IBM SPSS Modeler 15 Handbuch für Skripterstellung und Automatisierung

Hinweis: Lesen Sie zunächst die allgemeinen Informationen unter Hinweise auf S. , bevor Sie dieses Informationsmaterial sowie das zugehörige Produkt verwenden.

Diese Ausgabe bezieht sich auf IBM SPSS Modeler 15 und alle nachfolgenden Versionen sowie Anpassungen, sofern dies in neuen Ausgaben nicht anders angegeben ist.

Vorwort

IBM® SPSS® Modeler ist die auf Unternehmensebene einsetzbare Data-Mining-Workbench von IBM Corp.. Mit SPSS Modeler können Unternehmen und Organisationen die Beziehungen zu ihren Kunden bzw. zu den Bürgern durch ein tief greifendes Verständnis der Daten verbessern. Organisationen benutzen die mithilfe von SPSS Modeler gewonnenen Erkenntnisse zur Bindung profitabler Kunden, zur Ermittlung von Cross-Selling-Möglichkeiten, zur Gewinnung neuer Kunden, zur Ermittlung von Betrugsfällen, zur Reduzierung von Risiken und zur Verbesserung der Verfügbarkeit öffentlicher Dienstleistungen.

Die visuelle Benutzeroberfläche von SPSS Modeler erleichtert die Anwendung des spezifischen Geschäftswissens der Benutzer, was zu leistungsstärkeren Vorhersagemodellen führt und die Zeit bis zur Lösungserstellung verkürzt. SPSS Modeler bietet zahlreiche Modellierungsverfahren, beispielsweise Algorithmen für Vorhersage, Klassifizierung, Segmentierung und Assoziationserkennung. Nach der Modellerstellung ermöglicht IBM® SPSS® Modeler Solution Publisher die unternehmensweite Bereitstellung für Entscheidungsträger oder in einer Datenbank.

Über IBM Business Analytics

IBM Business Analytics-Software bietet vollständige, einheitliche und genaue Informationen, auf die Entscheidungsträger vertrauen, um die Unternehmensleistung zu steigern. Ein umfassendes Portfolio von Anwendungen für Unternehmensinformationen, Vorhersageanalysen, Verwaltung der Finanzleistung und Strategie sowie Analysen bietet sofort klare und umsetzbare Einblicke in die aktuelle Leistung und ermöglicht die Vorhersage zukünftiger Ergebnisse. In Kombination mit umfassenden Branchenlösungen, bewährten Vorgehensweisen und professionellen Dienstleistungen können Unternehmen jeder Größe optimale Produktivität erreichen, die Entscheidungsfindung zuverlässig automatisieren und bessere Ergebnisse erzielen.

Als Teil dieses Portfolios unterstützt die IBM SPSS Predictive Analytics-Software Unternehmen dabei, zukünftige Ereignisse vorherzusagen und aktiv auf diese Erkenntnisse zu reagieren, um bessere Geschäftsergebnisse zu erzielen. Kunden aus den Bereichen Wirtschaft, Behörden und Bildung aus aller Welt verlassen sich auf die IBM SPSS-Technologie. Sie bringt Ihnen beim Gewinnen, Halten und Ausbauen neuer Kundenbeziehungen einen Wettbewerbsvorteil und verringert gleichzeitig das Betrugs- sowie andere Risiken. Durch Integration der IBM SPSS-Software in den täglichen Betrieb können diese Unternehmen qualifizierte Vorhersagen treffen und dadurch die Entscheidungsfindung so ausrichten und automatisieren, dass Geschäftsziele erreicht werden und ein messbarer Wettbewerbsvorteil entsteht. Wenn Sie weitere Informationen wünschen oder einen Mitarbeiter kontaktieren möchten, ist dies unter *http://www.ibm.com/spss* möglich.

Technischer Support

Kunden mit Wartungsvertrag können den technischen Support in Anspruch nehmen. Kunden können sich an den technischen Support wenden, wenn sie Hilfe bei der Arbeit mit IBM Corp.-Produkten oder bei der Installation in einer der unterstützten Hardware-Umgebungen benötigen. Die Kontaktdaten des Technischen Supports finden Sie auf der IBM Corp.-Website

unter *http://www.ibm.com/support*. Sie müssen bei der Kontaktaufnahme Ihren Namen, Ihre Organisation und Ihre Supportvereinbarung angeben.

Inhalt

16 Modellierungsknoten – Eigenschaften 201

Kapitel 1

Informationen zu IBM SPSS Modeler

IBM® SPSS® Modeler ist ein Set von Data Mining-Tools, mit dem Sie auf der Grundlage Ihres Geschäftswissens schnell und einfach Vorhersagemodelle erstellen und zur Erleichterung der Entscheidungsfindung in die Betriebsabläufe einbinden können. SPSS Modeler, das auf der Grundlage des den Industrienormen entsprechenden Modells CRISP-DM entwickelt wurde, unterstützt den gesamten Data Mining-Prozess, von den Daten bis hin zu besseren Geschäftsergebnissen.

SPSS Modeler bietet eine Vielzahl von Modellbildungsmethoden, die aus dem maschinellen Lernen, der künstlichen Intelligenz und der Statistik stammen. Mit den in der Modellierungspalette verfügbaren Methoden können Sie aus Ihren Daten neue Informationen ableiten und Vorhersagemodelle erstellen. Jede Methode besitzt ihre Stärken und eignet sich besonders für bestimmte Problemtypen.

SPSS Modeler kann als Standalone-Produkt oder als Client in Verbindung mit SPSS Modeler Server erworben werden. Außerdem ist eine Reihe von Zusatzoptionen verfügbar, die in den folgenden Abschnitten kurz dargelegt werden. Weitere Informationen finden Sie unter *http://www.ibm.com/software/analytics/spss/products/modeler/*.

IBM SPSS Modeler-Produkte

Zur IBM® SPSS® Modeler-Produktfamilie und der zugehörigen Software gehören folgende Elemente.

- IBM SPSS Modeler
- IBM SPSS Modeler Server
- IBM SPSS Modeler Administration Console
- IBM SPSS Modeler Batch
- IBM SPSS Modeler Solution Publisher
- IBM SPSS Modeler Server-Adapter für IBM SPSS Collaboration and Deployment Services

IBM SPSS Modeler

SPSS Modeler ist eine funktionell in sich abgeschlossene Produktversion, die Sie auf Ihrem PC installieren und ausführen können. Sie können SPSS Modeler im lokalen Modus als Standalone-Produkt oder im verteilten Modus zusammen mit IBM® SPSS® Modeler Server verwenden, um bei Daten-Sets die Leistung zu verbessern.

Mit SPSS Modeler können Sie schnell und intuitiv genaue Vorhersagemodelle erstellen, und das ohne Programmierung. Mithilfe der speziellen visuellen Benutzeroberfläche können Sie ganz einfach den Data Mining-Prozess visualisieren. Mit der Unterstützung der in das Produkt

eingebetteten erweiterten Analyseprozesse können Sie zuvor verborgene Muster und Trends in Ihren Daten aufdecken. Sie können Ergebnisse modellieren und Einblick in die Faktoren gewinnen, die Einfluss auf diese Ergebnisse haben, wodurch Sie in die Lage versetzt werden, Geschäftschancen zu nutzen und Risiken abzuschwächen.

SPSS Modeler ist in zwei Editionen erhältlich: SPSS Modeler Professional und SPSS Modeler Premium. Für weitere Informationen siehe Thema IBM SPSS Modeler-Editionen in *IBM SPSS Modeler 15 Benutzerhandbuch*.

IBM SPSS Modeler Server

SPSS Modeler verwendet eine Client/Server-Architektur zur Verteilung von Anforderungen für ressourcenintensive Vorgänge an leistungsstarke Serversoftware, wodurch bei größeren Daten-Sets eine schnellere Leistung erzielt werden kann.

SPSS Modeler Server ist ein separat lizenziertes Produkt, das durchgehend im verteilten Analysemodus auf einem Server-Host in Verbindung mit einer oder mehreren IBM® SPSS® Modeler-Installationen ausgeführt wird. Auf diese Weise bietet SPSS Modeler Server eine herausragende Leistung bei großen Daten-Sets, da speicherintensive Vorgänge auf dem Server ausgeführt werden können, ohne Daten auf den Client-Computer herunterladen zu müssen. IBM® SPSS® Modeler Server bietet außerdem Unterstützung für SQL-Optimierung sowie Möglichkeiten zur Modellierung innerhalb der Datenbank, was weitere Vorteile hinsichtlich Leistung und Automatisierung mit sich bringt.

IBM SPSS Modeler Administration Console

Die Modeler Administration Console ist eine grafische Anwendung zur Verwaltung einer Vielzahl der SPSS Modeler Server-Konfigurationsoptionen, die auch mithilfe einer Optionsdatei konfiguriert werden können. Die Anwendung bietet eine Konsolen-Benutzeroberfläche zur Überwachung und Konfiguration der SPSS Modeler Server-Installationen und steht aktuellen SPSS Modeler Server-Kunden kostenlos zur Verfügung. Die Anwendung kann nur unter Windows installiert werden. Der von ihr verwaltete Server kann jedoch auf einer beliebigen unterstützten Plattform installiert sein.

IBM SPSS Modeler Batch

Data Mining ist zwar für gewöhnlich ein interaktiver Vorgang, es ist jedoch auch möglich, SPSS Modeler über eine Befehlszeile auszuführen, ohne dass die grafische Benutzeroberfläche verwendet werden muss. Beispielsweise kann es sinnvoll sein, langwierige oder repetitive Aufgaben ohne Eingreifen des Benutzers durchzuführen. SPSS Modeler Batch ist eine spezielle Version des Produkts, die die vollständigen Analysefunktionen von SPSS Modeler ohne Zugriff auf die reguläre Benutzeroberfläche bietet. Zur Verwendung von SPSS Modeler Batch ist eine SPSS Modeler Server-Lizenz erforderlich.

IBM SPSS Modeler Solution Publisher

SPSS Modeler Solution Publisher ist ein Tool, mit dem Sie eine gepackte Version eines SPSS Modeler-Streams erstellen können, der durch eine externe Runtime-Engine ausgeführt oder in eine externe Anwendung eingebettet werden kann. Auf diese Weise können Sie vollständige SPSS Modeler-Streams für die Verwendung in Umgebungen veröffentlichen und bereitstellen, in denen SPSS Modeler nicht installiert ist. SPSS Modeler Solution Publisher wird als Teil des IBM SPSS Collaboration and Deployment Services - Scoring-Diensts verteilt, für den eine separate Lizenz erforderlich ist. Mit dieser Lizenz erhalten Sie SPSS Modeler Solution Publisher Runtime, womit Sie die veröffentlichen Streams ausführen können.

IBM SPSS Modeler Server-Adapter für IBM SPSS Collaboration and Deployment Services

Es ist eine Reihe von Adaptern für IBM® SPSS® Collaboration and Deployment Services verfügbar, mit denen SPSS Modeler und SPSS Modeler Server mit einem IBM SPSS Collaboration and Deployment Services-Repository interagieren können. Auf diese Weise kann ein im Repository bereitgestellter SPSS Modeler-Stream von mehreren Benutzern gemeinsam verwendet werden. Auch der Zugriff über die Thin-Client-Anwendung IBM SPSS Modeler Advantage ist möglich. Sie installieren den Adapter auf dem System, das als Host für das Repository fungiert.

IBM SPSS Modeler-Editionen

SPSS Modeler ist in den folgenden Editionen erhältlich.

SPSS Modeler Professional

SPSS Modeler Professional bietet sämtliche Tools, die Sie für die Arbeit mit den meisten Typen von strukturierten Daten benötigen, beispielsweise in CRM-Systemen erfasste Verhaltensweisen und Interaktionen, demografische Daten, Kaufverhalten und Umsatzdaten.

SPSS Modeler Premium

SPSS Modeler Premium ist ein separat lizenziertes Produkt, das SPSS Modeler Professional für die Arbeit mit spezialisierten Daten erweitert, wie beispielsweise den Daten, die für Entitätsanalysen oder soziale Netzwerke verwendet werden, sowie für die Arbeit mit unstrukturierten Textdaten. SPSS Modeler Premium umfasst die folgenden Komponenten.

IBM® SPSS® Modeler Entity Analytics fügt eine völlig neue Dimension zu den IBM® SPSS® Modeler-Vorhersageanalysen hinzu. Während bei Vorhersageanalysen versucht wird, zukünftiges Verhalten aus früheren Daten vorherzusagen, liegt der Schwerpunkt bei der Entitätsanalyse auf der Verbesserung von Kohärenz und Konsistenz der aktuellen Daten, indem Identitätskonflikte innerhalb der Datensätze selbst aufgelöst werden. Bei der Identität kann es sich um die Identität einer Person, einer Organisation, eines Objekts oder einer anderen Entität handelt, bei der Unklarheiten bestehen könnten. Die Identitätsauflösung kann in einer Reihe von Bereichen

entscheidend sein, darunter Customer Relationship Management, Betrugserkennung, Bekämpfung der Geldwäsche sowie nationale und internationale Sicherheit.

IBM SPSS Modeler Social Network Analysis transformiert Informationen zu Beziehungen in Felder, die das Sozialverhalten von Einzelpersonen und Gruppen charakterisieren. Durch die Verwendung von Daten, die die Beziehungen beschreiben, die sozialen Netzwerken zugrunde liegen, ermittelt IBM® SPSS® Modeler Social Network Analysis Führer in sozialen Netzwerken, die das Verhalten anderer Personen im Netzwerk beeinflussen. Außerdem können Sie feststellen, welche Personen am meisten durch andere Teilnehmer im Netzwerk beeinflusst werden. Durch die Kombination dieser Ergebnisse mit anderen Maßzahlen können Sie aussagekräftige Profile für Einzelpersonen, die Sie als Grundlage für Ihre Vorhersagemodelle verwenden können. Modelle, die diese sozialen Informationen berücksichtigen, sind leistungsstärker als Modelle, die dies nicht tun.

Text Analytics for IBM® SPSS® Modeler verwendet hoch entwickelte linguistische Technologien und die Verarbeitung natürlicher Sprache (Natural Language Processing, NLP), um eine schnelle Verarbeitung einer großen Vielfalt an unstrukturierten Textdaten zu ermöglichen, um die Schlüsselkonzepte zu extrahieren und zu ordnen und um diese Konzepte in Kategorien zusammenzufassen. Extrahierte Konzepte und Kategorien können mit bestehenden strukturierten Daten, beispielsweise demografischen Informationen, kombiniert und mithilfe der vollständigen Suite der Data-Mining-Tools von SPSS Modeler auf die Modellierung angewendet werden, um bessere und fokussiertere Entscheidungen zu ermöglichen.

IBM SPSS Modeler-Dokumentation

Dokumentation im Online-Hilfe-Format finden Sie im Hilfe-Menü von SPSS Modeler. Dazu gehören die Dokumentation für SPSS Modeler, SPSS Modeler Server und SPSS Modeler Solution Publisher sowie das Anwendungshandbuch und weiteres Material zur Unterstützung.

Die vollständige Dokumentation für die einzelnen Produkte (einschließlich Installationsanweisungen) steht im PDF-Format im Ordner *\Documentation* auf der jeweiligen Produkt-DVD zur Verfügung. Installationsdokumente können auch aus dem Internet unter *http://www-01.ibm.com/support/docview.wss?uid=swg27023172* heruntergeladen werden:

Dokumentation in beiden Formaten steht auch im SPSS Modeler Information Center unter *http://publib.boulder.ibm.com/infocenter/spssmodl/v15r0m0/* zur Verfügung.

SPSS Modeler Professional-Dokumentation

Die SPSS Modeler Professional-Dokumentationssuite (ohne Installationsanweisungen) umfasst folgende Dokumente:

- **IBM SPSS Modeler-Benutzerhandbuch.** Allgemeine Einführung in die Verwendung von SPSS Modeler, in der u. a. die Erstellung von Daten-Streams, der Umgang mit fehlenden Werten, die Erstellung von CLEM-Ausdrücken, die Arbeit mit Projekten und Berichten sowie das Packen von Streams für das Deployment in IBM SPSS Collaboration and Deployment Services, Predictive Applications (Prognoseanwendungen) oder IBM SPSS Modeler Advantage beschrieben werden.

- **Quellen-, Prozess- und Ausgabeknoten in IBM SPSS Modeler.** Beschreibung aller Knoten, die zum Lesen, zum Verarbeiten und zur Ausgabe von Daten in verschiedenen Formaten verwendet werden. Im Grunde sind sie alle Knoten, mit Ausnahme der Modellierungsknoten.
- **IBM SPSS Modeler Modellierungsknoten.** Beschreibungen sämtlicher für die Erstellung von Data Mining-Modellen verwendeter Knoten. IBM® SPSS® Modeler bietet eine Vielzahl von Modellbildungsmethoden, die aus dem maschinellen Lernen, der künstlichen Intelligenz und der Statistik stammen. Für weitere Informationen siehe Thema Überblick über Modellierungsknoten in Kapitel 3 in *IBM SPSS Modeler 15 Modellierungsknoten*.
- **IBM SPSS Modeler-Algorithmushandbuch.** Beschreibung der mathematischen Grundlagen der in SPSS Modeler verwendeten Modellierungsmethoden. Dieses Handbuch steht nur im PDF-Format zur Verfügung.
- **IBM SPSS Modeler-Anwendungshandbuch.** Die Beispiele in diesem Handbuch bieten eine kurze, gezielte Einführung in bestimmte Modellierungsmethoden und -verfahren. Eine Online-Version dieses Handbuchs kann auch über das Hilfe-Menü aufgerufen werden. Für weitere Informationen siehe Thema Anwendungsbeispiele in *IBM SPSS Modeler 15 Benutzerhandbuch*.
- **Skripterstellung und Automatisierung in IBM SPSS Modeler.** Informationen zur Automatisierung des Systems über Skripterstellung, einschließlich der Eigenschaften, die zur Bearbeitung von Knoten und Streams verwendet werden können.
- **IBM SPSS Modeler Deployment-Handbuch.** Informationen zum Ausführen von SPSS Modeler-Streams und -Szenarien als Schritte bei der Verarbeitung von Jobs im IBM® SPSS® Collaboration and Deployment Services Deployment Manager.
- **IBM SPSS Modeler CLEF-Entwicklerhandbuch.** CLEF bietet die Möglichkeit, Drittanbieterprogramme, wie Datenverarbeitungsroutinen oder Modellierungsalgorithmen, als Knoten in SPSS Modeler zu integrieren.
- **In-Database Mining-Handbuch für IBM SPSS Modeler.** Informationen darüber, wie Sie Ihre Datenbank dazu einsetzen, die Leistung zu verbessern, und wie Sie die Palette der Analysefunktionen über Drittanbieteralgorithmen erweitern.
- **IBM SPSS Modeler Server-Verwaltungs- und -Leistungshandbuch.** Informationen zur Konfiguration und Verwaltung von IBM® SPSS® Modeler Server.
- **IBM SPSS Modeler Administration Console – Benutzerhandbuch.** Informationen zur Installation und Nutzung der Konsolen-Benutzeroberfläche zur Überwachung und Konfiguration von SPSS Modeler Server. Die Konsole ist als Plugin für die Deployment Manager-Anwendung implementiert.
- **IBM SPSS Modeler Solution Publisher-Handbuch.** SPSS Modeler Solution Publisher ist eine Zusatzkomponente, mit der Unternehmen Streams zur Verwendung außerhalb der SPSS Modeler-Standardumgebung veröffentlichen können.
- **IBM SPSS Modeler-Handbuch zu CRISP-DM.** Schritt-für-Schritt-Anleitung für das Data Mining mit SPSS Modeler unter Verwendung der CRISP-DM-Methode.
- **IBM SPSS Modeler Batch-Benutzerhandbuch.** Vollständiges Handbuch für die Verwendung von IBM SPSS Modeler im Batch-Modus, einschließlich Details zur Ausführung des Batch-Modus und zu Befehlszeilenargumenten. Dieses Handbuch steht nur im PDF-Format zur Verfügung.

SPSS Modeler Premium-Dokumentation

Die SPSS Modeler Premium-Dokumentationssuite (ohne Installationsanweisungen) umfasst folgende Dokumente:

- **IBM SPSS Modeler Entity Analytics – Benutzerhandbuch.** Information zur Verwendung von Entitätsanalysen mit SPSS Modeler, unter Behandlung von Repository-Installation und -Konfiguration, Entity Analytics-Knoten und Verwaltungsaufgaben.
- **IBM SPSS Modeler Social Network Analysis – Benutzerhandbuch.** Ein Handbuch zur Durchführung sozialer Netzwerkanalyse mit SPSS Modeler, einschließlich Gruppenanalyse und Diffusionsanalyse.
- **Text Analytics for SPSS Modeler – Benutzerhandbuch.** Informationen zur Verwendung von Textanalysen mit SPSS Modeler, unter Behandlung der Text Mining-Knoten, der interaktiven Workbench sowie von Vorlagen und anderen Ressourcen.
- **Text Analytics for IBM SPSS Modeler Administration Console – Benutzerhandbuch.** Informationen zur Installation und Nutzung der Konsolen-Benutzeroberfläche zur Überwachung und Konfiguration von IBM® SPSS® Modeler Server für die Verwendung mit Text Analytics for SPSS Modeler. Die Konsole ist als Plugin für die Deployment Manager-Anwendung implementiert.

Anwendungsbeispiele

Mit den Data-Mining-Tools in SPSS Modeler kann eine große Bandbreite an geschäfts- und unternehmensbezogenen Problemen gelöst werden; die Anwendungsbeispiele dagegen bieten jeweils eine kurze, gezielte Einführung in spezielle Modellierungsmethoden und -verfahren. Die hier verwendeten Daten-Sets sind viel kleiner als die riesigen Datenbestände, die von einigen Data-Mining-Experten verwaltet werden müssen, die zugrunde liegenden Konzepte und Methoden sollten sich jedoch auch auf reale Anwendungen übertragen lassen.

Sie können auf die Beispiele zugreifen, indem Sie im Menü “Hilfe” in SPSS Modeler auf die Option Anwendungsbeispiele klicken. Die Datendateien und Beispiel-Streams wurden im Ordner *Demos*, einem Unterordner des Produktinstallationsverzeichnisses, installiert. Für weitere Informationen siehe Thema Ordner “Demos” in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Beispiele für die Datenbank-Modellierung. Die Beispiele finden Sie im *IBM SPSS Modeler In-Database Mining-Handbuch*.

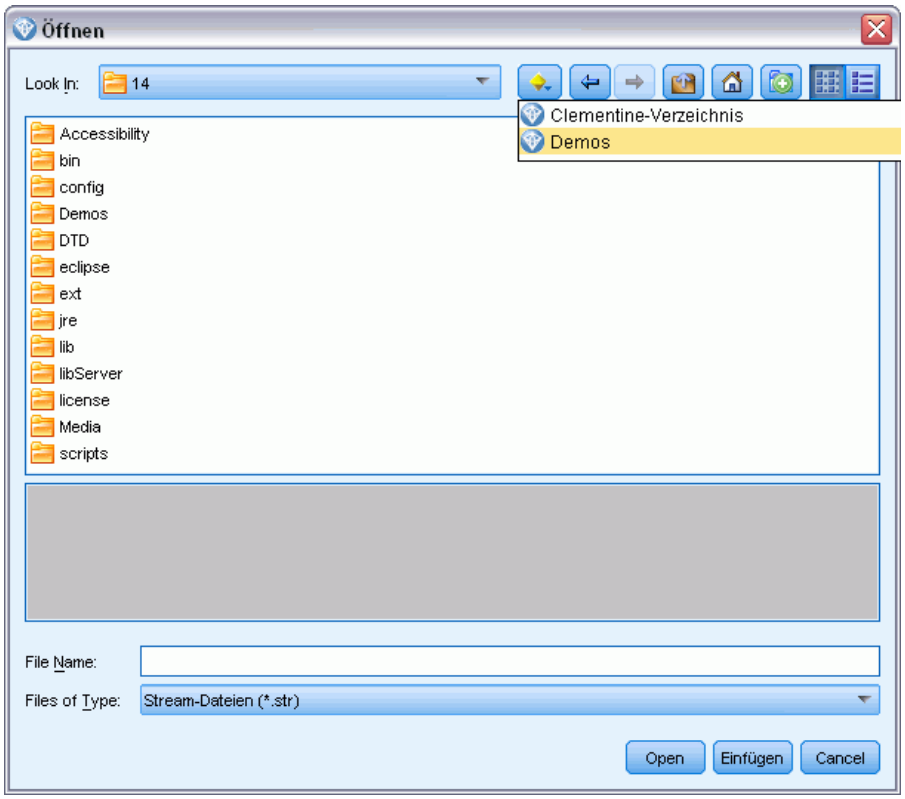
Skriptbeispiele. Die Beispiele finden Sie im *IBM SPSS Modeler Handbuch für die Skripterstellung und Automatisierung*.

Ordner “Demos”

Die in den Anwendungsbeispielen verwendeten Datendateien und Beispiel-Streams wurden im Ordner *Demos*, einem Unterordner des Produktinstallationsverzeichnisses, installiert. Auf diesen Ordner können Sie auch über die Programmgruppe IBM SPSS Modeler 15 im Windows-Startmenü

oder durch Klicken auf *Demos* in der Liste der zuletzt angezeigten Verzeichnisse im Dialogfeld "Datei öffnen" zugreifen.

Abbildung 1-1
Auswahl des Ordners "Demos" in der Liste der zuletzt angezeigten Verzeichnisse



Teil I:
Skripts und die Skriptsprache

Kapitel
2

Skripterstellung – Überblick

Die Skripterstellung in IBM® SPSS® Modeler ist ein leistungsstarkes Tool, mit dem Prozesse in der Benutzeroberfläche automatisiert werden. Skripts 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.

Skripts können zu folgenden Zwecken verwendet werden:

- Eine bestimmte Reihenfolge für die Knotenausführung in einem Stream erzwingen.
- Die Eigenschaften von Knoten festlegen und Ableitungen durchführen, indem Sie eine Untergruppe von CLEM (Control Language for Expression Manipulation) verwenden.
- 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 Skripts auf der Stream-Ebene, Standalone-Skripts und Skripts innerhalb von Superknoten auf der SPSS Modeler-Benutzeroberfläche. Weitere Informationen zu Skriptsprache, Syntax und Befehlen finden Sie in den nachfolgenden Kapiteln.

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

Skripttypen

IBM® SPSS® Modeler verwendet drei Skripttypen:

- **Stream-Skripts** werden als Stream-Eigenschaft gespeichert und daher zusammen mit einem bestimmten Stream gespeichert und geladen. Beispielsweise können Sie ein Stream-Skript schreiben, das das Trainieren und Anwenden eines Modell-Nuggets automatisiert. Außerdem können Sie angeben, dass bei jeder Ausführung eines bestimmten Streams statt des Inhalts des Stream-Zeichenbereichs das Skript ausgeführt werden soll.

- **Standalone-Skripts** sind mit keinem bestimmten Stream verknüpft und werden in externen Textdateien gespeichert. Mit einem Standalone-Skript können beispielsweise mehrere Streams gemeinsam bearbeitet werden.
- **Superknoten-Skripts** werden als Stream-Eigenschaft von Superknoten gespeichert. Superknoten-Skripts stehen nur in End-Superknoten zur Verfügung. Mit Superknoten-Skripts kann die Ausführungssequenz der Superknoten-Inhalte 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-Skript definieren.

Stream-Skripts

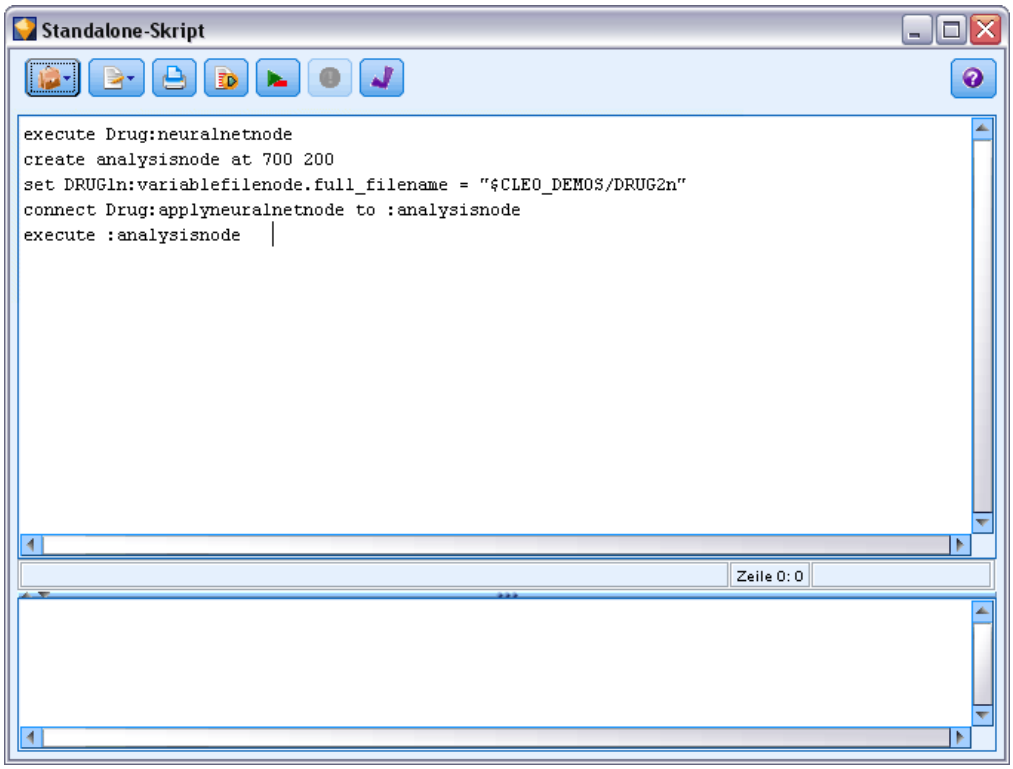
Mit Skripts können in einem bestimmten Stream enthaltene Operationen angepasst und zusammen mit dem Stream gespeichert werden. Stream-Skripts können verwendet werden, um eine bestimmte Ausführungsreihenfolge der in einem Stream enthaltenen Endknoten vorzugeben. Die Bearbeitung des mit dem aktuellen Stream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

So können Sie im Dialogfeld “Stream-Eigenschaften” auf die Registerkarte für das Stream-Skript zugreifen:

▶ Wählen Sie im Menü “Extras” folgende Optionsfolge aus:
Stream-Eigenschaften > Skript...

▶ Klicken Sie auf die Registerkarte Skript, um mit den Skripts des aktuellen Streams zu arbeiten.

Abbildung 2-1
Dialogfeld "Skript" des Streams



Mit den Symbolleistensymbolen oben in diesem Dialogfeld können Sie folgende Operationen durchführen.

- Inhalte eines bereits vorhandenen Standalone-Skripts in das Fenster importieren.
- Skript als Textdatei speichern.
- Skript drucken.
- Standardskript anhängen.
- Gesamtes aktuelles Skript ausführen.
- Ausgewählte Zeilen eines Skripts ausführen.
- Syntax des Skripts überprüfen und etwaige Fehler zur Untersuchung im unteren Fensterbereich des Dialogfelds anzeigen.

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

Beispiel für Stream-Skript: Trainieren eines neuronalen Netzes

Ein Stream kann dazu genutzt werden, um bei der Ausführung ein neuronales Netzwerkmodell zu trainieren. Um das Modell zu testen, müssen Sie den Modellknoten ausführen, um das Modell in den Stream einzufügen, die entsprechenden Verbindungen herzustellen und einen Analyseknoten auszuführen.

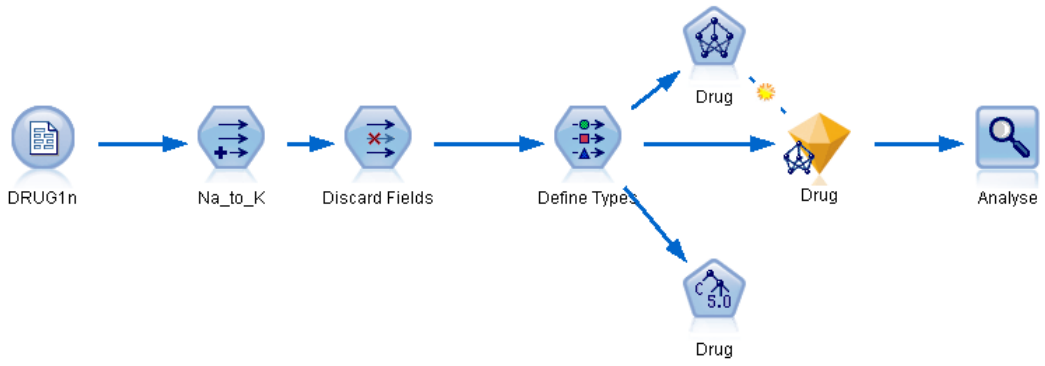
Mit einem IBM® SPSS® Modeler-Skript können Sie das Testen des Modell-Nuggets nach seiner Erstellung automatisieren. Beispielsweise kann folgendes Skript für den Demo-Stream *druglearn.str* (verfügbar im Ordner */Demos/streams/* unter Ihrer SPSS Modeler-Installation) aus dem Dialogfeld "Stream-Eigenschaften" (Extras > Stream-Eigenschaften > Skript) ausgeführt werden:

```
execute Drug:neuralnetworknode
create analysisnode at 700 200
set DRUG1n:variablefilenode.full_filename = "$CLEO_DEMOS/DRUG2n"
connect :applyneuralnetworknode to :analysisnode
execute :analysisnode
```

In der folgenden Auflistung werden die einzelnen Zeilen dieses Skriptbeispiels beschrieben.

