandwerte die Messungen für stetige Eingabefelder standardisiert.
`use_coresets`	*boolesch*	Wenn "True", wird Stichprobennahme mit Core-Sets verwendet, um die Berechnung bei großen Datasets zu beschleunigen.

Netezza - Divisives Clustering

Die folgenden Eigenschaften sind für Knoten des Typs `netezzadivcluster` verfügbar.

Tabelle 178. Eigenschaften des Knotens "netezzadivcluster".

Eigenschaften des Knotens **netezzadivcluster**	Werte	Eigenschaftsbeschreibung
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
max_iterations	*ganze Zahl*	Maximale Anzahl an Algorithmusiterationen, die durchgeführt werden sollen, bevor das Modelltraining beendet wird; Standardwert ist 5.
max_tree_depth	*ganze Zahl*	Maximale Anzahl an Ebenen, in die das Dataset unterteilt werden kann; Standardwert ist 3.
rand_seed	*ganze Zahl*	Für die Reproduktion von Analysen verwendeter Zufallsstartwert; Standardwert ist 12345.
min_instances_split	*ganze Zahl*	Mindestanzahl von Datensätzen, die aufgeteilt werden können; Standardwert ist 5.
level	*ganze Zahl*	Hierarchieebene, auf der die Datensätze gescort werden; Standardwert ist -1.

Netezza-PCA

Die folgenden Eigenschaften sind für Knoten des Typs netezzapca verfügbar.

Tabelle 179. Eigenschaften des Knotens "netezzapca".

Eigenschaften des Knotens **netezzapca**	Werte	Eigenschaftsbeschreibung
center_data	*boolesch*	Wenn "True" (Standard), wird vor der Analyse Datenzentrierung (auch als Mittelwertsubtraktion bezeichnet) durchgeführt.
perform_data_scaling	*boolesch*	Wenn "True", wird vor der Analyse eine Datenskalierung durchgeführt. Auf diese Weise wird die Analyse eventuell weniger arbiträr, wenn verschiedene Variablen in verschiedenen Einheiten gemessen werden.
force_eigensolve	*boolesch*	Wenn "True", wird eine weniger genaue, jedoch schnellere Methode zur Ermittlung der Hauptkomponenten verwendet.
pc_number	*ganze Zahl*	Anzahl an Hauptkomponenten, auf die das Dataset reduziert werden soll; Standardwert ist 1.

Netezza-Regressionsbaum

Die folgenden Eigenschaften sind für Knoten des Typs netezzaregtree verfügbar.

Tabelle 180. Eigenschaften des Knotens "netezzaregtree".

Eigenschaften des Knotens **netezzaregtree**	Werte	Eigenschaftsbeschreibung
max_tree_depth	*ganze Zahl*	Maximale Anzahl an Ebenen, auf die ein Baum unterhalb des Stammknotens erweitert werden kann; Standardwert ist 10.

Tabelle 180. Eigenschaften des Knotens "netezzaregtree" (Forts.).

Eigenschaften des Knotens netezzaregtree	**Werte**	**Eigenschaftsbeschreibung**
split_evaluation_measure	Variance	Unreinheitsmaß für die Klasse, das verwendet wird, um die beste Position für eine Baumteilung zu ermitteln; Standardwert (und einzige derzeit mögliche Option) ist Variance.
min_improvement_splits	*Zahl*	Mindestwert der Unreinheitsreduzierung, bevor eine neue Aufteilung des Baums erfolgt.
min_instances_split	*ganze Zahl*	Mindestanzahl an Datensätzen, die aufgeteilt werden kann.
pruning_measure	mse r2 pearson spearman	Für die Reduzierung zu verwendende Methode.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	In der Standardeinstellung wird allTrainingData zur Schätzung der Modellgenauigkeit verwendet. Verwenden Sie partitionTrainingData, um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdataset aus einer angegebenen Datenbanktabelle zu verwenden.
perc_training_data	*Zahl*	Wenn prune_tree_options auf PercTrainingData gesetzt ist, wird der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	*ganze Zahl*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wenn prune_tree_options auf PercTrainingData gesetzt ist. Die Standardeinstellung ist 1.
pruning_table	*Zeichenfolge*	Tabellenname eines separaten Datasets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	*boolesch*	Wenn "True", wird angegeben, ob die Varianz zugewiesener Klassen in die Ausgabe aufgenommen werden soll.

Netezza - Lineare Regression

Die folgenden Eigenschaften sind für Knoten des Typs netezzalineregression verfügbar.

Tabelle 181. Eigenschaften des Knotens "netezzalineregression".

Eigenschaften des Knotens **netezzalineregression**	Werte	Eigenschaftsbeschreibung
use_svd	*boolesch*	Wenn "True", wird anstelle der ursprünglichen Matrix die Matrix zur Einzelwertzerlegung verwendet, um eine höhere Geschwindigkeit und numerische Genauigkeit zu erreichen.
include_intercept	*boolesch*	Wenn "True" (Standard), wird die Gesamtgenauigkeit der Lösung erhöht.
calculate_model_diagnostics	*boolesch*	Wenn "True", werden Diagnosedaten für das Modell berechnet.

Netezza-Zeitreihe

Die folgenden Eigenschaften sind für Knoten des Typs netezzatimeseries verfügbar.

Tabelle 182. Eigenschaften des Knotens "netezzatimeseries".

Eigenschaften des Knotens **netezzatimeseries**	Werte	Eigenschaftsbeschreibung
time_points	*Feld*	Das Eingabefeld, das die Datums- bzw. Zeitwerte für die Zeitreihe enthält.
time_series_ids	*Feld*	Eingabefeld mit Zeitreihen-IDs. Verwenden Sie das Feld, wenn die Eingabe mehrere Zeitreihen enthält.
model_table	*Feld*	Der Name der Datenbanktabelle, in der das Netezza-Zeitreihenmodell gespeichert werden soll.
description_table	*Feld*	Name der Eingabetabelle mit Zeitreihennamen und Beschreibungen.
seasonal_adjustment_table	*Feld*	Name der Ausgabetabelle, in der saisonal angepasste Werte gespeichert werden, die durch exponentielles Glätten oder Algorithmen zur saisonalen Zerlegung in Trends berechnet werden.
algorithm_name	SpectralAnalysis oder spectral ExponentialSmoothing oder esmoothing ARIMA (X11 ARIMA) SeasonalTrendDecomposition oder std	Für die Modellierung von Zeitreihen zu verwendender Algorithmus.
trend_name	N A DA M DM	Trendtyp für exponentielles Glätten: N - keiner A - additiv DA - gedämpft additiv M - multiplikativ DM - gedämpft multiplikativ
seasonality_type	N A M	Saisonalitätstyp für exponentielles Glätten: N - keiner A - additiv M - multiplikativ

Tabelle 182. Eigenschaften des Knotens "netezzatimeseries" (Forts.).

Eigenschaften des Knotens netezzatimeseries	**Werte**	**Eigenschaftsbeschreibung**
interpolation_method	linear cubicspline exponentialspline	Zu verwendende Interpolationsmethode.
timerange_setting	SD SP	Einstellung für den zu verwendenden Zeitbereich: SD - systembestimmt (verwendet den vollständigen Bereich der Zeitreihendaten) SP - benutzerdefiniert über earliest_time und latest_time
earliest_time	*Datum*	Start- und Endzeitpunkte, wenn timerange_setting auf SP gesetzt ist. Format: <jjjj>-<mm>-<tt>
latest_time		
arima_setting	SD SP	Einstellung für den ARIMA-Algorithmus (nur verwendet, wenn algorithm_name auf ARIMA gesetzt ist): SD - systembestimmt SP - benutzerdefiniert Wenn arima_setting = SP angegeben ist, verwenden Sie die folgenden Parameter, um die saisonalen und nicht saisonalen Werte festzulegen.
p_symbol	less eq lesseq	ARIMA - Operator für die Parameter p, d, q, sp, sd und sq: less - kleiner als eq - gleich lesseq - kleiner-gleich
d_symbol		
q_symbol		
sp_symbol		
sd_symbol		
sq_symbol		
p (Missing Values)	*ganze Zahl*	ARIMA - nicht saisonale Autokorrelationsmaße.
q	*ganze Zahl*	ARIMA - nicht saisonaler Ableitungswert.
d	*ganze Zahl*	ARIMA - nicht saisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.
sp	*ganze Zahl*	ARIMA - saisonale Autokorrelationsmaße.
sq	*ganze Zahl*	ARIMA - saisonaler Ableitungswert.
sd	*ganze Zahl*	ARIMA - saisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.

Tabelle 182. Eigenschaften des Knotens "netezzatimeseries" (Forts.).

Eigenschaften des Knotens netezzatimeseries	**Werte**	**Eigenschaftsbeschreibung**
advanced_setting	SD SP	Legt fest, wie erweiterte Einstellungen behandelt werden: SD - systembestimmt SP - benutzerdefiniert über period, units_period und forecast_setting.
period	*ganze Zahl*	Länge des saisonalen Zyklus, die in Verbindung mit units_period angegeben wird. Wird nicht für Spektralanalyse verwendet.
units_period	ms s min h d wk q y	Einheiten für period: ms - Millisekunden s - Sekunden min - Minuten h - Stunden d - Tage wk - Wochen q - Quartale y - Jahre Beispiel: Verwenden Sie für einen wöchentliche Zeitreihe 1 für period und wk für units_period.
forecast_setting	forecasthorizon forecasttimes	Gibt an, wie Vorhersagen gemacht werden.
forecast_horizon	*Zeichenfolge*	Wenn forecast_setting = forecasthorizon angegeben ist, wird ein Endpunkt für die Vorhersage angegeben. Format: <jjjj>-<mm>-<tt>
forecast_times	[{'*Datum*'}, {'*Datum*'},..., {'*Datum*'}]	Wenn forecast_setting = forecasttimes angegeben ist, werden die für die Vorhersagen zu verwendenden Zeiten angegeben. Format: <jjjj>-<mm>-<tt>
include_history	*boolesch*	Gibt an, ob historische Werte bei der Ausgabe berücksichtigt werden sollen.
include_interpolated_values	*boolesch*	Gibt an, ob interpolierte Werte bei der Ausgabe berücksichtigt werden sollen. Wird nicht verwendet, wenn include_history auf false gesetzt ist.

Verallgemeinertes lineares Netezza-Modell

Die folgenden Eigenschaften sind für Knoten des Typs netezzaglm verfügbar.

Tabelle 183. Eigenschaften des Knotens "netezzaglm".

Eigenschaften des Knotens `netezzaglm`	**Werte**	**Eigenschaftsbeschreibung**

`dist_family`	`bernoulli`
	`gaussian`
	`poisson`
	`negativebinomial`
	`wald`
	`gamma`
		Verteilungstyp. Die Standardeinstellung ist `bernoulli`.

`dist_params`	*Zahl*	Zu verwendender Wert für Verteilungsparameter. Wird nur verwendet, wenn `distribution` auf `Negativebinomial` gesetzt ist.
`trials`	*ganze Zahl*	Wird nur verwendet, wenn `distribution` auf `Binomial` gesetzt ist. Wenn es sich bei der Zielantwort um eine Reihe von Ereignissen handelt, die während Tests auftreten, enthält das Feld `target` die Anzahl der Ereignisse und das Feld `trials` die Anzahl der Tests.
`model_table`	*Feld*	Der Name der Datenbanktabelle, in der das verallgemeinerte lineare Netezza-Modell gespeichert werden soll.
`maxit`	*ganze Zahl*	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden sollen. Der Standardwert ist 20.
`eps`	*Zahl*	Der maximale Fehlerwert (in wissenschaftlicher Notation), bei dem der Algorithmus die Suche nach dem am besten passenden Modell beenden soll. Der Standardwert ist -3, d. h. 1E-3 bzw. 0,001.
`tol`	*Zahl*	Der Wert (in wissenschaftlicher Notation), unterhalb dessen Fehler so behandelt werden, als hätten sie den Wert 0. Der Standardwert ist -7, es werden also Fehlerwerte unter 1E-7 (bzw. 0,0000001) als nicht signifikant gewertet.

Tabelle 183. Eigenschaften des Knotens "netezzaglm" (Forts.).

Eigenschaften des Knotens netezzaglm	**Werte**	**Eigenschaftsbeschreibung**
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Zu verwendende Verknüpfungsfunktion. Die Standardeinstellung ist logit.
link_params	*Zahl*	Für die Verknüpfungsfunktion zu verwendender Parameterwert. Wird nur verwendet, wenn link_function auf power oder oddspower gesetzt ist.
interaction	[{[*Spaltennamen1*],[*Niveaus1*]}, {[*Spaltennamen2*], [*Niveaus2*]},...,{[*SpaltennamenN*], [*NiveausN*]},]	Gibt die Interaktionen zwischen Feldern an. *Spaltennamen* ist eine Liste von Eingabefeldern und *Niveau* ist für jedes Feld immer 0.
intercept	*boolesch*	Wenn true gesetzt ist, wird konstanter Term in das Modell einbezogen.

Eigenschaften von Netezza-Modellnuggets

Folgende Eigenschaften haben alle Modellnuggets von Netezza-Datenbanken gemeinsam.

Tabelle 184. Allgemeine Eigenschaften von Netezza-Modellnuggets.

Allgemeine Eigenschaften von Netezza-Modellnuggets	**Werte**	**Eigenschaftsbeschreibung**
connection	*Zeichenfolge*	Die Verbindungszeichenfolge für die Netezza-Datenbank, in der das Modell gespeichert ist.
table_name	*Zeichenfolge*	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.

Die anderen Eigenschaften des Modellnuggets stimmen mit denen für den zugehörigen Modellierungsknoten überein.

Die Scriptnamen des Modellnuggets lauten wie folgt.

Tabelle 185. Scriptnamen von Netezza-Modellnuggets.

Modellnugget	**Scriptname**
Entscheidungsbaum	applynetezzadectree
K-Means	applynetezzakmeans

Tabelle 185. Scriptnamen von Netezza-Modellnuggets (Forts.).

Modellnugget	**Scriptname**
Bayes-Netz	`applynetezzabayes`
Naive Bayes	`applynetezzanaivebayes`
KNN	`applynetezzaknn`
Divisives Clustering	`applynetezzadivcluster`
PCA	`applynetezzapca`
Regressionsbaum	`applynetezzaregtree`
Lineare Regression	`applynetezzalineregression`
Zeitreihen	`applynetezzatimeseries`
Verallgemeinert linear (GenLin)	`applynetezzaglm`

Kapitel 16. Eigenschaften von Ausgabeknoten

Die Eigenschaften von Ausgabeknoten unterscheiden sich von denen anderer Knotentypen. Statt auf eine bestimmte Knotenoption zu verweisen, speichern Ausgabeknoten-Eigenschaften eine Referenz zum Ausgabeobjekt. Dies ist nützlich, wenn ein Wert aus einer Tabelle als Streamparameter festgelegt wird.

In diesem Abschnitt werden die für Ausgabeknoten verfügbaren Scripteigenschaften beschrieben.

Eigenschaften von Analyseknoten

Der Analyseknoten evaluiert die Fähigkeit von Vorhersagemodellen, genaue Vorhersagen zu generieren. Mit Analyseknoten werden verschiedene Vergleiche zwischen den vorhergesagten Werten und den tatsächlichen Werten für ein oder mehrere Modellnuggets angestellt. Sie können außerdem Vorhersagemodelle miteinander vergleichen.

Tabelle 186. Eigenschaften von Analyseknoten.

Eigenschaften von Analyseknoten	Datentyp	Eigenschaftsbeschreibung
`output_mode`	`Screen` `File`	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
`use_output_name`	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
`output_name`	*Zeichenfolge*	Wenn `use_output_name` wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
`output_format`	`Text` (*.txt*) `HTML` (*.html*) `Output` (*.cou*)	Dient zur Angabe des Ausgabetyps.
`by_fields`	[*Feld Feld Feld*]	
`full_filename`	*Zeichenfolge*	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
`coincidence`	*boolesch*	
`performance`	*boolesch*	
`evaluation_binary`	*boolesch*	
`confidence`	*boolesch*	
`threshold`	*Zahl*	
`improve_accuracy`	*Zahl*	
`inc_user_measure`	*boolesch*	
`user_if`	*Ausdr*	
`user_then`	*Ausdr*	
`user_else`	*Ausdr*	
`user_compute`	`[Mean Sum Min Max SDev]`	

Eigenschaften des Knotens "dataaudit"

Der Data Audit-Knoten bietet einen umfassenden ersten Einblick in die Daten mit statistischen Funktionen, Histogrammen und der Verteilung für die einzelnen Felder sowie Informationen zu Ausreißern, fehlenden Werten und Extremwerten. Die Ergebnisse werden in einer übersichtlichen Matrix dargestellt, die sortiert werden kann und als Grundlage für die Erzeugung normal großer Diagramme und Datenvorbereitungsknoten dient.

Tabelle 187. Eigenschaften des Knotens "dataaudit".

Eigenschaften des Knotens `dataaudit`	Datentyp	Eigenschaftsbeschreibung
`custom_fields`	*boolesch*	
`fields`	*[Feld1 ... FeldN]*	
`overlay`	*Feld*	
`display_graphs`	*boolesch*	Dient zur Aktivierung bzw. Inaktivierung der Anzeige von Diagrammen in der Ausgabematrix.
`basic_stats`	*boolesch*	
`advanced_stats`	*boolesch*	
`median_stats`	*boolesch*	
`calculate`	`Count` `Breakdown`	Dient zur Berechnung fehlender Werte. Sie können eine der beiden Berechnungsmethoden, beide Methoden oder auch keine der Methoden auswählen.
`outlier_detection_method`	`std` `iqr`	Dient zur Angabe der Erkennungsmethode für Ausreißer und Extremwerte.
`outlier_detection_std_outlier`	*Zahl*	Wenn für `outlier_detection_method` die Option `std` verwendet wird, wird die Zahl angegeben, die für die Definition der Ausreißer verwendet werden soll.
`outlier_detection_std_extreme`	*Zahl*	Wenn für `outlier_detection_method` die Option `std` verwendet wird, wird die Zahl angegeben, die für die Definition der Extremwerte verwendet werden soll.
`outlier_detection_iqr_outlier`	*Zahl*	Wenn für `outlier_detection_method` die Option `iqr` verwendet wird, wird die Zahl angegeben, die für die Definition der Ausreißer verwendet werden soll.
`outlier_detection_iqr_extreme`	*Zahl*	Wenn für `outlier_detection_method` die Option `iqr` verwendet wird, wird die Zahl angegeben, die für die Definition der Extremwerte verwendet werden soll.

Tabelle 187. Eigenschaften des Knotens "dataaudit" (Forts.).

Eigenschaften des Knotens **dataaudit**	Datentyp	Eigenschaftsbeschreibung
use_output_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*Zeichenfolge*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Dient zur Angabe des Ausgabetyps.
paginate_output	*boolesch*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	*Zahl*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	*Zeichenfolge*	

Eigenschaften des Knotens "matrix"

Der Matrixknoten erstellt eine Tabelle, die die Beziehungen zwischen den Feldern aufzeigt. Dieser Knoten dient am häufigsten zur Darstellung der Beziehung zwischen zwei symbolischen Feldern, kann jedoch auch zum Aufzeigen der Beziehungen zwischen Flagfeldern oder numerischen Feldern herangezogen werden.

Tabelle 188. Eigenschaften des Knotens "matrix".

Eigenschaften des Knotens **matrix**	Datentyp	Eigenschaftsbeschreibung
fields	Ausgewählt Flags Numerics	
row	*Feld*	
column	*Feld*	
include_missing_values	*boolesch*	Gibt an, ob benutzerdefiniert fehlende Werte (leer) und systemdefiniert fehlende Werte (null) in die Zeilen- und Spaltenausgabe eingeschlossen werden sollen.
cell_contents	CrossTabs Function	
function_field	*Zeichenfolge*	
function	Sum Mean Min Max SDev	

Tabelle 188. Eigenschaften des Knotens "matrix" (Forts.).

Eigenschaften des Knotens `matrix`	Datentyp	Eigenschaftsbeschreibung

`sort_mode`	`Unsorted`
	`Ascending`
	`Descending`	

`highlight_top`	*Zahl*	Wenn ungleich 0, dann ist die Eigenschaft wahr.
`highlight_bottom`	*Zahl*	Wenn ungleich 0, dann ist die Eigenschaft wahr.

`display`	`[Counts`
	`Expected`
	`Residuen`
	`RowPct`
	`ColumnPct`
	`TotalPct]`	

`include_totals`	*boolesch*	
`use_output_name`	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
`output_name`	*Zeichenfolge*	Wenn `use_output_name` wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
`output_mode`	`Screen` `File`	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
`output_format`	`Formatted` (*.tab*) `Delimited` (*.csv*) `HTML` (*.html*) `Output` (*.cou*)	Dient zur Angabe des Ausgabetyps. Sowohl für das Format `Formatted` als auch für das Format `Delimited` kann der Modifikator `transposed` verwendet werden, der die Zeilen und Spalten in der Tabelle transponiert.
`paginate_output`	*boolesch*	Wenn `output_format` auf `HTML` gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
`lines_per_page`	*Zahl*	Bei Verwendung mit `paginate_output` wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
`full_filename`	*Zeichenfolge*	

Eigenschaften des Knotens "means"

Der Mittelwertknoten vergleicht die Mittelwerte zwischen unabhängigen Gruppen oder zwischen Paaren von in Bezug stehenden Feldern, um zu testen, ob ein signifikanter Unterschied vorliegt. So können Sie beispielsweise die Einnahmen vor und nach der Durchführung einer Werbeaktion vergleichen oder die Einnahmen, die von Kunden stammen, die keine Werbezettel erhielten, mit den Einnahmen von Kunden vergleichen, die von der Werbeaktion erreicht wurden.

Tabelle 189. Eigenschaften des Knotens "means".