- In der ersten Zeile wird der Netzwerkknoten Drug ausgeführt, der bereits im Demo-Stream enthalten ist, um ein Modell-Nugget zu generieren und es in den Stream-Zeichenbereich zu platzieren, der mit dem bereits im Stream vorhandenen Typknoten verbunden ist.
- In Zeile 2 erstellt das Skript einen Analyseknoten und fügt ihn an der Position 700 x 200 im Zeichenbereich ein.
- In Zeile 3 wird von der ursprünglich im Stream verwendeten Datenquelle zu einem Test-Daten-Set mit der Bezeichnung DRUG2n gewechselt.
- In Zeile 4 wird das Netzwerk-Modell-Nugget mit dem Analyseknoten verbunden. Beachten Sie, dass zur Angabe des Netzwerk-Modell-Nuggets und des Analyseknotens keine Namen verwendet werden, da keine ähnlichen Knoten im Stream vorhanden sind.
- Schließlich wird der Analyseknoten ausgeführt, um den Analysebericht zu erstellen.

Abbildung 2-2
Ergebnis-Stream



Dieser Stream ist für die Zusammenarbeit mit einem bestehenden Stream gedacht, da er voraussetzt, dass bereits ein Netzwerkknoten mit dem Namen *Drug* (Medikament) vorhanden ist. Außerdem können Sie mithilfe eines Skripts einen völlig neuen Stream, ausgehend von einem leeren Zeichenbereich, erstellen und ausführen. Weitere Informationen zur Skriptsprache im Allgemeinen finden Sie unter Skriptsprache – Überblick auf S. 21.Weitere Informationen zu Skriptbefehlen im Besonderen finden Sie unter Skriptbefehle auf S. 33.

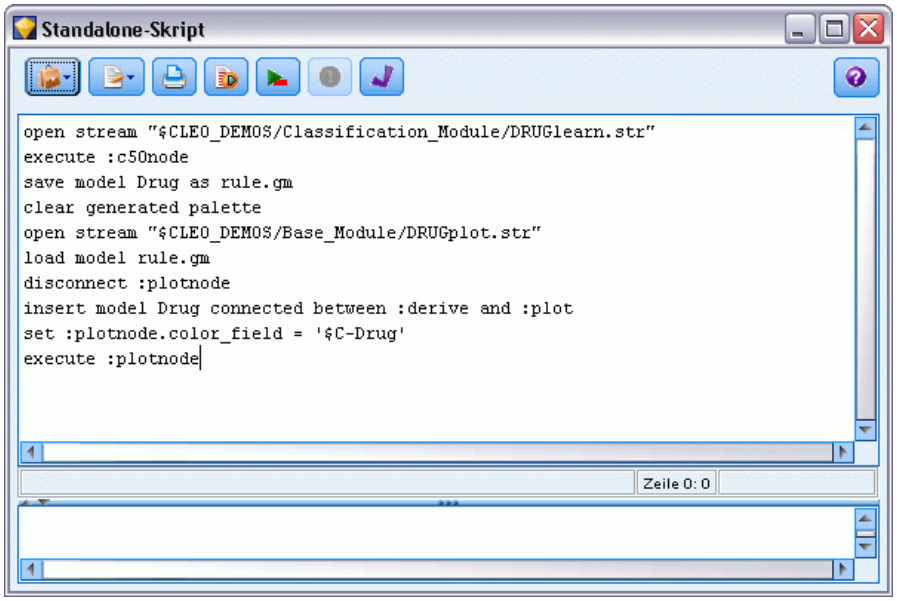
Standalone-Skripts

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

So öffnen Sie das Dialogfeld für Standalone-Skripts:

▶ Wählen Sie im Hauptmenü Folgendes:
Werkzeuge > Standalone-Skript

Abbildung 2-3
Dialogfeld “Standalone-Skript”



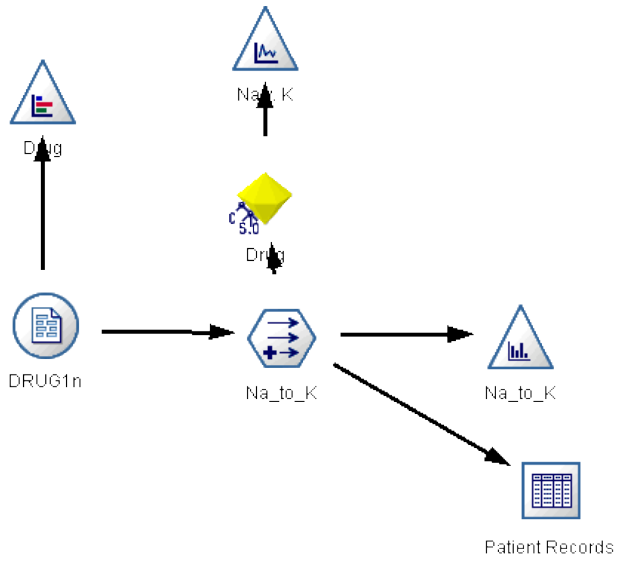
Für Standalone-Skripts stehen dieselbe Symbolleiste und dieselben Optionen zur Überprüfung der Skript-Syntax zur Verfügung wie für Stream-Skripts. Für weitere Informationen siehe Thema Stream-Skripts auf S. 10.

Beispiel für Standalone-Skript: Speichern und Laden von Modellen

Standalone-Skripts sind bei der Stream-Bearbeitung hilfreich. Nehmen wir an, Sie besitzen zwei Streams – einen, der ein Modell erstellt, und einen anderen, der die aus dem ersten Stream mit vorhandenen Datenfeldern generierte Regelmenge anhand von Diagrammen untersucht. Ein Standalone-Skript für dieses Szenario könnte wie folgt aussehen:

```
open stream "$CLEO_DEMOS/streams/druglearn.str"
execute :c50node
save model Drug as rule.gm
clear generated palette
open stream "$CLEO_DEMOS/streams/drugplot.str"
load model rule.gm
disconnect :plotnode
insert model Drug connected between :derive and :plot
set :plotnode.color_field = '$C-Drug'
execute :plotnode
```

Abbildung 2-4
Ergebnis-Stream



Hinweis: Weitere Informationen zur Skriptsprache im Allgemeinen finden Sie unter Skriptsprache – Überblick auf S. 21.Weitere Informationen zu Skriptbefehlen im Besonderen finden Sie unter Skriptbefehle auf S. 33.

Beispiel für Standalone-Skript: Generieren eines Merkmalsauswahlmodells

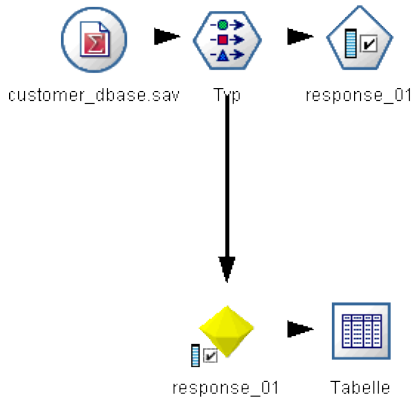
In diesem Beispiel wird ausgehend von einem leeren Zeichenbereich ein Stream erstellt, der ein Merkmalsauswahlmodell generiert, das Modell anwendet und eine Tabelle erstellt, in der die 15 wichtigsten Ziele in Bezug auf das angegebene Ziel aufgeführt werden.

```
create stream 'featureselection'
create statisticsimportnode
```

```
position :statisticsimportnode at 50 50
set :statisticsimportnode.full_filename = "$CLEO_DEMOS/customer_dbase.sav"

create typenode
position :typenode at 150 50
set :typenode.direction.'response_01' = Target
connect :statisticsimportnode to :typenode

create featureselectionnode
position :featureselectionnode at 250 50
set :featureselectionnode.screen_missing_values=true
set :featureselectionnode.max_missing_values=80
set :featureselectionnode.criteria = Likelihood
set :featureselectionnode.important_label = "Check Me Out!"
set :featureselectionnode.selection_mode = TopN
set :featureselectionnode.top_n = 15
connect :typenode to :featureselectionnode
execute :featureselectionnode

create tablenode
position :tablenode at 250 250
connect response_01:applyfeatureselectionnode to :tablenode
execute :tablenode
```

Das Skript erstellt einen Quellenknoten zum Einlesen der Daten, verwendet einen Typknoten, um die Rolle (Verwendung) des Felds *response_01* auf Target zu setzen und erstellt anschließend einen Merkmalsauswahlknoten und führt diesen aus. Außerdem verbindet das Skript die Knoten und positioniert sie im Stream-Zeichenbereich, um ein lesbares Layout zu erstellen. Anschließend wird das resultierende Modell-Nugget mit einem Tabellenknoten verbunden, in dem die 15 wichtigsten Felder aufgeführt sind, die durch die Eigenschaften selection_mode und top_n bestimmt wurden. Für weitere Informationen siehe Thema Eigenschaften von “featureselectionnode” in Kapitel 16 auf S. 226.

Abbildung 2-5
Ergebnis-Stream



Superknoten-Skripts

Mithilfe der Skriptsprache von IBM® SPSS® Modeler können Sie Skripts innerhalb jedes beliebigen End-Superknotens erstellen und speichern. Diese Skripts stehen ausschließlich für End-Superknoten zur Verfügung und werden häufig beim Erstellen von Vorlagen-Streams oder zum Erzwingen einer bestimmten Ausführungsreihenfolge für die Superknoten-Inhalte verwendet. Mit Superknoten-Skripts können Sie außerdem mehrere Skripts 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 Superknoten-Skript 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 Superknoten-Skripts können Sie Knoteneigenschaften auf dieselbe Weise angeben wie bei anderen Skripts. Alternativ können Sie die Eigenschaften für einen beliebigen Superknoten oder den darin verkapselten Knoten direkt über ein Stream-Skript ändern und definieren. Für weitere Informationen siehe Thema Superknoten-Eigenschaften in Kapitel 22 auf S. 326. Diese Methode funktioniert für Quellen- und Prozess-Superknoten ebenso wie für End-Superknoten.

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

So öffnen Sie das Dialogfeld "Superknoten-Skript" im Hauptzeichenbereich:

- Wählen Sie einen End-Superknoten im Stream-Zeichenbereich aus und wählen Sie folgende Option im Menü "Superknoten":

  Superknoten-Skript...

So öffnen Sie das Dialogfeld "Superknoten-Skript" im vergrößerten Superknoten-Zeichenbereich:

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

  Superknoten-Skript...

Für weitere Informationen siehe Thema Superknoten und Skripts in Kapitel 9 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel für Superknoten-Skript

Das folgende Superknoten-Skript gibt die Reihenfolge an, in der die Endknoten innerhalb des Superknotens ausgeführt werden sollen. Mit dieser Reihenfolge wird sichergestellt, dass der Globalwerteknoten zuerst ausgeführt wird und somit die von diesem Knoten berechneten Werte bei der Ausführung eines weiteren Knotens verwendet werden können.

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
```

```
execute 'age v. $CC-pep'
execute 'Table'
```

Ausführen und unterbrechen von Skripts

Es gibt mehrere Möglichkeiten zur Ausführung von Skripts. Beispielsweise führt die Schaltfläche “Dieses Skript ausführen” im Dialogfeld für das Stream-Skript oder Standalone-Skript das vollständige Skript aus:

Abbildung 2-6
Schaltfläche “Dieses Skript ausführen”

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

Abbildung 2-7
Schaltfläche “Ausgewählte Zeilen ausführen”

Zum Ausführen von Skripts stehen folgende Methoden zur Auswahl:

- Klicken Sie im Dialogfeld für ein Stream-Skript oder ein Standalone-Skript auf die Schaltfläche “Dieses Skript ausführen” oder “Ausgewählte Zeilen ausführen”.
- Ausführen eines Streams mit Dieses Skript ausführen als Standard-Ausführungsmethode.
- Verwenden Sie das Flag `-execute` beim Start im interaktiven Modus. Für weitere Informationen siehe Thema Verwenden von Befehlszeilenargumenten in Kapitel 7 auf S. 72.

Hinweis: Ein Superknoten-Skript wird bei der Ausführung des Superknotens ausgeführt, sofern Sie Dieses Skript ausführen im Superknoten-Skriptdialogfeld ausgewählt haben.

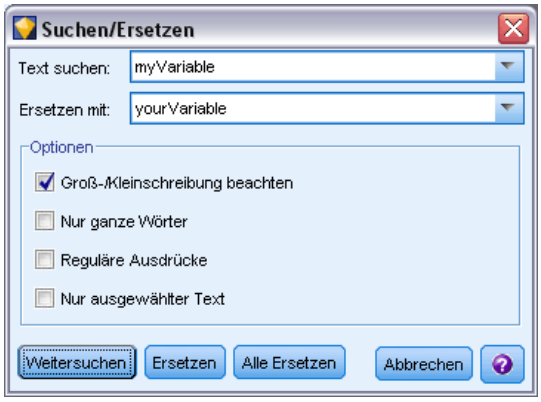
Unterbrechen der Skriptausführung

Während der Skriptausführung ist im Dialogfeld “Stream-Skript” die rote Stoppschaltfläche aktiviert. Mit dieser Schaltfläche können Sie die Ausführung des Skripts und aller aktuellen Streams abbrechen.

Suchen und ersetzen

Das Dialogfeld “Suchen/Ersetzen” ist in Situationen verfügbar, in denen Sie Skript- oder Ausdruckstext bearbeiten, u. a. im Skript-Editor, im CLEM Expression Builder 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 in 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.

Abbildung 2-8
Dialogfeld "Suchen/Ersetzen"



- Achten Sie darauf, dass sich der Cursor in einem Textfeld befindet, und drücken Sie Strg+F, um das Dialogfeld "Suchen/Ersetzen" aufzurufen.
- Geben Sie den Text ein, nach dem Sie suchen möchten, oder treffen Sie eine Auswahl aus der Dropdown-Liste der kürzlich durchsuchten Elemente.
- Geben Sie, falls erforderlich, den Ersatztext ein.
- Klicken Sie auf Weitersuchen, um die Suche zu starten.
- Klicken Sie auf Ersetzen, um die aktuelle Auswahl zu ersetzen, oder Alle ersetzen, um alle bzw. die ausgewählten Instanzen zu ersetzen.
- 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* Ausdrücke 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 deaktiviert 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. Folgende Arten von Ausdrücken werden unterstützt:

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 Zeichen für Zeilenvorschub (‘\u000A’)
\r	Das Zeichen für Wagenrücklauf (‘\u000D’)
\f	Das Zeichen für Seitenvorschub (‘\u000C’)
\a	Das Alarmzeichen (‘\u0007’)
\e	Das Escape-Zeichen (‘\u001B’)
\cx	Das Steuerzeichen, das x entspricht

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]

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]

Vordefinierte Zeichenklassen	Übereinstimmung/Bedeutung
\S	Ein Nicht-Leerzeichen: [^\s]
\w	Buchstabe: [a-zA-Z_0-9]
\W	Nichtbuchstabe: [^\w]

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

Kapitel 3

Skriptsprache

Skriptsprache – Überblick

Die IBM® SPSS® Modeler-Skriptsprache besteht aus folgenden Elementen:

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

In diesem Abschnitt wird die Grundsyntax für die Verwendung der Skriptsprache beschrieben. Informationen zu speziellen Eigenschaften und Befehlen finden Sie in den nachfolgenden Abschnitten.

Skriptsyntax

Um eine größere Klarheit bei der Analyse zu erreichen, sollten bei der Arbeit mit Skripts in IBM® SPSS® Modeler folgende Regeln eingehalten werden:

- Variablennamen, wie income oder referrerID, dürfen nicht in Anführungszeichen stehen.
- Variablennamen, wie ^mystream, wird ein Winkelzeichen (^) vorangestellt, wenn eine bestehende Variable referenziert wird, deren Wert bereits festgelegt ist. Das Winkelzeichen wird jedoch nicht beim Deklarieren oder Festlegen des Werts der Variablen verwendet. Für weitere Informationen siehe Thema Referenzieren von Knoten auf S. 22.
- Verweise auf Sitzungs-, Stream- und Superknoten-Parameter, wie beispielsweise '$P-Maxvalue', sollten in einzelnen Anführungszeichen stehen.
- Wenn doppelte Anführungszeichen verwendet werden, wird ein Ausdruck als Literalzeichenkette behandelt – z. B. "Web graph of BP and Drug". Dies kann zu unerwarteten Ergebnissen führen, wenn einzelne und doppelte Anführungszeichen nicht sorgfältig unterschieden werden. "$P-Maxvalue" beispielsweise ist eine Zeichenkette und nicht eine Referenz auf den in einem Parameter gespeicherten Wert.
- Dateinamen, wie beispielsweise "druglearn.str", sollten in doppelten Anführungszeichen stehen.
- Knotennamen, wie databasenode oder Na_to_K können ohne Anführungszeichen oder in einfachen Anführungszeichen stehen. *Hinweis*: Namen müssen in Anführungszeichen stehen, wenn sie Leerzeichen oder Sonderzeichen enthalten. Sie dürfen jedoch keinen Knotennamen in einem Skript verwenden, wenn der Name mit einer Zahl beginnt, wie beispielsweise '2a_referrerID'.

- Flag-Eigenschaften sollten anhand der Werte true und false (in Kleinbuchstaben) gelesen bzw. festgelegt werden. (Variationen mit Off, OFF, off, No, NO, no, n, N, f, F, False, FALSE oder 0 werden beim Festlegen von Werten ebenfalls erkannt, können jedoch beim Lesen von Eigenschaftswerten in manchen Fällen zu Fehlern führen. Alle anderen Werte werden als true betrachtet. Durch die durchgängige Verwendung von true und false können Verwechslungen vermieden werden.)
- Literalzeichenketten oder Blöcke, die Zeilenumbrüche, Leerzeichen oder einzelne oder doppelte Anführungszeichen innerhalb des Blocks enthalten, können in dreifache Anführungszeichen eingeschlossen werden. Für weitere Informationen siehe Thema Blöcke mit Literaltext auf S. 31.
- CLEM-Ausdrücke, wie beispielsweise "Age >= 55", sollten in doppelten Anführungszeichen stehen. Beispiel:

  ```
  set :derivenode.flag_expr = "Age >= 55"
  ```

- Bei Verwendung von Anführungszeichen innerhalb von CLEM-Ausdrücken muss jedem Anführungszeichen ein umgekehrter Schrägstrich (\) vorangestellt werden. Beispiel:

  ```
  set :node.parameter = "BP = \"HIGH\""
  ```

Diese Richtlinien müssen zwar nicht in allen Fällen unbedingt eingehalten werden, sind jedoch zur Verbesserung der Deutlichkeit und Klarheit empfohlen. Die in allen Skriptdialogfeldern zur Verfügung stehende Skriptprüfung kennzeichnet mehrdeutige Syntax.

Referenzieren von Knoten

In Skripts gibt es mehrere Möglichkeiten, eine Referenz zu Knoten herzustellen:

- Sie können die Knoten nach Namen angeben, beispielsweise DRUG1n. Sie können den Namen durch Typ genauer angeben. So bezieht sich Drug:neuralnetworknode auf einen Netzwerkknoten (neuronales Netzwerk) namens Drug und auf keine andere Art von Knoten.
- Sie können Knoten auch nur nach Typ angeben. So bezieht sich :neuralnetworknode beispielsweise auf alle Netzwerkknoten (neuronales Netzwerk). Jeder gültige Knotentyp kann verwendet werden, z. B. samplenode, neuralnetworknode und kmeansnode. Das Suffix node ist optional und kann weggelassen werden. Es wird jedoch empfohlen, es zu verwenden, da dies das Auffinden von Fehlern in Skripts erleichtert.
- Sie können auf die einzelnen Knoten anhand ihrer eindeutigen ID Bezug nehmen, die auf der Registerkarte “Anmerkungen” für die einzelnen Knoten angezeigt wird. Verwenden Sie ein “@“-Symbol und danach die ID, z. B. @id5E5GJK23L.custom_name = "My Node". Für weitere Informationen siehe Thema Anmerkungen in Kapitel 5 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Generierte Modelle. Dieselben Regeln gelten für generierte Modellknoten. Sie können den Namen des Knotens verwenden, der er auf der Palette der generierten Modelle im Manager-Fenster angezeigt wird, oder Sie können die generierten Modellknoten nach Typ referenzieren. Beachten Sie: die zur Referenzierung der generierten Modelle im Manager verwendeten Namen unterscheiden sich von den Namen für Modelle, die zum Zwecke des Scorings zu einem Stream hinzugefügt wurden (bei den letzteren wird das Präfix “apply” verwendet). Für weitere Informationen siehe Thema Modell-Nugget-Namen in Kapitel 4 auf S. 45.

Referenzieren von Knoten mithilfe von Variablen

Sie können Knotennamen und Typen als Werte lokaler Skriptvariablen angeben, indem Sie die Winkelzeichensyntax (^) verwenden. Beispiel: Wenn ein Knotenname erforderlich ist, wird durch ^n der Knoten angegeben, dessen Name in der Variablen n gespeichert ist, und durch Drug:^t der Knoten namens Drug, dessen Typ in der Variablen t gespeichert ist.

Knotenreferenzen können in lokalen Skriptvariabeln (mit var-Anweisung erklärt) gespeichert werden, nicht jedoch in Stream-, Sitzungs- oder Superknoten-Parametern. Um eindeutige Referenzen auf Knoten zu gewährleisten, weisen Sie beim Erstellen des Knotens einer Variablen eine eindeutige Knoten-ID zu.

```
var x
set x = create typenode
set ^x.custom_name = "Mein Knoten"
```

- Die Zeile erstellt eine Variable mit dem Namen x.
- Die zweite Zeile erstellt einen neuen Typknoten und speichert eine Referenz auf den Knoten in x. Hinweis: x speichert eine Referenz auf den Knoten selbst, nicht auf den Knotennamen.
- Die dritte Zeile legt den Wert der Eigenschaft custom_name für den Knoten auf "My Node" fest. Das Winkelzeichen gibt an, dass es sich bei x um den Namen einer Variablen und nicht um einen Knoten handelt. (Ohne das Winkelzeichen würde das System nach einem Knoten mit dem Namen x suchen. Das Winkelzeichen ist beim Deklarieren bzw. Festlegen der Variablen nicht erforderlich, da als Objekt eines var-Befehls nur eine Variable infrage kommt. In der dritten Zeile könnte x jedoch auch ein Knotenname anstatt einer Variablen sein. Daher muss das Winkelzeichen verwendet werden, um die beiden Möglichkeiten unterscheiden zu können.)

Ein häufiger Fehler ist, dass versucht wird, eine Referenz auf einen Knoten in einer Variablen zu speichern, ohne diese zuvor zu deklarieren.

```
set x = create typenode
set ^x.custom_name = "Mein Knoten"
```

In diesem Fall versucht der SET-Befehl, x als Stream-, Sitzungs- oder Superknoten-Parameter zu erstellen anstatt als Variable und gibt einen Fehler aus, da eine Referenz auf einen Knoten nicht in einem Parameter gespeichert werden kann.

Referenzieren von Knoten anhand der ID

Sie können auch eine eindeutige Knoten-ID in einer Variablen speichern. Beispiel:

```
var n
set n = "id5E5GJK23L"
set @^n.custom_name = "Mein Knoten"
```

Durchlaufen von Knoten in einem Stream. Mit der Eigenschaft stream.nodes können Sie alle Knoten in einem Stream auflisten und anschließend diese Liste in einer Schleife durchlaufen, um auf die einzelnen Knoten zuzugreifen. Für weitere Informationen siehe Thema Stream-Bericht in Kapitel 6 auf S. 69.

Beispiele

NAME:TYP

NAME ist der Name eines Knotens und TYPE ist sein Typ. Sie müssen mindestens entweder NAME oder TYPE angeben. Sie können auf eines dieser Elemente verzichten, nicht jedoch auf beide. Der folgende Befehl beispielsweise erstellt einen neuen Ableitungsknoten zwischen einem bestehenden Knoten vom Typ "Datei (var.)" mit der Bezeichnung drug1n und einem bestehenden Plotknoten (bei neuen Knoten wird kein Doppelpunkt verwendet):

```
create derivenode connected between drug1n and :plotnode
```

Sie können NAME oder TYPE ein Winkelzeichen (^) voranstellen, um das Vorliegen eines Parameters anzuzeigen. Beispiel:

```
Medikament:^t
```

Diese Referenz bezieht sich auf einen Knoten namens Drug. Dabei ist t ein Parameter, der den Knotentyp angibt. Wenn ^t beispielsweise den Wert c50node aufweist, lässt sich die oben angegebene Referenz wie folgt übersetzen:

```
Medikament:c50node
```

Ebenso kann ein Parameter als Knotenname verwendet werden. Die folgenden Elemente können beispielsweise beide in einem Kontext verwendet werden, in dem ein Knotenname erforderlich ist:

```
^n:derivenode
^n
```

Abrufen von Objekten

Der Befehl get gibt eine Referenz auf einen Stream, einen Knoten oder ein Ausgabeobjekt aus, wodurch es möglich wird, diese Objekte mithilfe von Skripts zu bearbeiten. Beispiel:

```
var mynode
set mynode = get node flag1:derivenode
position ^mynode at 400 400

var mytable = get output :tableoutput
export output ^mytable as c:/mytable.htm format html

set stream = get stream 'Stream1'
set ^stream.execute_method = "Script"
```

Festlegen des aktuellen Objekts

Folgende Sondervariablen können zur Referenzierung aktueller Objekte verwendet werden:

- node
- stream

- output
- project

Mit der Ausnahme von project können sie auch neu festgesetzt werden, um den aktuellen Kontext zu ändern. Im Gegensatz zu anderen Skriptvariablen müssen Sie nicht zuerst mit dem Befehl var deklariert werden, da sie vordefiniert sind.

```
set node = create typenode
rename ^node as "mytypenode"

set output = get output :statisticsoutput
export output ^output as c:/myoutput.htm format html
```

Da diese Sondervariablen mit den Namen der von ihnen referenzierten Objekten übereinstimmen, kann der Unterschied zwischen Variable und Objekt in einigen Fällen verdunkelt sein, was zu feinen Verwendungsunterschieden führt. Für weitere Informationen siehe Thema Befehl “set” in Kapitel 4 auf S. 35.

Kommentare

Wenn einer Sondervariablen ein Wert mit einem falschen Typ zugewiesen wird (beispielsweise wenn ein Knotenobjekt auf die Variable stream gesetzt wird) wird ein Laufzeitfehler ausgegeben.

Wenn die Sondervariable verwendet werden kann, kann jede beliebige andere Variable ebenfalls verwendet werden. Das Speichern des aktuellen Streams kann beispielsweise mit folgendem Befehl durchgeführt werden:

```
save stream as "C:/My Streams/Churn.str"
```

Ebenfalls gültig ist:

```
save my_stream as 'C:/My Streams/Churn.str'
```

wobei my_stream zuvor ein Stream-Wert zugewiesen wurde.

Öffnen von Streams und anderen Objekten

Bei einem Standalone-Skript können Sie einen Stream öffnen, indem Sie den Dateinamen und den Speicherort der Datei angeben. Beispiel:

```
open stream "c:/demos/druglearn.str"
```

Andere Objekttypen können mithilfe des Befehls load geöffnet werden. Beispiel:

```
load node c:/mynode.nod

load model c:/mymodel.gm
```

“Open stream” und “load stream” im Vergleich Mit dem Befehl load stream wird der angegebene Stream zum Zeichenbereich hinzugefügt, ohne die Knoten aus dem aktuellen Stream zu löschen. Dieser Befehl wurde in früheren Versionen häufiger verwendet und wurde größtenteils durch die

Möglichkeit ersetzt, Knoten zu öffnen, zu verwalten und zwischen verschiedenen Streams zu kopieren.

Arbeiten mit mehreren Streams

Abgesehen von den Befehlen, die zum Zugriff auf Streams über das Dateisystem oder über IBM® SPSS® Collaboration and Deployment Services Repository verwendet werden (open, load und retrieve), werden die meisten Skriptbefehle automatisch auf den aktuellen Stream angewendet. Bei Standalone-Skripts können Sie jedoch mehrere Streams aus demselben Skript öffnen und bearbeiten. Dazu können Sie eine Referenz auf jeden beliebigen offenen Stream festlegen oder mithilfe des Befehls with... endwith den aktuellen Stream vorübergehend neu zuweisen.

Um beispielsweise einen anderen Stream als den aktuellen Stream zu schließen, können Sie mithilfe des Befehls get stream den gewünschten Stream referenzieren:

```
set stream = get stream "druglearn"
close stream
```

Dieses Skript weist die Sondervariable “Stream” dem Stream druglearn zu (wodurch dieser im Grunde zum aktuellen Stream wird) und schließt den Stream.

Alternativ kann der aktuelle Stream mithilfe der Anweisung with stream vorübergehend neu zugewiesen werden. Beispiel:

```
with stream 'druglearn'
  create typenode
  execute_script
endwith
```

Die oben angegebenen Anweisungen führen die Aktion create aus und führen das Skript des Streams aus, wobei der angegebene Stream als aktueller Stream fungiert. Der ursprüngliche aktuelle Stream wird wiederhergestellt, sobald die einzelnen Anweisungen ausgeführt wurden. Außerdem können bedingte Anweisungen und Schleifenkonstrukte aufgenommen werden. Beispiel:

```
with stream 'druglearn'
  create tablenode at 500 400
  create selectnode connected between :typenode and :tablenode
  for I from 1 to 5
    set :selectnode.condition = 'Age > ' >< (I * 10)
    execute :selectnode
  endfor
endwith
```

Die oben angegebenen Anweisungen setzen den aktuellen Stream für alle Ausdrücke innerhalb der Schleife auf STREAM und stellen nach Abschluss der Schleife den ursprünglichen Wert wieder her.

Lokale Skriptvariablen

Lokale Skriptvariablen werden mit dem Befehl var deklariert und nur für das aktuelle Skript festgelegt. Variablen unterscheiden sich von Parametern, die für Sitzungen, Streams oder Superknoten festgelegt werden können und nur Zeichenketten oder Zahlen enthalten dürfen.