Eigenschaften des Knotens means	Datentyp	Eigenschaftsbeschreibung
means_mode	BetweenGroups BetweenFields	Gibt den Typ der Mittelwertstatistik an, die für die Daten ausgeführt werden soll.
test_fields	[Feld1 ... Feldn]	Gibt das Testfeld an, wenn means_mode auf BetweenGroups gesetzt ist.
grouping_field	*Feld*	Gibt das Gruppierungsfeld an.
paired_fields	[{Feld1 Feld2} {Feld3 Feld4} ...]	Gibt die zu verwendenden Feldpaare an, wenn means_mode auf BetweenFields gesetzt ist.
label_correlations	*boolesch*	Gibt an, ob Korrelationsbeschriftungen in der Ausgabe angezeigt werden sollen. Diese Einstellung gilt nur, wenn means_mode auf BetweenFields gesetzt ist.
correlation_mode	Wahrscheinlichkeits- funktionen Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
weak_label	*Zeichenfolge*	
medium_label	*Zeichenfolge*	
strong_label	*Zeichenfolge*	
weak_below_probability	*Zahl*	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	*Zahl*	Trennwert für starke Korrelationen.
weak_below_absolute	*Zahl*	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	*Zahl*	Trennwert für starke Korrelationen.
unimportant_label	*Zeichenfolge*	
marginal_label	*Zeichenfolge*	
important_label	*Zeichenfolge*	
unimportant_below	*Zahl*	Trennwert für niedrige Feldwichtigkeit. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
important_above	*Zahl*	
use_output_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*Zeichenfolge*	Zu verwendender Name.

Tabelle 189. Eigenschaften des Knotens "means" (Forts.).

Eigenschaften des Knotens means	Datentyp	Eigenschaftsbeschreibung
output_mode	Screen File	Gibt den Zielort für die vom Ausgabeknoten erstellte Ausgabe an.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Gibt den Ausgabetyp an.
full_filename	*Zeichenfolge*	
output_view	Simple Advanced	Gibt an, ob die einfache oder die erweiterte Ansicht in der Ausgabe angezeigt werden soll.

Eigenschaften des Knotens "report"

Der Berichtknoten erstellt formatierte Berichte, die sowohl festen Text als auch Daten und andere aus den Daten abgeleitete Ausdrücke enthalten. Das Format des Berichts wird mithilfe von Textvorlagen festgelegt, mit denen der feste Text und die Datenausgabekonstruktionen definiert werden. Sie können eine benutzerdefinierte Textformatierung angeben; hierzu stehen HTML-Tags in der Vorlage sowie Optionen auf der Registerkarte "Ausgabe" zur Verfügung. Sie können Datenwerte und andere bedingte Ausgaben mithilfe von CLEM-Ausdrücken in der Vorlage aufnehmen.

Tabelle 190. Eigenschaften des Knotens "report".

Eigenschaften des Knotens report	Datentyp	Eigenschaftsbeschreibung
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Dient zur Angabe des Ausgabetyps.
use_output_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*Zeichenfolge*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
text	*Zeichenfolge*	
full_filename	*Zeichenfolge*	
highlights	*boolesch*	
title	*Zeichenfolge*	
lines_per_page	*Zahl*	

Eigenschaften des Knotens "Routput"

Mit dem Knoten "Routput" können Sie Daten und die Ergebnisse des Modellscorings mithilfe Ihres eigenen benutzerdefinierten R-Scripts analysieren. Die Ausgabe von der Analyse kann Text oder grafisch sein. Die Ausgabe wird der Registerkarte **Ausgabe** des Managerbereichs hinzugefügt. Alternativ kann die Ausgabe in eine Datei umgeleitet werden.

Tabelle 191. Eigenschaften des Knotens "Routput".

Eigenschaften des Knotens `Routput`	Datentyp	Eigenschaftsbeschreibung
`Syntax`	*Zeichenfolge*	
`convert_flags`	`StringsAndDoubles` `LogicalValues`	
`convert_datetime`	*boolesch*	
`convert_datetime_class`	`POSIXct` `POSIXlt`	
`convert_missing`	*boolesch*	
`output_name`	`Auto` `Custom`	
`custom_name`	*Zeichenfolge*	
`output_to`	`Screen` `File`	
`output_type`	`Graph` `Text`	
`full_filename`	*Zeichenfolge*	
`graph_file_type`	`HTML` `COU`	

`text_file_type`	`HTML`
	`TXT`
	`COU`	

Eigenschaften des Knotens "setglobals"

Mit dem Globalwerteknoten werden die Daten gescannt und Übersichtswerte berechnet, die in CLEM-Ausdrücken herangezogen werden können. Mit diesem Knoten können Sie beispielsweise die Statistiken für das Feld *Alter* berechnen und dann den Gesamtmittelwert für *Alter* in CLEM-Ausdrücken verwenden. Fügen Sie hierzu die Funktion `@GLOBAL_MEAN(alter)` ein.

Tabelle 192. Eigenschaften des Knotens "setglobals".

Eigenschaften des Knotens `setglobals`	Datentyp	Eigenschaftsbeschreibung
`globals`	`[Sum Mean Min Max SDev]`	Strukturierte Eigenschaft
`clear_first`	*boolesch*	
`show_preview`	*boolesch*	

Eigenschaften des Knotens "simeval"

Der Simulationsevaluierungsknoten wertet ein angegebenes vorausgesagtes Zielfeld aus und stellt Verteilungs- und Korrelationsinformationen über das Zielfeld dar.

Tabelle 193. Eigenschaften des simeval-Knotens.

Eigenschaften des Knotens `simeval`	Datentyp	Eigenschaftsbeschreibung
`target`	*Feld*	
`iteration`	*Feld*	
`presorted_by_iteration`	*boolesch*	
`max_iterations`	*Zahl*	
`tornado_fields`	*[Feld1...FeldN]*	
`plot_pdf`	*boolesch*	
`plot_cdf`	*boolesch*	
`show_ref_mean`	*boolesch*	
`show_ref_median`	*boolesch*	
`show_ref_sigma`	*boolesch*	
`num_ref_sigma`	*Zahl*	
`show_ref_pct`	*boolesch*	
`ref_pct_bottom`	*Zahl*	
`ref_pct_top`	*Zahl*	
`show_ref_custom`	*boolesch*	
`ref_custom_values`	*[Zahl1...ZahlN]*	

`category_values`	`Category`
	`Probabilities`
	`Both`	

`category_groups`	`Categories` `Iterations`	
`create_pct_table`	*boolesch*	

`pct_table`	`Quartiles`
	`Intervals`
	`Custom`	

`pct_intervals_num`	*Zahl*	
`pct_custom_values`	*[Zahl1...ZahlN]*	

Eigenschaften des Knotens "simfit"

Der Simulationsanpassungsknoten untersucht die statistische Verteilung der Daten in jedem Feld und generiert (oder aktualisiert) einen Simulationsgenerierungsknoten, wobei jedem Feld die am besten passende Verteilung zugewiesen wird. Der Simulationsgenerierungsknoten kann dann zum Generieren simulierter Daten verwendet werden.

Tabelle 194. Eigenschaften des Knotens "simfit".

Eigenschaften des Knotens simfit	Datentyp	Eigenschaftsbeschreibung
build	Node XMLExport Both	
use_source_node_name	*boolesch*	
source_node_name	*Zeichenfolge*	Der benutzerdefinierte Name des Quellenknotens, der generiert oder aktualisiert wird.
use_cases	All LimitFirstN	
use_case_limit	*ganze Zahl*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*ganze Zahl*	
parameter_xml_filename	*Zeichenfolge*	
generate_parameter_import	*boolesch*	

Eigenschaften des Knotens "statistics"

Der Statistikknoten liefert grundlegende Übersichtsdaten zu numerischen Feldern. Er berechnet Übersichtsstatistiken für einzelne Felder und für die Korrelationen zwischen den Feldern.

Tabelle 195. Eigenschaften des Knotens "statistics".

Eigenschaften des Knotens statistics	Datentyp	Eigenschaftsbeschreibung
use_output_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*Zeichenfolge*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Dient zur Angabe des Ausgabetyps.
full_filename	*Zeichenfolge*	
examine	[*Feld Feld Feld*]	
correlate	[*Feld Feld Feld*]	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Wahrscheinlichkeits-funktionen Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
label_correlations	*boolesch*	

Tabelle 195. Eigenschaften des Knotens "statistics" (Forts.).

Eigenschaften des Knotens statistics	Datentyp	Eigenschaftsbeschreibung
weak_label	*Zeichenfolge*	
medium_label	*Zeichenfolge*	
strong_label	*Zeichenfolge*	
weak_below_probability	*Zahl*	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	*Zahl*	Trennwert für starke Korrelationen.
weak_below_absolute	*Zahl*	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	*Zahl*	Trennwert für starke Korrelationen.

Eigenschaften des Knotens "statisticsoutput"

Mit dem Statistics-Ausgabeknoten können Sie eine IBM SPSS Statistics-Prozedur aufrufen, um Ihre IBM SPSS Modeler-Daten zu analysieren. Es stehen zahlreiche IBM SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften des Knotens "statisticsoutput"" auf Seite 246.

Eigenschaften des Knotens "table"

Der Tabellenknoten zeigt die Daten in Tabellenform an, die auch in eine Datei geschrieben werden kann. Diese Vorgehensweise empfiehlt sich immer dann, wenn die Datenwerte überprüft oder in leicht lesbarer Form exportiert werden sollen.

Tabelle 196. Eigenschaften des Knotens "table".

Eigenschaften des Knotens table	Datentyp	Eigenschaftsbeschreibung
full_filename	*Zeichenfolge*	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
use_output_name	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*Zeichenfolge*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.

Tabelle 196. Eigenschaften des Knotens "table" (Forts.).

Eigenschaften des Knotens **table**	Datentyp	Eigenschaftsbeschreibung
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Dient zur Angabe des Ausgabetyps.
transpose_data	*boolesch*	Transponiert die Daten vor dem Export, sodass die Zeilen Felder und die Spalten Datensätze darstellen.
paginate_output	*boolesch*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	*Zahl*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
highlight_expr	*Zeichenfolge*	
output	*Zeichenfolge*	Eine schreibgeschützte Eigenschaft, die eine Referenz zur letzten vom Knoten erstellten Tabelle enthält.
value_labels	*[{Wert Beschriftungszeichenfolge} {Wert Beschriftungszeichenfolge} ...]*	Gibt Beschriftungen für Wertpaare an.
display_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld bei der Anzeige fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Mit dem Wert -1 wird der Streamstandard verwendet.
export_places	*ganze Zahl*	Legt die Dezimalstellen für das Feld beim Exportieren fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Mit dem Wert -1 wird der Streamstandard verwendet.
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL).

Tabelle 196. Eigenschaften des Knotens "table" (Forts.).

Eigenschaften des Knotens table	Datentyp	Eigenschaftsbeschreibung
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Legt das Datumsformat für das Feld fest (gilt nur für Felder mit dem Speichertyp DATE oder 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"	Legt das Zeitformat für das Feld fest (gilt nur für Felder mit dem Speichertyp TIME oder TIMESTAMP).
column_width	*ganze Zahl*	Legt die Spaltenbreite für das Feld fest. Mit dem Wert -1 wird die Spaltenbreite auf Auto (Automatisch) gesetzt.
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest.

Eigenschaften des Knotens "transform"

Mit dem Transformationsknoten können Sie die Ergebnisse von Transformationen auswählen und in einer Vorschau anzeigen, bevor Sie sie auf ausgewählte Felder anwenden.

Tabelle 197. Eigenschaften des Knotens "transform".

Eigenschaften des Knotens `transform`	Datentyp	Eigenschaftsbeschreibung
`fields`	[*Feld1... FieldN*]	Die bei der Transformation zu verwendenden Felder.
`formula`	`All` `Select`	Gibt an, ob alle oder nur die ausgewählten Transformationen berechnet werden sollen.
`formula_inverse`	*boolesch*	Gibt an, ob die inversen Transformation verwendet werden soll.
`formula_inverse_offset`	*Zahl*	Gibt eine relative Datenadresse an, die für die Formel verwendet werden soll. Standardmäßig auf 0 gesetzt, sofern vom Benutzer nicht anders angegeben.
`formula_log_n`	*boolesch*	Gibt an, ob die $\log_n$-Transformation verwendet werden soll.
`formula_log_n_offset`	*Zahl*	
`formula_log_10`	*boolesch*	Gibt an, ob die $\log_{10}$-Transformation verwendet werden soll.
`formula_log_10_offset`	*Zahl*	
`formula_exponential`	*boolesch*	Gibt an, ob die exponentielle Transformation (e^x) verwendet werden soll.
`formula_square_root`	*boolesch*	Gibt an, ob die Quadratwurzeltransformation verwendet werden soll.
`use_output_name`	*boolesch*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
`output_name`	*Zeichenfolge*	Wenn use_output_name true ist, gibt diese Eigenschaft den zu verwendenden Namen an.
`output_mode`	`Screen` `File`	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
`output_format`	`HTML` (*.html*) `Output` (*.cou*)	Dient zur Angabe des Ausgabetyps.
`paginate_output`	*boolesch*	Wenn `output_format` auf `HTML` gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
`lines_per_page`	*Zahl*	Bei Verwendung mit `paginate_output` wird die Anzahl der Zeilen pro Ausgabeseite angegeben.

Tabelle 197. Eigenschaften des Knotens "transform" (Forts.).

Eigenschaften des Knotens **transform**	Datentyp	Eigenschaftsbeschreibung
`full_filename`	*Zeichenfolge*	Gibt den für die Dateiausgabe zu verwendenden Dateinamen an.

Kapitel 17. Eigenschaften von Exportknoten

Allgemeine Eigenschaften von Exportknoten

Folgende Eigenschaften haben alle Exportknoten gemeinsam:

Tabelle 198. Allgemeine Eigenschaften von Exportknoten.

Eigenschaft	Werte	Eigenschaftsbeschreibung
`publish_path`	*Zeichenfolge*	Geben Sie den Stammnamen für die veröffentlichten Image- und Parameterdateien an.
`publish_metadata`	*boolesch*	Gibt an, ob eine Metadatendatei erzeugt wird, welche die Ein- und Ausgaben des Bilds und der zugehörigen Datenmodelle beschreibt.
`publish_use_parameters`	*boolesch*	Gibt an, ob Streamparameter in der *.par-Datei enthalten sind.
`publish_parameters`	*Zeichenfolgeliste*	Geben Sie die Parameter an, die eingeschlossen werden sollen.
`execute_mode`	`export_data` `publish`	Gibt an, ob der Knoten ohne Veröffentlichen des Streams ausgeführt wird oder ob der Stream automatisch beim Ausführen des Knotens veröffentlicht wird.

Eigenschaften des Knotens "asexport"

Der Analytic Server-Export ermöglicht Ihnen die Ausführung eines Streams unter HDFS (Hadoop Distributed File System).

Tabelle 199. Eigenschaften des Knotens "asexport".

Eigenschaften des Knotens `asexport`	Datentyp	Eigenschaftsbeschreibung
`data_source`	*Zeichenfolge*	Der Name der Datenquelle.
`export_mode`	*Zeichenfolge*	Gibt an, ob die exportierten Daten an die vorhandene Datenquelle angehängt (**`append`**) werden oder die vorhandene Datenquelle überschreiben (**`overwrite`**).
`host`	*Zeichenfolge*	Der Name des Analytic Server-Hosts.
`Port`	*ganze Zahl*	Der Port, an dem Analytic Server empfangsbereit ist.
`tenant`	*Zeichenfolge*	In einer Multi-Tenant-Umgebung der Name des Nutzers (Tenants), zu dem Sie gehören. In einer Single-Tenant-Umgebung ist dies standardmäßig **`ibm`**.
`set_credentials`	*boolesch*	Wenn die Benutzerauthentifizierung für Analytic Server und den SPSS Modeler-Server identisch ist, setzen Sie diesen Parameter auf **`false`**. Geben Sie andernfalls **`true`** an.

Tabelle 199. Eigenschaften des Knotens "asexport" (Forts.).

Eigenschaften des Knotens asexport	Datentyp	Eigenschaftsbeschreibung
user_name	*Zeichenfolge*	Der Benutzername für die Anmeldung bei Analytic Server. Nur erforderlich, wenn set_credentials auf "true" gesetzt ist.
password	*Zeichenfolge*	Das Kennwort für die Anmeldung bei Analytic Server. Nur erforderlich, wenn set_credentials auf "true" gesetzt ist.

Eigenschaften des Knotens "cognosexport"

Der IBM Cognos BI-Exportknoten exportiert Daten in einem Format, das von Cognos BI-Datenbanken gelesen werden kann.

Hinweis: Für diesen Knoten müssen Sie eine Cognos-Verbindung und eine ODBC-Verbindung definieren.

Cognos-Verbindung

Im Folgenden finden Sie die Eigenschaften für die Cognos-Verbindung.

Tabelle 200. Eigenschaften des Knotens "cognosexport".

Eigenschaften des Knotens cognosexport	Datentyp	Eigenschaftsbeschreibung
cognos_connection	*{"Feld","Feld", ... ,"Feld"}*	Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: {"*Cognos-Server-URL*", *Anmeldemodus*, "*Namespace*", "*Benutzername*", "*Kennwort*"} Dabei gilt: *Cognos-Server-URL* ist die URL des Cognos-Servers, auf den Sie exportieren. *Anmeldemodus* gibt an, ob eine anonyme Anmeldung verwendet wird, und ist entweder true oder false; bei Angabe von true sollten die folgenden Felder auf "" gesetzt werden. *Namespace* gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. *Benutzername* und *Kennwort* sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.
cognos_package_name	*Zeichenfolge*	Pfad und Name des Cognos-Pakets, an die Sie Daten exportieren, z. B.: /Public Folders/MyPackage
cognos_datasource	*Zeichenfolge*	
cognos_export_mode	Publish ExportFile	
cognos_filename	*Zeichenfolge*	

ODBC-Verbindung

Die Eigenschaften für die ODBC-Verbindung sind identisch mit denen, die im nächsten Bereich für `databaseexport` aufgelistet sind, mit der Ausnahme, dass die Eigenschaft `datasource` nicht gültig ist.

Eigenschaften des Knotens "databaseexport"

Der Datenbankexportknoten schreibt Daten in eine ODBC-kompatible relationale Datenquelle. Um Daten in eine ODBC-Datenquelle schreiben zu können, muss die betreffende Datenquelle bereits vorhanden sein und Sie benötigen Schreibzugriff dafür.

Tabelle 201. Eigenschaften des Knotens "databaseexport".

Eigenschaften des Knotens `databaseexport`	**Datentyp**	**Eigenschaftsbeschreibung**
`datasource`	*Zeichenfolge*	
`username`	*Zeichenfolge*	
`password`	*Zeichenfolge*	
`epassword`	*Zeichenfolge*	Dieser Slot ist während der Ausführung schreibgeschützt. Um ein codiertes Kennwort zu erstellen, verwenden Sie das Kennworttool im Menü "Extras". Weitere Informationen finden Sie im Thema „Erstellen eines verschlüsselten Kennworts” auf Seite 49.
`table_name`	*Zeichenfolge*	

`write_mode`	`Create`
	`Append`
	`Merge`	

`map`	*Zeichenfolge*	Ordnet einen Streamfeldnamen zu einer Datenbankspalte zu (nur gültig, wenn `write_mode` auf `Merge` eingestellt ist). Für eine Zusammenführung müssen alle Felder zugeordnet sein, damit sie exportiert werden. Feldnamen, die in der Datenbank nicht vorhanden sind, werden als neue Spalten hinzugefügt.
`key_fields`	*[Feld Feld ... Feld]*	Gibt an, dass das Streamfeld für "key" verwendet wird. Die `map`-Eigenschaft zeigt, welche Entsprechung in der Datenbank vorhanden ist.
`join`	`Database` `Add`	
`drop_existing_table`	*boolesch*	
`delete_existing_rows`	*boolesch*	
`default_string_size`	*ganze Zahl*	

Tabelle 201. Eigenschaften des Knotens "databaseexport" (Forts.).

Eigenschaften des Knotens databaseexport	**Datentyp**	**Eigenschaftsbeschreibung**
type		Strukturierte Eigenschaft, mit der das Schema festgelegt wird.
generate_import	*boolesch*	
use_custom_create_table_command	*boolesch*	Mit dem Slot *custom_create_table* ändern Sie den SQL-Standardbefehl CREATE TABLE.
custom_create_table_command	*Zeichenfolge*	Gibt eine Befehlszeichenfolge an, die statt des SQL-Standardbefehls CREATE TABLE verwendet werden soll.
use_batch	*boolesch*	Bei den folgenden Eigenschaften handelt es sich um erweiterte Optionen für das Massenladen von Datenbanken. Mit dem Wert "wahr" für Use_batch werden zeilenweise Commits zur Datenbank inaktiviert.
batch_size	*Zahl*	Gibt Anzahl der Datensätze an, die an die Datenbank gesendet werden sollen, bevor die Übertragung in den Speicher erfolgt.
bulk_loading	Off ODBC External	Gibt den Typ für das Massenladen an. Zusätzliche Optionen für ODBC und External sind unten aufgeführt.
not_logged	*boolesch*	
odbc_binding	Row Column	Geben Sie eine zeilen- oder spaltenweise Bindung für das Massenladen über ODBC an.
loader_delimit_mode	Tab Space Other	Geben Sie für das Massenladen über ein externes Programm das Trennzeichen an. Wählen Sie Other in Verbindung mit der Eigenschaft loader_other_delimiter zur Angabe von Trennzeichen, wie z. B. Komma (,).
loader_other_delimiter	*Zeichenfolge*	
specify_data_file	*boolesch*	Mit dem Flag "true" wird die unten genannte Eigenschaft data_file aktiviert, mit der Sie den Dateinamen und den Pfad für das Speichern beim Massenladen in die Datenbank angeben können.
data_file	*Zeichenfolge*	
specify_loader_program	*boolesch*	Mit dem Flag "true" wird die unten genannte Eigenschaft loader_program aktiviert, mit der Sie den Namen und den Speicherort eines externen Ladescripts oder Ladeprogramms angeben können.
loader_program	*Zeichenfolge*	

Tabelle 201. Eigenschaften des Knotens "databaseexport" (Forts.).