```
var my_node
set my_node = create distributionnode
rename ^my_node as "Distribution of Flag"
```

Bei der Referenzierung bestehender Variablen müssen Sie unbedingt das Winkelzeichen (^) vor dem Parameternamen verwenden. Beim oben angegebenen Skript gilt beispielsweise Folgendes:

- Die erste Zeile deklariert die Variable.
- Die zweite Zeile legt ihren Wert fest.
- Die dritte Zeile benennt den von der Variablen referenzierten Knoten um (nicht jedoch die Variable selbst). Das Winkelzeichen gibt an, dass es sich bei ^my_node um den Namen einer Variablen und nicht um den Literalnamen des Knotens handelt. (Ohne das Winkelzeichen würde der Befehl rename nach einem Knoten mit dem Namen my_node suchen. In den ersten beiden Zeilen wird das Winkelzeichen nicht benötigt, da nur Variablen als Objekt eines var-Befehls fungieren können. Das Winkelzeichen wird nur bei der Referenzierung von bereits festgelegten Variablen verwendet. In diesem Fall würde sein Weglassen zu einer mehrdeutigen Referenz führen.)
- Beim Auflösen von Variablenreferenzen wird zuerst die Liste der lokalen Variablen durchsucht und dann erst die Liste der Sizungs-, Stream- bzw. Superknoten-Parameter. Beispiel: Wenn eine Variable x als lokale Variable und als Sitzungsparameter vorhanden ist, würde durch die Verwendung der Syntax '$P-X' in einer Skriptanweisung sichergestellt, dass der Sitzungsparameter verwendet wird und nicht die lokale Variable.

Hinweis: In der Praxis bedeutet dies: Wenn Sie eine Variable festlegen, ohne sie zuvor mithilfe eines var-Befehls zu deklarieren, wird je nach Kontext des aktuellen Skripts ein Stream-, Sitzungs- oder SuperNode-Parameter erstellt. Beispielsweise erstellt der folgende Code eine lokale Skriptvariable mit dem Namen z und setzt ihren Wert auf [1 2 3]:

```
var z
set z = [1 2 3]
```

Wenn der Befehl var weggelassen wird (und vorausgesetzt, dass noch keine Variable bzw. kein Knoten mit diesem Namen vorhanden ist), wird z statt als Variable als Parameter erstellt.

Stream-, Sitzungs- und Superknoten-Parameter

Parameter können für die Verwendung in CLEM-Ausdrücken und Skripts definiert werden. Es handelt sich dabei im Grunde um benutzerdefinierte Variablen, die mit dem aktuellen Stream, der aktuellen Sitzung bzw. dem aktuellen Superknoten gespeichert und persistent gemacht werden. Ein Zugriff auf diese Variablen ist über die Benutzeroberfläche und über Skripts möglich. Wenn Sie also einen Stream speichern, werden alle für diesen Stream festgelegten Parameter ebenfalls gespeichert. (Dies unterscheidet sie von lokalen Skriptvariablen, die nur in dem Skript verwendet

werden können, in dem sie deklariert sind.) Parameter werden häufig in Skripts als Teil eines CLEM-Ausdrucks verwendet, bei dem der Parameterwert im Skript angegeben wird.

Der Bereich eines Parameters hängt davon ab, wo er festgelegt wird:

- Stream-Parameter können in einem Stream-Skript oder im Dialogfeld "Stream-Eigenschaften" festgelegt werden und sie stehen für alle Knoten im Stream zur Verfügung. Sie werden in der Parameterliste im Expression Builder angezeigt.
- Sitzungsparameter können in einem Standalone-Skript oder im Dialogfeld "Sitzungsparameter" festgelegt werden. Sie stehen allen Streams zur Verfügung, die in der aktuellen Sitzung verwendet werden (alle Streams auf der Registerkarte "Streams" im Manager-Bereich).

Parameter können auch für Superknoten festgelegt werden. In diesem Fall sind sie nur für Knoten sichtbar, die in dem betreffenden Superknoten verkapselt sind. Für weitere Informationen siehe Thema Festlegen von Superknoten-Parametern in Kapitel 9 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Festlegen von Parametern in Skripts

Mithilfe des Befehls set und der folgenden Syntax können Sie Parameter in Skripts festlegen:

```
set Lebensmittel = Pizza
```

Wenn im aktuellen Skript keine Knoten oder Variablen mit dem Namen foodtype deklariert sind, erstellt dieser Befehl einen Parameter namens foodtype mit dem Standardwert pizza.

Benutzeroberfläche. Alternativ können Parameter über die Benutzeroberfläche festgelegt oder angezeigt werden. Wählen Sie dazu im Menü "Extras" die Option Stream-Eigenschaften bzw. Sitzungsparameter festlegen aus. Mithilfe dieser Dialogfelder können Sie auch weitere Optionen, wie beispielsweise den Speichertyp, angeben, die nicht über die Skripterstellung verfügbar sind. Für weitere Informationen siehe Thema Festlegen der Stream- und Sitzungsparameter in Kapitel 5 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Befehlszeile. Die Parameter können auch über die Befehlszeile festgelegt werden. In diesem Fall werden sie als Sitzungsparameter erstellt.

Verweise auf Parameter in Skripts

Sie können auf zuvor erstellte Parameter Bezug nehmen, indem Sie sie in einzelne Anführungszeichen einschließen und ihnen die Zeichenkette $P voranstellen. Beispiel: '$P-minvalue'. Außerdem können Sie einfach auf den Parameternamen (z. B. minvalue) Bezug nehmen. Der Wert für einen Parameter ist immer eine Zeichenkette oder eine Zahl. Sie können beispielsweise den Parameter foodtype referenzieren und mit folgender Syntax einen neuen Wert festlegen:

```
set Lebensmittel = Nudeln
```

Außerdem können Sie auf Parameter im Kontext eines CLEM-Ausdrucks, der in einem Skript verwendet wird, Bezug nehmen. Das folgende Stream-Skript ist ein Beispiel. Es legt die Eigenschaften für einen Auswahlknoten so fest, dass Datensätze aufgenommen werden, bei denen der Wert für Age über dem von dem Stream-Parameter cutoff angegebenen Wert liegt. Der Parameter wird in einem CLEM-Ausdruck mit der ordnungsgemäßen Syntax für CLEM verwendet – '$P-cutoff':

```
set :selectnode {
mode = "Include"
condition = "Age >= '$P-cutoff'"
}
```

Das oben angegebene Skript verwendet den Standardwert für den Stream-Parameter cutoff. Sie können einen neuen Parameterwert angeben, indem Sie folgende Syntax zu den oben stehenden Auswahlknoten-Spezifikationen hinzufügen:

```
set cutoff = 50
```

Das sich daraus ergebende Skript wählt alle Datensätze aus, bei denen der Wert von Age größer ist als 50.

Für weitere Informationen siehe Thema Stream-, Sitzungs- und Superknoten-Parameter in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Steuern der Skriptausführung

Bei der Skriptausführung werden normalerweise die Anweisungen einzeln nacheinander verarbeitet. Sie können jedoch diese Ausführungsreihenfolge durch eine if-Anweisung und mehrere Arten von for-Schleifen außer Kraft setzen. Beispiel:

```
if s.maxsize > 10000 then
s.maxsize = 10000
connect s to :derive
endif
```

Für die for-Schleife ist eine Vielzahl von Formen möglich:

```
for PARAMETER in LISTE
 ANWEISUNGEN
endfor
```

Das obige Skript führt STATEMENTS einmal für jeden Wert in LIST aus, der PARAMETER zugewiesen ist. Dabei wird die Reihenfolge der Liste verwendet. Die Liste weist keine einschließenden Klammern auf, da es sich bei ihrem Inhalt um Konstanten handelt. Außerdem steht eine Reihe weiterer Formen zur Verfügung. Für weitere Informationen siehe Thema Allgemeine Skriptbefehle in Kapitel 4 auf S. 33.

Operatoren in Skripts

Zusätzlich zu den üblichen CLEM-Operatoren können Sie mit den Operatoren "+" und "–" lokale Skriptvariablen (mithilfe eines var-Befehls deklariert) bearbeiten. Der Operator "+" fügt ein Element zur Liste hinzu und der Operator "–" entfernt ein Element. Hier ein Beispiel:

```
var z       # Neue lokale Variable erstellen
set z = [1 2 3]       # sie in die Liste mit 1, 2 und 3 setzen
set z = z + 4       # ein Element hinzufügen; z ist jetzt gleich [1 2 3 4]
```

Diese Operatoren können nicht mit Stream-, Superknoten- oder Sitzungsparametern (mithilfe des set-Befehls in Skripts definiert) oder außerhalb von Skripts in allgemeinen CLEM-Ausdrücken (beispielsweise als Formel in einem Ableitungsknoten) verwendet werden.

CLEM-Ausdrücke in Skripts

In IBM® SPSS® Modeler-Skripts können CLEM-Ausdrücke, -Funktionen und -Operatoren verwendet werden. Die Skriptausdrücke dürfen jedoch keine Aufrufe für @-Funktionen, Datums-/Uhrzeitfunktionen oder bitweise Operationen enthalten. Außerdem gelten folgende Regeln für CLEM-Ausdrücke in Skripts:

- Parameter müssen in einfachen Anführungszeichen und mit dem Präfix $P- angegeben werden.
- CLEM-Ausdrücke müssen in Anführungszeichen eingeschlossen werden. Wenn der CLEM-Ausdruck selbst Zeichenketten oder Feldnamen in Anführungszeichen enthält, so muss den eingebetteten Anführungszeichen ein umgekehrter Schrägstrich (\) vorangestellt werden. Für weitere Informationen siehe Thema Skriptsyntax auf S. 21.

Globale Werte, beispielsweise GLOBAL_MEAN(Age), können in Skripts verwendet werden. Die Funktion @GLOBAL selbst kann jedoch nicht innerhalb der Skriptumgebung verwendet werden.

Hier einige Beispiele für in Skripts verwendete CLEM-Ausdrücke:

```
set :balancenode.directives = [{1.3 "Alter > 60"}]

set :fillernode.condition = "(Alter > 60) and (BP = \"Hoch\")"

set :derivenode.formula_expr = "substring(5, 1, Medikament)"

set Flag:derivenode.flag_expr = "Medikament = X"

set :selectnode.condition = "Alter >= '$P-cutoff'"

set :derivenode.formula_expr = "Alter - GLOBAL_MEAN(Alter)"
```

Einfügen von Kommentaren und Fortsetzungen

Folgende Zeichen werden in Skripts zur Kennzeichnung von Kommentaren und Fortsetzungen verwendet:

Zeichen	Verwendung	Beispiel
#	Das Rautenzeichen zeigt einen Kommentar an. Der Rest der Zeile wird ignoriert.	#Dies ist ein einzeiliger Kommentar.
\	Ein umgekehrter Schrägstrich am Ende der Zeile zeigt an, dass die Anweisung auf der nächsten Zeile fortgesetzt wird.	Siehe unten stehendes Beispiel.
/*	Die Zeichenfolge /* zeigt den Anfang eines Kommentars an. Es wird alles ignoriert, bis die Markierung für das Kommentarende (*/) gefunden wird.	Siehe unten stehendes Beispiel.
"""	Literalzeichenketten oder Blöcke, die Zeilenumbrüche, Leerzeichen oder einzelne oder doppelte Anführungszeichen innerhalb des Blocks enthalten, können in dreifache Anführungszeichen eingeschlossen werden. Für weitere Informationen siehe Thema Blöcke mit Literaltext auf S. 31.	

Beispiele

```
/* Dies ist ein
mehrzeiliger
Kommentar
*/


#das Folgende ist eine mehrzeilige
Anweisungt set :fixedfilenode.fields = [{"Alter" 1 3}\
 {"Geschlecht" 5 7} {"BP" 9 10} {"Cholesterin" 12 22}\
 {"Na" 24 25} {"K" 27 27} {"Medikament" 29 32}]
```

Blöcke mit Literaltext

Literaltextblöcke, die Leerzeichen, Tabulatoren und Zeilenumbrüche enthalten, können in Skripts aufgenommen werden, indem Sie sie in dreifache Anführungszeichen einschließen. Jeder Text in dem in Anführungszeichen eingeschlossenen Block wird als Literaltext beibehalten, einschließlich Leerzeichen, Zeilenumbrüchen und eingebetteten einfachen und doppelten Anführungszeichen. Es sind keine Fortsetzungs- oder Escape-Zeichen erforderlich.

Dieses Verfahren können Sie beispielsweise verwenden, um eine Reihe von Strukturerweiterungsrichtlinien in ein Skript einzubetten:

```
set :cartnode.tree_directives = """
  Create Root_Node
  Grow Node Index 0 Children 1 2 SplitOn ("DRUG",
    Group ( "drugA", "drugB", "drugC" )
    Group ( "drugY", "drugX" ))
  End Tree
```

"""

Dies ist auch für Pfade, Anmerkungen und andere Verwendungszwecke sinnvoll. Beispiel:

```
set :node.annotation = """Dieser Knoten wurde erstellt, um zu ermitteln, welcher der Faktoren
   Milchprodukte
   Fisch
   Gemüse
   Fleisch
   Teigwaren
   Süßigkeiten
ein ungewöhnliches Absatzverhalten aufweist"""
```

Zeilenumbrüche nach dem öffnenden Literalkennzeichen werden von IBM® SPSS® Modeler ignoriert. So entspricht folgendes Beispiel dem vorangegangenen:

```
set :node.annotation = """
Dieser Knoten wurde erstellt, um zu ermitteln, welcher der Faktoren
usw.
"""
```

Kapitel
4

Skriptbefehle

Dieser Abschnitt bietet einen Überblick über die Befehle, die in IBM® SPSS® Modeler-Skripts verwendet werden können (nach Objekttyp geordnet). Weitere Informationen zur Skriptsprache finden Sie hier: Kapitel 3. Weitere Informationen zu Knoten-, Stream-, Projekt- und Superknoteneigenschaften finden Sie hier: Kapitel 9 bis Kapitel 22.

Allgemeine Skriptbefehle

Sofern nicht anders angegeben, stehen folgende Befehle in allen Standalone-, Stream-, und Superknoten-Skripts zur Verfügung.

execute_all

```
execute_all
```

Führt alle Endknoten im aktuellen Stream aus.

```
open stream "c:/demos/druglearn.str"
execute_all
```

execute_script

```
execute_script
```

Nur Standalone-Skripts. Führt das dem aktuellen Stream zugeordnete Stream-Skript aus. (Auf Standalone-Skripts beschränkt, da andernfalls das Stream-Skript sich selbst aufrufen würde.)

```
open stream "c:/demos/mysample.str"
execute_script
```

exit

```
exit CODE
```

Führt das aktuelle Skript aus. Mit dem Beenden-Code (exit) kann das Skript oder die Bedingung eines Streams oder Knotens evaluiert werden. Beispiel:

```
create tablenode
create variablefilenode
connect :variablefilenode to :tablenode

set :variablefilenode.full_filename = "$CLEO_DEMOS/DRUG1n"
execute 'Table'

set param  = value :tablenode.output at 1 1
```

```
if ^param = 23 then
  create derivenode
  else exit 2
endif
```

for...endfor

Der Befehl for...endfor durchläuft auf der Grundlage einer Bedingung als Schleife eine Menge von Anweisungen. Der Befehl kann verschiedene Formen annehmen, die jedoch alle dieselbe allgemeine Struktur aufweisen.

```
for PARAMETER in LIST
  STATEMENTS
endfor
```

for PARAMETER in LISTE.Führt STATEMENTS einmal für jeden Wert in LIST aus, der PARAMETER zugewiesen ist. Dabei wird die Reihenfolge der Liste verwendet. Beispielsweise kann die Eigenschaft Filter.include wie folgt für mehrere Felder auf true gesetzt werden:

```
for f in Age Sex
  set Filter.include.^f=true
endfor
```

for PARAMETER from N to M. Führt STATEMENTS einmal für jede Ganzzahl von N bis einschließlichM aus. Beispiel:

```
for I from 1 to 5
  set :selectnode.condition = 'Age > ' >< (I * 10)
  execute :selectnode
endfor
```

for PARAMETER in_fields_to NODE. Führt STATEMENTS einmal für jedes Feld auf der aufwärts gelegenen Seite von NODE aus. Das folgende Skript beispielsweise legt für die Eigenschaft include für alle Felder den Wert true fest, auch für Felder, die zuvor auf false gesetzt waren:

```
for f in_fields_to Filter
  set Filter.include.^f = "true"
endfor
```

Hinweis: In Fällen, bei denen ein Knoten mehrere Eingabefelder mit demselben Namen enthalten kann – beispielsweise bei Merge- oder Anhangknoten – gibt diese Methode nicht die Liste der aufwärts-, sondern die der abwärtsgelegenen Felder aus, um mögliche Konflikte zu vermeiden.

for PARAMETER in_fields_at NODE. Führt STATEMENTS einmal für jedes Feld aus, das vom (oder unterhalb vom) angegebenen NODE ausgeht. Wenn der Knoten also ein Filter ist, werden nur Felder aufgenommen, die den Filter passiert haben, und der Knoten sollte kein Endknoten sein, da in diesem Fall keine Felder ausgegeben werden. Beispielsweise hätte das folgende Skript im

Gegensatz zu dem oben stehenden keine Wirkung, da die Schleife nur für diejenigen Felder ausgeführt wird, die bereits auf true gesetzt sind:

```
for f in_fields_at Filter
  set Filter.include.^f = "true"
endfor
```

for PARAMETER in_models. Führt STATEMENTS einmal für jedes Modell-Nugget in der Modellpalette aus. Beispielsweise fügt das folgende Skript die einzelnen Modelle aus der Palette in den aktuellen Stream ein. (Mit der Variablen xpos wird verhindert, dass die Knoten im Stream-Zeichenbereich übereinander gestapelt werden.)

```
var xpos
set xpos = 100
for m in_models
 set xpos = xpos + 100
 insert model ^m at ^xpos 100
endfor
```

for PARAMETER in_streams. *Nur Standalone-Skripts.* Führt STATEMENTS einmal für jeden geladenen Stream aus (wie in der Stream-Palette aufgeführt). Wenn es sich bei PARAMETER um die Sondervariable "stream" handelt, wird der aktuelle Stream für STATEMENTS in der Schleife festgelegt. Der ursprüngliche Wert von "stream" wird bei Beendigung der Schleife wiederhergestellt.

if...then...else...

```
if EXPR then
   STATEMENTS 1
 else
   STATEMENTS 2
endif
```

Führt STATEMENTS 1 aus, wenn der angegebene Ausdruck wahr ist, und STATEMENTS 2, wenn der Ausdruck falsch ist. Die Klausel else ist optional.

```
if :samplenode.use_max_size = true then
   set x = "yes"
 else
   set x = "no"
endif
```

Befehl "set"

```
set VARIABLE = AUSDRUCK
set PARAMETER = AUSDRUCK
set EIGENSCHAFT = AUSDRUCK
```

Legt den Wert einer lokalen Skriptvariablen, einer Sondervariablen, eines Parameters oder einer Eigenschaft fest.

Festlegen von Variablen

Um den Wert einer lokalen Skriptvariablen festzulegen, müssen Sie zunächst die Variable mithilfe des Befehls var deklarieren. Beispiel:

```
var xpos
var ypos
set xpos = 100
set ypos = 100
```

Der Wert der Variabel kann ein für die Skripterstellung gültiger CLEM-Ausdruck sein, ein Skriptbefehl, der einen Wert ergibt (z. B. load, create oder get) oder ein Literalwert.

```
set xpos = ^xpos + 50
```

```
var x
set x = create typenode
```

```
var s
set s = get stream 'Druglearn'
```

Festlegen von Sondervariablen für Referenzobjekte

Die Sondervariablen node, stream, output und project dienen zur Referenzierung des “aktuellen” Objekts in den einzelnen Kontexten. Mit der Ausnahme von project können sie auch neu festgesetzt werden, um den aktuellen Kontext zu ändern. Im Gegensatz zu anderen Skriptvariablen müssen sie nicht zuerst mit dem Befehl var deklariert werden, da sie vordefiniert sind.

```
set node = create typenode
rename ^node as "mytypenode"
```

```
set output = get output :statisticsoutput
export output ^output as c:/myoutput.htm format html
```

Diese Variablen sind zwar nützlich, weisen jedoch einige leichte Unterschiede in der Verwendung auf, wie im folgenden Beispiel zu sehen:

```
set stream = get stream 'Stream7'
set ^stream.execute_method = "Script"
save stream as c:/sample7.str
close stream
```

- Die erste Zeile setzt den aktuellen Stream neu fest oder konkreter, legt den Wert der Sondervariablen stream fest. (Mit anderen Worten: stream ist eine Variable und nicht Teil des Befehls.)
- Die zweite Zeile verwendet diese Variable, um eine Eigenschaft für den aktuellen Stream festzulegen (weitere Eigenschaften finden Sie im Folgenden). Das Winkelzeichen gibt an, dass es sich bei ^stream um den Namen einer Variablen und nicht um ein Objekt, wie

beispielsweise einen Knoten, handelt. (Ohne das Winkelzeichen würde der Befehl set nach einem Knoten mit dem Namen *stream* suchen.

- Die letzten beiden Zeilen speichern und schließen den aktuellen Stream. Wie zuvor ist stream eine Variable. In diesem Fall wird jedoch kein Winkelzeichen verwendet, da die Befehle save und close in diesem Beispiel nur auf einen Stream angewendet werden können. (Das Winkelzeichen wird nur dann verwendet, wenn seine Entfernung zu einer mehrdeutigen Referenz führen würde.)

Referenzieren des aktuellen Projekts. Die Sondervariable project kann zur Referenzierung des aktuellen Projekts verwendet werden (siehe unten angegebenes Beispiel für die Festlegung der Projekteigenschaften). Der Wert von project kann nicht neu festgesetzt werden, da jeweils nur ein einziges Projekt geöffnet (und damit aktuell) sein kann.

Festlegen von Parametern

Stream-, Sitzungs- und Superknotenparameter können auf dieselbe Weise festgelegt werden wie Variablen, jedoch ohne den Befehl var zu verwenden.

```
set p = 1
set minvalue = 21
```

Hinweis: In der Praxis bedeutet dies: Wenn das Objekt eines set-Befehls nicht mit dem Namen einer deklarierten Variablen, einer Sondervariablen oder eines bestehenden Objekts, wie beispielsweise einem Knoten, übereinstimmt, wird ein Parameter erstellt. Für weitere Informationen siehe Thema Stream-, Sitzungs- und Superknoten-Parameter in Kapitel 3 auf S. 27.

Festlegen von Knoten-, Stream- und Projekteigenschaften

Es können auch Eigenschaften für Knoten, Streams und Projekte festgelegt werden. Beispiel:

```
set :variablefilenode.full_filename = "$CLEO_DEMOS/DRUG1n"

set ^stream.execute_method = "Script"

load project "C:/myproject.cpj"
set ^project.structure = Phase
```

Eine vollständige Liste der für Knoten, Streams und Projekte verfügbaren Eigenschaften finden Sie hier: *Eigenschaftsverweis* auf S. 119.

Festlegen mehrerer Eigenschaften. Sie können den Eigenschaften für Knoten oder Objekte in einer einzigen Operation mehrere Ausdrücke zuweisen. Diese Methode wird verwendet, wenn an einem Knoten mehrere Änderungen vorgenommen werden müssen, bevor das Datenmodell bestimmt wird. Das für die Festlegung mehrerer Eigenschaften verwendete Format lautet:

```
set KNOTEN {
  KNOTENEIGENSCHAFT1 = AUSDRUCK1
  KNOTENEIGENSCHAFT2 = AUSDRUCK2
}
```

Beispiel:

```
set :samplenode {
 max_size = 200
 mode = "Include"
 sample_type = "First"
}

set ^project {
 summary = "Erste Modellierungsarbeiten an den aktuellen Daten"
 ordering = NameAddedType
}
```

Festlegen der Flag-Werte ("true" und "false"). Beim Lesen bzw. Schreiben der Eigenschaften vom Typ "Flag" sollten die Werte true und false in Kleinbuchstaben geschrieben werden. Beispiel:

```
set :variablefilenode.read_field_names = true
```

Hinweis: (Variationen mit Off, OFF, off, No, NO, no, n, N, f, F, false, False, FALSE oder 0 werden beim Festlegen von Werten ebenfalls erkannt, können jedoch beim Lesen von Eigenschaftswerten in manchen Fällen zu Fehlern führen. Alle anderen Werte werden als wahr betrachtet. Durch die durchgängige Verwendung von true und false können Verwechslungen vermieden werden.)

Beispiel: Festlegen der Knoteneigenschaften

Es gibt zahlreiche knotenspezifische Eigenschaften (zuweilen als Slot-Parameter bezeichnet), die zur Festlegung der Optionen verwendet werden, die in den Dialogfeldern der Benutzeroberfläche für die einzelnen Knoten gefunden wurden. Um beispielsweise einen Stream zu erstellen und Optionen für die einzelnen Knoten festzulegen, kann ein Skript der folgenden Art verwendet werden: Weitere Informationen zu Knoten-, Stream-, Projekt- und Superknoteneigenschaften finden Sie hier: Kapitel 9 bis Kapitel 22.

```
create varfilenode at 100 100
set :varfilenode {
full_filename = "demos/drug1n"
read_field_names = true
}
create tablenode at 400 100
create samplenode connected between :varfilenode and :tablenode
set :samplenode {
max_size = 200
mode = "Include"
sample_type = "First"
}
create plotnode at 300 300
create derivenode connected between drug1n and :plotnode
set :derivenode {
new_name = "Ratio of Na to K"
formula_expr = "'Na' / 'K'"
}
set :plotnode {
x_field = 'Ratio of Na to K'
```

```
y_field = 'Age'
color_field = 'BP'
}
```

Befehl "var"

```
var VARNAME
```

Deklariert eine lokale Skriptvariable.

```
var my_node
set my_node = create distributionnode
rename ^my_node as "Distribution of Flag"
```

Variablen unterscheiden sich von Parametern, die für Sitzungen, Streams oder Superknoten festgelegt werden können und nur Zeichenketten oder Zahlen enthalten dürfen. In der Praxis bedeutet dies: Wenn Sie eine Variable festlegen, ohne sie zuvor mithilfe eines VAR-Befehls zu deklarieren, wird je nach Kontext des aktuellen Skripts ein Stream-, Sitzungs- oder SuperNode-Parameter erstellt. Für weitere Informationen siehe Thema Lokale Skriptvariablen in Kapitel 3 auf S. 27.

Knotenobjekte

Die folgenden Skriptbefehle stehen für Knotenobjekte zur Verfügung.

create NODE

```
create NODE
create NODE at X Y
create NODE between NODE1 and NODE2
create NODE connected between NODE1 and NODE2
```

Erstellt einen Knoten mit dem angegebenen Typ. Beispiel:

```
create statisticsimportnode
```

Optional können auch Positions- und Verbindungsoptionen angegeben werden:

```
create featureselectionnode at 400 100
```

```
create typenode between :statisticsimportnode and :featureselectionnode
```

```
create selectnode connected between :typenode and :featureselectionnode
```

Außerdem können Sie einen Knoten mit Variablen erstellen, um Mehrdeutigkeiten zu vermeiden. In dem unten stehenden Beispiel wird ein Typknoten erstellt und die Referenzvariable *x* wird so festgelegt, dass sie eine Referenz zu diesem Typknoten enthält. Anschließend können Sie mit der Variablen *x* das von *x* referenzierte Objekt ausgeben (in diesem Fall den Typknoten) und weitere Operationen durchführen, beispielsweise den neuen Knoten umbenennen, positionieren oder verbinden.

```
var x
set x = create typenode
rename ^x as "mytypenode"
position ^x at 200 200
var y
set y = create varfilenode
rename ^y as "mydatasource"
position ^y at 100 200
connect ^y to ^x
```

Im oben angegebenen Beispiel werden zwei Knoten erstellt, jeder davon umbenannt, sie werden positioniert und schließlich im Stream-Zeichenbereich verbunden.

Abbildung 4-1
Mithilfe von Variablen erstellte Knoten

Alternativ kann die (vordefinierte) Sondervariable node auf ähnliche Weise wie die Variablen x und y im obigen Beispiel verwendet werden. In diesem Fall muss die Variable nicht über den Befehl var deklariert werden (da sie vordefiniert ist) und das entstehende Skript ist möglicherweise etwas leichter lesbar.

```
set node = create typenode
rename ^node as "mytypenode"
position ^node at 200 200
set node = create varfilenode
rename ^node as "mydatasource"
position ^node at 100 200
connect mydatasource to mytypenode
```

Hinweis: Sondervariablen, beispielsweise node, können für die Referenzierung mehrerer Knoten wiederverwendet werden. Verwenden Sie einfach den Befehl set, um das von der Variablen referenzierte Objekt neu festzusetzen. Für weitere Informationen siehe Thema Festlegen des aktuellen Objekts in Kapitel 3 auf S. 24.

Duplizieren von Knoten. Alternativ können Sie den Befehl duplicate verwenden, um einen bestehenden Knoten zu duplizieren. Für weitere Informationen siehe Thema duplicate KNOTEN auf S. 41.

connect KNOTEN

```
connect KNOTEN1 to KNOTEN2
connect KNOTEN1 between KNOTEN2 and KNOTEN3
```

Verbindet NODE1 gemäß der Angabe mit anderen Knoten.

```
connect :statisticsimportnode to :typenode
```

```
connect :selectnode between :typenode and :featureselectionnode
```

delete KNOTEN

delete KNOTEN

Löscht den angegebenen Knoten aus dem aktuellen Stream.

delete :statisticsimportnode

delete DRUG1N:variablefilenode

KNOTEN deaktivieren

KNOTEN deaktivieren

Löscht den angegebenen Knoten aus dem aktuellen Stream mit dem Ergebnis, dass der Knoten während der Ausführung des Streams ignoriert wird. Auf diese Weise müssen Sie den Knoten nicht löschen oder umgehen und können seine Verbindung mit den restlichen Knoten bestehen lassen. Die Knoteneinstellungen können nach wie vor bearbeitet werden. Änderungen werden jedoch erst dann wirksam, wenn Sie den Knoten wieder aktivieren.

disable :statisticsimportnode

disable DRUG1N:variablefilenode

disconnect KNOTEN

disconnect KNOTEN
disconnect KNOTEN1 from KNOTEN2
disconnect KNOTEN1 between KNOTEN2 and KNOTEN3

Trennt den angegebenen Knoten von allen anderen Knoten (Standard) oder von den angegebenen Knoten.

disconnect :typenode

disconnect :typenode from :selectnode

duplicate KNOTEN

duplicate KNOTEN as NEUERNAME

Erstellt einen neuen Knoten als Duplikat des angegebenen Knotens. Optional kann die Position auch in absoluter oder relativer Form angegeben werden.

duplicate :derivenode as flag1 at 100 400

duplicate flag1 as flag2 connected between flag1 and flag3

KNOTEN aktivieren

KNOTEN aktivieren

Aktiviert einen zuvor deaktivierten Knoten im aktuellen Stream mit dem Ergebnis, dass der Knoten während der Ausführung des Streams mit eingeschlossen wird. Die Änderungen werden nicht wirksam, wenn Sie die Knoteneinstellungen bei deaktiviertem Knoten bearbeitet haben.

```
enable :statisticsimportnode

enable DRUG1N:variablefilenode
```

execute KNOTEN

```
execute KNOTEN
```

Führt den angegebenen Knoten aus. Beispiel:

```
execute :neuralnetworknode
```

Wenn es sich nicht um einen Endknoten handelt, entspricht die Ausführung der Option Ab hier ausführen aus dem Popup-Menü.

So führen Sie alle Endknoten im aktuellen Stream aus:

```
execute_all
```

Nur Standalone-Skripts. So führen Sie das dem aktuellen Stream zugeordnete Stream-Skript aus:

```
execute_script
```

Hinweis: Skripts, die unterschiedlichen Streams zugeordnet sind, können ausgeführt werden, indem der Stream als aktueller Stream festgelegt wird, oder über den Befehl with. Für weitere Informationen siehe Thema Arbeiten mit mehreren Streams in Kapitel 3 auf S. 26.

export KNOTEN as DATEI

```
export node KNOTEN in VERZEICHNIS format FORMAT
export node KNOTEN as DATEI format FORMAT
```

PMML-Export. So exportieren Sie ein generiertes Modell im PMML-Format:

```
export Drug as c:/mymodel.xml format pmml
```

SQL-Export. So exportieren Sie ein generiertes Modell im SQL-Format:

```
export Drug in c:/mymodels format sql

export Drug as c:/mymodel.txt format sql
```

Knotendetails. So exportieren Sie Knotendetails im HTML- bzw. Textformat:

```
export Drug as c:\mymodel.htm format html

export Drug as c:\mymodel.txt format text
```

Knotenübersicht. So exportieren Sie die Knotenübersicht im HTML- bzw. Textformat:

```
export Drug summary in c:/mymodels format html

export Drug summary as c:/mymodel.txt format text

export 'assocapriori' as 'C:/temp/assoc_apriori' format html
```

flush KNOTEN

```
flush KNOTEN
```

Leert den Cache am angegebenen Knoten oder allen Knoten im Stream. Wenn der Cache nicht aktiviert oder für einen bestimmten Knoten nicht voll ist, hat diese Operation keine Auswirkungen.

```
flush :mergenode
```

So leeren Sie alle Knoten im aktuellen Stream:

```
flush_all
```

get node KNOTEN

```
get node KNOTEN
```

Ruft eine Referenz zu einem bestehenden Knoten ab. Dies kann eine nützliche Methode sein, um sicherzustellen, dass die Referenzen auf Knoten nicht mehrdeutig sind.

```
var mynode
set mynode = get node flag1:derivenode
position ^mynode at 400 400
```

load node DATEINAME

```
load node DATEINAME
```

Lädt einen gespeicherten Knoten in den aktuellen Stream.

```
load node c:/mynode.nod
```

position KNOTEN

```
position KNOTEN at X Y
position KNOTEN between KNOTEN1 and KNOTEN2
position KNOTEN connected between KNOTEN1 and KNOTEN2
```

Positioniert einen Knoten in absoluter oder relativer Form im Stream-Zeichenbereich. Optional können auch Verbindungsoptionen angegeben werden:

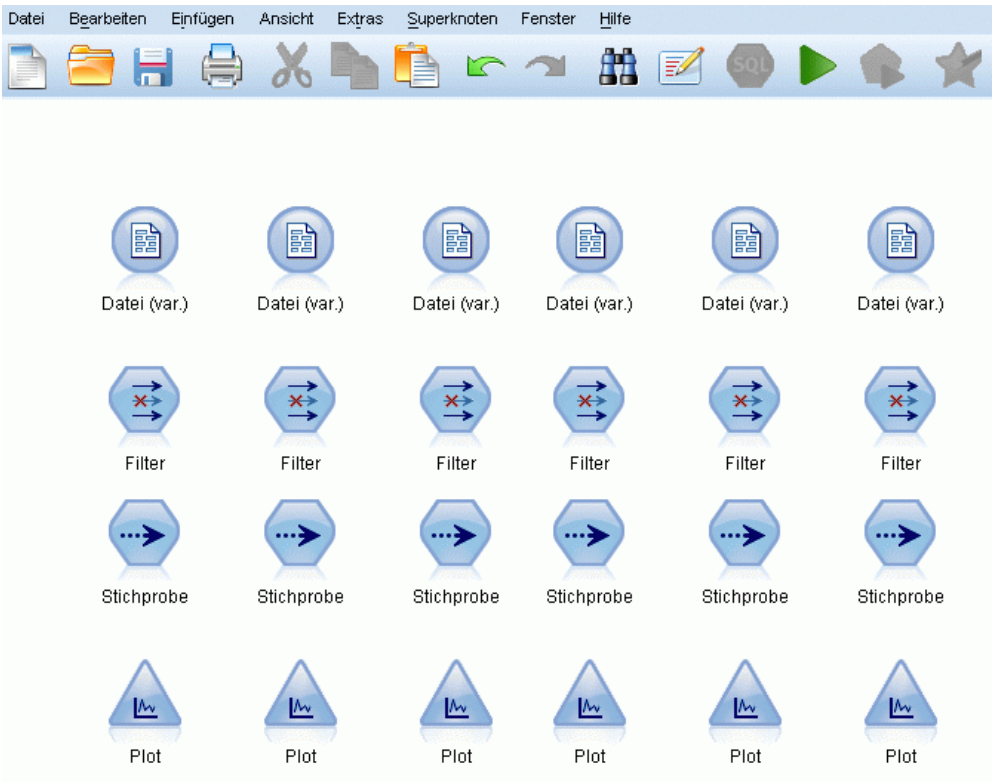
```
position DRUG1n:variablefilenode at 100 100

position Drug:net between DRUG2n and analysis
```

position :typenode connected between :variablefilenode and :tablenode

Positionierungskoordinaten

Für die Positionierung im Stream-Zeichenbereich wird ein unsichtbares *x*-*y*-Gitter verwendet. Sie können das unten stehende Bild als Referenz für die *x*-*y*-Gitterkoordinaten verwenden.

Abbildung 4-2
An der mithilfe der x-y-Koordinaten angegebenen Position erstellte Knoten



rename NODE as NEWNAME

rename NODE as NEWNAME

Benennt den angegebenen Knoten um.

rename :derivenode as 'Flag1'

rename :varfilenode as 'testdata'

retrieve node REPOSITORY_PFAD

retrieve node REPOSITORY_PFAD {label LABEL | version VERSION}

Ruft den angegebenen Knoten aus IBM® SPSS® Collaboration and Deployment Services Repository ab. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
retrieve node "/samples/drugtypenode"
```

save node KNOTEN as DATEINAME

```
save node KNOTEN as DATEINAME
```

Speichert den angegebenen Knoten.

```
save node :statisticsimportnode as c:/mynode.nod
```

store node KNOTEN as REPOSITORY_PFAD

```
store node KNOTEN as REPOSITORY_PFAD {label LABEL}
```

Speichert einen Knoten in IBM® SPSS® Collaboration and Deployment Services Repository. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
store node DRUG1n as "/samples/drug1ntypenode"
```

```
store node :typenode as "/samples/drugtypenode"
```

Modellobjekte

Die folgenden Skriptbefehle stehen für Modellobjekte zur Verfügung.

Modell-Nugget-Namen

Modell-Nuggets (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 Modell-Nuggets 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. Für weitere Informationen siehe Thema Modell-Nugget-Knoten – Eigenschaften in Kapitel 17 auf S. 260.

Das folgende Skript beispielsweise fügt ein Modell-Nugget zum aktuellen Stream hinzu, verbindet es mit einem Typknoten und erstellt einen Tabellenknoten und führt ihn aus. Beachten Sie, dass ein anderer Name verwendet wird, um das Modell aus der Palette einzufügen, als der Name, der nach dem Hinzufügen zum Stream zur Referenzierung des “apply”-Modell-Knotens verwendet wird (:featureselection bzw. :applyfeatureselectionnode).