Eigenschaften des Knotens databaseexport	**Datentyp**	**Eigenschaftsbeschreibung**
gen_logfile	*boolesch*	Mit dem Flag "true" wird die unten genannte Eigenschaft logfile_name aktiviert, mit der Sie den Namen einer Datei zur Erzeugung eines Fehlerprotokolls auf dem Server angeben können.
logfile_name	*Zeichenfolge*	
check_table_size	*boolesch*	Mit dem Flag "true" wird die Tabellenprüfung ermöglicht, um sicherzustellen, dass die Zunahme der Größe der Datenbanktabelle der Anzahl der aus IBM SPSS Modeler exportierten Zeilen entspricht.
loader_options	*Zeichenfolge*	Legen Sie für das Ladeprogramm zusätzliche Argumente fest, z. B. -comment und -specialdir.
export_db_primarykey	*boolesch*	Gibt an, ob es sich bei einem bestimmten Feld um einen Primärschlüssel handelt.
use_custom_create_index_command	*boolesch*	Bei true wird hiermit benutzerdefinierte SQL für alle Indizes aktiviert.
custom_create_index_command	*Zeichenfolge*	Gibt den SQL-Befehl an, der zum Erstellen von Indizes verwendet wird, wenn benutzerdefinierte SQL aktiviert ist. (Dieser Wert kann für bestimmte Indizes außer Kraft gesetzt werden (siehe unten).)
indexes.INDEXNAME.fields		Erstellt bei Bedarf den angegebenen Index und listet die in diesen Index aufzunehmenden Feldnamen auf.
indexes.INDEXNAME.use_custom_create_⌂index_command	*boolesch*	Wird zum Aktivieren bzw. Inaktivieren der benutzerdefinierten SQL für einen bestimmten Index verwendet.
indexes.INDEXNAME.custom_create_command		Gibt die für den angegebenen Index verwendete benutzerdefinierte SQL an.
indexes.INDEXNAME.remove	*boolesch*	Bei true wird hiermit der angegebene Index aus der Menge der Indizes entfernt.
table_space	*Zeichenfolge*	Gibt den zu erstellenden Tabellenbereich an.
use_partition	*boolesch*	Gibt an, dass das Verteilungs-Hashfeld verwendet werden soll.
partition_field	*Zeichenfolge*	Gibt den Inhalt des Verteilungs-Hashfelds an.

Hinweis: Bei einigen Datenbanken können Sie angeben, dass Datenbanktabellen für den Export mit Komprimierung erstellt werden sollen (z. B. die Entsprechung von `CREATE TABLE MYTABLE (...) COMPRESS YES;` in SQL). Die Eigenschaften `use_compression` und `compression_mode` werden zur Unterstützung dieser Funktion wie folgt bereitgestellt.

Tabelle 202. Eigenschaften des Knotens "databaseexport" mit Komprimierungsfunktionen.

Eigenschaften des Knotens `databaseexport`	**Datentyp**	**Eigenschaftsbeschreibung**
`use_compression`	*boolesch*	Wenn auf `true` gesetzt, werden Tabellen für den Export mit Komprimierung erstellt.
`compression_mode`	`Row` `Page`	Legt das Komprimierungsniveau für SQL Server-Datenbanken fest.

	`Default`
	`Direct_Load_Operations`
	`All_Operations`
	`Basic`
	`OLTP`
	`Query_High`
	`Query_Low`
	`Archive_High`
	`Archive_Low`
		Legt das Komprimierungsniveau für Oracle-Datenbanken fest. Beachten Sie, dass für die Werte `OLTP`, `Query_High`, `Query_Low`, `Archive_High` und `Archive_Low` mindestens Oracle 11gR2 erforderlich ist.

Eigenschaften des Knotens "datacollectionexport"

Der IBM SPSS Data Collection-Exportknoten gibt Daten in dem von der Marktforschungssoftware IBM SPSS Data Collection verwendeten Format aus. Um diesen Knoten verwenden zu können, muss die IBM SPSS Data Collection Data Library installiert sein.

Tabelle 203. Eigenschaften des Knotens "datacollectionexport".

Eigenschaften des Knotens `datacollectionexport`	**Datentyp**	**Eigenschaftsbeschreibung**
`metadata_file`	*Zeichenfolge*	Name der zu exportierenden Metadatendatei.
`merge_metadata`	`Overwrite` `MergeCurrent`	
`enable_system_variables`	*boolesch*	Gibt an, ob die exportierte *.mdd*-Datei IBM SPSS Data Collection-Systemvariablen enthalten soll.
`casedata_file`	*Zeichenfolge*	Der Name der *.sav*-Datei, in die die Falldaten exportiert werden.
`generate_import`	*boolesch*	

Eigenschaften des Knotens "excelexport"

Der Excel-Exportknoten gibt Daten im Microsoft Excel-Format (*.xls*) aus. Optional können Sie auswählen, dass bei der Ausführung des Knotens Excel automatisch gestartet und die exportierte Datei geöffnet werden soll.

Tabelle 204. Eigenschaften des Knotens "excelexport".

Eigenschaften des Knotens **excelexport**	Datentyp	Eigenschaftsbeschreibung
full_filename	*Zeichenfolge*	
excel_file_type	Excel2003 Excel2007	
export_mode	Create Append	
inc_field_names	*boolesch*	Gibt an, ob Feldnamen in die erste Zeile des Arbeitsblattes eingefügt werden sollen.
start_cell	*Zeichenfolge*	Gibt die Startzelle für den Export an.
worksheet_name	*Zeichenfolge*	Name des zu schreibenden Arbeitsblattes.
launch_application	*boolesch*	Gibt an, ob Excel für die resultierende Datei aufgerufen werden soll. Beachten Sie, dass der Pfad für den Start von Excel im Dialogfeld "Hilfsanwendungen" (Menü "Extras", "Hilfsanwendungen") angegeben werden muss.
generate_import	*boolesch*	Gibt an, ob ein Excel-Importknoten generiert werden soll, der die exportierte Datendatei liest.

Eigenschaften des Knotens "outputfile"

Der Flatfile-Export gibt Daten in einer Textdatei mit Trennzeichen aus. Diese Vorgehensweise eignet sich für das Exportieren von Daten, die von anderen Analyse- oder Tabellenkalkulationsprogrammen gelesen werden sollen.

Tabelle 205. Eigenschaften des Knotens "outputfile".

Eigenschaften des Knotens **outputfile**	Datentyp	Eigenschaftsbeschreibung
full_filename	*Zeichenfolge*	Name der Ausgabedatei.
write_mode	Overwrite Append	
inc_field_names	*boolesch*	
use_newline_after_records	*boolesch*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*Zeichen*	
quote_mode	None Single Double Other	

Tabelle 205. Eigenschaften des Knotens "outputfile" (Forts.).

Eigenschaften des Knotens **outputfile**	Datentyp	Eigenschaftsbeschreibung
other_quote	*boolesch*	
generate_import	*boolesch*	
encoding	StreamDefault SystemDefault "UTF-8"	

Eigenschaften des Knotens "sasexport"

Mit dem SAS-Exportknoten werden Daten in das SAS-Format ausgegeben, die dann in SAS oder in SAS-kompatible Softwarepakete eingelesen werden können. Drei SAS-Dateiformate sind verfügbar: SAS für Windows/OS2, SAS für UNIX sowie SAS Version 7/8.

Tabelle 206. Eigenschaften des Knotens "sasexport".

Eigenschaften des Knotens **sasexport**	Datentyp	Eigenschaftsbeschreibung
format	Windows UNIX SAS7 SAS8	Beschriftungsfelder für die Eigenschaft "Variante".
full_filename	*Zeichenfolge*	
export_names	NamesAndLabels NamesAsLabels	Dient der Zuordnung von Feldnamen von IBM SPSS Modeler zu IBM SPSS Statistics- oder SAS-Variablennamen nach dem Export.
generate_import	*boolesch*	

Eigenschaften des Knotens "statisticsexport"

Der Statistikexportknoten gibt Daten im Format IBM SPSS Statistics *.sav* aus. Die *.sav*-Dateien können von IBM SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cachedateien in IBM SPSS Modeler verwendet.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften des Knotens "statisticsexport"" auf Seite 246.

Eigenschaften des Knotens "xmlexport"

Der XML-Exportknoten gibt Daten an eine Datei im XML-Format aus. Optional können Sie einen XML-Quellenknoten erstellen, um die exportierten Daten wieder in der Stream einzulesen.

Tabelle 207. Eigenschaften des Knotens "xmlexport".

Eigenschaften des Knotens `xmlexport`	Datentyp	Eigenschaftsbeschreibung
`full_filename`	*Zeichenfolge*	(erforderlich) Vollständiger Pfad und Dateiname der XML-Exportdatei.
`use_xml_schema`	*boolesch*	Legt fest, ob ein XML-Schema (XSD- oder DTD-Datei) für die Steuerung der Struktur der exportierten Daten verwendet wird.
`full_schema_filename`	*Zeichenfolge*	Vollständiger Pfad und Dateiname der zu verwendenden XSD- oder DTD-Datei. Erforderlich, wenn `use_xml_schema` auf "wahr" gesetzt ist.
`generate_import`	*boolesch*	Generiert einen XML-Quellenknoten, der die exportierten Daten wieder in den Stream einliest.
`records`	*Zeichenfolge*	XPath-Ausdruck, der die Datensatzgrenze angibt.
`map`	*Zeichenfolge*	Ordnet den Feldnamen der XML-Struktur zu.

Kapitel 18. IBM SPSS Statistics-Knoteneigenschaften

Eigenschaften des Knotens "statisticsimport"

Der Statistikdateiknoten liest Daten aus dem Dateiformat *.sav* ein, das von IBM SPSS Statistics verwendet wird, sowie in IBM SPSS Modeler gespeicherte Cache-Dateien, die ebenfalls dasselbe Format verwenden.

Tabelle 208. Eigenschaften des Knotens "statisticsimport".

Eigenschaften des Knotens **statisticsimport**	Datentyp	Eigenschaftsbeschreibung
`full_filename`	*Zeichenfolge*	Der vollständige Dateiname mit Pfad.
`import_names`	`NamesAndLabels` `LabelsAsNames`	Methode für die Behandlung von Variablennamen und -beschriftungen.
`import_data`	`DataAndLabels` `LabelsAsData`	Methode für die Behandlung von Werten und Beschriftungen.
`use_field_format_for_storage`	*boolesch*	Gibt an, ob IBM SPSS Statistics-Feldformatinformationen beim Import verwendet werden.

Eigenschaften des Knotens "statisticstransform"

Der Statistics-Transformationsknoten führt eine Auswahl von IBM SPSS Statistics-Syntaxbefehlen für Datenquellen in IBM SPSS Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Tabelle 209. Eigenschaften des Knotens "statisticstransform".

Eigenschaften des Knotens **statisticstransform**	Datentyp	Eigenschaftsbeschreibung
`syntax`	*Zeichenfolge*	
`check_before_saving`	*boolesch*	Überprüft die eingegebene Syntax vor dem Speichern der Einträge. Zeigt eine Fehlernachricht an, wenn die Syntax ungültig ist.
`default_include`	*boolesch*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.
`include`	*boolesch*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.
`new_name`	*Zeichenfolge*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.

Eigenschaften des Knotens "statisticsmodel"

Mithilfe des Statistics-Modellknotens können Sie Ihre Daten analysieren und bearbeiten, indem Sie IBM SPSS Statistics-Prozeduren ausführen, die PMML erzeugen. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eigenschaften des Knotens `statisticsmodel`	Datentyp	Eigenschaftsbeschreibung
`syntax`	*Zeichenfolge*	
`default_include`	*boolesch*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.
`include`	*boolesch*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.
`new_name`	*Zeichenfolge*	Weitere Informationen finden Sie im Thema „Eigenschaften des Knotens "filter"" auf Seite 105.

Eigenschaften des Knotens "statisticsoutput"

Mit dem Statistics-Ausgabeknoten können Sie eine IBM SPSS Statistics-Prozedur aufrufen, um Ihre IBM SPSS Modeler-Daten zu analysieren. Es stehen zahlreiche IBM SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Tabelle 210. Eigenschaften des Knotens "statisticsoutput".

Eigenschaften des Knotens `statisticsoutput`	Datentyp	Eigenschaftsbeschreibung
`mode`	`Dialog` `Syntax`	Wählt die Option "IBM SPSS Statistics-Dialogfeld" oder Syntaxeditor aus
`syntax`	*Zeichenfolge*	
`use_output_name`	*boolesch*	
`output_name`	*Zeichenfolge*	
`output_mode`	`Screen` `File`	
`full_filename`	*Zeichenfolge*	

`file_type`	`HTML`
	`SPV`
	`SPW`	

Eigenschaften des Knotens "statisticsexport"

Der Statistikexportknoten gibt Daten im Format IBM SPSS Statistics *.sav* aus. Die *.sav*-Dateien können von IBM SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cachedateien in IBM SPSS Modeler verwendet.

Tabelle 211. Eigenschaften des Knotens "statisticsexport".

Eigenschaften des Knotens **statisticsexport**	Datentyp	Eigenschaftsbeschreibung
full_filename	*Zeichenfolge*	
launch_application	*boolesch*	
export_names	NamesAndLabels NamesAsLabels	Dient der Zuordnung von Feldnamen von IBM SPSS Modeler zu IBM SPSS Statistics- oder SAS-Variablennamen nach dem Export.
generate_import	*boolesch*	

Kapitel 19. Superknoteneigenschaften

In den folgenden Tabellen werden die für Superknoten spezifischen Eigenschaften beschrieben. Beachten Sie, dass allgemeine Knoteneigenschaften auch für Superknoten gelten.

Tabelle 212. Eigenschaften von Endsuperknoten.

Eigenschaftsname	Eigenschaftstyp/Liste der Werte	Eigenschaftsbeschreibung
`execute_method`	`Script` `Normal`	
`script`	*Zeichenfolge*	
`script_language`	`Python` `Legacy`	Legt die Scriptsprache für das Superknotenscript fest.

Superknotenparameter

Mithilfe der Funktionen, die zum Ändern von Streamparametern verwendet werden, können Sie Scripts zum Erstellen oder Festlegen von Superknotenparametern verwenden. Weitere Informationen finden Sie im Thema „Stream-, Sitzungs- und Superknotenparameter" auf Seite 42.

Festlegen von Eigenschaften für gekapselte Knoten

Zum Festlegen der Eigenschaften bei Knoten im Superknoten müssen Sie auf das Diagramm zugreifen, das diesem Superknoten gehört, und dann die verschiedenen `find`-Methoden (wie `findByName()` und `findByID()`) verwenden, um die Knoten zu suchen. Beispiel: In einem Superknotenscript, das einen einzigen Knotentyp umfasst:

```
supernode = modeler.script.supernode()
diagram = supernode.getCompositeProcessorDiagram()
# Find the type node within the supernode internal diagram
typenode = diagram.findByName("type", None)
typenode.setKeyedProperty("direction", "Drug", "Input")
typenode.setKeyedProperty("direction", "Age", "Target")
```

Einschränkungen bei Superknotenscripts. Superknoten können andere Streams nicht bearbeiten und den aktuellen Stream nicht ändern.

Anhang A. Knotennamenreferenz

In diesem Abschnitt erhalten Sie eine Referenz für die Scriptnamen der Knoten in IBM SPSS Modeler.

Modellnuggetnamen

Modellnuggets (auch als "generierte Modelle" bezeichnet) können ebenso wie Knoten- und Ausgabeobjekte nach Typ referenziert werden. In der folgenden Tabelle werden die Referenznamen für Modellobjekte aufgeführt.

Beachten Sie, dass diese Namen speziell zur Referenzierung von Modellnuggets in der Modellpalette (in der rechten oberen Ecke des IBM SPSS Modeler-Fensters) verwendet werden. Zur Referenzierung von Modellknoten, die zu Scoring-Zwecken zu einem Stream hinzugefügt wurden, wird ein anderes Set von Namen mit dem Präfix `apply...` verwendet. Weitere Informationen finden Sie in Kapitel 14, „Eigenschaften von Modellnuggetknoten", auf Seite 181.

Hinweis: Unter normalen Umständen wird die Referenzierung von Modellen sowohl anhand des Namens *als auch* anhand des Typs empfohlen, um Verwirrungen zu vermeiden.

Tabelle 213. Namen von Modellnuggets (Modellierungspalette).

Modellname	**Modell**
anomalydetection	Anomalie
apriori	Apriori
autoclassifier	Automatisches Klassifikationsmerkmal
autocluster	Automatisches Clustering
autonumeric	Auto-Numerisch
bayesnet	Bayes-Netz
c50	C5.0
carma	Carma
cart	C&R-Baum
chaid	CHAID
coxreg	Cox-Regression
decisionlist	Entscheidungsliste
Diskriminanz	Diskriminanz
Faktor	Faktor/PCA
featureselection	Merkmalauswahl
genlin	Verallgemeinerte lineare Regression
glmm	GLMM
kmeans	K-Means
knn	*k*-Nächste-Nachbarn
kohonen	Kohonen
linear	Linear
logreg	Logistische Regression
neuralnetwork	Netz

Tabelle 213. Namen von Modellnuggets (Modellierungspalette) (Forts.).

Modellname	**Modell**
quest	QUEST
regression	Lineare Regression
sequence	Sequenz
slrm	Lernfähiges Antwortmodell
statisticsmodel	IBM SPSS Statistics-Modell
svm	Support Vector Machine
timeseries	Zeitreihen
twostep	Two Step

Tabelle 214. Namen von Modellnuggets (Datenbankmodellierungspalette).

Modellname	**Modell**
db2imcluster	IBM ISW-Clustering
db2imlog	IBM ISW Logistische Regression
db2imnb	IBM ISW Naive Bayes
db2imreg	IBM ISW-Regression
db2imtree	IBM ISW-Entscheidungsbaum
msassoc	MS-Assoziationsregeln
msbayes	MS Naive Bayes
mscluster	MS-Clustering
mslogistic	MS - Logistische Regression
msneuralnetwork	MS - Neuronales Netz
msregression	MS - Lineare Regression
mssequencecluster	MS-Sequenzclustering
mstimeseries	MS Time Series
mstree	MS-Entscheidungsbaum
netezzabayes	Netezza-Bayes-Netz
netezzadectree	Netezza-Entscheidungsbaum
netezzadivcluster	Netezza - Divisives Clustering
netezzaglm	Verallgemeinertes lineares Netezza-Modell
netezzakmeans	Netezza-K-Means
netezzaknn	Netezza-KNN
netezzalineregression	Netezza - Lineare Regression
netezzanaivebayes	Netezza - Naive Bayes
netezzapca	Netezza-PCA
netezzaregtree	Netezza-Regressionsbaum
netezzatimeseries	Netezza-Zeitreihe
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Oracle Decision Tree
oraglm	Oracle GLM

Tabelle 214. Namen von Modellnuggets (Datenbankmodellierungspalette) (Forts.).

Modellname	Modell
orakmeans	Oracle *k*-Means
oranb	Oracle Naive Bayes
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

Vermeidung doppelter Modellnamen

Bei der Verwendung von Scripts zur Bearbeitung generierter Modelle sollten Sie sich bewusst sein, dass das Zulassen doppelter Modellnamen zu mehrdeutigen Referenzen führen kann. Um dies zu vermeiden, sollten bei der Scripterstellung eindeutige Namen für die generierten Modelle erforderlich sein.

So legen Sie Optionen für doppelte Modellnamen fest:

1. Wählen Sie die folgenden Befehle aus den Menüs aus:

   Extras > **Benutzeroptionen**
2. Klicken Sie auf die Registerkarte **Benachrichtigungen**.
3. Wählen Sie die Option **Bisheriges Modell ersetzen**, um die Vergabe doppelter Namen für generierte Modelle zu beschränken.

Das Verhalten der Scriptausführung kann zwischen SPSS Modeler und IBM SPSS Collaboration and Deployment Services variieren, wenn mehrdeutige Modellverweise vorliegen. Der SPSS Modeler-Client beinhaltet die Option "Bisheriges Modell ersetzen", bei dem automatisch Modelle mit demselben Namen ersetzt werden (z. B. wenn ein Script in mehreren Iterationen eine Schleife durchläuft und jedes Mal ein anderes Modell erstellt). Diese Option steht jedoch nicht zur Verfügung, wenn dasselbe Script in IBM SPSS Collaboration and Deployment Services ausgeführt wird. Sie können diese Situation vermeiden, indem Sie entweder das in den einzelnen Iterationen generierte Modell umbenennen, um mehrdeutige Verweise auf Modelle zu vermeiden, oder indem Sie das aktuelle Modell vor dem Ende der Schleife löschen (z. B. durch Hinzufügen der Anweisung `clear generated palette`).

Namen der Ausgabetypen

In der folgenden Tabelle werden alle Ausgabeobjekttypen und die Knoten, von denen Sie erstellt werden, aufgelistet. Eine vollständige Liste der für die einzelnen Ausgabeobjekttypen verfügbaren Exportformate finden Sie in der Eigenschaftsbeschreibung für den Knoten, der den Ausgabetyp erstellt. Diese Beschreibung finden Sie in „Allgemeine Eigenschaften von Diagrammknoten" auf Seite 121 und Kapitel 16, „Eigenschaften von Ausgabeknoten", auf Seite 221.

Tabelle 215. Ausgabeobjekttypen und die Knoten, von denen sie erstellt werden.

Ausgabeobjekttyp	Knoten
analysisoutput	Analyse
collectionoutput	Sammlung
dataauditoutput	Data Audit
distributionoutput	Verteilung
evaluationoutput	Evaluation
histogramoutput	Histogramm
matrixoutput	Matrix
meansoutput	Mittelwerte

Tabelle 215. Ausgabeobjekttypen und die Knoten, von denen sie erstellt werden (Forts.).