```
insert model :featureselection at 150 250
connect Type to :applyfeatureselectionnode
```

```
create tablenode at 250 250
connect :applyfeatureselectionnode to :tablenode
execute :tablenode
```

Hinweis: Dies ist lediglich ein Beispiel. Unter normalen Umständen wird die Referenzierung von Modellen sowohl nach Namen *als auch* nach Typ empfohlen, um Verwirrungen zu vermeiden (z. B. response_01:featureselection).

Namen von Modell-Nuggets (Modellierungspalette)

Modellname	**Modell**
anomalydetection	Anomalie
apriori	A Priori
autoclassifier	Automatischer Klassifizierer
autocluster	Automatisches Clustering
autonumeric	Auto-Numerisch
bayesnet	Bayes-Netzwerk
c50	C5.0
carma	Carma
cart	C&R-Baum
chaid	CHAID
coxreg	Cox-Regression
decisionlist	Entscheidungsliste
Diskriminanz	Diskriminanz
Faktor	Faktor/PCA
featureselection	Funktionsauswahl
genlin	Verallgemeinerte lineare Regression
kmeans	K-Means
knn	*k*-Nächste-Nachbarn
kohonen	Kohonen
linear	Linear
logreg	Logistische Regression
neuralnetwork	Netzwerk
quest	QUEST
regression	Lineare Regression
sequence	Sequenz
slrm	Selbstlern-Antwortmodell
statisticsmodel	IBM® SPSS® Statistics-Modell
svm	Support Vector Machine
timeseries	Zeitreihen
twostep	Two Step

Namen von Modell-Nuggets (Datenbank-Modellierungspalette)

Modellname	Modell
db2imassoc	IBM ISW-Assoziation
db2imcluster	IBM ISW-Clustering
db2imreg	IBM ISW-Regression
db2imsequence	IBM ISW-Sequenz
db2imtree	IBM ISW-Entscheidungsbaum
msassoc	MS-Assoziationsregeln
msbayes	MS Naive Bayes
mscluster	MS-Clustering
mslogistic	MS – Logistische Regression
msneuralnetwork	MS – Neuronales Netzwerk
msregression	MS – Lineare Regression
mssequencecluster	MS Sequenz-Clustering
mstimeseries	MS Time Series
mstree	MS-Entscheidungsbaum
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oraapriori	Oracle Apriori
oradecisiontree	Oracle Decision Tree
oraglm	Oracle GLM
orakmeans	Oracle *k*-Means
oramdl	Oracle MDL
oranb	Oracle Naive Bayes
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

Vermeidung doppelter Modellnamen

Bei der Verwendung von Skripts 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 Skripterstellung eindeutige Namen für die generierten Modelle erforderlich sein.

So legen Sie Optionen für doppelte Modellnamen fest:

- Wählen Sie die folgenden Befehle aus den Menüs aus:
  Werkzeuge > Benutzeroptionen
- Klicken Sie auf die Registerkarte Benachrichtigungen.
- Wählen Sie die Option Bisheriges Modell ersetzen, um die Vergabe doppelter Namen für generierte Modelle zu beschränken.

Das Verhalten der Skriptausführung 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 Skript in mehreren Iterationen eine Schleife durchläuft und jedes Mal ein anderes Modell erstellt). Diese Option steht jedoch nicht zur Verfügung, wenn dasselbe Skript 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).

delete model MODELL

```
delete model MODELL
```

Löscht ein angegebenes Modell (oder alle Modelle) aus der Palette der Modell-Nuggets.

```
delete model Drug
```

```
delete model Drug:c50
```

So löschen Sie das zuletzt vom aktuellen Skript eingefügte Modell:

```
delete last model
```

Damit diese letzte Anweisung funktioniert, muss die Anweisung insert model in der aktuellen Skriptausführung mindestens einmal ausgeführt worden sein.

So löschen Sie alle Modell-Nuggets aus der Modellpalette:

```
clear generated palette
```

export model MODELL as DATEI

```
export model MODELL in VERZEICHNIS format FORMAT
export model MODELL as DATEI format FORMAT
```

PMML-Export. So exportieren Sie das generierte Modell im PMML-Format:

```
export model Drug in c:/mymodels format pmml
```

```
export model Drug as c:/mymodel.xml format pmml
```

Für weitere Informationen siehe Thema Importieren und Exportieren von Modellen als PMML in Kapitel 10 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

SQL-Export. So exportieren Sie ein generiertes Modell im SQL-Format:

```
export Drug in c:/mymodels format sql
```

```
export Drug as c:/mymodel.txt format sql
```

Hinweis: SQL-Export ist nur für bestimmte Modelltypen verfügbar. Für weitere Informationen siehe Thema Durchsuchen von Modell-Nuggets in Kapitel 3 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Modelldetails. So exportieren Sie Modelldetails (wie beim Durchsuchen des Modell-Nuggets auf der Registerkarte “Modell” angegeben) im HTML- bzw. Textformat:

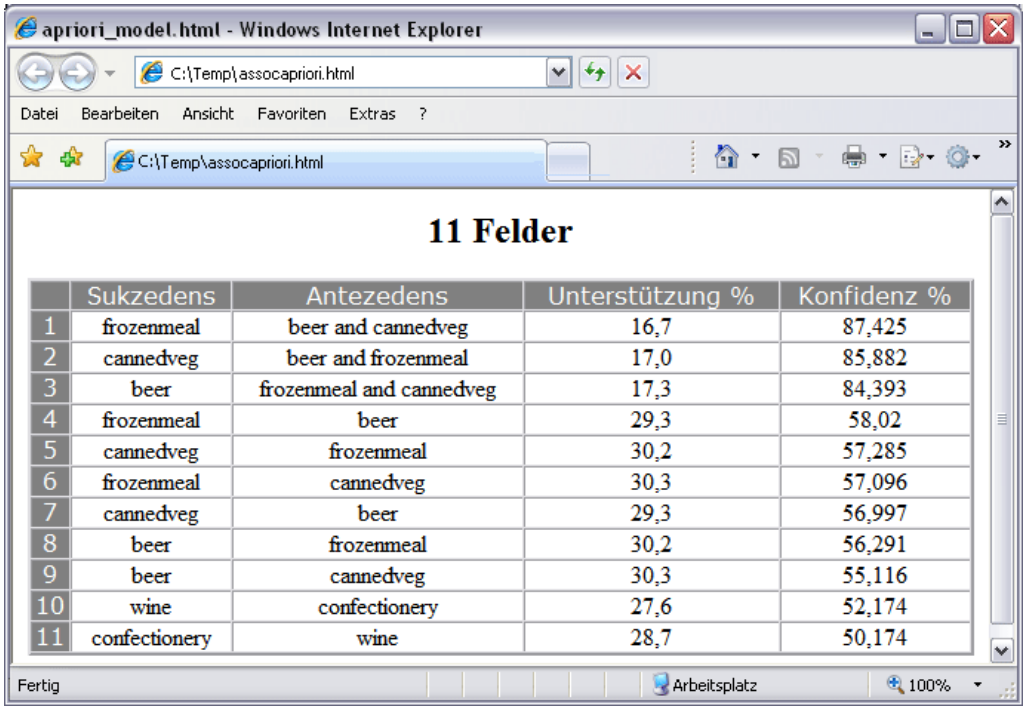
```
export model Drug as c:\mymodel.htm format html

export model Drug as c:\mymodel.txt format text
```

Hinweis: Diese Formate sind für Modelle, die keine Registerkarte “Modell” aufweisen, nicht verfügbar.

Modellzusammenfassung. So exportieren Sie die Modellübersicht (Registerkarte “Übersicht” beim Durchsuchen des Modell-Nuggets) im HTML- bzw. Textformat:

```
export model Drug summary in c:/mymodels format html

export model Drug summary as c:/mymodel.txt format text

export model 'assocapriori' as 'C:/temp/assoc_apriori' format html
```

Abbildung 4-3
Registerkarte “Assoziationsmodell” als HTML exportiert



11 Felder

	Sukzedens	Antezedens	Unterstützung %	Konfidenz %
1	frozenmeal	beer and cannedveg	16,7	87,425
2	cannedveg	beer and frozenmeal	17,0	85,882
3	beer	frozenmeal and cannedveg	17,3	84,393
4	frozenmeal	beer	29,3	58,02
5	cannedveg	frozenmeal	30,2	57,285
6	frozenmeal	cannedveg	30,3	57,096
7	cannedveg	beer	29,3	56,997
8	beer	frozenmeal	30,2	56,291
9	beer	cannedveg	30,3	55,116
10	wine	confectionery	27,6	52,174
11	confectionery	wine	28,7	50,174

delete model MODEL

```
insert model MODELL
insert model MODELL at X Y
```

```
insert model MODELL between KNOTEN1 and KNOTEN2
insert model MODELL connected between KNOTEN1 and KNOTEN2
```

Fügt das Modell zum aktuellen Stream hinzu. Optional können auch Positions- und Verbindungsoptionen angegeben werden.

```
insert model Kohonen between :typenode and :analysisnode
```

```
insert model Drug:neuralnetwork connected between 'Define Types' and 'Analysis'
```

load model DATEINAME

```
load model DATEINAME
```

Lädt ein gespeichertes Modell in die Modellpalette.

```
load model c:/mymodel.gm
```

retrieve model REPOSITORY_PFAD

```
retrieve model REPOSITORY_PFAD {label LABEL | version VERSION}
```

Ruft ein gespeichertes Modell aus IBM® SPSS® Collaboration and Deployment Services Repository ab. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
retrieve model "/my folder/Kohonen.gm"
```

save model MODELL as DATEINAME

```
save model MODELL as DATEINAME
```

Speichert das angegebene Modell als generierte Modelldatei

```
save model Drug as c:/mymodel.gm
```

store model MODELL as REPOSITORY_PFAD

```
store model MODELL as REPOSITORY_PFAD {label LABEL}
```

Speichert das angegebene Modell in IBM® SPSS® Collaboration and Deployment Services Repository. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
store model Kohonen as "/my folder/Kohonen.gm"
```

Die Erweiterung (*.*gm*) ist optional, muss jedoch beim Speichern und Abrufen des Modells konsistent verwendet werden. Wenn das Modell beispielsweise einfach als “Kohonen” gespeichert wurde, muss es mit demselben Namen abgerufen werden. (Anders ausgedrückt: Die Erweiterung (sofern verwendet) ist einfach ein Teil des Modellnamens.)

Stream-Objekte

Die folgenden Skriptbefehle stehen für Stream-Objekte zur Verfügung.

create stream STANDARD_DATEINAME

```
create stream STANDARD_DATEINAME
```

Nur Standalone-Skripts. Erstellt einen neuen Stream mit dem angegebenen Namen im Arbeitsspeicher. Der Stream wird nicht automatisch gespeichert.

```
create stream 'Druglearn'
```

close STREAM

```
close STREAM
```

Nur Standalone-Skripts. Schließt den angegebenen Stream.

Um den aktuellen Stream zu schließen, geben Sie den Befehl wie folgt vollständig in Kleinbuchstaben ein:

```
close stream
```

Standalone-Skripts

Wenn Sie mit mehreren Streams arbeiten, müssen Sie beachten, dass stream (in Kleinbuchstaben) eigentlich eine Sondervariable ist, die zur Referenzierung des aktuellen Streams verwendet wird. Um einen anderen Stream zu schließen, kann diese Variable auf einen anderen Wert gesetzt werden:

```
set stream = get stream 'Stream5'
close stream
```

Alternativ kann jede deklarierte Variable angegeben werden, die einen Stream referenziert. Beispiel:

```
var s
set s = get stream 'Stream2'
save s as c:/stream2.str
close s
```

Schließlich kann der aktuelle Stream mithilfe des Befehls with stream vorübergehend neu zugewiesen werden. Beispiel:

```
with stream 'Stream1'
 close stream
endwith
```

clear stream

```
clear stream
```

Entfernt alle Knoten aus dem aktuellen Stream.

get stream STREAM

```
get stream STREAM
```

Nur Standalone-Skripts. Zum Abrufen einer Referenz auf den angegebenen Stream verwendet, der einer lokalen Variablen (oder der Sondervariablen stream) zugewiesen werden kann. Der angegebene Stream muss bereits geöffnet sein.

```
var s
set s = get stream 'Druglearn'
close s
```

load stream DATEINAME

```
load stream DATEINAME
```

Nur Standalone-Skripts. Fügt den angegebenen Stream zum Zeichenbereich hinzu, ohne die Knoten aus dem aktuellen Stream zu löschen.

```
load stream "c:/demos/druglearn.str"
```

"Open stream" und "load stream" im Vergleich Der Befehl load stream fügt den angegebenen Stream zum Zeichenbereich hinzu, ohne die Knoten aus dem aktuellen Stream zu löschen. Dieser Befehl wurde in früheren IBM® SPSS® Modeler-Versionen häufiger verwendet und wurde in den neueren Versionen größtenteils durch die Möglichkeit ersetzt, Knoten zu öffnen, zu verwalten und zwischen verschiedenen Streams zu kopieren.

open stream FILENAME

```
open stream FILENAME
```

Nur Standalone-Skripts. Öffnet den angegebenen Stream.

```
open stream "c:/demos/druglearn.str"
```

retrieve stream REPOSITORY_PFAD

```
retrieve stream REPOSITORY_PATH {label LABEL | version VERSION}
retrieve stream URI [(#m.marker | #l.label)]
```

Ruft den angegebenen Stream aus IBM® SPSS® Collaboration and Deployment Services Repository ab. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
retrieve stream "/myfolder/druglearn.str"
```

```
retrieve stream "spsscr:///models/drug%20model.gm#m.0:2005-10-12%2014:15:41.281"
```

save STREAM as DATEINAME

```
save STREAM
save STREAM as FILENAME
```

Um Änderungen, am aktuellen Stream zu speichern (vorausgesetzt, er wurde zuvor gespeichert), geben Sie den Befehl wie folgt vollständig in Kleinbuchstaben ein:

```
save stream
```

So speichern Sie einen Stream erstmals unter einem neuen Dateinamen:

```
create stream nifty
create featureselectionnode
save stream as c:/nifty.str
```

Standalone-Skripts

Wenn Sie mit mehreren Streams in einem Standalone-Skript arbeiten, müssen Sie beachten, dass stream (in Kleinbuchstaben) eigentlich eine Sondervariable ist, die zur Referenzierung des aktuellen Streams verwendet wird. Um einen anderen Stream zu speichern, kann diese Variable auf einen anderen Wert gesetzt werden:

```
set stream = get stream 'Stream5'
save stream
```

Alternativ kann jede deklarierte Variable angegeben werden, die einen Stream referenziert. Beispiel:

```
var s
set s = get stream 'Stream2'
save s as c:/stream2.str
close s
```

Schließlich kann der aktuelle Stream mithilfe des Befehls with stream vorübergehend neu zugewiesen werden. Beispiel:

```
with stream 'Stream1'
 save stream
endwith
```

Für weitere Informationen siehe Thema Arbeiten mit mehreren Streams in Kapitel 3 auf S. 26.

store stream as REPOSITORY_PFAD

```
store stream as REPOSITORY_PFAD {label LABEL}
store stream as URI [#l.label]

store stream as "/folder_1/folder_2/mystream.str"
```

Speichert den aktuellen Stream in IBM® SPSS® Collaboration and Deployment Services Repository. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
store stream as "/folder_1/folder_2/druglearn.str"
store stream as "spsscr:///folder_1/folder_2/mystream.str"
```

Standalone-Skripts

Wenn Sie mit mehreren Streams in einem Standalone-Skript arbeiten, müssen Sie beachten, dass stream (in Kleinbuchstaben) eigentlich eine Sondervariable ist, die zur Referenzierung des aktuellen Streams verwendet wird. Um einen anderen Stream zu speichern, kann diese Variablen auf einen anderen Wert gesetzt werden:

```
set stream = get stream 'Stream5'
store stream as "/folder_1/mystream.str"
```

Alternativ kann jede deklarierte Variable angegeben werden, die einen Stream referenziert, oder der aktuelle Stream kann mithilfe des Befehls with stream vorübergehend neu zugewiesen werden:

```
with stream 'Stream6'
 store stream as "/folder_1/mystream.str"
endwith
```

with stream STREAM

```
with stream STREAM
 STATEMENTS
endwith
```

Nur Standalone-Skripts. Führt STATEMENTS aus, wobei der angegebene STREAM als aktueller Stream festgelegt ist. Der ursprüngliche aktuelle Stream wird wiederhergestellt, sobald die Anweisungen ausgeführt wurden.

```
with stream 'druglearn'
 create typenode
 execute_script
endwith
```

Projektobjekte

Die folgenden Skriptbefehle stehen für Projektobjekte zur Verfügung.

Die Erweiterung (**.cpj*) ist optional, muss jedoch beim Speichern und Abrufen des jeweiligen Projekts konsistent verwendet werden.

execute_project

```
execute_project
```

Generiert den Standardbericht des aktuellen Projekts.

load project DATEINAME

load project DATEINAME

Öffnet das angegebene Projekt.

```
load project "C:/clemdata/DrugData.cpj"
set ^project.summary="Erste Modellierungsarbeiten an den aktuellen Daten."
set ^project.ordering=NameAddedType
execute_project
```

retrieve project REPOSITORY_PFAD

retrieve project REPOSITORY_PFAD {label LABEL | version VERSION}

Ruft ein Projekt aus IBM® SPSS® Collaboration and Deployment Services Repository ab. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
retrieve project "/CRISPDM/DrugExample.cpj"
```

save project as DATEINAME

save project
save project as DATEINAME

Speichert das aktuelle Projekt.

store project as REPOSITORY_PFAD

store project as REPOSITORY_PFAD {label LABEL}

Speichert das aktuelle Projekt in IBM® SPSS® Collaboration and Deployment Services Repository. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
store project as "/CRISPDM/DrugExample.cpj"
```

Statusobjekte

Ein gespeicherter Status kann mithilfe des Befehls load state geladen werden.

load state DATEINAME

load state DATEINAME

Lädt den angegebenen Status.

```
load state "c:/data/myproject.cst"
```

Ergebnisobjekte

Ein Zugriff auf Ergebnisse ist über den Befehl value möglich.

value ERGEBNIS

value RESULT at ROW COLUMN

Endknoten enthalten einen schreibgeschützten Parameter namens output, der zum Zugriff auf das aktuellste generierte Objekt verwendet werden kann. Bei Knoten, die Tabellenausgaben in Zeilen und Spalten erstellen, wird es dadurch möglich, auf den Wert für eine angegebene Zelle zuzugreifen. Beispiel:

```
execute :tablenode
set last_row = :tablenode.output.row_count
set last_column = :tablenode.output.column_count
set last_value = value :tablenode.output at ^last_row ^last_column
var myresults
set myresults = open create 'C:/myresults.txt'
write myresults 'The value in the last cell is ' >< ^last_value
```

Bei Zeile und Spalte handelt es sich um ein Offset von 1. Wenn das Ausgabeobjekt nicht vorhanden ist, wird ein Fehler ausgegeben.

Eigenschaften des Objektergebnisses

Die folgenden Eigenschaften gelten für alle Ergebnisobjekte (z. B. Tabellen- und Matrix-Ergebnisse), die Daten in Zeilen und Spalten enthalten:

Eigenschaft	Beschreibung
row_count	Ergibt die Anzahl der Zeilen in den Daten.
column_count	Ergibt die Anzahl der Spalten in den Daten.

Dateiobjekte

Die folgenden Skriptbefehle stehen für Dateiobjekte zur Verfügung.

close DATEI

close DATEI

Mit der obigen Anweisung wird die angegebene Datei geschlossen.

open DATEI

```
open create DATEINAME
open append DATEINAME
```

Mit den obigen Anweisungen wird die angegebene Datei geöffnet.

- **create.** Erstellt die Datei, wenn sie noch nicht vorhanden ist, bzw. überschreibt sie, wenn sie vorhanden ist.
- **append.** Hängt die Datei an eine bestehende Datei an. Erstellt einen Fehler, wenn die Datei nicht vorhanden ist.

Dabei wird die Dateikennung für die geöffnete Datei ausgegeben.

```
var file
set file = open create 'C:/script.out'
for I from 1 to 3
 write file 'Stream ' >< I
endfor
close file
```

write DATEI

```
write DATEI TEXTAUSDRUCK
writeln DATEI TEXTAUSDRUCK
```

Durch die obigen Ausdrücke wird der Textausdruck in die Datei geschrieben. Die erste Anweisung schreibt den Text, wie er ist, während die zweite außerdem nach dem Ausdruck eine neue Zeile einfügt. Es wird eine Fehlermeldung ausgegeben, wenn es sich bei FILE nicht um ein geöffnetes Dateiobjekt handelt.

```
var file
set file = open create 'C:/hello.txt'
writeln file 'Hello'
writeln file 'World'
write file 'Would you like to play a game?'
close file
```

Ausgabeobjekte

Die folgenden Skriptbefehle stehen für Ausgabeobjekte zur Verfügung.

Namen der Ausgabetypen

In der folgenden Tabelle werden alle Ausgabeobjekttypen und die Knoten, die sie erstellen, 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 hier: Kapitel 15, *Diagrammknoten – Eigenschaften*, und Kapitel 19, *Ausgabeknoten – Eigenschaften*.

Ausgabeobjekttyp	Knoten
analysisoutput	Analyse
collectionoutput	Sammlung
dataauditoutput	Data Audit
distributionoutput	Verteilung
evaluationoutput	Evaluation

Ausgabeobjekttyp	Knoten
histogramoutput	Histogramm
matrixoutput	Matrix
meansoutput	Mittelwerte
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

delete output AUSGABE

```
delete output AUSGABE
```

Löscht die angegebene Ausgabe aus der Manager-Palette. Beispiel:

```
delete output :statisticsoutput
```

So löschen Sie alle Ausgabeelemente aus der Manager-Palette:

```
clear outputs
```

export output AUSGABE

```
export output AUSGABE as DATEI format FORMAT
```

Exportiert die Ausgabe im angegebenen Format. Beachten Sie, dass die verfügbaren Formate vom Ausgabetyp abhängen, aber dennoch denjenigen entsprechen sollten, die beim Durchsuchen der angegebenen Ausgabe im Exportmenü verfügbar sind.

```
export output :statisticsoutput as "C:/output/statistics.html" format html
export output :matrixoutput as "C:/output/matrix.csv" format delimited
export output :tableoutput as "C:/output/table.tab" format transposed formatted
```

get output AUSGABE

```
get output AUSGABE
```

Ruft eine Referenz zur angegebenen Ausgabe ab. Beispielsweise kann eine Schleife verwendet werden, um eine Reihe von Ausgabeobjekten abzurufen und nacheinander zu exportieren.

```
execute_all
for item in statisticsoutput matrixoutput tableoutput
 var theoutput
 set theoutput = get output :^item
 set filename = 'c:/'><^item ><'.htm'
 export output ^theoutput as ^filename format html
endfor
```

load output DATEINAME

load output DATEINAME

Lädt die angegebene Ausgabe.

```
load output 'c:/matrix.cou'
```

retrieve output REPOSITORY_PFAD

retrieve output REPOSITORY_PFAD {label LABEL | version VERSION}

Ruft die angegebene Ausgabe aus IBM® SPSS® Collaboration and Deployment Services Repository ab. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
retrieve output "/results/mytable"
```

save output AUSGABE as DATEINAME

save output as DATEINAME

Speichert die angegebene Ausgabe.

```
save output :matrixoutput as 'c:/matrix.cou'
```

store output AUSGABE as REPOSITORY_PFAD

store output AUSGABE as REPOSITORY_PFAD {label LABEL}

Speichert die angegebene Ausgabe in IBM® SPSS® Collaboration and Deployment Services Repository. Für weitere Informationen siehe Thema Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61.

```
store output "Datenaudit von [6 Felder]" as "/eigener Ordner/Mein Audit"
```

```
store output :tableoutput as "/results/mytable"
```

Kapitel 5

Tipps zur Skripterstellung

In diesem Abschnitt erhalten Sie einen Überblick über Tipps und Verfahren für die Verwendung von Skripts, wie beispielsweise die Änderung der Stream-Ausführung, die Verwendung von verschlüsselten Passwörtern in Skripts und den Zugriff auf Objekte in IBM® SPSS® Collaboration and Deployment Services Repository.

Ändern der Stream-Ausführung

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 "Stream-Eigenschaften" auf der Registerkarte "Skript" folgende Schritte aus:

- Starten Sie mit einem leeren Skript.
- Klicken Sie in der Symbolleiste auf die Schaltfläche Standardskript anhängen, um ein Standard-Stream-Skript hinzuzufügen.
- Bringen Sie die im Standard-Stream-Skript enthaltenen Anweisungen in die für die Ausführung gewünschte Reihenfolge.

Verwendung von Schleifen bei Knoten

Sie können eine for-Schleife in Verbindung mit der Eigenschaft ^stream.nodes verwenden, um alle Knoten in einem Stream in einer Schleife zu durchlaufen. Das folgende Skript beispielsweise durchläuft alle Knoten in einer Schleife und ändert dabei die Feldnamen in allen Filterknoten in Großbuchstaben.

Dieses Skript kann in jedem Stream verwendet werden, der einen Filterknoten enthält, selbst wenn tatsächlich gar keine Felder gefiltert werden. Fügen Sie einfach einen Filterknoten hinzu, der alle Felder weitergibt, um die Feldnamen durchgängig in Großbuchstaben zu ändern.

```
var my_node
var loop_me
var var_name

for my_node in ^stream.nodes
 if ^my_node.node_type = filternode then
  for loop_me in_fields_to ^my_node:filternode
   set var_name = lowertoupper(^my_node:filternode.new_name.^loop_me)
   set ^my_node:filternode.new_name.^loop_me = ^var_name
  endfor
 else
 endif
```

```
endfor
```

Das Skript durchläuft alle Knoten im aktuellen Stream, wie von der Eigenschaft ^stream.nodes ausgegeben, und prüft jeweils, ob es sich bei den einzelnen Knoten um einen Filter handelt. Wenn ja, durchläuft das Skript die einzelnen Felder im Knoten und verwendet die Funktion lowertoupper() , um den Namen in Großbuchstaben zu ändern.

Tipp: Wenn Sie die Feldnamen in Kleinbuchstaben ändern möchten, verwenden Sie stattdessen die Funktion uppertolower().

Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository

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

Wenn Sie IBM® SPSS® Collaboration and Deployment Services Repository lizenziert haben, können Sie mithilfe von Skriptbefehlen Objekte im Repository speichern, abrufen, sperren und entsperren. Mit dem Repository können Sie die Lebensdauer von Data-Mining-Modellen und verwandten Vorhersageobjekten im Zusammenhang mit Unternehmensanwendungen, Tools und Lösungen verwalten. Für weitere Informationen siehe Thema Informationen zu IBM SPSS Collaboration and Deployment Services Repository in Kapitel 9 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Verbindung mit dem IBM SPSS Collaboration and Deployment Services Repository

Um auf das Repository zugreifen zu können, müssen Sie zunächst eine gültige Verbindung einrichten, entweder über das Menü “Extras” der IBM® SPSS® Modeler-Benutzeroberfläche oder über die Befehlszeile. (Für weitere Informationen siehe Thema Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung in Kapitel 7 auf S. 75.)

Speichern und Abrufen von Objekten

Innerhalb eines Skripts können Sie mit den Befehlen retrieve und store auf verschiedene Objekte zugreifen, beispielsweise auf Streams, Modelle, Ausgaben, Knoten und Projekte. Die Syntax lautet wie folgt:

```
store object as REPOSITORY_PATH {label LABEL}
store object as URI [#l.label]

retrieve object REPOSITORY_PATH {label LABEL | version VERSION}
retrieve object URI [(#m.marker | #l.label)]
```

REPOSITORY_PATH gibt die Position des Objekts im Repository an. Der Pfad muss in Anführungszeichen eingeschlossen sein und es müssen normale Schrägstriche als Trennzeichen verwendet werden. Die Groß- und Kleinschreibung wird nicht berücksichtigt.

```
store stream as "/folder_1/folder_2/mystream.str"
store model Drug as "/myfolder/drugmodel"
```

```
store model Drug as "/myfolder/drugmodel.gm" label "final"
store node DRUG1n as "/samples/drug1ntypenode"
store project as "/CRISPDM/DrugExample.cpj"
store output "Data Audit of [6 fields]" as "/my folder/My Audit"
```

Optional können Erweiterungen wie *.str* oder *.gm* in den Objektnamen aufgenommen werden. Dies ist jedoch nicht erforderlich, solange der Name konsistent ist. Wenn beispielsweise ein Modell ohne Erweiterung gespeichert wird, muss es mit demselben Namen wieder abgerufen werden:

```
store model "/myfolder/drugmodel"
retrieve model "/myfolder/drugmodel"
```

gegenüber:

```
store model "/myfolder/drugmodel.gm"
retrieve model "/myfolder/drugmodel.gm" version "0:2005-10-12 14:15:41.281"
```

Beachten Sie: Beim Abrufen von Objekten wird immer die aktuellste Version des Objekts ausgegeben, es sei denn, Sie geben eine Version oder eine Beschriftung an. Beim Abrufen eines Knotenobjekts wird der Knoten automatisch in den aktuellen Stream eingefügt. Beim Abrufen eines Stream-Objekts müssen Sie ein Standalone-Skript verwenden. Stream-Objekte können nicht aus Stream-Skripts abgerufen werden.

Sperren und Entsperren von Objekten

Mit einem Skript können Sie ein Objekt sperren, um zu verhindern, dass andere Benutzer seine bestehenden Versionen aktualisieren oder neue Versionen erstellen. Außerdem können Sie ein Objekt entsperren, das Sie gesperrt haben.

Die Syntax zum Sperren und Entsperren eines Objekts:

```
lock REPOSITORY_PATH
lock URI

unlock REPOSITORY_PATH
unlock URI
```

Wie beim Speichern und Abrufen von Objekten gibt REPOSITORY_PATH die Position des Objekts im Repository an. Der Pfad muss in Anführungszeichen eingeschlossen sein und es müssen normale Schrägstriche als Trennzeichen verwendet werden. Die Groß- und Kleinschreibung wird nicht berücksichtigt.

```
lock "/myfolder/Stream1.str"

unlock "/myfolder/Stream1.str"
```

Alternativ können Sie statt eines Repository-Pfads einen URI (Uniform Resource Identifier) verwenden, um die Position des Objekts anzugeben. Der URI muss das Präfix spsscr: enthalten und muss vollständig in Anführungszeichen eingeschlossen sein. Nur normale Schrägstriche sind als Pfadtrennzeichen zulässig und Leerzeichen müssen kodiert werden. Statt eines Leerzeichens

muss im Pfad %20 verwendet werden. Die Groß- und Kleinschreibung wird beim URI nicht berücksichtigt. Beispiele:

```
lock "spsscr:///myfolder/Stream1.str"

unlock "spsscr:///myfolder/Stream1.str"
```

Beachten Sie, dass das Sperren von Objekten für alle Versionen eines Objekts gilt – Sie können keine einzelnen Versionen sperren oder entsperren.