Ausgabeobjekttyp	**Knoten**
multiplotoutput	Multiplot
plotoutput	Diagramm
qualityoutput	Qualität
reportdocumentoutput	Dieser Objekttyp stammt nicht aus einem Knoten; es handelt sich um die von einem Projektbericht erstellte Ausgabe.
reportoutput	Bericht
statisticsprocedureoutput	Statistics-Ausgabe
statisticsoutput	Statistics
tableoutput	Tabelle
timeplotoutput	Zeitdiagramm
weboutput	Internet

Anhang B. Migration von traditionellem Scripting zu Python-Scripting

Übersicht über die Migration traditioneller Scripts

In diesem Abschnitt erhalten Sie eine Zusammenfassung der Unterschiede zwischen Python-Scripting und traditionellem Scripting in IBM SPSS Modeler sowie Informationen zur Migration von traditionellen Scripts in Python-Scripts. In diesem Abschnitt finden Sie eine Liste der traditionellen SPSS Modeler-Standardbefehle und der entsprechenden Python-Befehle.

Allgemeine Unterschiede

Traditionelles Scripting beruht stark auf Betriebssystembefehlsscripts. Traditionelles Scripting ist zeilenorientiert. Obwohl es einige Blockstrukturen wie z. B. `if...then...else...endif` und `for...endfor` gibt, hat eine Einrückung in der Regel keine Bedeutung.

Bei Python-Scripting ist die Einrückung von Bedeutung. Zeilen, die zum selben logischen Block gehören, müssen gleich weit eingerückt sein.

Anmerkung: Darauf müssen Sie beim Kopieren und Einfügen von Python-Code achten. Eine Zeile, die mit Tabstopps eingerückt ist, könnte im Editor genauso aussehen wie eine Zeile, die mit Leerzeichen eingerückt ist. Das Python-Script generiert jedoch einen Fehler, da diese Zeilen nicht als gleich weit eingerückt gelten.

Scriptingkontext

Der Scriptingkontext definiert die Umgebung, in der das Script ausgeführt wird, z. B. den Stream oder den Superknoten, der das Script ausführt. Bei traditionellem Scripting ist der Kontext implizit, d. h. es wird angenommen, dass alle Knotenreferenzen in einem Stream-Script in dem Stream liegen, der das Script ausführt.

Bei Python-Scripting wird der Scriptingkontext explizit über das Modul `modeler.script` angegeben. Beispiel: ein Python-Stream-Script kann mit dem folgenden Code auf den Stream zugreifen, der das Script ausführt:

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

Funktionen im Zusammenhang mit Streams können dann über das zurückgegebene Objekt aufgerufen werden.

Befehle und Funktionen

Traditionelles Scripting ist befehlsorientiert. Das heißt, dass jede Zeile des Scripts typischerweise mit dem auszuführenden Befehl beginnt und dann die Parameter folgen. Beispiel:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python verwendet Funktionen, die üblicherweise über ein Objekt (Modul, Klasse oder Objekt) aufgerufen werden, das die Funktion definiert. Beispiel:

```
stream = modeler.script.stream()
typenode = stream.findByName("type", "Type")
filternode = stream.findByName("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Literale und Kommentare

Einige Literal- und Kommentarbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 216. Zuordnung von traditionellem Scripting zu Python-Scripting für Literale und Kommentare.

Traditionelles Scripting	**Python-Scripting**
Ganzzahl, z. B. `4`	Gleich
Gleitkommazahl, z. B. `0.003`	Gleich
Zeichenfolgen in einfachen Anführungszeichen, z. B. `'Hallo'`	Gleich **Anmerkung:** Zeichenfolgeliteralen mit Nicht-ASCII-Zeichen muss ein `u` vorangestellt werden, damit sie als Unicode dargestellt werden.
Zeichenfolgen in doppelten Anführungszeichen, z. B. `"Nochmal Hallo"`	Gleich **Anmerkung:** Zeichenfolgeliteralen mit Nicht-ASCII-Zeichen muss ein `u` vorangestellt werden, damit sie als Unicode dargestellt werden.

Lange Zeichenfolgen, z. B.
`"""Dies ist eine`
`Zeichenfolge, die mehrere Zeilen`
`umfasst"""`
	Gleich

Listen, z. B. `[1 2 3]`	`[1, 2, 3]`
Variablenreferenzen, z. B. `set x = 3`	`x = 3`

Zeilenfortsetzung (\), z. B.
`set x = [1 2 \`
`  3 4]`
	`x = [ 1, 2,\`
	`  3, 4]`

Blockkommentar, z. B.
`/* Dies ist ein langer Kommentar`
`in einer Zeile. */`
	`/* Dies ist ein langer Kommentar`
	`in einer Zeile. */`

Zeilenkommentar, z. B. `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Operatoren

Einige Operatorbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 217. Zuordnung von traditionellem Scripting zu Python-Scripting für Operatoren.

Traditionelles Scripting	**Python-Scripting**

`ZAHL1 + ZAHL2`
`LISTE + ELEMENT`
`LISTE1 + LISTE2`
	`ZAHL1 + ZAHL2`
	`LIST.append(ELEMENT)`
	`LIST1.extend(LISTE2)`

`ZAHL1 - ZAHL2` `LISTE - ELEMENT`	`ZAHL1 - ZAHL2` `LIST.remove(ELEMENT)`
`ZAHL1 * ZAHL2`	`ZAHL1 * ZAHL2`

Tabelle 217. Zuordnung von traditionellem Scripting zu Python-Scripting für Operatoren (Forts.).

Traditionelles Scripting	Python-Scripting
ZAHL1 / ZAHL2	ZAHL1 / ZAHL2
= ==	==
/= /==	!=
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(AUSDR)	and or not AUSDR

Bedingte Befehle und Schleifenbefehle

Für einige bedingte Befehle und Schleifenbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 218. Zuordnung von traditionellem Scripting zu Python-Scripting für bedingte Befehle und Schleifenbefehle.

Traditionelles Scripting	Python-Scripting

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	ODER
	VAR = INT1
	while VAR <= INT2:
	...
	VAR += 1

for VAR in LISTE ... endfor	for VAR in LISTE: ...
for VAR in_fields_to KNOTEN ... endfor	for VAR in KNOTEN.getInputDataModel(): ...
for VAR in_fields_at KNOTEN ... endfor	for VAR in KNOTEN.getOutputDataModel(): ...

if...then
...
elseif...then
...
else
...
endif
	if ...:
	...
	elif ...:
	...
	else:
	...

with TYPE OBJECT ... endwith	Keine Entsprechung

Tabelle 218. Zuordnung von traditionellem Scripting zu Python-Scripting für bedingte Befehle und Schleifenbefehle (Forts.).

Traditionelles Scripting	Python-Scripting
`var VAR1`	Variablendeklaration nicht erforderlich

Variablen

Bei traditionellem Scripting werden Variablen deklariert, bevor sie referenziert werden. Beispiel:

```
var mynode
set mynode = create typenode at 96 96
```

Bei Python-Scripting werden Variablen bei ihrer ersten Referenzierung erstellt. Beispiel:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

Bei traditionellem Scripting müssen Referenzen auf Variablen explizit mit dem Operator ^ entfernt werden. Beispiel:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Wie bei den meisten Scriptsprachen ist dies bei Python-Scripting nicht erforderlich. Beispiel:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Knoten-, Ausgabe- und Modelltypen

Bei traditionellem Scripting wird für die unterschiedlichen Objekttypen (Knoten, Ausgabe und Modell) in der Regel der Typ an den Objekttyp angehängt. Beispiel: der Knoten "derive" hat den Typ `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

Die IBM SPSS Modeler-API in Python schließt das Suffix `node` nicht ein, sodass der Knoten "derive" den Typ `derive` hat. Beispiel:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

Typnamen bei traditionellem Scripting und bei Python-Scripting unterscheiden sich nur darin, dass bei Python-Scripting das Typsuffix fehlt.

Eigenschaftsnamen

Eigenschaftsnamen sind bei traditionellem und bei Python-Scripting gleich. Beispiel: im Variablendateiknoten lautet die Eigenschaft, die die Dateiposition definiert, in beiden Scripting-Umgebungen `full_filename`.

Knotenreferenzen

Viele traditionelle Scripts verwenden eine implizite Suche, um den zu ändernden Knoten zu suchen und darauf zuzugreifen. Beispiel: die folgenden Befehle durchsuchen den aktuellen Stream nach einem Typknoten mit der Beschriftung "Type" und legen dann für die Richtung (Modellierungsrolle) das Feld "Age" als Eingabe und das Feld "Drug" als Ziel (vorherzusagender Wert) fest:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

Bei Python-Scripting müssen Knotenobjekte explizit gesucht werden, bevor die Funktion zum Festlegen des Eigenschaftswert aufgerufen wird. Beispiel:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Anmerkung: In diesem Fall muss "Target" in Anführungszeichen für Zeichenfolgen stehen.

Python-Scripts können alternativ die Aufzählung `ModelingRole` im Paket `modeler.api` verwenden.

Obwohl Scripts bei Python-Scripting länger sein können, führt dies zu einer besseren Laufzeitleistung, da die Suche nach dem Knoten in der Regel nur einmal erfolgt. Beim Beispiel mit traditionellem Scripting wird die Suche nach dem Knoten bei jedem Befehl ausgeführt.

Die Suche nach Knoten anhand der ID wird ebenfalls unterstützt. (Die Knoten-ID ist auf der Registerkarte "Anmerkungen" des Knotendialogs zu sehen.) Beispiel bei traditionellem Scripting:

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

Das folgende Script zeigt dasselbe Beispiel bei Python-Scripting:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Abrufen und Festlegen von Eigenschaften

Traditionelles Scripting verwendet den Befehl `set` zum Festlegen eines Wertes. Der Ausdruck nach dem Befehl `set` kann eine Eigenschaftsdefinition sein. Das folgende Script zeigt zwei mögliche Scriptformate zum Festlegen einer Eigenschaft:

```
set <Knotenreferenz>.<Eigenschaft> = <Wert>
set <Knotenreferenz>.<verschlüsselte_Eigenschaft>.<Schlüssel> = <Wert>
```

Bei Python-Scripting wird dasselbe Ergebnis mit den Funktionen `setPropertyValue()` und `setKeyedPropertyValue()` erreicht. Beispiel:

```
Objekt.setPropertyValue(Eigenschaft, Wert)
Objekt.setKeyedPropertyValue(verschlüsselte_Eigenschaft, Schlüssel, Wert)
```

Bei traditionellem Scripting kann mit dem Befehl `get` auf Eigenschaftswerte zugegriffen werden. Beispiel:

```
var n v
set n = get node :filternode
set v = ^n.name
```

Bei Python-Scripting wird dasselbe Ergebnis mit der Funktion `setPropertyValue()` erreicht. Beispiel:

```
n = stream.findByName("filter", None)
v = n.getPropertyValue("name")
```

Bearbeiten von Streams

Bei traditionellen Scripting wird der Befehl `create` zum Erstellen eines neuen Knotens verwendet. Beispiel:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

Bei Python-Scripting gibt es in Streams verschiedene Methoden zur Erstellung von Knoten. Beispiel:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

Bei traditionellen Scripting wird der Befehl connect zum Herstellen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
connect ^agg to ^select
```

Bei Python-Scripting wird die Methode link zum Herstellen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
stream.link(agg, select)
```

Bei traditionellen Scripting wird der Befehl disconnect zum Entfernen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
disconnect ^agg from ^select
```

Bei Python-Scripting wird die Methode unlink zum Entfernen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
stream.unlink(agg, select)
```

Bei traditionellen Scripting wird der Befehl position zur Positionierung von Knoten im Streamerstellungsbereich oder zwischen anderen Knoten verwendet. Beispiel:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

Bei Python-Scripting wird dasselbe Ergebnis mit zwei separaten Methoden erreicht: setXYPosition und setPositionBetween. Beispiel:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Knotenoperationen

Einige Befehle für Knotenoperationen, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 219. Zuordnung von traditionellem Scripting zu Python-Scripting für Knotenoperationen.

Traditionelles Scripting	**Python-Scripting**
create *Knotenspezifikation* at x y	*Stream*.create(*Typ*, *Name*) *Stream*.createAt(*Typ*, *Name*, x, y) *Stream*.createBetween(*Typ*, *Name*, preNode, postNode) *Stream*.createModelApplier(*Modell*, *Name*)
connect *Ausgangsknoten* to *Zielknoten*	*Stream*.link(*Ausgangsknoten*, *Zielknoten*)
delete *Knoten*	*Stream*.delete(*Knoten*)
disable *Knoten*	*Stream*.setEnabled(*Knoten*, False)
enable *Knoten*	*Stream*.setEnabled(*Knoten*, True)
disconnect *Ausgangsknoten* from *Zielknoten*	*Stream*.unlink(*Ausgangsknoten*, *Zielknoten*) *Stream*.disconnect(*Knoten*)
duplicate *Knoten*	*Knoten*.duplicate()
execute *Knoten*	*Stream*.runSelected(*Knoten*, *Ergebnisse*) *Stream*.runAll(*Ergebnisse*)
flush *Knoten*	*Knoten*.flushCache()
position *Knoten* at x y	*Knoten*.setXYPosition(x, y)
position *Knoten* between *Knoten1* and *Knoten2*	*Knoten*.setPositionBetween(*Knoten1*, *Knoten2*)
rename *Knoten* as *Name*	*Knoten*.setLabel(*Name*)

Verwendung von Schleifen

Bei traditionellem Scripting werden es zwei Hauptschleifenoptionen unterstützt:

- *Gezählte Schleifen*, bei denen eine Indexvariable sich zwischen zwei ganzzahligen Grenzen bewegt.
- *Sequenzschleifen*, die eine Folge von Werten in einer Schleife durchlaufen und den aktuellen Wert an die Schleifenvariable binden.

Das folgende Script ist ein Beispiel für eine gezählte Schleife bei traditionellem Scripting:

```
for i from 1 to 10
 println ^i
endfor
```

Das folgende Script ist ein Beispiel für eine Sequenzschleife bei traditionellem Scripting:

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

Es können auch andere Typen von Schleifen verwendet werden:

- Durchlaufen der Modelle in der Modellpalette oder der Ausgaben in der Ausgabepalette.
- Durchlaufen der Felder, die in einen Knoten eintreten oder aus einem Knoten austreten.

Python-Scripting unterstützt auch unterschiedliche Schleifentypen. Das folgende Script ist ein Beispiel für eine gezählte Schleife bei Python-Scripting:

```
i = 1
while i <= 10:
 print i
 i += 1
```

Das folgende Script ist ein Beispiel für eine Sequenzschleife bei Python-Scripting:

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

Die Sequenzschleife ist sehr flexibel. Wenn sie mit API-Methoden von IBM SPSS Modeler kombiniert wird, kann sie die Mehrzahl der Anwendungsfälle bei traditionellem Scripting unterstützen. Das folgende Beispiel zeigt, wie mit einer Sequenzschleife bei Python-Scripting die Felder durchlaufen werden können, die aus einem Knoten austreten:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

Streams ausführen

Während der Streamausführung werden generierte Modelle oder Ausgabeobjekte zu einem der Objektmanager hinzugefügt. Bei traditionellen Scripting muss das Script entweder die erstellten Objekte im Objektmanager finden oder auf die zuletzt generierte Ausgabe aus dem Knoten zugreifen, der die Ausgabe erstellt hat.

Die Streamausführung in Python unterscheidet sich darin, dass alle von der Ausführung generierten Modell- oder Ausgabeobjekte in einer Liste zurückgegeben werden, die an die Ausführungsfunktion übergeben wird. Dadurch kann leichter auf die Ergebnisse der Streamausführung zugegriffen werden.

Traditionelles Scripting unterstützt drei Befehle zur Streamausführung:

- `execute_all` - führt alle ausführbaren Endknoten im Stream aus.
- `execute_script` - führt das Stream-Script unabhängig von der Einstellung der Scriptausführung aus.
- `execute` *`Knoten`* - führt den angegebenen Knoten aus.

Python-Scripting unterstützt eine ähnliche Gruppe von Funktionen:

- *`Stream`*`.runAll(`*`Ergebnisliste`*`)` - führt alle ausführbaren Endknoten im Stream aus.
- *`Stream`*`.runScript(`*`Ergebnisliste`*`)` - führt das Stream-Script unabhängig von der Einstellung der Scriptausführung aus.
- *`Stream`*`.runSelected(`*`Knotenarray`*`, `*`Ergebnisliste`*`)` - führt das angegebene Set von Knoten in der aufgelisteten Reihenfolge auf.
- *`Knoten`*`.run(`*`Ergebnisliste`*`)` - führt den angegebenen Knoten aus.

Bei traditionellem Scripting kann eine Streamausführung mit dem Befehl `exit` und einem optionalen ganzzahligen Code beendet werden. Beispiel:

```
exit 1
```

Bei Python-Scripting wird dasselbe Ergebnis mit dem folgenden Script erreicht:

```
modeler.script.exit(1)
```

Zugriff auf Objekte über das Dateisystem und das Repository

Bei traditionellem Scripting können Sie einen vorhandenen Stream, ein vorhandenes Modell oder ein vorhandenes Ausgabeobjekt mit dem Befehl `open` öffnen: Beispiel:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

Bei Python-Scripting gibt es die Klasse `TaskRunner`, auf die von der Sitzung zugegriffen werden kann und mit der ähnliche Tasks ausgeführt werden können. Beispiel:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Wenn Sie bei traditionellen Scripting ein Objekt speichern wollen, können Sie den Befehl `save` verwenden. Beispiel:

```
save stream s as "c:/my streams/new_modeling.str"
```

Die entsprechende Vorgehensweise bei einem Python-Script ist die Verwendung der Klasse `TaskRunner`. Beispiel:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Auf IBM SPSS Collaboration and Deployment Services Repository basierende Operationen werden bei traditionellen Scripting über die Befehle `retrieve` und `store` unterstützt. Beispiel:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

Bei Python-Scripting wird die entsprechende Funktionalität über das der Sitzung zugeordnete Repository-Objekt abgerufen. Beispiel:

```
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)
```

Anmerkung: Für den Repository-Zugriff muss die Sitzung mit einer gültigen Repository-Verbindung konfiguriert worden sein.

Streamoperationen

Für einige Befehle für Streamoperationen, die üblicherweise in IBM SPSS Modeler verwendet werden, gibt es funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 220. Zuordnung von traditionellem Scripting zu Python-Scripting für Streamoperationen.

Traditionelles Scripting	Python-Scripting
`create stream STANDARDDATEINAME`	*Task-Runner*.createStream(*Name*, *autoVerbindung*, *autoVerwaltung*)
`close stream`	*Stream*.close()
`clear stream`	*Stream*.clear()
`get stream Stream`	Keine Entsprechung
`load stream Pfad`	Keine Entsprechung
`open stream Pfad`	*Task-Runner*.openStreamFromFile(*Pfad*, *autoVerwaltung*)
`save Stream as Pfad`	*Task-Runner*.saveStreamToFile(*Stream*, *Pfad*)
`retreive stream Pfad`	*Repository*.retreiveStream(*Pfad*, *Version*, *Beschriftung*, *autoVerwaltung*)
`store Stream as Pfad`	*Repository*.storeStream(*Stream*, *Pfad*, *Beschriftung*)

Modelloperationen

Für einige Befehle für Modelloperationen, die in IBM SPSS Modeler häufig verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 221. Zuordnung von traditionellem Scripting zu Python-Scripting für Modelloperationen.

Traditionelles Scripting	Python-Scripting
`open model Pfad`	*Task-Runner*.openModelFromFile(*Pfad*, *autoVerwaltung*)
`save Modell as Pfad`	*Task-Runner*.saveModelToFile(*Modell*, *Pfad*)
`retrieve model Pfad`	*Repository*.retrieveModel(*Pfad*, *Version*, *Beschriftung*, *autoVerwaltung*)
`store Modell as Pfad`	*Repository*.storeModel(*Modell*, *Pfad*, *Beschriftung*)

Dokumentausgabeoperationen

Für einige Befehle für Dokumentausgabeoperationen, die in IBM SPSS Modeler häufig verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 16 in Python-Scripts zu konvertieren.

Tabelle 222. Zuordnung von traditionellem Scripting zu Python-Scripting für Dokumentausgabeoperationen.

Traditionelles Scripting	Python-Scripting
`open output Pfad`	*Task-Runner*.openDocumentFromFile(*Pfad*, *autoVerwaltung*)
`save Ausgabe as Pfad`	*Task-Runner*.saveDocumentToFile(*Ausgabe*, *Pfad*)
`retrieve output Pfad`	*Repository*.retrieveDocument(*Pfad*, *Version*, *Beschriftung*, *autoVerwaltung*)

Tabelle 222. Zuordnung von traditionellem Scripting zu Python-Scripting für Dokumentausgabeoperationen (Forts.).

Traditionelles Scripting	Python-Scripting
`store Ausgabe as Pfad`	`Repository.storeDocument(Ausgabe, Pfad, Beschriftung)`

Weitere Unterschiede zwischen traditionellem und Python-Scripting

Traditionelle Scripts bieten Unterstützung zur Manipulation von IBM SPSS Modeler-Projekten. Python-Scripting unterstützt diese Funktion derzeit nicht.

Traditionelles Scripting bietet eine gewisse Unterstützung beim Laden von *Statusobjekten* (Kombinationen von Streams und Modellen). Statusobjekte werden seit IBM SPSS Modeler 8.0 nicht weiter unterstützt. Python-Scripting unterstützt keine Statusobjekte.

Python-Scripting bietet die folgenden zusätzlichen Funktionen, die bei traditionellem Scripting nicht verfügbar sind:

- Klassen- und Funktionsdefinitionen
- Fehlerbehandlung
- Fortgeschrittenere Ein-/Ausgabe-Unterstützung
- Externe und Fremdanbieter-Module

Bemerkungen

Diese Informationen wurden für weltweit angebotene Produkte und Dienstleistungen erarbeitet.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
USA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corp in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Markten oder eingetragene Marken der Intel Corporation oder der Tochtergesellschaften des Unternehmens in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA, anderen Ländern oder beidem.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z