Erstellen eines verschlüsselten Passworts

In bestimmten Fällen müssen Sie möglicherweise ein Passwort in ein Skript aufnehmen, beispielsweise um auf eine passwortgeschützte Datenquelle zuzugreifen. Verschlüsselte Passwörter können in folgenden Elementen verwendet werden:

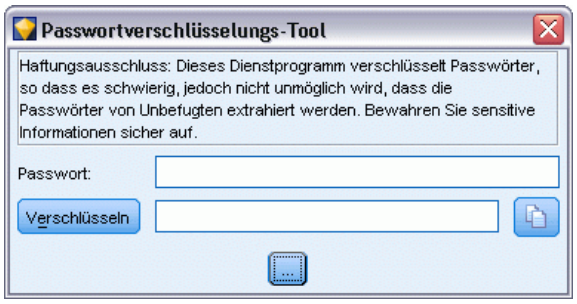
- Knoteneigenschaften für Datenbankquellenknoten und Ausgabeknoten
- Befehlszeilenargumente für die Anmeldung beim Server
- Die Datenbankverbindungseigenschaften, die in einer *.par*-Datei (die über die Registerkarte "Veröffentlichen" eines Exportknotens generierte Parameterdatei) gespeichert sind.

Über die Benutzeroberfläche steht ein Tool zur Verfügung, mit dem Sie verschlüsselte Passwörter 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 Passwort in Skriptdateien und Befehlszeilenargumente kopieren und dort speichern. In der Knoteneigenschaft `epassword`, die für `databasenode` und `databaseexportnode` verwendet wird, wird das verschlüsselte Passwort gespeichert.

▶ Um ein verschlüsseltes Passwort zu erstellen, wählen Sie im Menü "Extras" folgende Befehlsfolge aus:

Passwort verschlüsseln...

Abbildung 5-1
Passwortverschlüsselungs-Tool



▶ Geben Sie ein Passwort im Textfeld "Passwort" ein.

▶ Klicken Sie auf Verschlüsseln, um eine Zufallsverschlüsselung des Passworts zu generieren.

▶ Klicken Sie auf die Schaltfläche "Kopieren", um das verschlüsselte Passwort in die Zwischenablage zu kopieren.

▶ Fügen Sie das Passwort in das gewünschte Skript bzw. den gewünschten Parameter ein.

Skriptprüfung

Die Syntax aller Skripttypen können Sie sehr schnell prüfen, indem Sie in der Symbolleiste des Dialogfelds “Standalone-Skript” auf die rote Prüfschaltfläche klicken.

Abbildung 5-2
Symbolleistenschaltflächen für Stream-Skripts

Die Skriptprüfung 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.

Skripts in der Befehlszeile

Mit Skripts können Sie Vorgänge ausführen, die normalerweise über die Benutzeroberfläche 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.txt -execute
```

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

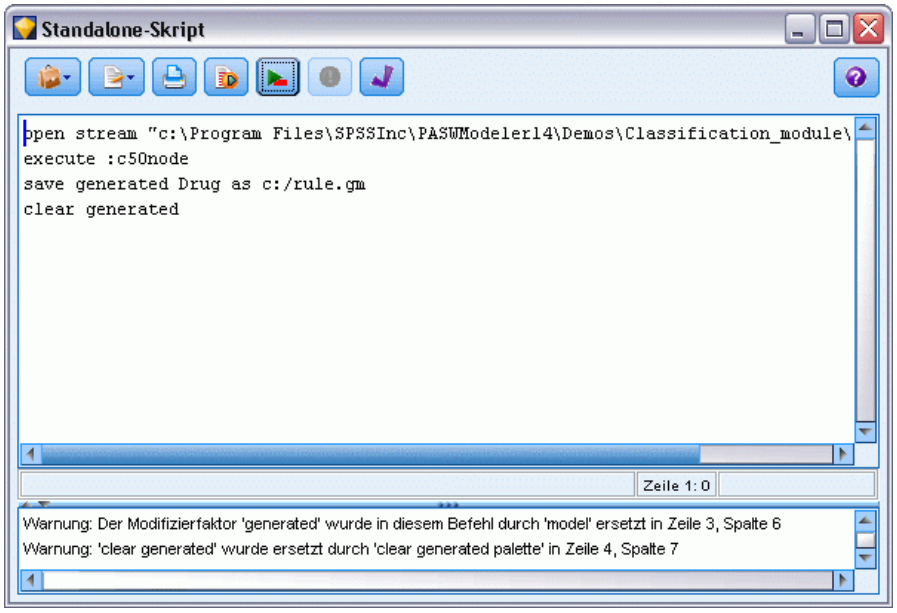
Kompatibilität zu früheren Versionen

In früheren IBM® SPSS® Modeler-Versionen erstellte Skripts laufen in der aktuellen Version normalerweise unverändert. Allerdings können nun automatisch Modell-Nuggets 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 Skript 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 Skripts funktionieren eventuell nicht in früheren Versionen.

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

Abbildung 5-3
Ausführung eines Skripts, das einen verworfenen Befehl verwendet.



Kapitel 6

Skriptbeispiele

In diesem Abschnitt finden Sie eine Reihe von Beispielen, die zeigen, wie Skripts in IBM® SPSS® Modeler verwendet werden können.

Typknotenbericht

Dieses Skript erstellt einen HTML-Bericht mit Informationen zu den Feldern im aktuellen Stream. Das Skript kann mit jedem Stream verwendet werden, der einen instanziierten Typknoten enthält, und es kann ggf. problemlos erweitert werden, um weitere Eigenschaften bzw. Knoten abzudecken.

- Standard-HTML-Tags werden verwendet, um die Ergebnisse zur Anzeige in einem Standardbrowser zu formatieren.
- Ein IBM® SPSS® Modeler-Typknoten dient zum Zugriff auf die Eigenschaften für die einzelnen Felder. Das Skript kann ggf. problemlos erweitert werden, um zusätzliche Eigenschaften aufzulisten, die durch den Typknoten aufgedeckt werden, wie beispielsweise fehlende Werte oder die Feldrolle. Für weitere Informationen siehe Thema Eigenschaften von “typenode” in Kapitel 14 auf S. 181.
- Mithilfe von SPSS Modeler-Skriptbefehlen werden die Ausgaben in eine Datei geschrieben und es wird ein Durchlauf durch die Felder durchgeführt, um auf die Eigenschaften der einzelnen Felder zugreifen zu können. Für weitere Informationen siehe Thema Skriptbefehle in Kapitel 4 auf S. 33.

Abbildung 6-1
Beispielskript für Typknotenbericht

```
# This script creates an HTML file and adds data from the Type node.
var myreport
set myreport = open create "C:/typenodereport.html"

# set up the HTML page
writeln myreport "<html>"
writeln myreport "<header>Typknoteninformationen von IBM SPSS Modeler</header>"
writeln myreport "<body><br/><br/>"

#create the table and write out the headers
writeln myreport "<table border=\"1\">"
writeln myreport "<tr bgcolor=\"C0C0C0\">"
writeln myreport "<td>Feld</td><td>Typ</td><td>Werte</td>"
writeln myreport "</tr>"

# loop through fields and add a row for each
var current_field
for current_field in_fields_at Type
  writeln myreport "<tr>"
    write myreport "<td>" >< ^current_field >< "</td>"
```

```
    write myreport "<td>" >< Type:typenode.type.^current_field >< "</td>"

    # add values for numeric fields
    if Type:typenode.type.^current_field = Range then
      writeln myreport "<td>" >< Type:typenode.values.^current_field >< "</td>"
    endif

    # add values for flag fields
    if Type:typenode.type.^current_field = Flag then
       writeln myreport "<td>" >< Type:typenode.values.^current_field >< "</td>"
    endif

    # add values for nominal fields
    if Type:typenode.type.^current_field = Set then
     writeln myreport "<td>"
     var current_value
     for current_value in Type:typenode.values.^current_field
       writeln myreport  ^current_value >< "<BR/>"
     endfor
     writeln myreport "</td>"
    endif

  writeln myreport "</tr>"
endfor
writeln myreport "</table>"
writeln myreport "</body>"
writeln myreport "</html>"
close myreport
```

Erstellen der Ausgabedatei

Das Skript erstellt zunächst eine neue HTML-Datei und fügt die Tags hinzu, die erforderlich sind, um eine Tabelle mit einer Kopfzeile zu erstellen, in der die Spaltentitel *Field* (Feld), *Type* (Typ) und *Values* (Werte) aufgeführt sind. (Jedes Tag-Paar vom Typ <td></td> erstellt eine Zelle innerhalb einer Tabellenzeile.) Diese Spalten werden für die einzelnen Felder anhand von Eigenschaften aus dem Typknoten ausgefüllt.

```
# This script creates an HTML file and adds data from the Type node.
var myreport
set myreport = open create "C:/typenodereport.html"

# set up the HTML page
writeln myreport "<html>"
writeln myreport "<header>Typknoteninformationen von IBM SPSS Modeler</header>"
writeln myreport "<body><br/><br/>"

#create the table and write out the headers
writeln myreport "<table border=\"1\">"
writeln myreport "<tr bgcolor=\"C0C0C0\">"
writeln myreport "<td>Feld</td><td>Typ</td><td>Werte</td>"
writeln myreport "</tr>"
```

Verwendung von Schleifen bei Feldern

Als Nächstes durchläuft das Skript in einer Schleife alle Felder im Typknoten und fügt eine Zeile für jedes Feld hinzu, in der Feldname und -typ aufgelistet werden.

```
# loop through fields and add a row for each
var current_field
for current_field in_fields_at Type
  writeln myreport "<tr>"
    write myreport "<td>" >< ^current_field >< "</td>"
    write myreport "<td>" >< Type:typenode.type.^current_field >< "</td>"
```

Werte für stetige und Flag-Felder

Bei stetigen Feldern (numerischer Bereich) ergibt die Eigenschaft typenode.values den niedrigsten und den höchsten Wert im Format [0.500517, 0.899774] (in der Tabelle angezeigt). Bei Flag-Feldern werden die Wahr-/Falsch-Werte in einem ähnlichen Format angezeigt.

```
# add values for numeric fields
    if Type:typenode.type.^current_field = Range then
      writeln myreport "<td>" >< Type:typenode.values.^current_field >< "</td>"
    endif

    # add values for flag fields
    if Type:typenode.type.^current_field = Flag then
       writeln myreport "<td>" >< Type:typenode.values.^current_field >< "</td>"
    endif
```

Werte für nominale Felder

Bei nominalen Feldern ergibt die Eigenschaft typenode.values die vollständige Liste der definierten Werte. Das Skript durchläuft diese Liste in einer Schleife für die einzelnen Felder, um jeden Wert nacheinander einzufügen. Dabei wird zwischen jeden Wert ein Zeilenumbruch (Tag
) eingefügt.

```
# add values for nominal fields
    if Type:typenode.type.^current_field = Set then
      writeln myreport "<td>"
      var current_value
      for current_value in Type:typenode.values.^current_field
        writeln myreport  ^current_value >< "<BR/>"
      endfor
      writeln myreport "</td>"
    endif
```

Schließen der Datei

Letztlich schließt das Skript die Zeile, schließt die Tags <table>, <body> und <html> und dann die Ausgabedatei.

```
writeln myreport "</tr>"
endfor
writeln myreport "</table>"
writeln myreport "</body>"
writeln myreport "</html>"
close myreport
```

Stream-Bericht

Dieses Skript erstellt einen HTML-Bericht mit Name, Typ und Anmerkung für jeden Knoten im aktuellen Stream. Zusätzlich zu den Grundlagen der Erstellung einer HTML-Datei und dem Zugriff auf Knoten- und Stream-Eigenschaften wird gezeigt, wie eine Schleife erstellt werden kann, die eine bestimmte Menge an Anweisungen für die einzelnen Knoten innerhalb eines Streams ausführt. Die Verwendung ist mit jedem Stream möglich.

Abbildung 6-2
Beispiel-Skript für Stream-Bericht

```
# Create the HTML page with heading
var myfile
set myfile = open create "c:\stream_report.html"
writeln myfile "<HTML>"
writeln myfile " <BODY>"
writeln myfile " <HEAD>Report for stream " >< ^stream.name >< ".str</HEAD>"
writeln myfile " <p>" >< ^stream.annotation >< "</p>"

#Create the table with header row
writeln myfile "<TABLE border=\"1\" width=\"90%\">"
writeln myfile " <tr bgcolor=\"lightgrey\" colspan=\"3\">"
writeln myfile "   <th>Node Name</th>"
writeln myfile "   <th>Type</th>"
writeln myfile "   <th>Annotation</th>"
writeln myfile " </tr>"

# Loop through nodes and add name, type, and annotation for each
# The ^stream.nodes property returns the list of nodes
var current_node
for current_node in ^stream.nodes
 writeln myfile "<tr>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.name
 writeln myfile "  </td>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.node_type
 writeln myfile "  </td>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.annotation >< " "
 writeln myfile "  </td>"
 writeln myfile "</tr>"
endfor
```

```
writeln myfile "</TABLE>"
writeln myfile "</BODY>"
writeln myfile "</HTML>"
close myfile
```

Erstellen des Berichts

Das Skript erstellt zunächst eine neue HTML-Datei mit den Elementen <BODY> und <HEAD>. Die Eigenschaft ^stream.name ergibt den Namen des aktuellen Streams, der in die Überschrift eingefügt wird. Der Operator >< dient dazu, Zeichenfolgen miteinander zu verketten.

```
# Create the HTML page with heading
var myfile
set myfile = open create "c:\stream_report.html"
writeln myfile "<HTML>"
writeln myfile " <BODY>"
writeln myfile " <HEAD>Report for stream " >< ^stream.name >< ".str</HEAD>"
writeln myfile " <p>" >< ^stream.annotation >< "</p>"
```

Als Nächstes erstellt das Skript eine HTML-Tabelle mit einer Überschriftstzeile, die die Spaltentitel column titles *Node Name* (Knotenname), *Type* (Typ) und *Annotation* (Anmerkung) auflistet. (Jedes Tag-Paar vom Typ <td></td> erstellt eine Zelle innerhalb einer Tabellenzeile.)

```
#Create the table with header row
writeln myfile "<TABLE border=\"1\" width=\"90%\">"
writeln myfile " <tr bgcolor=\"lightgrey\" colspan=\"3\">"
writeln myfile "   <th>Node Name</th>"
writeln myfile "   <th>Type</th>"
writeln myfile "   <th>Annotation</th>"
writeln myfile " </tr>"
```

Anschließend durchläuft das Skript alle Knoten im aktuellen Stream. Für jeden Knoten wird eine Zeile zur Tabelle hinzugefügt, in der Name, Typ und Anmerkung aufgeführt werden. Ein unsichtbares umbruchgeschütztes Leerzeichen ( ) wird nach der Anmerkung eingefügt, um zu vermeiden, dass eine leere Zelle erstellt wird, wenn für einen bestimmten Knoten keine Anmerkung angegeben wurde. (Leere Zellen können bei der Anzeige der Tabelle zu Formatierungsproblemen führen.)

```
# Loop through nodes and add name, type, and annotation for each
# The ^stream.nodes property returns the list of nodes
var current_node
for current_node in ^stream.nodes
 writeln myfile "<tr>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.name
 writeln myfile "  </td>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.node_type
 writeln myfile "  </td>"
 writeln myfile "  <td>"
 writeln myfile     ^current_node.annotation >< " "
```

```
 writeln myfile "  </td>"
 writeln myfile "</tr>"
endfor
```

Schließlich fügt das Skript die zum Schließen des Dokuments erforderlichen HTML-Tags hinzu und schließt die Datei.

```
writeln myfile "</TABLE>"
writeln myfile "</BODY>"
writeln myfile "</HTML>"
close myfile
```

Kapitel
7

Befehlszeilenargumente

Aufrufen der Software

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

- Öffnen Sie auf einem Computer, auf dem IBM® SPSS® Modeler installiert ist, ein DOS- oder Befehlszeilenfenster.
- Um die SPSS Modeler-Schnittstelle im interaktiven Modus zu starten, geben Sie den Befehl modelerclient und dann die gewünschten Argumente ein, z. B.:

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

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

Verwenden von Befehlszeilenargumenten

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

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 Client-Installation sind die Argumente für die Serververbindung nicht erforderlich.

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

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

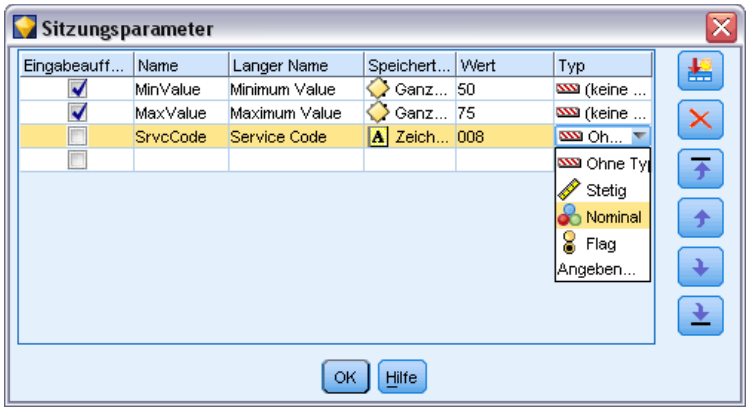
Sie können auch SPSS Modeler-Statusmodi und -Skripts auf diese Weise ausführen, nämlich mit den Flags -state bzw. -script.

Fehlersuche bei Befehlszeilenargumenten

Um die Fehlersuche in einer Befehlszeile durchzuführen, starten Sie SPSS Modeler mithilfe des Befehls modelerclient mit den gewünschten Argumenten. Dadurch können Sie gewährleisten, dass 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.

Abbildung 7-1
Definieren von Parametern für die Sitzung



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 <commandFileName> referenzierten Datei angegeben sind.

```
modelerclient @<commandFileName>
```

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

```
modelerclient @ "C:\Programme\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 @CommandFile-Argument in eine Befehlsdatei ein.

Argumente zum Herstellen einer Server-Verbindung

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 wird festgelegt, auf welche Weise 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. Für weitere Informationen siehe Thema Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung auf S. 75.

Argument	Verhalten/Beschreibung
-server	Führt SPSS Modeler im Servermodus aus. 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 <number>	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 Cluster-Namen 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. Für weitere Informationen siehe Thema Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung auf S. 75.
-username <name>	Benutzername, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar.
-password <password>	Passwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. *Hinweis*: Falls das Argument -password nicht vorliegt, werden Sie aufgefordert, ein Passwort einzugeben.

Argument	Verhalten/Beschreibung
-epassword <encodedpasswordstring>	Kodiertes Passwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. *Hinweis*: Kodierte Passwörter können in 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 (Slot-Parameter) 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 im Dokument *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 werden die Argumente aufgeführt, die zum Einrichten der Verbindung verwendet werden können:

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 <number>	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 <password>	Passwort, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
-spsscr_epassword <kodiertes Passwort>	Kodiertes Passwort, 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 – Sie müssen es nur verwenden, wenn Sie sich mithilfe von LDAP oder Active Directory anmelden.

Systemargumente

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

Argument	**Verhalten/Beschreibung**
@ <commandFile>	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. Für weitere Informationen siehe Thema Kombinieren mehrerer Argumente auf S. 73.
-directory <dir>	Bestimmt das Standard-Arbeitsverzeichnis. Im lokalen Modus wird dieses Verzeichnis sowohl für Daten als auch für die Ausgabe herangezogen.
-server_directory <dir>	Bestimmt das Server-Standardverzeichnis für Daten. Das Arbeitsverzeichnis, das mithilfe des Flag -directory angegeben wird, wird für die Ausgabe genutzt.
-execute	Nach dem Starten: Alle Streams, Statusangaben oder Skripts ausführen, die beim Starten geladen waren. Wird ein Skript zusätzlich zu einem Stream oder einem Status geladen, wird nur das Skript 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-Skript laden. Sie können dieses Skript zusätzlich zu einem Stream oder einem Status angeben (siehe unten); beim Starten kann jedoch nur ein einziges Skript geladen werden.
-model <Modell>	Beim Starten: Angegebenes generiertes Modell (Datei im Format *.gm*) laden.
-state <Status>	Beim Starten: Angegebenen gespeicherten Status laden.
-project <project>	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 (Slot-Parameter) herangezogen werden.

Hinweis: Die Standardverzeichnisse können auch in der Benutzeroberfläche 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 dem IBM SPSS Collaboration and Deployment Services Repository

Da Sie bestimmte Objekte aus einer Datei oder aus dem IBM® SPSS® Collaboration and Deployment Services Repository (sofern lizenziert) laden können, gibt das Dateinamenspräfix spsscr: und optional file: (für Objekte auf Datenträgern) an, wo IBM® SPSS® Modeler nach dem Objekt suchen 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 Präfix spsscr: ist es erforderlich, dass im selben 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
```

Weitere Details zu URIs für Objekte im IBM SPSS Collaboration and Deployment Services Repository finden Sie unter Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository in Kapitel 5 auf S. 61. Beachten Sie: In der Befehlszeile *müssen* Sie einen URI verwenden. Das einfachere REPOSITORY_PATH wird nicht unterstützt. (Es funktioniert nur innerhalb von Skripts.)

Parameter-Argumente

Bei der Ausführung von IBM® SPSS® Modeler über die Befehlszeile können Parameter als Flags herangezogen werden. Die Parameter werden in den Befehlszeilen mit dem Flag -P gekennzeichnet: -P *<Name>=<Wert>*.

Die folgenden Parameter stehen zur Auswahl:

- **Einfache Parameter** (oder Parameter, die direkt in CLEM-Ausdrücken verwendet werden).
- **Slot-Parameter** (auch als **Knoteneigenschaften** bezeichnet). Mit diesen Parametern werden die Einstellungen für die Knoten im Stream bearbeitet. Für weitere Informationen siehe Thema Überblick über Knoteneigenschaften in Kapitel 9 auf S. 122.
- **Befehlszeilenparameter** dienen zum Ändern der Vorgehensweise beim Aufrufen von SPSS Modeler.

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

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

Das Format stimmt mit dem datasource-Parameter der Knoteneigenschaft databasenode überein. Für weitere Informationen siehe Thema Eigenschaften von “databasenode” in Kapitel 12 auf S. 132.

Kapitel
8

CLEM-Sprachreferenz

CLEM-Referenz – Überblick

In diesem Abschnitt wird CLEM (Control Language for Expression Manipulation) beschrieben, ein leistungsstarkes Tool zur Analyse und Bearbeitung der in IBM® SPSS® Modeler-Streams verwendeten Daten. CLEM kann in Knoten zur Ausführung verschiedener Aufgaben eingesetzt werden, vom Auswerten von Bedingungen über das Ableiten von Werten bis hin zum Einfügen von Daten in Berichte. Für weitere Informationen siehe Thema Informationen zu CLEM in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Eine Untergruppe der CLEM-Sprache kann außerdem in Skripts in der Benutzeroberfläche verwendet werden. Dadurch können Sie viele gleichartige Datenbearbeitungen automatisieren. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

CLEM-Ausdrücke bestehen aus Werten, Feldnamen, Operatoren und Funktionen. Mit der richtigen Syntax können Sie eine Vielzahl leistungsstarker Datenoperationen erstellen. Für weitere Informationen siehe Thema CLEM-Beispiele in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

CLEM-Datentypen

CLEM-Datentypen können aus folgenden Elementen bestehen:

- Ganze Zahlen
- Reelle Zahlen
- Zeichen
- Zeichenketten
- Listen
- Felder
- Datum/Uhrzeit

Regeln für die Verwendung von Anführungszeichen

IBM® SPSS® Modeler ist zwar flexibel bei der Bestimmung der in einem CLEM-Ausdruck verwendeten Felder, Werte, Parameter und Zeichenketten. Dennoch ist es ratsam, sich beim Erstellen von Ausdrücken an die unten angegebenen allgemeinen Regeln zu halten.

- Strings – Setzen Sie Zeichenketten immer in doppelte Anführungszeichen, wie beispielsweise "Type 2". Bei einfachen Anführungszeichen besteht die Gefahr, dass sie mit in Anführungszeichen gesetzten Feldern verwechselt werden.

- Fields – Verwenden Sie einzelne Anführungszeichen, nur wenn es erforderlich ist, um Leerzeichen oder andere Sonderzeichen einzuschließen. Beispiel: 'Order Number'. Felder, die in Anführungszeichen gesetzt sind, im Daten-Set jedoch nicht definiert sind, werden irrtümlich als Zeichenketten interpretiert.
- Parameter – Setzen Sie Parameter immer in einfache Anführungszeichen, wie beispielsweise '$P-threshold'.
- Zeichen – Zeichen müssen stets in einfache umgekehrte Anführungszeichen (`) gesetzt werden. Beispiel: stripchar(`d`, "drugA").

Für weitere Informationen siehe Thema Werte und Datentypen in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*. Diese Regeln werden außerdem in folgenden Themenabschnitten eingehend behandelt.

Ganze Zahlen

Ganze Zahlen werden als Folge von Dezimalziffern dargestellt. Optional können Sie der ganzen Zahl ein Minuszeichen (−) voranstellen, um eine negative Zahl anzugeben. Beispiele: 1234, 999, −77.

Die CLEM-Sprache kann mit ganzen Zahlen beliebiger Genauigkeit umgehen. Die maximale Größe der ganzen Zahlen hängt von Ihrer Plattform ab. Wenn die Werte zu groß für die Anzeige in einem Feld ganzer Zahlen sind, kann der Wert normalerweise durch Ändern des Feldtyps in Real wiederhergestellt werden.

Reelle Zahlen

Reelle Zahlen sind Gleitkommazahlen. Reelle Zahlen bestehen aus einer oder mehreren Ziffern, gefolgt von einem Punkt als Dezimaltrennzeichen, gefolgt von mindestens einer weiteren Ziffer. Bei CLEM gilt für die reellen Zahlen doppelte Genauigkeit.

Optional können Sie der reellen Zahl ein Minuszeichen (−) voranstellen, um eine negative Zahl anzugeben. Beispiele: 1.234, 0.999, −77.001. Mit dem Format <*Zahl*> e <*Exponent*> können Sie eine reelle Zahl in Exponentialnotation angeben. Beispiel: 1234.0e5, 1.7e−2. Wenn die IBM® SPSS® Modeler-Anwendung numerische Zeichenketten aus Dateien einliest und sie automatisch in Zahlen umwandelt, werden Zahlen ohne führende Ziffer vor dem Dezimalpunkt oder ohne Ziffer nach dem Punkt akzeptiert. Beispiel: 999. oder .11. Bei CLEM-Ausdrücken sind diese Formate jedoch nicht zulässig.

Hinweis: Bei der Verwendung reeller Zahlen in CLEM-Ausdrücken muss ein Punkt als Dezimaltrennzeichen verwendet werden, unabhängig von den verwendeten Ländereinstellungen und den Einstellungen für den aktuellen Stream. So müssen Sie beispielsweise angeben:

```
Na > 0.6
```

Und nicht:

```
Na > 0,6
```

Dies gilt auch dann, wenn im Dialogfeld für die Stream-Eigenschaften das Komma als Dezimaltrennzeichen ausgewählt wurde. Diese Vorgehensweise entspricht der allgemeinen Richtlinie, dass Code-Syntax unabhängig von bestimmten Ländereinstellungen bzw. Konventionen sein sollte.

Zeichen

Zeichen (in der Regel als CHAR angezeigt) werden in CLEM-Ausdrücken normalerweise zur Durchführung von Tests an Zeichenketten verwendet. Mit der Funktion isuppercode können Sie beispielsweise ermitteln, ob das erste Zeichen einer Zeichenkette ein Großbuchstabe ist. Im folgenden CLEM-Ausdruck wird ein Zeichen verwendet, um anzugeben, dass der Test für das erste Zeichen der Zeichenkette durchgeführt werden soll.

```
isuppercode(subscrs(1, "MyString"))
```

Um den Code (im Gegensatz zur Position) eines bestimmten Zeichens in einem CLEM-Ausdruck anzugeben, verwenden Sie einzelne umgekehrte Anführungszeichen in der Form `<*Zeichen*>`. Beispiel: `A`, `Z`.

Hinweis: Bei Feldern gibt es keinen Speichertyp CHAR. Wenn also ein Feld abgeleitet oder mit einem Ausdruck ausgefüllt wird, der zu CHAR führt, wird das Ergebnis in eine Zeichenkette umgewandelt.

Zeichenketten

Im Allgemeinen sollten Zeichenketten in doppelte Anführungszeichen gesetzt werden. Beispiele: "c35product2" und "referrerID". Zur Kennzeichnung von Sonderzeichen in Zeichenketten werden umgekehrte Schrägstriche verwendet, z. B. "\$65443". (Zur Kennzeichnung eines umgekehrten Schrägstrichs wird ein doppelter umgekehrter Schrägstrich verwendet: \\.) Zeichenketten können auch in einfache Anführungszeichen gesetzt werden. Das Ergebnis lässt sich jedoch nicht von einem in Anführungszeichen gesetzten Feld unterscheiden ('referrerID'). Für weitere Informationen siehe Thema String-Funktionen in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Listen

Eine Liste ist eine geordnete Abfolge von Elementen, die verschiedenen Typen angehören können. Listen werden in eckige Klammern ([]) eingeschlossen. Beispiele: [1 2 4 16] und ["abc" "def"]. Listen werden nicht als Wert von IBM® SPSS® Modeler-Feldern verwendet. Sie werden verwendet, um Argumente für Funktionen, beispielsweise member und oneof, bereitzustellen.

Felder

Namen in CLEM-Ausdrücken, bei denen es sich nicht um Namen von Funktionen handelt, werden als Feldnamen betrachtet. Diese können einfach als Power, val27, state_flag usw. geschrieben werden, wenn der Name jedoch mit einer Ziffer beginnt oder nicht alphabetische Zeichen wie Leerzeichen enthält (mit Ausnahme des Unterstrichs), setzen Sie den Namen in einfache Anführungszeichen. Beispiel: 'Power Increase', '2nd answer', '#101', '$P-NextField'.

Hinweis: Felder, die in Anführungszeichen gesetzt sind, im Daten-Set jedoch nicht definiert sind, werden irrtümlich als Zeichenketten interpretiert.

Datumsangaben

Die Datumsberechnungen beruhen auf einem "Basisdatum", das im Dialogfeld "Stream-Eigenschaften" angegeben wird. Standardmäßig wird als Basisdatum der 1. Januar 1900 verwendet. Für weitere Informationen siehe Thema Festlegen von allgemeinen Optionen für Streams in Kapitel 5 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Die CLEM-Sprache unterstützt folgende Datumsformate.

Format	Beispiele
DDMMYY	150163
MMDDYY	011563
YYMMDD	630115
YYYYMMDD	19630115
YYYYDDD	Vierstellige Jahreszahl, gefolgt von einer dreistelligen Zahl für den Tag im Jahr – so steht 2000032 für den 32. Tag des Jahres 2000, also für den 1. Februar 2000.
DAY	Tag der Woche in der aktuellen Ländereinstellung – z. B. Monday, Tuesday usw. für Deutsch.
MONTH	Monat in der aktuellen Ländereinstellung – z. B. January, February, ….
DD/MM/YY	15/01/63
DD/MM/YYYY	15/01/1963
MM/DD/YY	01/15/63
MM/DD/YYYY	01/15/1963
DD-MM-YY	15-01-63
DD-MM-YYYY	15-01-1963
MM-DD-YY	01-15-63
MM-DD-YYYY	01-15-1963
DD.MM.YY	15.01.63
DD.MM.YYYY	15.01.1963
MM.DD.YY	01.15.63
MM.DD.YYYY	01.15.1963
DD-MON-YY	15-JAN-63, 15-jan-63, 15-Jan-63
DD/MON/YY	15/JAN/63, 15/jan/63, 15/Jan/63
DD.MON.YY	15.JAN.63, 15.jan.63, 15.Jan.63
DD-MON-YYYY	15-JAN-1963, 15-jan-1963, 15-Jan-1963
DD/MON/YYYY	15/JAN/1963, 15/jan/1963, 15/Jan/1963
DD.MON.YYYY	15.JAN.1963, 15.jan.1963, 15.Jan.1963
MON YYYY	Jan 2004

Format	Beispiele
q Q YYYY	Datum als Ziffer (1–4) für das Quartal, gefolgt vom Buchstaben *Q* und einer vierstelligen Jahresangabe. So wird beispielsweise das Datum 25. Dezember 2004 als 4 Q 2004 dargestellt.
ww WK YYYY	Zweistellige Zahl für die Kalenderwoche, gefolgt von den Buchstaben *WK* und dann von einer vierstelligen Jahreszahl. Bei der Berechnung der Kalenderwoche wird davon ausgegangen, dass Montag der erste Wochentag ist, und dass mindestens ein Tag in der ersten Kalenderwoche liegt.

Zeit

Die CLEM-Sprache unterstützt folgende Zeitformate.

Format	Beispiele
HHMMSS	120112, 010101, 221212
HHMM	1223, 0745, 2207
MMSS	5558, 0100
HH:MM:SS	12:01:12, 01:01:01, 22:12:12
HH:MM	12:23, 07:45, 22:07
MM:SS	55:58, 01:00
(H)H:(M)M:(S)S	12:1:12, 1:1:1, 22:12:12
(H)H:(M)M	12:23, 7:45, 22:7
(M)M:(S)S	55:58, 1:0
HH.MM.SS	12.01.12, 01.01.01, 22.12.12
HH.MM	12.23, 07.45, 22.07
MM.SS	55.58, 01.00
(H)H.(M)M.(S)S	12.1.12, 1.1.1, 22.12.12
(H)H.(M)M	12.23, 7.45, 22.7
(M)M.(S)S	55.58, 1.0

CLEM Operatoren

Folgende Operatoren stehen zur Verfügung.

Operation	Kommentare	Rangfolge (siehe nächsten Abschnitt)
or	Wird zwischen zwei CLEM-Ausdrücken verwendet. Ergibt den Wert “Wahr”, wenn mindestens einer der beiden Ausdrücke wahr ist.	10
and	Wird zwischen zwei CLEM-Ausdrücken verwendet.	9

Operation	Kommentare	Rangfolge (siehe nächsten Abschnitt)
	Ergibt den Wert “Wahr”, wenn beide Ausdrücke “wahr” sind.	
=	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 gleich ELEMENT2 ist.	7
==	Identisch mit =.	7
/=	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 *nicht* gleich ELEMENT2 ist.	7
/==	Identisch mit /=.	7
>	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 größer als ELEMENT2 ist.	6
>=	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 größer oder gleich ELEMENT2 ist.	6
<	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 kleiner als ELEMENT2 ist.	6
<=	Wird zwischen zwei beliebigen miteinander vergleichbaren Elementen verwendet. Ergibt den Wert “Wahr”, wenn ELEMENT1 kleiner oder gleich ELEMENT2 ist.	6
&&=_0	Wird zwischen zwei ganzen Zahlen verwendet. Entspricht dem Boole’schen Ausdruck GANZZ1 && GANZZ2 = 0.	6
&&/=_0	Wird zwischen zwei ganzen Zahlen verwendet. Entspricht dem Boole’schen Ausdruck GANZZ1 && GANZZ2 /= 0.	6
+	Addiert zwei Zahlen: ZAHL1 + ZAHL2.	5
><	Verkettet zwei Zeichenfolgen; beispielsweise STRING1 >< STRING2.	5
-	Subtrahiert eine Zahl von einer anderen Zahl: ZAHL1 - ZAHL2. Kann außerdem vor einer Zahl verwendet werden: -ZAHL.	5
*	Dient zur Multiplikation zweier Zahlen: ZAHL1 * ZAHL2.	4

Operation	Kommentare	Rangfolge (siehe nächsten Abschnitt)
&&	Wird zwischen zwei ganzen Zahlen verwendet. Das Ergebnis liefert das bitweise “Und” der ganzen Zahlen GANZZ1 und GANZZ2.	4
&&~~	Wird zwischen zwei ganzen Zahlen verwendet. Das Ergebnis liefert das bitweise “Und” von GANZZ1 und das bitweise Komplement von GANZZ2.	4
||	Wird zwischen zwei ganzen Zahlen verwendet. Das Ergebnis liefert das bitweise “Oder” von GANZZ1 und GANZZ2.	4
~~	Wird vor einer ganzen Zahl verwendet. Liefert das bitweise Komplement von GANZZ.	4
||/&	Wird zwischen zwei ganzen Zahlen verwendet. Das Ergebnis liefert das bitweise “exklusive Oder” von GANZZ1 und GANZZ2.	4
INT1 << N	Wird zwischen zwei ganzen Zahlen verwendet. Ergibt das Bitmuster von GANZZ, um N Positionen nach links verschoben.	4
INT1 >> N	Wird zwischen zwei ganzen Zahlen verwendet. Ergibt das Bitmuster von GANZZ, um N Positionen nach rechts verschoben.	4
/	Dient zur Division zwischen zwei Zahlen: ZAHL1 / ZAHL2.	4
**	Wird zwischen zwei Zahlen verwendet. BASIS ** POTENZ. Ergibt BASIS hoch POTENZ.	3
rem	Wird zwischen zwei ganzen Zahlen verwendet: GANZZ1 rem GANZZ2. Ergibt den Rest: GANZZ1 - (GANZZ1 div GANZZ2) * GANZZ2.	2
div	Wird zwischen zwei ganzen Zahlen verwendet: GANZZ1 div GANZZ2. Führt eine Division zwischen zwei ganzen Zahlen durch.	2

Rangfolge der Operatoren

Rangfolgen bestimmen die Analyse komplexer Ausdrücke, insbesondere von Ausdrücken ohne Klammern mit mehreren Infix-Operatoren. Beispiel:

```
3 + 4 * 5
```

Analysiert als `3 + (4 * 5)` statt als `(3 + 4) * 5`, da die relativen Rangfolgen vorgeben, dass * vor + analysiert wird. Jedem Operator in der CLEM-Sprache ist ein Rangfolgewert zugeordnet; je niedriger dieser Wert ist, desto wichtiger ist er auf der Analyseliste, was bedeutet, dass er vor anderen Operatoren mit höheren Rangfolgewerten verarbeitet wird.

Funktionsreferenz

Folgende CLEM-Funktionen stehen bei der Arbeit mit Daten in IBM® SPSS® Modeler zur Verfügung. Diese Funktionen können in einer Reihe von Dialogfeldern (z. B. “Ableitungsknoten” und “Dichotomknoten”) als Code eingegeben werden. Alternativ können Sie mit dem Expression Builder gültige CLEM-Ausdrücke erstellen, ohne sich Funktionslisten oder Feldnamen einprägen zu müssen.

Funktionstyp	Beschreibung
Information	Dient dazu, Erkenntnisse über Feldwerte zu gewinnen. Beispiel: Die Funktion is_string gibt für alle Datensätze vom Typ “Zeichenkette” (String) einen Wahr-Wert zurück.
Umwandlung	Dient zur Erstellung neuer Felder oder zur Konvertierung des Speichertyps. Die Funktion to_timestamp konvertiert das ausgewählte Feld beispielsweise in einen Zeitstempel.
Vergleich	Dient zum Vergleichen von Feldwerten untereinander oder mit einer angegebenen Zeichenkette. Beispiel: Mit <= wird verglichen, ob die Werte zweier Felder kleiner oder gleich sind.
Logisch	Dient zur Durchführung logischer Operationen wie if (wenn), then (dann), else (sonst).
Numerisch	Dient zur Durchführung numerischer Berechnungen, wie dem natürlichen Logarithmus von Feldwerten.
Trigonometrisch	Dient zur Durchführung trigonometrischer Berechnungen, wie Berechnung des Arkuskosinus eines bestimmten Winkels.
Probability	Rückgabe von Wahrscheinlichkeiten auf der Grundlage verschiedener Verteilungen, z. B. die Wahrscheinlichkeit, dass ein Wert aus der *t*-Verteilung in der Studentenversion kleiner ist als ein bestimmter Wert.
Bitweise	Dient zur Bearbeitung von ganzen Zahlen als Bitmuster.
Random	Dient zur zufälligen Auswahl von Elementen oder Generierung von Zahlen.
Zeichenfolge	Dient zur Durchführung diverser Operationen an Zeichenketten, wie beispielsweise stripchar zum Entfernen eines angegebenen Zeichens.
SoundEx	Dient zum Auffinden von Zeichenketten, deren genaue Schreibweise nicht bekannt ist, und zwar auf der Grundlage fonetischer Annahmen, wie bestimmte Buchstaben ausgesprochen werden.
Datum und Uhrzeit,	Dient zur Durchführung verschiedenster Operationen von Datums-, Zeit- und Zeitstempelfeldern.
Sequenz	Dient zur Gewinnung von Erkenntnissen über die Datensatzsequenz eines Daten-Sets oder zur Durchführung von Operationen basierend auf dieser Sequenz.
Globalwert	Ermöglicht den Zugriff auf Globalwerte, die von einem Globalwerteknoten erstellt wurden. Beispiel: @MEAN wird als Verweis auf den mittleren Durchschnitt aller Werte für ein Feld im gesamten Daten-Set verwendet.
Leerstellen und Nullen	Dient zum Zugreifen, Kennzeichnen und häufig zum Ausfüllen benutzerdefinierter Leerstellen oder systemdefiniert fehlender Werte. Beispiel: @BLANK(FIELD) wird verwendet, um Datensätze mit Leerstellen als “wahr” zu kennzeichnen.
Sonderfelder	Dient zur Angabe der gesonderten Felder, die untersucht werden. @FIELD wird beispielsweise beim Ableiten mehrerer Felder verwendet.

Konventionen bei Funktionsbeschreibungen

Folgende Konventionen werden überall in diesem Dokument für Elemente in einer Funktion verwendet.

Konvention	Beschreibung
BOOL	Ein Boole'scher Wert bzw. Flag, wie "wahr" oder "falsch".
ZAHL, *ZAHL1*, *ZAHL2*	Eine beliebige Zahl.
REELL, *REELL1*, *REELL2*	Eine reelle Zahl, wie 1.234 oder –77.01.
GANZZ, *GANZZ1*, *GANZZ2*	Eine ganze Zahl, wie 1 oder –77.
CHAR	Eine Zeichencode, wie beispielsweise `A`.
STRING	Ein Zeichenkette, wie beispielsweise "referrerID".
LISTE	Eine Liste mit Elementen, wie beispielsweise ["abc" "def"].
ELEMENT	Ein Feld, wie beispielsweise Customer oder extract_concept.
DATE	Ein Datumsfeld, wie beispielsweise start_date, mit Werten in einem Format wie DD-MON-YYYY.
ZEIT	Ein Zeitfeld, wie beispielsweise power_flux, mit Werten in einem Format wie HHMMSS.

Die Funktionen in diesem Handbuch sind wie folgt aufgeführt: Die Funktion in der ersten Spalte, der Ergebnistyp (ganze Zahl, String usw.) in der zweiten und gegebenenfalls eine Beschreibung in der dritten Spalte. Folgendes Beispiel ist eine Beschreibung der Funktion rem.

Funktion	Ergebnis	Beschreibung
INT1 rem INT2	*Number*	Ergibt den Rest des Quotienten aus *GANZZ1* und *GANZZ2*. Beispiel: INT1 – (INT1 div INT2) * INT2.

Einzelheiten zur den Verwendungskonventionen, beispielsweise zur Auflistung von Elementen oder zur Angabe von Zeichen in einer Funktion, werden an anderer Stelle beschrieben. Für weitere Informationen siehe Thema CLEM-Datentypen in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Informationsfunktionen

Informationsfunktionen werden verwendet, um einen Einblick in die Werte eines bestimmten Felds zu gewinnen. Sie werden üblicherweise verwendet, um Flag-Felder abzuleiten. Beispiel: Mit der Funktion @BLANK können Sie ein Flag-Feld erstellen, das die Datensätze angibt, deren

Werte für das ausgewählte Feld leer sind. In ähnlicher Weise können Sie den Speichertyp für ein Feld mit einer der Speichertypfunktionen überprüfen, beispielsweise mit is_string.

Funktion	Ergebnis	Beschreibung
@BLANK(FIELD)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Werte gemäß den Regeln zum Umgang mit Leerstellen, die in einem weiter oben im Stream gelegenen Typknoten oder Quellenknoten (Registerkarte “Typen”) festgelegt wurden, Leerstellen sind. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
@NULL(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Werte nicht definiert sind. Nichtdefinierte Werte sind systemdefinierte Nullwerte, die in IBM® SPSS® Modeler als $null$ angezeigt werden. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
is_date(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ ein Datum ist.
is_datetime(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ “Datum”, “Zeit” oder “Zeitstempel” ist.
is_integer(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ eine ganze Zahl ist.
is_number(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ eine Zahl ist.
is_real(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ eine reelle Zahl ist.
is_string(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ eine Zeichenkette ist.
is_time(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ eine Zeitangabe ist.
is_timestamp(ITEM)	*Boolesch*	Ergibt den Wert “Wahr” für alle Datensätze, deren Typ ein Zeitstempel ist.

Konvertierungsfunktionen

Mit Konvertierungsfunktionen können Sie neue Felder erstellen und den Speichertyp bestehender Felder umwandeln. Beispielsweise können Sie neue Zeichenketten bilden, indem Sie Zeichenketten miteinander verbinden oder zerlegen. Zwei Zeichenketten können mit dem Operator >< miteinander verbunden werden. Wenn das Feld Site beispielsweise den Wert "BRAMLEY" hat, dann ergibt "xx" >< Site als Ergebnis "xxBRAMLEY". Das Ergebnis von >< ist immer eine Zeichenkette, selbst wenn die einzelnen Argumente keine Zeichenketten sind. Wenn das Feld V1 den Wert 3 und das Feld V2 den Wert 5 hat, dann gibt V1 >< V2 folglich als Ergebnis "35" zurück (eine Zeichenkette, keine Zahl).

Konvertierungsfunktionen (und alle anderen Funktionen, für die ein spezieller Eingabetyp, wie beispielsweise ein Wert für Datum oder Uhrzeit, erforderlich ist) hängen von den aktuell im Dialogfeld für die Stream-Optionen angegebenen Formaten ab. Wenn Sie beispielsweise ein Zeichenkettenfeld mit den Werten *Jan 2003*, *Feb 2003* usw. konvertieren möchten, müssen Sie das zugehörige Datumsformat, MON JJJJ, als Standard-Datumsformat für den Stream auswählen.

Für weitere Informationen siehe Thema Festlegen von allgemeinen Optionen für Streams in Kapitel 5 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Funktion	Ergebnis	Beschreibung
ITEM1 >< ITEM2	*Zeichenfolge*	Zieht die Werte der beiden Felder zusammen und ergibt die resultierende Zeichenkette als *ELEMENT1ELEMENT2*.
to_integer(ITEM)	*Ganzzahl*	Konvertiert den Speichertyp des angegebenen Felds in eine ganze Zahl.
to_real(ITEM)	*Reelle Zahl*	Konvertiert den Speichertyp des angegebenen Felds in eine reelle Zahl.
to_number(ITEM)	*Number*	Konvertiert den Speichertyp des angegebenen Felds in eine Zahl.
to_string(ITEM)	*Zeichenfolge*	Konvertiert den Speichertyp des angegebenen Felds in eine Zeichenkette.
to_time(ITEM)	*Zeit*	Konvertiert den Speichertyp des angegebenen Felds in eine Zeit.
to_date(ITEM)	*Datum*	Konvertiert den Speichertyp des angegebenen Felds in ein Datum.
to_timestamp(ITEM)	*Zeitstempel*	Konvertiert den Speichertyp des angegebenen Felds in einen Zeitstempel.
to_datetime(ITEM)	*Datetime*	Konvertiert den Speichertyp des angegebenen Felds in einen Wert vom Typ “Datum”, “Zeit” oder “Zeitstempel”.
datetime_date(ITEM)	*Datum*	Gibt den Datumswert für eine *Zahl*, eine *Zeichenkette* oder einen *Zeitstempel* zurück. Beachten Sie, dass dies die einzige Funktion ist, mit der Sie eine Zahl (in wenigen Sekunden) wieder in ein Datum zurückkonvertieren können. Wenn es sich bei ITEM um eine Zeichenkette handelt, wird ein Datum erstellt, indem eine Zeichenkette im aktuellen Datumsformat analysiert wird. Das im Dialogfeld für die Stream-Eigenschaften angegebene Datumsformat muss korrekt sein, damit diese Funktion erfolgreich ausgeführt werden kann. Wenn es sich bei ITEM um eine Zahl handelt, wird sie als Anzahl von Sekunden seit dem Basisdatum (Epoche) interpretiert. Die Bruchteile eines Tages werden gekürzt. Wenn es sich bei ITEM um einen Zeitstempel handelt, wird der Datumsteil des Zeitstempels ausgegeben. Wenn es sich bei ITEM um ein Datum handelt, wird es unverändert ausgegeben.

Vergleichsfunktionen

Mit Vergleichsfunktionen werden Feldwerte miteinander oder mit einer angegebenen Zeichenkette verglichen. Beispielsweise können Sie Zeichenketten mit = auf Gleichheit überprüfen. Ein Beispiel für die Überprüfung der Zeichenkettengleichheit ist: Class = "class 1".

Bei numerischen Vergleichen bedeutet *größer* näher an der positiven Unendlichkeit und *kleiner* näher an der negativen Unendlichkeit. Negativen Zahlen sind also stets kleiner als jede positive Zahl.

Funktion	Ergebnis	Beschreibung
count_equal(ITEM1, LIST)	*Ganzzahl*	Ergibt die Anzahl der Werte aus einer Felderliste, die gleich *ELEMENT1* sind, bzw. den Wert null, wenn *ELEMENT1* gleich null ist. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
count_greater_than(ITEM1, LIST)	*Ganzzahl*	Ergibt die Anzahl der Werte aus einer Felderliste, die größer als *ELEMENT1* sind, bzw. den Wert null, wenn *ELEMENT1* gleich null ist.
count_less_than(ITEM1, LIST)	*Ganzzahl*	Ergibt die Anzahl der Werte aus einer Felderliste, die kleiner als *ELEMENT1* sind, bzw. den Wert null, wenn *ELEMENT1* gleich null ist.
count_not_equal(ITEM1, LIST)	*Ganzzahl*	Ergibt die Anzahl der Werte aus einer Felderliste, die ungleich *ELEMENT1* sind, bzw. den Wert null, wenn *ELEMENT1* gleich null ist.
count_nulls(LIST)	*Ganzzahl*	Ergibt die Anzahl der Nullwerte aus einer Felderliste.
count_non_nulls(LIST)	*Ganzzahl*	Ergibt die Anzahl der Nicht-Nullwerte aus einer Felderliste.
date_before(DATE1, DATE2)	*Boolesch*	Dient zur Überprüfung der alphabetischen Sortierung von Datumswerten. Ergibt den Wert "Wahr", wenn *DATUM1* vor *DATUM2* liegt.
first_index(ITEM, LIST)	*Ganzzahl*	Ergibt den Index des ersten Felds aus einer LISTE von Feldern, das ELEMENT enthält, oder "0", wenn der Wert nicht gefunden wird. Nur für die Typen "Zeichenkette", "Ganze Zahl" und "Reelle Zahl" unterstützt. Für weitere Informationen siehe Thema Arbeiten mit Mehrfachantwortdaten in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
first_non_null(LIST)	*Jede*	Ergibt den ersten Nicht-Null-Wert in der angegebenen Felderliste. Alle Speichertypen werden unterstützt.
first_non_null_index(LIST)	*Ganzzahl*	Ergibt den Index des erstens Felds in der angegebenen LISTE, das einen Nicht-Null-Wert enthält, oder "0", wenn alle Werte null sind. Alle Speichertypen werden unterstützt.
ITEM1 = ITEM2	*Boolesch*	Ergibt den Wert "Wahr" für Datensätze, bei denen *ELEMENT1* gleich *ELEMENT2* ist.
ITEM1 /= ITEM2	*Boolesch*	Ergibt den Wert "Wahr", wenn die beiden Zeichenketten nicht identisch sind, oder "0", wenn sie identisch sind.
ITEM1 < ITEM2	*Boolesch*	Ergibt den Wert "Wahr" für Datensätze, bei denen *ELEMENT1* kleiner als *ELEMENT2* ist.
ITEM1 <= ITEM2	*Boolesch*	Ergibt den Wert "Wahr" für Datensätze, bei denen *ELEMENT1* kleiner oder gleich *ELEMENT2* ist.
ITEM1 > ITEM2	*Boolesch*	Ergibt den Wert "Wahr" für Datensätze, bei denen *ELEMENT1* größer als *ELEMENT2* ist.
ITEM1 >= ITEM2	*Boolesch*	Ergibt den Wert "Wahr" für Datensätze, bei denen *ELEMENT1* größer oder gleich *ELEMENT2* ist.

Funktion	Ergebnis	Beschreibung
last_index(ITEM, LIST)	*Ganzzahl*	Ergibt den Index des letzten Felds aus einer LISTE von Feldern, das ELEMENT enthält, oder "0", wenn der Wert nicht gefunden wird. Nur für die Typen "Zeichenkette", "Ganze Zahl" und "Reelle Zahl" unterstützt. Für weitere Informationen siehe Thema Arbeiten mit Mehrfachantwortdaten in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
last_non_null(LIST)	*Jede*	Ergibt den letzten Nicht-Null-Wert in der angegebenen Felderliste. Alle Speichertypen werden unterstützt.
last_non_null_index(LIST)	*Ganzzahl*	Ergibt den Index des letzten Felds in der angegebenen LISTE, das einen Nicht-Null-Wert enthält, oder "0", wenn alle Werte null sind. Alle Speichertypen werden unterstützt.
max(ITEM1, ITEM2)	*Jede*	Ergibt das größere der beiden Elemente: *ELEMENT1* oder *ELEMENT2*.
max_index(LIST)	*Ganzzahl*	Ergibt den Index des Felds mit dem größten Wert aus einer Liste numerischer Felder bzw. "0", wenn alle Feldwerte null sind. Wenn beispielsweise das dritte Feld in der Liste den größten Wert enthält, wird der Indexwert 3 ausgegeben. Wenn mehrere Felder den größten Wert enthalten, wird das zuerst aufgelistete (von links nach rechts) ausgegeben. Für weitere Informationen siehe Thema Arbeiten mit Mehrfachantwortdaten in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
max_n(LIST)	*Number*	Ergibt den größten Wert aus einer Liste numerischer Felder bzw. den Wert null, wenn alle Feldwerte gleich null sind. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
member(ITEM, LIST)	*Boolesch*	Ergibt den Wert "Wahr", wenn *ELEMENT* ein Mitglied der angegebenen *LISTE* ist. Andernfalls ergibt sich ein Falsch-Wert. Auch eine Liste mit Feldnamen kann angegeben werden. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
min(ITEM1, ITEM2)	*Jede*	Ergibt das kleinere der beiden Elemente: *ELEMENT1* oder *ELEMENT2*.
min_index(LIST)	*Ganzzahl*	Ergibt den Index des Felds mit dem kleinsten Wert aus einer Liste numerischer Felder bzw. "0", wenn alle Feldwerte null sind. Wenn beispielsweise das dritte Feld in der Liste den kleinsten Wert enthält, wird der Indexwert 3 ausgegeben. Wenn mehrere Felder den kleinsten Wert enthalten, wird das zuerst aufgelistete (von links nach rechts) ausgegeben. Für weitere Informationen siehe Thema Arbeiten mit Mehrfachantwortdaten in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
min_n(LIST)	*Number*	Ergibt den kleinsten Wert aus einer Liste numerischer Felder bzw. den Wert null, wenn alle Feldwerte gleich null sind.

Funktion	Ergebnis	Beschreibung
time_before(TIME1, TIME2)	*Boolesch*	Dient zur Überprüfung der alphabetischen Sortierung von Zeitwerten. Ergibt den Wert "Wahr", wenn *ZEIT1* vor *ZEIT2* liegt.
value_at(INT, LIST)		Gibt den Wert jedes aufgelisteten Felds bei Offset GANZZ zurück oder NULL, wenn das Offset außerhalb des Bereichs der gültigen Werte liegt (d. h. kleiner als 1 oder größer als die Anzahl der aufgelisteten Felder ist). Alle Speichertypen werden unterstützt.

Logische Funktionen

CLEM-Ausdrücke können zur Durchführung logischer Operationen verwendet werden.

Funktion	Ergebnis	Beschreibung
COND1 and COND2	*Boolesch*	Bei dieser Operation handelt es sich um eine logische Konjunktion, die einen Wahr-Wert ergibt, wenn sowohl *BEDG1* als auch *BEDG2* wahr sind. Wenn *BEDG1* falsch ist, wird *BEDG2* nicht ausgewertet; dadurch sind Konjunktionen möglich, bei denen *BEDG1* zuerst testet, ob eine Operation in *BEDG2* zulässig ist. Beispiel: length(Label) >=6 und Label(6) = 'x'.
COND1 or COND2	*Boolesch*	Bei dieser Operation handelt es sich um eine logische (inklusive) Disjunktion, die einen Wahr-Wert ergibt, wenn entweder *BEDG1* oder *BEDG2* wahr ist oder wenn beide wahr sind. Wenn *BEDG1* wahr ist, wird *BEDG2* nicht ausgewertet.
not(COND)	*Boolesch*	Bei dieser Operation handelt es sich um eine logische Negation, die einen Wahr-Wert ergibt, wenn *BEDG* falsch ist. Anderenfalls gibt diese Operation den Wert 0 zurück.
if COND then EXPR1 else EXPR2 endif	*Jede*	Bei dieser Operation handelt es sich um eine bedingte Auswertung. Wenn *BEDG* wahr ist, ergibt die Operation das Ergebnis von *AUSDR1*. Andernfalls ergibt sich das Ergebnis der Auswertung von *AUSDR2*.
if COND1 then EXPR1 elseif COND2 then EXPR2 else EXPR_N endif	*Jede*	Bei dieser Operation handelt es sich um eine bedingte Auswertung mit mehreren Zweigen. Wenn *BEDG1* wahr ist, ergibt die Operation das Ergebnis von *AUSDR1*. Andernfalls, wenn *BEDG2* wahr ist, ergibt die Operation das Ergebnis der Auswertung von *AUSDR2*. Andernfalls ergibt sich das Ergebnis der Auswertung von *AUSDR_N*.

Numerische Funktionen

CLEM enthält eine Reihe häufig verwendeter numerischer Funktionen.

Funktion	Ergebnis	Beschreibung
–NUM	*Number*	Wird zur Umkehrung des Vorzeichens von *ZAHL* verwendet. Gibt die entsprechende Zahl mit dem entgegengesetzten Vorzeichen zurück.
NUM1 + NUM2	*Number*	Ergibt die Summe von *ZAHL1* und *ZAHL2*.
Code –NUM2	*Number*	Ergibt die Differenz von *ZAHL1* und *ZAHL2*.
NUM1 * NUM2	*Number*	Ergibt das Produkt aus *ZAHL1* und *ZAHL2*.

Funktion	Ergebnis	Beschreibung
NUM1 / NUM2	*Number*	Ergibt den Quotienten aus *ZAHL1* und *ZAHL2*.
INT1 div INT2	*Number*	Wird zur Durchführung einer Division zwischen zwei ganzen Zahlen verwendet. Ergibt den Wert von *GANZZ1* dividiert durch *GANZZ2*.
INT1 rem INT2	*Number*	Ergibt den Rest des Quotienten aus *GANZZ1* und *GANZZ2*. Beispiel: INT1 – (INT1 div INT2) * INT2.
INT1 mod INT2	*Number*	Diese Funktion wurde verworfen. Verwenden Sie stattdessen die Funktion rem.
BASE ** POWER	*Number*	Ergibt den Wert von *BASIS* hoch *POTENZ*. Für beide Werte ist eine beliebige Zahl möglich (mit der Ausnahme, dass *BASIS* nicht null sein darf, wenn *POTENZ* einen anderen Nullwert als die ganze Zahl 0 aufweist). Wenn *POTENZ* eine ganze Zahl ist, erfolgt die Berechnung, indem *BASIS* mehrmals nacheinander mit sich selbst multipliziert wird. Wenn *BASIS* also eine ganze Zahl ist, ist das Ergebnis ebenfalls eine ganze Zahl. Wenn *POTENZ* die ganze Zahl 0 ist, ist das Ergebnis immer eine 1 vom selben Typ wie *BASIS*. Andernfalls, wenn *POTENZ* keine ganze Zahl ist, wird das Ergebnis als exp(POWER * log(BASE)) berechnet.
abs(NUM)	*Number*	Ergibt den Absoluten Wert von *ZAHL*, bei dem es sich immer um eine Zahl desselben Typs handelt.
exp(NUM)	*Reelle Zahl*	Ergibt den Wert von *e* hoch *ZAHL*. Dabei ist *e* die Basis eines natürlichen Logarithmus.
fracof(NUM)	*Reelle Zahl*	Ergibt den Bruchbereich (Nachkommastellen) von *ZAHL*, der definiert ist als NUM–intof(NUM).
intof(NUM)	*Ganzzahl*	Kürzt das Argument zu einer ganzen Zahl. Ergibt eine ganze Zahl mit demselben Vorzeichen wie *ZAHL* und mit dem größten Absolutwert, sodass abs(INT) <= abs(NUM).
log(NUM)	*Reelle Zahl*	Ergibt den natürlichen Logarithmus (Basis *e*) von *ZAHL* (darf keine Art von Nullwert sein).
log10(NUM)	*Reelle Zahl*	Ergibt den Logarithmus zur Basis 10 von *ZAHL* (darf keine Art von Nullwert sein). Diese Funktion ist definiert als log(NUM) / log(10).
negate(NUM)	*Number*	Wird zur Umkehrung des Vorzeichens von *ZAHL* verwendet. Ergibt die entsprechende Zahl mit dem entgegengesetzten Vorzeichen.
round(NUM)	*Ganzzahl*	Dient zum Runden von *ZAHL* auf eine ganze Zahl. Dabei wird intof(NUM+0.5) verwendet, wenn *ZAHL* positiv ist, oder intof(NUM–0.5), wenn *ZAHL* negativ ist.
sign(NUM)	*Number*	Wird zur Ermittlung des Vorzeichens von *ZAHL* verwendet. Diese Operation ergibt –1, 0 oder 1, wenn *ZAHL* eine ganze Zahl ist. Wenn *ZAHL* eine reelle Zahl ist, ergibt sich –1,0; 0,0 oder 1,0, je nachdem ob *ZAHL* negativ, null oder positiv ist.
sqrt(NUM)	*Reelle Zahl*	Ergibt die Quadratwurzel von *ZAHL*. *ZAHL* muss positiv sein.
sum_n(LIST)	*Number*	Ergibt die Summe der Werte aus einer Liste numerischer Felder bzw. den Wert null, wenn alle Feldwerte gleich null sind. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Funktion	Ergebnis	Beschreibung
mean_n(LIST)	*Number*	Ergibt den Mittelwert aus einer Liste numerischer Felder bzw. den Wert null, wenn alle Feldwerte gleich null sind.
sdev_n(LIST)	*Number*	Ergibt die Standardabweichung aus einer Liste numerischer Felder bzw. den Wert null, wenn alle Feldwerte gleich null sind.

Trigonometrische Funktionen

Alle Funktionen in diesem Abschnitt verwenden entweder einen Winkel als Argument oder ergeben einen Winkel als Ergebnis. In beiden Fällen richtet sich die für den Winkel verwendete Einheit (Radianten oder Grad) nach die Einstellung in der entsprechenden Stream-Option.

Funktion	Ergebnis	Beschreibung
arccos(NUM)	*Reelle Zahl*	Berechnet den Arcuscosinus des angegebenen Winkels.
arccosh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Arcuscosinus des angegebenen Winkels.
arcsin(NUM)	*Reelle Zahl*	Berechnet den Arcussinus des angegebenen Winkels.
arcsinh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Arcussinus des angegebenen Winkels.
arctan(NUM)	*Reelle Zahl*	Berechnet den Arcustangens des angegebenen Winkels.
arctan2(NUM_Y, NUM_X)	*Reelle Zahl*	Berechnet den Arcustangens von NUM_Y / NUM_X und verwendet die Vorzeichen der beiden Zahlen zur Ableitung von Quadranteninformationen. Das Ergebnis ist eine reelle Zahl im Bereich von - pi < ANGLE <= pi (radians) – 180 < ANGLE <= 180 (degrees) .
arctanh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Arcustangens des angegebenen Winkels.
cos(NUM)	*Reelle Zahl*	Berechnet den Cosinus des angegebenen Winkels.
cosh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Cosinus des angegebenen Winkels.
pi	*Reelle Zahl*	Diese Konstante ist die beste reelle Annäherung an Pi.
sin(NUM)	*Reelle Zahl*	Berechnet den Sinus des angegebenen Winkels.
sinh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Sinus des angegebenen Winkels.
tan(NUM)	*Reelle Zahl*	Berechnet den Tangens des angegebenen Winkels.
tanh(NUM)	*Reelle Zahl*	Berechnet den hyperbolischen Tangens des angegebenen Winkels.

Wahrscheinlichkeitsfunktionen

Wahrscheinlichkeitsfunktionen ergeben Wahrscheinlichkeiten auf der Grundlage verschiedener Verteilungen, z. B. die Wahrscheinlichkeit, dass ein Wert aus der *t*-Verteilung in der Studentenversion kleiner ist als ein bestimmter Wert.

Funktion	Ergebnis	Beschreibung
cdf_chisq(NUM, DF)	*Reelle Zahl*	Ergibt die Wahrscheinlichkeit, dass ein Wert aus der Chi-Quadrat-Verteilung mit den angegebenen Freiheitsgraden kleiner ist als die angegebene Zahl.
cdf_f(NUM, DF1, DF2)	*Reelle Zahl*	Ergibt die Wahrscheinlichkeit, dass ein Wert aus der *F*-Verteilung mit den angegebenen Freiheitsgraden *DF1* und *DF2* kleiner ist als die angegebene Zahl.
cdf_normal(NUM, MEAN, STDDEV)	*Reelle Zahl*	Ergibt die Wahrscheinlichkeit, dass ein Wert aus der Normalverteilung mit dem angegebenen Mittelwert und der angegebenen Standardabweichung kleiner ist als die angegebene Zahl.
cdf_t(NUM, DF)	*Reelle Zahl*	Ergibt die Wahrscheinlichkeit, dass ein Wert aus der Chi-Quadrat-Verteilung aus einer *t*-Verteilung in der Studentenversion mit den angegebenen Freiheitsgraden kleiner ist als die angegebene Zahl.

Bitweise Operationen mit ganzen Zahlen

Mit diesen Funktionen können Ganzzahlen als Bitmuster behandelt werden, die die Zweierkomplement-Werte darstellen, bei denen die Bit-Position N den Wert 2**N aufweist. Bits werden von 0 aufwärts nummeriert. Diese Operationen funktionieren so, als ob das Vorzeichen-Bit einer ganzen Zahl endlos nach links erweitert wird. Daher weisen positive ganze Zahlen überall oberhalb des werthöchsten 0-Bits auf, negative ganze Zahlen 1-Bits.

Hinweis: Die bitweisen Funktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
~~ INT1	*Ganzzahl*	Liefert das bitweise Komplement der ganzen Zahl *GANZZ1*. Im Ergebnis steht also eine 1 für jede Bitposition, bei der *GANZZ1* den Wert 0 aufweist. Es gilt immer: ~~ INT = –(INT + 1). Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
INT1 || INT2	*Ganzzahl*	Das Ergebnis dieser Operation ist das bitweise "inklusive Oder" von *GANZZ1* und *GANZZ2*. Im Ergebnis steht also eine 1 für jede Bitposition, bei der bei *GANZZ1* und/oder bei *GANZZ2* der Wert 1 steht.
INT1 ||/& INT2	*Ganzzahl*	Das Ergebnis dieser Operation ist das bitweise "exklusive Oder" von *GANZZ1* und *GANZZ2*. Im Ergebnis steht also eine 1 für jede Bitposition, bei der entweder bei *GANZZ1* oder bei *GANZZ2* der Wert 1 steht, nicht jedoch bei beiden.

Funktion	Ergebnis	Beschreibung
INT1 && INT2	*Ganzzahl*	Liefert das bitweise “Und” der ganzen Zahlen *GANZZ1* und *GANZZ2*. Im Ergebnis steht also eine 1 für jede Bitposition, bei der sowohl bei *GANZZ1* als auch bei *GANZZ2* der Wert 1 steht.
INT1 &&~~ INT2	*Ganzzahl*	Liefert das bitweise “Und” von *GANZZ1* und das bitweise Komplement von *GANZZ2*. Im Ergebnis steht also eine 1 für jede Bitposition, bei der bei *GANZZ1* der Wer 1 und bei*GANZZ2* der Wert 0 steht. Diese Operation ist identisch mit INT1&& (~~INT2) und ist sinnvoll zum Bereinigen von in *GANZZ2* gesetzten Bits von *GANZZ1*.
INT << N	*Ganzzahl*	Ergibt das Bitmuster von *GANZZ1*, um *N* Positionen nach links verschoben. Ein negativer Wert von *N* führt zu einer Verschiebung nach rechts.
INT >> N	*Ganzzahl*	Ergibt das Bitmuster von *GANZZ1*, um *N* Positionen nach rechts verschoben. Ein negativer Wert von *N* führt zu einer Verschiebung nach links.
INT1 &&=_0 INT2	*Boolesch*	Entspricht dem Boole’schen Ausdruck INT1 && INT2 /== 0, ist jedoch effizienter.
INT1 &&/=_0 INT2	*Boolesch*	Entspricht dem Boole’schen Ausdruck INT1 && INT2 == 0, ist jedoch effizienter.
integer_bitcount(INT)	*Ganzzahl*	Zählt die Anzahl der 1- oder 0-Bits in der Zweierkomplement-Darstellung von *GANZZ*. Wenn *GANZZ* nichtnegativ ist, ist *N* die Anzahl der 1-Bits. Wenn *GANZZ* negativ ist, gibt dieser Wert die Anzahl der 0-Bits an. Aufgrund der Vorzeichenerweiterung gibt es unendlich Viele 0-Bits in nichtnegativen Ganzzahlen bzw. 1-Bits in negativen Ganzzahlen. Es gilt immer: integer_bitcount(INT) = integer_bitcount(-(INT+1)).
integer_leastbit(INT)	*Ganzzahl*	Gibt die Bitposition *N* des wertniedrigsten Bits an, das in der ganzen Zahl *GANZZ* gesetzt wurde. *N* ist die höchste Potenz von 2, durch die sich *GANZZ* ohne Rest teilen lässt.
integer_length(INT)	*Ganzzahl*	Ergibt die Länge von *GANZZ* als ganze Zweierkomplement-Zahl in Bit. Im Ergebnis ist also *N* die kleinste Ganzzahl, sodass INT < (1 << N) if INT >= 0 INT >= (–1 << N) if INT < 0. Wenn *GANZZ* nichtnegativ ist, ist für die Darstellung von *GANZZ* als vorzeichenlose ganze Zahl ein Feld mit mindestens *N* Bit erforderlich. Alternativ sind, unabhängig vom Vorzeichen, mindestens *N+1* Bit erforderlich, um *GANZZ* als ganze Zahl mit Vorzeichen darzustellen.
testbit(INT, N)	*Boolesch*	Testet das Bit an Position *N* in der ganzen Zahl *GANZZ* und ergibt den Status von Bit *N* als Boole’schen Wert, der bei 1 “wahr” und bei 0 “falsch” ist.

Zufallsfunktionen

Folgende Funktionen dienen zur zufälligen Auswahl von Elementen oder zur Generierung von Zufallszahlen.

Funktion	Ergebnis	Beschreibung
oneof(LIST)	*Jede*	Ergibt ein zufällig ausgewähltes Element von *LISTE*. Die Listenelemente sollten wie folgt eingegeben werden [ITEM1,ITEM2,...,ITEM_N]. Auch eine Liste mit Feldnamen kann angegeben werden. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
random(NUM)	*Number*	Ergibt eine gleichmäßig verteilte Zufallszahl desselben Typs (*GANZZ* oder *REELL*). Es gilt der Bereich von 1 bis *ZAHL*. Bei Verwendung einer ganzen Zahl ergeben sich nur ganze Zahlen. Bei Verwendung einer reellen Zahl (Dezimalzahl), ergeben sich reelle Zahlen (die Anzahl der Dezimalstellen werden in den Stream-Optionen festgelegt). Die größte Zufallszahl, die die Funktion ergibt, kann gleich *ZAHL* sein.
random0(NUM)	*Number*	Diese Operation hat dieselben Eigenschaften wie random(NUM), der Bereich beginnt jedoch bei 0. Die größte Zufallszahl, die die Funktion ergibt, ist nie gleich *X*.

String-Funktionen

In CLEM können Sie folgende Operationen an Zeichenketten durchführen:

- Vergleichen von Zeichenketten
- Erstellen von Zeichenketten
- Zugreifen auf Zeichen

In CLEM-Ausdrücken ist eine Zeichenkette jede Folge von Zeichen zwischen zusammengehörenden (doppelten) Anführungszeichen ("string quotes"). Zeichen (CHAR) können beliebige alphanumerische Zeichen sein. Sie werden in CLEM-Ausdrücken unter Verwendung von einfachen umgekehrten Anführungszeichen in folgendem Format deklariert: `<*Zeichen*>`. Beispiele: `z`, `A` oder `2`. Zeichen, die außerhalb des zulässigen Bereichs liegen, oder negative Indizes bei einer Zeichenkette führen zu einem undefinierten Verhalten.

Anmerkung. Vergleiche zwischen Zeichenketten, die SQL-Pushback verwenden oder nicht, können unterschiedliche Ergebnisse generieren, wenn nachgestellte Leerzeichen vorhanden sind.

Funktion	Ergebnis	Beschreibung
allbutfirst(N, STRING)	*Zeichen-folge*	Ergibt eine Zeichenkette, bei der es sich um *ZEICHENKETTE* nach der Entfernung der ersten *N* Zeichen handelt.
allbutlast(N, STRING)	*Zeichen-folge*	Ergibt eine Zeichenkette, bei der es sich um *ZEICHENKETTE* nach der Entfernung der letzten N Zeichen handelt.
alphabefore(STRING1, STRING2)	*Boolesch*	Dient zur Überprüfung der alphabetischen Sortierung von Zeichenketten. Ergibt den Wert “Wahr”, wenn *ZEICHENKETTE1* der *ZEICHENKETTE2* vorausgeht.

Funktion	Ergebnis	Beschreibung
endstring(LENGTH, STRING)	*Zeichen-folge*	Extrahiert die letzten *N* Zeichen aus der angegebenen Zeichenkette. Ist eine Zeichenkette kleiner oder gleich der angegebenen Länge, dann bleibt die Zeichenkette unverändert.
hasendstring(STRING, SUBSTRING)	*Ganzzahl*	Diese Funktion entspricht isendstring(SUBSTRING, STRING)..
hasmidstring(STRING, SUBSTRING)	*Ganzzahl*	Diese Funktion entspricht ismidstring(SUBSTRING, STRING) (eingebettete Teilzeichenkette).
hasstartstring(STRING, SUBSTRING)	*Ganzzahl*	Diese Funktion entspricht isstartstring(SUBSTRING, STRING).
hassubstring(STRING, N, SUBSTRING)	*Ganzzahl*	Diese Funktion entspricht issubstring(SUBSTRING, N, STRING), wobei für *N* standardmäßig der Wert 1 verwendet wird.
count_substring(STRING, SUBSTRING)	*Ganzzahl*	Ergibt den Wert, wie oft die angegebene Teilzeichenkette in der Zeichenkette auftritt. Beispiel: count_substring("foooo.txt", "oo") ergibt 3.
hassubstring(STRING, SUBSTRING)	*Ganzzahl*	Diese Funktion entspricht issubstring(SUBSTRING, 1, STRING), wobei für *N* standardmäßig der Wert 1 verwendet wird.
isalphacode(CHAR)	*Boolesch*	Ergibt den Wert "Wahr", wenn *CHAR* ein Zeichen in der angegebenen Zeichenkette (häufig ein Feldname) ist, dessen Zeichencode ein Buchstabe ist. Andernfalls ergibt diese Funktion den Wert 0. Beispiel: isalphacode(produce_num(1)).
isendstring(SUBSTRING, STRING)	*Ganzzahl*	Wenn die Zeichenkette *ZEICHENKETTE* mit der Teilzeichenkette *TEILZEICHENKETTE* endet, ergibt diese Funktion den ganzzahligen Index von *TEILZEICHENKETTE* in *ZEICHENKETTE*. Andernfalls ergibt die Funktion den Wert 0.
islowercode(CHAR)	*Boolesch*	Ergibt den Wert "Wahr", wenn *CHAR* ein Kleinbuchstabe der angegebenen Zeichenkette (häufig ein Feldname) ist. Andernfalls ergibt die Funktion den Wert 0. Beispielsweise sind sowohl islowercode(``) als auch islowercode(country_name(2)) gültige Ausdrücke.
ismidstring(SUBSTRING, STRING)	*Ganzzahl*	Wenn *TEILZEICHENKETTE* eine Teilzeichenkette von *ZEICHENKETTE* ist, jedoch nicht beim ersten Zeichen von *ZEICHENKETTE* beginnt oder mit dem letzten Zeichen endet, ergibt diese Funktion den Index, bei dem die Teilzeichenkette beginnt. Andernfalls gibt die Funktion den Wert 0 zurück.

Funktion	Ergebnis	Beschreibung
isnumbercode(CHAR)	*Boolesch*	Ergibt den Wert "Wahr", wenn *CHAR* bei der angegebenen Zeichenkette (häufig ein Feldname) ein Zeichen ist, dessen Zeichencode eine Ziffer ist. Andernfalls ergibt diese Funktion den Wert 0. Beispiel: isnumbercode(product_id(2)).
isstartstring(SUBSTRING, STRING)	*Ganzzahl*	Wenn die Zeichenkette *ZEICHENKETTE* mit der Teilzeichenkette *TEILZEICHENKETTE* beginnt, ergibt diese Funktion den Index 1. Andernfalls ergibt die Funktion den Wert 0.
issubstring(SUBSTRING, N, STRING)	*Ganzzahl*	Durchsucht die Zeichenkette *ZEICHENKETTE* beginnend beim *N*. Zeichen nach einer Teilzeichenkette, die mit *TEILZEICHENKETTE* übereinstimmt. Wenn die Teilzeichenkette gefunden wird, ergibt diese Funktion den Index, bei dem die Teilzeichenkette beginnt. Andernfalls ergibt diese Funktion den Wert 0. Wenn *N* nicht angegeben ist, wird standardmäßig der Wert 1 verwendet.
issubstring(SUBSTRING, STRING)	*Ganzzahl*	Durchsucht die Zeichenkette *ZEICHENKETTE* beginnend beim *N*. Zeichen nach einer Teilzeichenkette, die mit *TEILZEICHENKETTE* übereinstimmt. Wenn die Teilzeichenkette gefunden wird, ergibt diese Funktion den Index, bei dem die Teilzeichenkette beginnt. Andernfalls ergibt diese Funktion den Wert 0. Wenn *N* nicht angegeben ist, wird standardmäßig der Wert 1 verwendet.
issubstring_count(SUBSTRING, N, STRING):	*Ganzzahl*	Ergibt den Index für das *N*-te Vorkommen von *TEILZEICHENKETTE* in der angegebenen *ZEICHENKETTE*. Falls *TEILZEICHENKETTE* weniger als *N*-mal auftritt, ergibt sich der Wert 0.
issubstring_lim(SUBSTRING, N, STARTLIM, ENDLIM, STRING)	*Ganzzahl*	Diese Funktion entspricht issubstring, der Treffer muss jedoch beim oder vor dem Index *STARTLIM* beginnen und beim oder vor dem Index *ENDLIM* enden. Die Beschränkungen *STARTLIM* bzw. *ENDLIM* können deaktiviert werden, indem ein Falsch-Wert für die Argumente angegeben wird. So ist beispielsweise issubstring_lim(SUBSTRING, N, false, false, STRING) mit issubstring identisch.
isuppercode(CHAR)	*Boolesch*	Ergibt den Wert "Wahr", wenn *CHAR* ein Großbuchstabe ist. Andernfalls ergibt die Funktion den Wert 0. Beispielsweise sind sowohl isuppercode(``) als auch isuppercode(country_name(2)) gültige Ausdrücke.
last(CHAR)	*Zeichen-folge*	Ergibt das letze Zeichen *CHAR* von *ZEICHENKETTE* (muss mindestens ein Zeichen lang sein).

Funktion	Ergebnis	Beschreibung
length(STRING)	*Ganzzahl*	Ergibt die Länge der Zeichenkette *ZEICHENKETTE*, d. h. die Anzahl der Zeichen in der Zeichenkette.
locchar(CHAR, N, STRING)	*Ganzzahl*	Mit dieser Funktion kann die Position der Zeichen in symbolischen Feldern ermittelt werden. Mit dieser Funktion wird die Zeichenkette *ZEICHENKETTE* nach dem Zeichen *CHAR* durchsucht, wobei der Suchvorgang beim *N*. Zeichen von *ZEICHENKETTE* gestartet wird. Diese Funktion ergibt einen Wert (mit Startwert *N*), der die Position angibt, an der das Zeichen gefunden wurde. Wenn das Zeichen nicht gefunden wurde, ergibt die Funktion den Wert 0. Wenn die Funktion ein ungültiges Offset *(N)* aufweist (z. B. ein Offset, das außerhalb der Länge des Strings liegt), ergibt die Funktion $null$. locchar(`n`, 2, web_page) beispielsweise durchsucht das Feld *Webseite* nach dem Zeichen `n` ausgehend vom zweiten Zeichen im Feldwert. *Hinweis*: Achten Sie darauf, das angegebene Zeichen in einzelne umgekehrte Anführungszeichen einzuschließen.
locchar_back(CHAR, N, STRING)	*Ganzzahl*	Ähnlich wie locchar, außer dass die Suche ausgehend vom *N*-ten Zeichen rückwärts durchgeführt wird. Beispiel: locchar_back(`n`, 9, web_page) durchsucht das Feld *Webseite* ausgehend vom 9. Zeichen rückwärts bis zum Anfang der Zeichenkette. Wenn die Funktion ein ungültiges Offset aufweist (z. B. ein Offset, das außerhalb der Länge des Strings liegt), ergibt die Funktion $null$. Idealerweise sollte locchar_back in Verbindung mit der Funktion length(<field>) verwendet werden, um die Länge des aktuellen Wertes des Feldes dynamisch zu verwenden. Beispiel: locchar_back(`n`, (length(web_page)), web_page).
lowertoupper(CHAR) lowertoupper (STRING)	*CHAR* oder *Zeichenkette*	Die Eingabe kann entweder eine Zeichenkette oder ein Zeichen sein. Diese wird in dieser Funktion verwendet, um ein neues Element desselben Typs auszugeben, wobei alle Kleinbuchstaben in die entsprechenden Großbuchstaben konvertiert wurden. lowertoupper(`a`), lowertoupper("My string") und lowertoupper(field_name(2)) sind beispielsweise alle gültige Ausdrücke.

Funktion	Ergebnis	Beschreibung
matches	*Boolesch*	Ergibt den Wert “wahr”, wenn eine Zeichenkette mit einem bestimmten Muster übereinstimmt. Das Muster muss aus einer Literalzeichenkette bestehen; der Name eines Felds, das ein Muster enthält, ist nicht zulässig. Das Fragezeichen (?) als Platzhalterzeichen kann die Stelle von genau einem beliebigen Zeichen einnehmen, das Sternchen (*) entspricht keinem, einem oder mehreren Zeichen. Soll ein Fragezeichen oder ein Sternchen nicht als Platzhalter fungieren, sondern bei der Suche berücksichtigt werden, geben Sie den umgekehrten Schrägstrich (\) als Escape-Zeichen ein.
replace(SUBSTRING, NEWSUBSTRING, STRING)	*Zeichen-folge*	Innerhalb der angegebenen *ZEICHENKETTE* werden alle Instanzen von *TEILZEICHENKETTE* durch *NEUE_TEILZEICHENKETTE* ersetzt.
replicate(COUNT, STRING)	*Zeichen-folge*	Ergibt eine Zeichenkette, die aus der ursprünglichen Zeichenkette besteht, die so oft wie angegeben kopiert wurde.
stripchar(CHAR,STRING)	*Zeichen-folge*	Mit dieser Funktion können Sie die angegebenen Zeichen aus einer Zeichenkette oder einem Feld entfernen. Mit dieser Funktion können Sie beispielsweise zusätzliche Symbole, wie Währungsangaben, aus den Daten entfernen, um eine einfache Zahl oder einen einfachen Namen zu erhalten. Mit der Syntax stripchar(`$`, 'Cost') beispielsweise erhalten Sie ein neues Feld, bei dem das Dollar-Zeichen aus allen Werten entfernt ist. *Hinweis*: Achten Sie darauf, das angegebene Zeichen in einzelne umgekehrte Anführungszeichen einzuschließen.
skipchar(CHAR, N, STRING)	*Ganzzahl*	Durchsucht die Zeichenkette *ZEICHENKETTE* nach allen Zeichen außer *CHAR* und beginnt dabei beim *N*. Zeichen. Diese Funktion ergibt eine ganzzahlige Teilzeichenkette, die den Punkt angibt, an dem ein solches Zeichen gefunden wurde, oder 0, wenn es sich bei jedem Zeichen vom *N*. Zeichen an um *CHAR* handelt. Wenn die Funktion ein ungültiges Offset aufweist (z. B. ein Offset, das außerhalb der Länge des Strings liegt), ergibt die Funktion $null$. locchar wird häufig in Verbindung mit den Funktionen vom Typ skipchar verwendet, um den Wert von *N* (die Stelle, an der mit der Suche begonnen werden soll) zu bestimmen. Beispiel: skipchar(`s`, (locchar(`s`, 1, "MyString")), "MyString").
skipchar_back(CHAR, N, STRING)	*Ganzzahl*	Ähnlich wie skipchar, außer dass die Suche ausgehend vom *N*-ten Zeichen **rückwärts** durchgeführt wird.

Funktion	Ergebnis	Beschreibung
startstring(LENGTH, STRING)	*Zeichenfolge*	Extrahiert die ersten *N* Zeichen aus der angegebenen Zeichenkette. Ist eine Zeichenkette kleiner oder gleich der angegebenen Länge, dann bleibt die Zeichenkette unverändert.
strmember(CHAR, STRING)	*Ganzzahl*	Entspricht locchar(CHAR, 1, STRING). Ergibt eine ganzzahlige Teilzeichenkette, die entweder den Punkt angibt, an dem *CHAR* zum ersten Mal auftritt, oder den Wert 0. Wenn die Funktion ein ungültiges Offset aufweist (z. B. ein Offset, das außerhalb der Länge des Strings liegt), ergibt diese Funktion $null$.
subscrs(N, STRING)	*CHAR*	Ergibt das *N*. Zeichen *CHAR* der Eingabezeichenkette *ZEICHENKETTE*. Diese Funktion kann auch in einer abgekürzten Form geschrieben werden, nämlich als STRING(N). lowertoupper("name"(1)) beispielsweise ist ein gültiger Ausdruck.
substring(N, LEN, STRING)	*Zeichenfolge*	Ergibt die Zeichenkette *TEILZEICHENKETTE*, die aus den *LÄNGE* Zeichen der Zeichenkette *ZEICHENKETTE* besteht, wobei mit dem Zeichen bei Index *N* begonnen wird.
substring_between(N1, N2, STRING)	*Zeichenfolge*	Ergibt die Teilzeichenkette von *ZEICHENKETTE*, die bei Index *N1* beginnt und bei Index *N2* endet.
trim(STRING)	*Zeichenfolge*	Entfernt führende und nachgestellte Leerzeichen aus der angegebenen Zeichenkette.
trim_start(STRING)	*Zeichenfolge*	Entfernt führende Leerzeichen aus der angegebenen Zeichenkette.
trimend(STRING)	*Zeichenfolge*	Entfernt nachgestellte Leerzeichen aus der angegebenen Zeichenkette.
unicode_char(NUM)	*CHAR*	Ergibt das Zeichen mit dem Unicode-Wert *ZAHL*.
unicode_value(CHAR)	*NUM*	Ergibt den Unicode-Wert für das Zeichen *CHAR*.
uppertolower(CHAR) uppertolower (STRING)	*CHAR* oder *Zeichenkette*	Die Eingabe kann entweder eine Zeichenkette oder ein Zeichen sein und wird in dieser Funktion verwendet, um ein neues Element desselben Typs auszugeben, wobei alle Großbuchstaben in die entsprechenden Kleinbuchstaben konvertiert wurden. *Hinweis*: Achten Sie darauf, Zeichenketten in doppelte Anführungszeichen und Zeichen in einzelne, umgekehrte Anführungszeichen zu setzen. Einzelne Feldnamen sollten ohne Anführungszeichen angegeben werden.

SoundEx-Funktionen

SoundEx ist eine Methode zum Auffinden von Zeichenketten, deren Klang bekannt ist, nicht jedoch deren genaue Schreibweise. Die Methode wurde im Jahr 1918 entwickelt. Hiermit werden Wörter mit ähnlichen Lauten ermittelt, und zwar auf der Grundlage phonetischer Annahmen, wie bestimmte Buchstaben ausgesprochen werden. Suchen Sie beispielsweise Namen in einer Datenbank, bei denen die Schreibweise und die Betonung ähnlicher Namen voneinander abweichen können. Der grundlegende SoundEx-Algorithmus ist in zahlreichen Schriften dokumentiert. Trotz bekannter Einschränkungen (führende Buchstaben und Buchstabenfolgen wie ph und f gelten beispielsweise nicht als Übereinstimmung, obwohl sie gleich klingen) wird diese Methode in den meisten Datenbanken in irgendeiner Form genutzt.

Funktion	Ergebnis	Beschreibung
soundex(STRING)	*Ganzzahl*	Ergibt den vierbuchstabigen SoundEx-Code für die angegebene *ZEICHENKETTE*.
soundex_difference(STRING1, STRING2)	*Ganzzahl*	Ergibt eine ganze Zahl zwischen 0 und 4, aus der die Anzahl der Zeichen hervorgeht, die im SoundEx-Code für die beiden Zeichenketten übereinstimmen. Der Wert 0 bedeutet dabei, dass gar keine Ähnlichkeit zwischen den Zeichenketten vorliegt, der Wert 4 weist auf sehr ähnliche oder sogar identische Zeichenketten hin.

Datums- und Uhrzeitfunktionen

CLEM beinhaltet eine Gruppe von Funktionen zum Umgang mit Feldern mit dem Speichertyp "Datum/Uhrzeit" von Stringvariablen, die für Datums- und Zeitangaben stehen. Die Formate für Datum und Uhrzeit sind für jeden Stream spezifisch und werden im Dialogfeld "Stream-Eigenschaften" angegeben. Die Datums- und Zeitfunktionen analysieren Datums- und Zeitzeichenketten gemäß dem aktuell ausgewählten Format.

Wenn Sie ein Jahr in einem Datumsformat angeben, bei dem nur zwei Ziffern verwendet werden (d. h., das Jahrhundert ist nicht angegeben), verwendet IBM® SPSS® Modeler das im Dialogfeld "Stream-Eigenschaften" angegebene Standard-Jahrhundert.

Hinweis: Die Datums- und Zeitfunktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
@TODAY	*Zeichenfolge*	Bei Auswahl von Tage/Minuten übertragen im Dialogfeld "Stream-Eigenschaften" ergibt diese Funktion das aktuelle Datum als Zeichenkette im aktuellen Datumsformat. Wenn Sie ein zweistelliges Datumsformat verwenden und nicht die Option Tage/Minuten übertragen verwenden, ergibt diese Funktion $null$ auf dem aktuellen Server. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
to_time(ITEM)	*Zeit*	Konvertiert den Speichertyp des angegebenen Felds in eine Zeit.

Funktion	Ergebnis	Beschreibung
to_date(ITEM)	*Datum*	Konvertiert den Speichertyp des angegebenen Felds in ein Datum.
to_timestamp(ITEM)	*Zeitstempel*	Konvertiert den Speichertyp des angegebenen Felds in einen Zeitstempel.
to_datetime(ITEM)	*Datetime*	Konvertiert den Speichertyp des angegebenen Felds in einen Wert vom Typ "Datum", "Zeit" oder "Zeitstempel".
datetime_date(ITEM)	*Datum*	Gibt den Datumswert für eine *Zahl*, eine *Zeichenkette* oder einen *Zeitstempel* zurück. Beachten Sie, dass dies die einzige Funktion ist, mit der Sie eine Zahl (in wenigen Sekunden) wieder in ein Datum zurückkonvertieren können. Wenn es sich bei ITEM um eine Zeichenkette handelt, wird ein Datum erstellt, indem eine Zeichenkette im aktuellen Datumsformat analysiert wird. Das im Dialogfeld für die Stream-Eigenschaften angegebene Datumsformat muss korrekt sein, damit diese Funktion erfolgreich ausgeführt werden kann. Wenn es sich bei ITEM um eine Zahl handelt, wird sie als Anzahl von Sekunden seit dem Basisdatum (Epoche) interpretiert. Die Bruchteile eines Tages werden gekürzt. Wenn es sich bei ITEM um einen Zeitstempel handelt, wird der Datumsteil des Zeitstempels ausgegeben. Wenn es sich bei ITEM um ein Datum handelt, wird es unverändert ausgegeben.
date_before(DATE1, DATE2)	*Boolesch*	Ergibt den Wert "Wahr", wenn *DATUM1* ein Datum oder einen Zeitstempel darstellt, das/der vor dem durch *DATUM2* dargestellten Datum liegt. Andernfalls ergibt die Funktion den Wert 0.
date_days_difference(DATE1, DATE2)	*Ganzzahl*	Ergibt die Zeit in Tagen (als ganze Zahl), die zwischen dem durch *DATUM1* und dem durch *DATUM2* angegebenen Datum bzw. Zeitstempel liegt. Wenn *DATUM2* vor *DATUM1* liegt, ergibt diese Funktion eine negative Zahl.
date_in_days(DATE)	*Ganzzahl*	Ergibt die Zeit in Tagen (als ganze Zahl), die zwischen dem Basisdatum und dem durch *DATUM* angegebenen Datum oder Zeitstempel liegt. Wenn *DATUM* vor dem Basisdatum liegt, ergibt diese Funktion eine negative Zahl. Es muss ein gültiges Datum angegeben werden, damit die Berechnung ordnungsgemäß erfolgen kann. Beispielsweise sollten Sie nicht den 29. Februar 2001 als Datum angeben. Da 2001 kein Schaltjahr war, gibt es dieses Datum nicht.
date_in_months(DATE)	*Reelle Zahl*	Ergibt die Zeit in Monaten (als reelle Zahl), die zwischen dem Basisdatum und dem durch *DATUM* angegebenen Datum oder Zeitstempel liegt. Dies ist ein Näherungswert, der auf einem Monat mit 30.4375 Tagen beruht. Wenn *DATUM* vor dem Basisdatum liegt, ergibt diese Funktion eine negative Zahl. Es muss ein gültiges Datum angegeben werden, damit die Berechnung ordnungsgemäß erfolgen kann. Beispielsweise sollten Sie nicht den 29. Februar 2001 als Datum angeben. Da 2001 kein Schaltjahr war, gibt es dieses Datum nicht.

Funktion	Ergebnis	Beschreibung
date_in_weeks(DATE)	*Reelle Zahl*	Ergibt die Zeit in Wochen (als reelle Zahl), die zwischen dem Basisdatum und dem durch *DATUM* angegebenen Datum oder Zeitstempel liegt. Der Wert basiert auf einer Woche mit 7,0 Tagen. Wenn *DATUM* vor dem Basisdatum liegt, ergibt diese Funktion eine negative Zahl. Es muss ein gültiges Datum angegeben werden, damit die Berechnung ordnungsgemäß erfolgen kann. Beispielsweise sollten Sie nicht den 29. Februar 2001 als Datum angeben. Da 2001 kein Schaltjahr war, gibt es dieses Datum nicht.
date_in_years(DATE)	*Reelle Zahl*	Ergibt die Zeit in Jahren (als reelle Zahl), die zwischen dem Basisdatum und dem durch *DATUM* angegebenen Datum oder Zeitstempel liegt. Dies ist ein Näherungswert, der auf einem Jahr mit 365.25 Tagen beruht. Wenn *DATUM* vor dem Basisdatum liegt, ergibt diese Funktion eine negative Zahl. Es muss ein gültiges Datum angegeben werden, damit die Berechnung ordnungsgemäß erfolgen kann. Beispielsweise sollten Sie nicht den 29. Februar 2001 als Datum angeben. Da 2001 kein Schaltjahr war, gibt es dieses Datum nicht.
date_months_difference (DATE1, DATE2)	*Reelle Zahl*	Ergibt die Zeit in Monaten (als reelle Zahl), die zwischen dem durch *DATUM1* und dem durch *DATUM2* angegebenen Datum bzw. Zeitstempel liegt. Dies ist ein Näherungswert, der auf einem Monat mit 30.4375 Tagen beruht. Wenn *DATUM2* vor *DATUM1* liegt, ergibt diese Funktion eine negative Zahl.
datetime_date(YEAR, MONTH, DAY)	*Datum*	Erstellt einen Datumswert für die angegeben Wert für *JAHR*, *MONAT* und *TAG*. Bei den Argumenten muss es sich um ganze Zahlen handeln.
datetime_day(DATE)	*Ganzzahl*	Ergibt den Tag im Monat ausgehend von einem angegebenen *DATUM* oder Zeitstempel. Das Ergebnis ist eine Ganzzahl im Bereich von 1 bis 31.
datetime_day_name(DAY)	*Zeichenfolge*	Ergibt den vollständigen Namen für den angegeben *TAG*. Das Argument muss eine ganze Zahl im Bereich von 1 (Sonntag) bis 7 (Samstag) sein.
datetime_hour(TIME)	*Ganzzahl*	Ergibt die Stunde aus einer *ZEIT* oder einem Zeitstempel. Das Ergebnis ist eine Ganzzahl im Bereich von 0 bis 23.
datetime_in_seconds(TIME)	*Reelle Zahl*	Ergibt den in *ZEIT* gespeicherten Sekundenabschnitt.
datetime_in_seconds(DATE), datetime_in_seconds(DATE-TIME)	*Reelle Zahl*	Ergibt die aus der Differenz zwischen dem aktuellen *DATUM* oder *DATUM/UHRZEIT* und dem Basisdatum (1900-01-01) berechneten Gesamtwert (umgewandelt in Sekunden).
datetime_minute(TIME)	*Ganzzahl*	Ergibt die Minute aus einer *ZEIT* oder einem Zeitstempel. Das Ergebnis ist eine Ganzzahl im Bereich von 0 bis 59.
datetime_month(DATE)	*Ganzzahl*	Ergibt den Monat aus einem *DATUM* oder einem Zeitstempel. Das Ergebnis ist eine Ganzzahl im Bereich von 1 bis 12.
datetime_month_name (MONTH)	*Zeichenfolge*	Ergibt den vollständigen Namen für den angegeben *MONAT*. Das Argument muss eine ganze Zahl im Bereich von 1 bis 12 sein.
datetime_now	*Zeitstempel*	Ergibt die aktuelle Zeit als Zeitstempel.

Funktion	Ergebnis	Beschreibung
datetime_second(TIME)	*Ganzzahl*	Ergibt die Sekunde aus einer *ZEIT* oder einem Zeitstempel. Das Ergebnis ist eine Ganzzahl im Bereich von 0 bis 59.
datetime_day_short_ name(DAY)	*Zeichenfolge*	Ergibt den abgekürzten Namen für den angegeben *TAG*. Das Argument muss eine ganze Zahl im Bereich von 1 (Sonntag) bis 7 (Samstag) sein.
datetime_month_short_ name(MONTH)	*Zeichenfolge*	Ergibt den abgekürzten Namen für den angegeben *MONAT*. Das Argument muss eine ganze Zahl im Bereich von 1 bis 12 sein.
datetime_time(HOUR, MINUTE, SECOND)	*Zeit*	Ergibt den Zeitwert für die angegebene *STUNDE*, *MINUTE* und *SEKUNDE*. Bei den Argumenten muss es sich um ganze Zahlen handeln.
datetime_time(ITEM)	*Zeit*	Ergibt den Zeitwert für das angegebene *ELEMENT*.
datetime_timestamp(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)	*Zeitstempel*	Ergibt den Zeitstempelwert für die angegeben Wert für *JAHR*, *MONAT*, *TAG*, *STUNDE*, *MINUTE* und *SEKUNDE*.
datetime_timestamp(DATE, TIME)	*Zeitstempel*	Ergibt den Zeitstempelwert für das angegebene *DATUM* und die angegebene *ZEIT*.
datetime_timestamp (NUMBER)	*Zeitstempel*	Gibt den Zeitstempelwert für die angegebene Anzahl von Sekunden zurück.
datetime_weekday(DATE)	*Ganzzahl*	Ergibt den Tag in der Woche t ausgehend von einem angegebenen *DATUM* oder Zeitstempel.
datetime_year(DATE)	*Ganzzahl*	Ergibt das Jahr aus einem *DATUM* oder einem Zeitstempel. Das Ergebnis ist eine ganze Zahl, wie beispielsweise 2002.
date_weeks_difference (DATE1, DATE2)	*Reelle Zahl*	Ergibt die Zeit in Wochen (als reelle Zahl), die zwischen dem durch *DATUM1* und dem durch *DATUM2* angegebenen Datum bzw. Zeitstempel liegt. Der Wert basiert auf einer Woche mit 7,0 Tagen. Wenn *DATUM2* vor *DATUM1* liegt, ergibt diese Funktion eine negative Zahl.
date_years_difference (DATE1, DATE2)	*Reelle Zahl*	Ergibt die Zeit in Jahren (als reelle Zahl), die zwischen dem durch *DATUM1* und dem durch *DATUM2* angegebenen Datum bzw. Zeitstempel liegt. Dies ist ein Näherungswert, der auf einem Jahr mit 365.25 Tagen beruht. Wenn *DATUM2* vor *DATUM1* liegt, ergibt diese Funktion eine negative Zahl.
time_before(TIME1, TIME2)	*Boolesch*	Ergibt den Wert “Wahr”, wenn *ZEIT1* eine Uhrzeit oder einen Zeitstempel darstellt, die/der vor der durch *ZEIT2* dargestellten Zeit liegt. Andernfalls ergibt die Funktion den Wert 0.
time_hours_difference (TIME1, TIME2)	*Reelle Zahl*	Ergibt den Zeitunterschied in Stunden (als reelle Zahl) zwischen den durch *ZEIT1* und *ZEIT2* angegebenen Uhrzeiten oder Zeitstempeln. Bei Auswahl von Tage/Minuten übertragen im Dialogfeld “Stream-Eigenschaften” wird ein höherer Wert von *ZEIT1* als Verweis auf den vorhergehenden Tag interpretiert. Wenn die Übertragungsoption nicht ausgewählt wird, führt ein höherer Wert von *ZEIT1* dazu, dass der ausgegebene Wert negativ ist.

Funktion	Ergebnis	Beschreibung
time_in_hours(TIME)	*Reelle Zahl*	Ergibt die durch *ZEIT* angegebene Zeit in Stunden als reelle Zahl. So wird beispielsweise beim Zeitformat HHMM der Ausdruck time_in_hours('0130') als 1,5 ausgewertet. *ZEIT* kann für eine Uhrzeit oder einen Zeitstempel stehen.
time_in_mins(TIME)	*Reelle Zahl*	Ergibt die durch *ZEIT* angegebene Zeit in Minuten als reelle Zahl. *ZEIT* kann für eine Uhrzeit oder einen Zeitstempel stehen.
time_in_secs(TIME)	*Ganzzahl*	Ergibt die durch *ZEIT* angegebene Zeit in Sekunden als ganze Zahl. *ZEIT* kann für eine Uhrzeit oder einen Zeitstempel stehen.
time_mins_difference(TIME1, TIME2)	*Reelle Zahl*	Ergibt den Zeitunterschied in Minuten (als reelle Zahl) zwischen den durch *ZEIT1* und *ZEIT2* angegebenen Uhrzeiten oder Zeitstempeln. Bei Auswahl von Tage/Minuten übertragen im Dialogfeld "Stream-Eigenschaften" wird ein höherer Wert von *ZEIT1* als Verweis auf den vorhergehenden Tag (bzw. die vorangegangene Stunde, sofern nur Minuten und Sekunden im aktuellen Format angegeben werden) interpretiert. Wenn die Übertragungsoption nicht ausgewählt wird, führt ein höherer Wert von *ZEIT1* dazu, dass der ausgegebene Wert negativ ist.
time_secs_difference(TIME1, TIME2)	*Ganzzahl*	Ergibt den Zeitunterschied in Sekunden (als ganze Zahl) zwischen den durch *ZEIT1* und *ZEIT2* angegebenen Uhrzeiten oder Zeitstempeln. Bei Auswahl von Tage/Minuten übertragen im Dialogfeld "Stream-Eigenschaften" wird ein höherer Wert von *ZEIT1* als Verweis auf den vorhergehenden Tag (bzw. die vorangegangene Stunde, sofern nur Minuten und Sekunden im aktuellen Format angegeben werden) interpretiert. Wenn die Übertragungsoption nicht ausgewählt wird, führt ein höherer Wert von *ZEIT1* dazu, dass der ausgegebene Wert negativ ist.

Konvertieren von Datums- und Zeitwerten.

Beachten Sie, dass die Konvertierungsfunktionen (und alle anderen Funktionen, für die ein spezieller Eingabetyp, wie beispielsweise ein Wert für Datum oder Uhrzeit, erforderlich ist) von den aktuell im Dialogfeld für die Stream-Optionen angegebenen Formaten abhängen. Ein Feld mit der Bezeichnung *DATUM* beispielsweise, das als Zeichenkette mit den Werten *Jan 2003*, *Feb 2003* usw. gespeichert ist, könnte wie folgt in einen Datumsspeicher konvertiert werden:

```
to_date(DATUM)
```

Damit diese Konvertierung funktionieren kann, müssen Sie das entsprechende Datumsformat MON JJJJ als Standarddatumsformat für den Stream festlegen. Für weitere Informationen siehe Thema Festlegen von allgemeinen Optionen für Streams in Kapitel 5 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Ein Beispiel, bei dem Zeichenkettenwerte mithilfe eines Füllerknotens in Datumswerte konvertiert werden, finden Sie im Stream *broadband_create_models.str*, der im Ordner *\Demos* unter dem Unterordner *streams* installiert ist. Für weitere Informationen siehe Thema Prognoseerstellung mit dem Zeitreihenknoten in Kapitel 14 in *IBM SPSS Modeler 15- Benutzerhandbuch*.

Als Zahlen gespeicherte Datumsangaben. Beachten Sie, dass *DATUM* im vorherigen Beispiel der Name eines Felds ist, während es sich bei to_date um eine CLEM-Funktion handelt. Wenn Datumsangaben als Zahlen gespeichert wurden, können Sie sie mithilfe der Funktion datetime_date konvertieren. Dabei wird die Zahl als eine Anzahl von Sekunden seit dem Basisdatum (Epoche) interpretiert.

```
datetime_date(DATUM)
```

Indem Sie ein Datum in eine Sekundenzahl (und zurück) konvertieren, können Sie Berechnungen wie die Addition oder Subtraktion einer Anzahl von Tagen zum bzw. vom aktuellen Datum durchführen. Beispiel:

```
datetime_date((date_in_days(DATUM)-7)*60*60*24)
```

Sequenzfunktionen

Bei einigen Operationen ist die Abfolge (Sequenz) der Ereignisse von Bedeutung. Die Anwendung ermöglicht die Arbeit mit folgenden Datensatzsequenzen:

- Sequenzen und Zeitreihen
- Sequenzfunktionen
- Datensatzindizierung
- Durchschnittsbildung, Summierung und Vergleich von Werten
- Überwachen von Änderungen – Differenzierung
- @SINCE
- Offest-Werte
- Zusätzliche Sequenzfunktionen

Bei vielen Anwendungen kann jeder Datensatz, der einen Stream durchläuft, als Einzelfall betrachtet werden, der von allen anderen Fällen unabhängig ist. In solchen Situationen ist die Reihenfolge der Datensätze normalerweise nicht von Bedeutung.

Bei einigen Problemklassen jedoch ist die Datensatzsequenz höchst wichtig. Dies ist üblicherweise bei Zeitreihen der Fall, bei denen die Sequenz der Datensätze eine geordnete Sequenz von Ereignissen oder Vorkommen darstellt. Jeder Datensatz stellt einen "Schnappschuss" zu einem bestimmten Zeitpunkt dar; ein Großteil der aussagekräftigsten Informationen sind jedoch möglicherweise nicht Augenblickswerten enthalten, sondern in der Art und Weise, wie sich diese Werte im Laufe der Zeit ändern und verhalten.

Der entscheidende Parameter muss natürlich nicht die Zeit sein. So könnten die Datensätze Analysen darstellen, die in Abstanden entlang einer Linie durchgeführt werden. Doch auch hier gelten dieselben Prinzipien.

Sequenz- und Sonderfunktionen lassen sich sofort an folgenden Merkmalen erkennen:

- Sie enthalten alle das Präfix @.
- Die Namen dieser Funktionen in Großbuchstaben.

Sequenzfunktionen können sich auf den derzeit von einem Knoten verarbeiteten Datensatz, auf die Datensätze, die einen Knoten bereits durchlaufen haben, und sogar, in einem Fall, auf Datensätze beziehen, die einen Knoten noch durchlaufen müssen. Sequenzfunktionen können nach Belieben

mit anderen Komponenten von CLEM-Ausdrücken gemischt werden; bei einigen gelten jedoch Einschränkungen in Bezug auf die verwendeten Argumente.

Beispiele

Es kann interessant sein zu wissen, wie lange es zurückliegt, dass ein bestimmtes Ereignis eintrat oder eine Bedingung wahr war. Dies kann mit der Funktion @SINCE ermittelt werden. Beispiel:

```
@SINCE(Einkommen > Ausgaben)
```

Diese Funktion ergibt das Offset des letzten Datensatzes, bei dem diese Bedingung erfüllt war – also wie viele Datensätze vor dem aktuellen Datensatz die Bedingung wahr war. Wenn die Bedingung niemals wahr war, gibt @SINCE den Wert @INDEX + 1 zurück.

Manchmal möchte man sich in dem von @SINCE verwendeten Ausdruck auf einen Wert des aktuellen Datensatzes beziehen. Dies ist mit der Funktion @THIS möglich, die angibt, dass ein Feldname immer für den aktuellen Datensatz gilt. Um das Offset des letzten Datensatzes zu ermitteln, bei dem der Wert des Felds Concentration mehr als zweimal so hoch war wie der des aktuellen Datensatzes, können Sie folgende Funktion verwenden:

```
@SINCE(Konzentration > 2 * @THIS(Konzentration))
```

In einigen Fällen ist die für @SINCE angegebene Bedingung beim aktuellen Datensatz per definitionem wahr. Beispiel:

```
@SINCE(ID == @THIS(ID))
```

Aus diesem Grund wertet @SINCE die Bedingung nicht für den aktuellen Datensatz aus. Mit einer ähnlichen Funktion, @SINCE0, können Sie die Bedingung für den aktuellen Datensatz und die vorangegangenen auswerten; wenn die Bedingung im aktuellen Datensatz wahr ist, ergibt @SINCE0 den Wert 0.

Hinweis: Die @-Funktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
MEAN(FIELD)	*Reelle Zahl*	Ergibt den mittleren Durchschnitt der Werte für das angegebene *FELD* bzw. die angegeben *FELDER*.
@MEAN(FIELD, EXPR)	*Reelle Zahl*	Ergibt den mittleren Durchschnitt der Werte für *FELD* in den letzten *AUSDR* Datensätzen, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich der Durchschnitt aus den bisher erhaltenen Datensätzen. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
@MEAN(FIELD, EXPR, INT)	*Reelle Zahl*	Ergibt den mittleren Durchschnitt der Werte für *FELD* in den letzten *AUSDR* Datensätzen, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich der Durchschnitt aus den bisher erhaltenen Datensätzen. *GANZZ* gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Dies ist wesentlich effizienter als die Verwendung von nur zwei Argumenten.
@DIFF1(FIELD)	*Reelle Zahl*	Ergibt das erste Differenzial von *FELD1*. Das Format mit einem einzelnen Argument gibt also einfach die Differenz zwischen dem aktuellen Wert und dem früheren Wert des Felds zurück. Ergibt 0, wenn die relevanten vorhergehenden Datensätze nicht vorhanden sind.
@DIFF1(FIELD1, FIELD2)	*Reelle Zahl*	Das Format mit einem zwei Argumenten ergibt das erste Differenzial von *FELD1* in Bezug auf *FELD2*. Ergibt 0, wenn die relevanten vorhergehenden Datensätze nicht vorhanden sind.
@DIFF2(FIELD)	*Reelle Zahl*	Ergibt das zweite Differenzial von *FELD1*. Das Format mit einem einzelnen Argument gibt also einfach die Differenz zwischen dem aktuellen Wert und dem früheren Wert des Felds zurück. Ergibt 0, wenn die relevanten vorhergehenden Datensätze nicht vorhanden sind.
@DIFF2(FIELD1, FIELD2)	*Reelle Zahl*	Das Format mit einem zwei Argumenten ergibt das erste Differenzial von *FELD1* in Bezug auf *FELD2*. Ergibt 0, wenn die relevanten vorhergehenden Datensätze nicht vorhanden sind.
@INDEX	*Ganzzahl*	Ergibt den Index des aktuellen Datensatzes. Indizes werden den Datensätzen zugewiesen, wenn sie am aktuellen Knoten ankommen. Der erste Datensatz erhält den Index 1. Dieser Index wird für jeden nachfolgenden Datensatz um den Wert 1 erhöht.
@LAST_NON_BLANK(FIELD)	*Jede*	Ergibt den letzten nichtleeren Wert für *FELD*, wie in einem aufwärts liegenden Quellen- oder Typknoten definiert. Wenn in den bisher gelesenen Datensätzen keine nichtleeren Werte für *FELD* vorhanden sind, wird $null$ zurückgegeben. Leerwerte (auch als benutzerdefiniert fehlende Werte bezeichnet) können für jedes Feld einzeln definiert werden.
@MAX(FIELD)	*Number*	Ergibt den Höchstwert für das angegebene *FELD*.
@MAX(FIELD, EXPR)	*Number*	Ergibt den Höchstwert für *FELD* in den letzten *AUSDR* Datensätzen, die bisher erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt.

Funktion	Ergebnis	Beschreibung
@MAX(FIELD, EXPR, INT)	*Number*	Ergibt den Höchstwert für *FELD* in den letzten *AUSDR* Datensätzen, die bisher erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich der Höchstwert aus den bisher erhaltenen Datensätzen. *GANZZ* gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Dies ist wesentlich effizienter als die Verwendung von nur zwei Argumenten.
@MIN(FIELD)	*Number*	Ergibt den Mindestwert für das angegebene *FELD*.
@MIN(FIELD, EXPR)	*Number*	Ergibt den Mindestwert für *FELD* in den letzten *AUSDR* Datensätzen, die bisher erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt.
@MIN(FIELD, EXPR, INT)	*Number*	Ergibt den Mindestwert für *FELD* in den letzten *AUSDR* Datensätzen, die bisher erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich der Mindestwert aus den bisher erhaltenen Datensätzen. *GANZZ* gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Dies ist wesentlich effizienter als die Verwendung von nur zwei Argumenten.
@OFFSET(FIELD, EXPR)	*Jede*	Ergibt den Wert von *FELD* in dem Datensatz, der gegenüber dem aktuellen Datensatz ein Offset mit dem Wert von *AUSDR* aufweist. Ein positives Offset bezieht sich auf einen Datensatz, der den Knoten bereits durchlaufen hat, ein negatives Offset gibt eine "Vorschau" auf einen Datensatz an, der noch nicht angekommen ist. Beispiel: @OFFSET(Status, 1) ergibt den Wert des Felds Status im vorangegangenen Datensatz, während @OFFSET(Status, –4) vier Datensätze nach vorn in der Sequenz blickt (also auf Datensätze, die den Knoten noch nicht durchlaufen haben), um den Wert zu ermitteln. *Beachten Sie, dass ein negatives Offset (Vorschau-Offset) als Konstante angegeben sein muss.* Bei positiven Offsets gilt Folgendes: Bei *AUSDR* kann es sich auch um einen beliebigen CLEM-Ausdruck handeln, der für den aktuellen Datensatz ausgewertet wird, um das Offset zu erhalten. In diesem Fall sollte die Version dieser Funktion mit drei Argumenten die Leistung verbessern (siehe nächste Funktion). Wenn der Ausdruck einen anderen Wert als eine nichtnegative ganze Zahl ergibt, wird ein Fehler ausgelöst. Berechnete Vorschau-Offsets sind also nicht zulässig. *Hinweis*: Eine selbstreferenzielle @OFFSET-Funktion kann keine Literalvorschau verwenden. In einem Füllerknoten beispielsweise kann der Wert von field1

Funktion	Ergebnis	Beschreibung
		nicht mithilfe eines Ausdrucks wie @OFFSET(field1,-2) ersetzt werden.
@OFFSET(FIELD, EXPR, INT)	*Jede*	Führt die gleiche Operation durch wie @OFFSET, allerdings mit einem zusätzlichen dritten Argument, *GANZZ*, das die maximale Anzahl der zurückliegenden Werte angibt, die betrachtet werden sollen. Wenn das Offset aus einem Ausdruck berechnet wird, sollte das dritte Argument zu einer Verbesserung der Leistung führen. In einem Ausdruck wie @OFFSET(Foo, Month, 12) beispielsweise wird das System angewiesen, nur die letzten 12 Werte von Foo zu behalten. Andernfalls müsste sicherheitshalber jeder Wert gespeichert werden. In Fällen, in denen es sich beim Offset-Wert um eine Konstante handelt – einschließlich der negativen "Vorschau"-Offsets, die konstant sein müssen – ist das dritte Argument sinnlos, weshalb hier die Version dieser Funktion mit zwei Argumenten verwendet werden sollte. Siehe auch den Hinweis zu selbstreferenziellen Funktionen für die Version mit zwei Argumenten weiter oben.
@SDEV(FIELD)	*Reelle Zahl*	Ergibt die Standardabweichung der Werte für das angegebene *FELD* bzw. die angegebenen *FELDER*.
@SDEV(FIELD, EXPR)	*Reelle Zahl*	Gibt die Standardabweichung der Werte für *FELD* in den letzten *AUSDR* Datensätzen zurück, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich die Standardabweichung aus allen bisher erhaltenen Datensätzen.
@SDEV(FIELD, EXPR, INT)	*Reelle Zahl*	Gibt die Standardabweichung der Werte für *FELD* in den letzten *AUSDR* Datensätzen zurück, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich die Standardabweichung aus allen bisher erhaltenen Datensätzen. *GANZZ* gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Dies ist wesentlich effizienter als die Verwendung von nur zwei Argumenten.
@SINCE(EXPR)	*Jede*	Gibt an, wie viele Datensätze den Knoten durchlaufen haben, seit *AUSDR*, ein beliebiger CLEM-Ausdruck, wahr war.
@SINCE(EXPR, INT)	*Jede*	Das zweite Argument, *GANZZ*, gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Wenn *AUSDR* niemals wahr war, ist *GANZZ* gleich @INDEX+1.
@SINCE0(EXPR)	*Jede*	Berücksichtigt den aktuellen Datensatz, was bei @SINCE nicht der Fall ist; @SINCE0 ergibt 0, wenn *AUSDR* für den aktuellen Datensatz wahr ist.

Funktion	Ergebnis	Beschreibung
@SINCE0(EXPR, INT)	*Jede*	Das zweite Argument, *GANZZ*, gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen.
@SUM(FIELD)	*Number*	Ergibt die Summe der Werte für das angegebene *FELD* bzw. die angegeben *FELDER*.
@SUM(FIELD, EXPR)	*Number*	Ergibt die Summe der Werte für *FELD* in den letzten *AUSDR* Datensätzen, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich die Summe aus den bisher erhaltenen Datensätzen.
@SUM(FIELD, EXPR, INT)	*Number*	Ergibt die Summe der Werte für *FELD* in den letzten *AUSDR* Datensätzen, die vom aktuellen Knoten erhalten wurden, einschließlich des aktuellen Datensatzes. *FELD* muss der Name eines numerischen Felds sein. *AUSDR* kann jeder Ausdruck sein, dessen Auswertung eine ganze Zahl größer als 0 ergibt. Wenn *AUSDR* weggelassen wird oder die Anzahl der bisher erhaltenen Datensätze übersteigt, ergibt sich die Summe aus den bisher erhaltenen Datensätzen. *GANZZ* gibt die maximale Anzahl von zurückliegenden Werten an, die betrachtet werden sollen. Dies ist wesentlich effizienter als die Verwendung von nur zwei Argumenten.
@THIS(FIELD)	*Jede*	Ergibt den Wert des Felds namens *FELD* im aktuellen Datensatz. Wird nur bei @SINCE-Ausdrücken verwendet.

Globale Funktionen

Die Funktionen @MEAN,@SUM, @MIN, @MAX und @SDEV verarbeiten maximal alle gelesenen Datensätzen bis einschließlich dem aktuellen. In einigen Fällen ist es jedoch von Vorteil, in der Lage zu sein, die Daten im aktuellen Datensatz mit Werten im gesamten Daten-Set zu vergleichen. Mit einem Globalwerteknoten können Sie Werte im gesamten Daten-Set generieren, auf die Sie dann mit den globalen Funktionen in einem CLEM-Ausdruck zugreifen können.

Beispiel:

```
@GLOBAL_MAX(Alter)
```

ergibt den höchsten Wert von Age im Daten-Set, während der Ausdruck

```
(Wert - @GLOBAL_MEAN(Wert)) / @GLOBAL_SDEV(Wert)
```

die Differenz zwischen dem Value dieses Datensatzes und dem globalen Mittelwert als Anzahl der Standardabweichungen angibt. Globale Werte können nur verwendet werden, nachdem sie von einem Globalwerteknoten berechnet wurden. Alle aktuellen Globalwerte können durch Klicken auf die Schaltfläche zum Löschen der Globalwerte auf der Registerkarte “Globalwerte” im Dialogfeld “Stream-Eigenschaften” entfernt werden.

Hinweis: Die @-Funktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
@GLOBAL_MAX(FIELD)	*Number*	Ergibt den Höchstwert für *FELD* im gesamten Daten-Set, das zuvor durch einen Globalwerteknoten erstellt wurde. *FELD* muss der Name eines numerischen Felds sein. Wenn der entsprechende Globalwert nicht festgelegt wurde, wird ein Fehler ausgegeben. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
@GLOBAL_MIN(FIELD)	*Number*	Ergibt den Höchstwert für *FELD* im gesamten Daten-Set, das zuvor durch einen Globalwerteknoten generiert wurde. *FELD* muss der Name eines numerischen Felds sein. Wenn der entsprechende Globalwert nicht festgelegt wurde, wird ein Fehler ausgegeben.
@GLOBAL_SDEV(FIELD)	*Number*	Ergibt die Standardabweichung der Werte für *FELD* im gesamten Daten-Set, das zuvor durch einen Globalwerteknoten generiert wurde. *FELD* muss der Name eines numerischen Felds sein. Wenn der entsprechende Globalwert nicht festgelegt wurde, wird ein Fehler ausgegeben.
@GLOBAL_MEAN(FIELD)	*Number*	Ergibt den mittleren Durchschnitt der Werte für *FELD* im gesamten Daten-Set, das zuvor durch einen Globalwerteknoten generiert wurde. *FELD* muss der Name eines numerischen Felds sein. Wenn der entsprechende Globalwert nicht festgelegt wurde, wird ein Fehler ausgegeben.
@GLOBAL_SUM(FIELD)	*Number*	Ergibt die Summe der Werte für *FELD* im gesamten Daten-Set, das zuvor durch einen Globalwerteknoten generiert wurde. *FELD* muss der Name eines numerischen Felds sein. Wenn der entsprechende Globalwert nicht festgelegt wurde, wird ein Fehler ausgegeben.

Funktionen zum Umgang mit Leerstellen und Nullwerten

Mit CLEM können Sie angeben, dass bestimmte Werte in einem Feld als “Leerstellen” oder fehlende Werte betrachtet werden sollen. Die nachstehenden Funktionen können auf Leerstellen angewendet werden.

Hinweis: Die @-Funktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
@BLANK(FIELD)	*Boolesch*	Ergibt den Wert "Wahr" für alle Datensätze, deren Werte gemäß den Regeln zum Umgang mit Leerstellen, die in einem weiter oben im Stream gelegenen Typknoten oder Quellenknoten (Registerkarte "Typen") festgelegt wurden, Leerstellen sind. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
@LAST_NON_BLANK(FIELD)	*Jede*	Ergibt den letzten nichtleeren Wert für *FELD*, wie in einem aufwärts liegenden Quellen- oder Typknoten definiert. Wenn in den bisher gelesenen Datensätzen keine nichtleeren Werte für *FELD* vorhanden sind, wird $null$ zurückgegeben. Leerwerte (auch als benutzerdefiniert fehlende Werte bezeichnet) können für jedes Feld einzeln definiert werden.
@NULL(FIELD)	*Boolesch*	Ergibt den Wert "Wahr", wenn der Wert von *FELD* der systemdefiniert fehlende Wert $null$. ist. Ergibt den Wert "Falsch" für alle anderen Werte, einschließlich benutzerdefinierten fehlenden Werten. Wenn Sie nach beiden suchen möchten, verwenden Sie @BLANK(FIELD) und @NULL(FIELD).
undef	*Jede*	Wird allgemein in CLEM zur Eingabe eines $null$-Werts verwendet – beispielsweise, um Leerwerte im Füllerknoten mit Nullen aufzufüllen.

Mithilfe des Füllerknotens können leere Felder "aufgefüllt" werden. Bei Füller- und Ableitungsknoten (nur beim Mehrfachmodus) bezieht sich die CLEM-Sonderfunktion @FIELD auf die aktuell untersuchten Felder.

Sonderfelder

Sonderfunktionen dienen zur Kennzeichnung der speziellen untersuchten Felder oder zum Erzeugen einer Felderliste als Eingabe. Wenn Sie beispielsweise mehrere Felder gleichzeitig ableiten, sollten Sie mit @FIELD angeben, dass diese Ableitungsaktion für die ausgewählten Felder durchgeführt werden soll. Mit dem Ausdruck log(@FIELD) wird ein neues Log-Feld für jedes ausgewählte Feld abgeleitet.

Hinweis: Die @-Funktionen können nicht aus Skripts heraus aufgerufen werden. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.

Funktion	Ergebnis	Beschreibung
@FIELD	*Jede*	Führt eine Aktion für alle im Ausdruckskontext angegebenen Felder durch. Beachten Sie, dass diese Funktion nicht über ein Skript aufgerufen werden kann. Für weitere Informationen siehe Thema CLEM-Ausdrücke in Skripts in Kapitel 3 auf S. 30.
@TARGET	*Jede*	Wenn ein CLEM-Ausdruck in einer benutzerdefinierten Analysefunktion verwendet wird, steht @TARGET für das Zielfeld bzw. den “korrekten Wert” für das analysierte Paar aus Zielwert und vorhergesagtem Wert. Diese Funktion wird häufig in Analyseknoten verwendet.
@PREDICTED	*Jede*	Wenn ein CLEM-Ausdruck in einer benutzerdefinierten Analysefunktion verwendet wird, steht @PREDICTED für den vorhergesagten Wert des analysierten Paars aus Zielwert und vorhergesagtem Wert. Diese Funktion wird häufig in Analyseknoten verwendet.
@PARTITION_FIELD	*Jede*	Ersetzt den Namen des aktuellen Partitionsfelds.
@TRAINING_PARTITION	*Jede*	Ergibt den Wert der aktuellen Trainingspartition. Beispiel: Zur Auswahl von Trainingsdatensätzen mit einem Auswahlknoten verwenden Sie den CLEM-Ausdruck: @PARTITION_FIELD = @TRAINING_PARTITION Dadurch wird gewährleistet, dass der Auswahlknoten immer funktioniert, unabhängig davon, welche Werte zur Darstellung der einzelnen Partitionen in den Daten verwendet werden.
@TESTING_PARTITION	*Jede*	Ergibt den Wert der aktuellen Testpartition.
@VALIDATION_PARTITION	*Jede*	Ergibt den Wert der aktuellen Validierungspartition.
@FIELDS_BETWEEN(start, end)	*Jede*	Ergibt die Liste der Feldnamen, die zwischen dem angegebenen Anfangs- und Endfeld (einschließlich dieser Felder) liegen, und zwar auf der Grundlage der natürlichen Reihenfolge der Felder in den Daten, also gemäß der Einfügereihenfolge. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Funktion	Ergebnis	Beschreibung
@FIELDS_MATCHING(pattern)	*Jede*	Ergibt eine Liste der Feldnamen, die mit dem angegebenen Muster übereinstimmen. Das Fragezeichen (?) als Platzhalterzeichen kann die Stelle von genau einem beliebigen Zeichen einnehmen, das Sternchen (*) entspricht keinem, einem oder mehreren Zeichen. Soll ein Fragezeichen oder ein Sternchen nicht als Platzhalter fungieren, sondern bei der Suche berücksichtigt werden, geben Sie den umgekehrten Schrägstrich (\) als Escape-Zeichen ein. Für weitere Informationen siehe Thema Zusammenfassen mehrerer Felder in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.
@MULTI_RESPONSE_SET	*Jede*	Ergibt die Liste der Felder im benannten Mehrfachantworten-Set. Für weitere Informationen siehe Thema Arbeiten mit Mehrfachantwortdaten in Kapitel 7 in *IBM SPSS Modeler 15 Benutzerhandbuch*.

Teil II: Eigenschaftsverweis

Kapitel 9

Eigenschaftsverweis

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 Stream-Operationen auf hoher Ebene wie Cachen oder das Verhalten von Superknoten. Der Zugriff auf Eigenschaften erfolgt über die Standardbenutzeroberfläche (z. B. beim Öffnen eines Dialogfelds zum Bearbeiten von Optionen für einen Knoten). Eigenschaften können auf vielfältige Weise verwendet werden.

- Eigenschaften lassen sich mit Skripts ändern, wie in diesem Abschnitt beschrieben. Weitere Informationen finden Sie unter Syntax für Eigenschaften.
- Knoteneigenschaften können in Superknotenparametern verwendet werden. Für weitere Informationen siehe Thema Verwenden von Superknotenparametern zum Zugriff auf Knoteneigenschaften in Kapitel 9 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.
- Knoteneigenschaften können auch als Teil einer Befehlszeilenoption (mit dem Flag -P) beim Starten von IBM® SPSS® Modeler verwendet werden.

Im Zusammenhang mit Skripts in SPSS Modeler werden Knoten- und Stream-Eigenschaften häufig als **Slot-Parameter** bezeichnet. In diesem Handbuch werden sie als Knoten- oder Stream-Eigenschaften beschrieben.

Weitere Informationen zur Skriptsprache finden Sie hier: Kapitel 3.

Syntax für Eigenschaften

Eigenschaften müssen folgende Syntaxstruktur aufweisen:

```
NAME:TYP.EIGENSCHAFT
```

Dabei ist NAME der Name eines Knotens und TYPE sein Typ (z. B. multiplotnode oder derivenode). Sie können entweder auf NAME oder auf TYPE verzichten, Sie müssen jedoch mindestens einen Wert angeben. PROPERTY ist der Name des Knotens oder Stream-Parameters, auf den sich Ihr Ausdruck bezieht. Beispiel: Mit folgender Syntax wird das Feld *Alter* aus Daten weiter unten im Stream gefiltert:

```
set mynode:filternode.include.Alter = false
```

Um einen benutzerdefinierten Wert für einen der Parameter (NAME, TYPE oder PROPERTY) zu verwenden, legen Sie zunächst den Wert in einer Anweisung fest, beispielsweise set derive.new_name = mynewfield. Anschließend können Sie den Wert mynewfield als Parameter

verwenden, indem Sie das Zeichen ^ voranstellen. Sie können den Typ für den oben genannten Ableitungsknoten beispielsweise mit folgender Syntax festlegen:

```
set ^mynewfield.result_type = "Conditional"
```

Alle in IBM® SPSS® Modeler verwendeten Knoten können im Parameter TYPE der Syntax NAME:TYPE.PROPERTY angegeben werden.

Strukturierte Eigenschaften

Es gibt zwei Möglichkeiten, wie Skripts strukturierte Eigenschaften verwenden können, um eine größere Klarheit bei der Analyse zu erreichen:

- Den Namen der Eigenschaften für komplexe Knoten, wie Typ-, Filter- und Balancierungsknoten, strukturieren.
- Ein Format zum Festlegen mehrerer Eigenschaften gleichzeitig angeben.

Strukturieren komplexer Benutzeroberflächen

Die Skripts für Knoten mit Tabellen und anderen komplexen Benutzeroberflächen (z. B. Typ-, Filter- und Balancierungsknoten) müssen eine bestimmte Struktur besitzen, damit die Analyse ordnungsgemäß erfolgt. Für diese strukturierten Eigenschaften ist ein komplexerer Name als für einen einzelnen Bezeichner erforderlich. Innerhalb eines Filterknotens wird beispielsweise jedes verfügbare Feld (auf der aufwärts im Stream liegenden Seite) ein- oder ausgeschaltet. Um auf diese Information zurückzugreifen, speichert der Filterknoten ein Informationselement pro Feld (ob das Feld jeweils wahr oder falsch ist) und der Zugriff und die Aktualisierung dieser Elemente erfolgt durch eine einzige Eigenschaft namens **Feld**. Dieses kann den Wert true oder false besitzen (bzw. zugewiesen bekommen). Angenommen, ein Filterknoten namens mynode weist (auf der abwärts im Stream liegenden Seite) ein Feld mit dem Namen *Alter* auf. Um dieses auszuschalten, legen Sie für die Eigenschaft mynode.include.Age wie folgt den Wert false fest:

```
set mynode.include.Alter = false
```

Strukturieren zum Festlegen mehrerer Eigenschaften

Für mehrere Knoten können Sie mehr als einen Knoten oder eine Stream-Eigenschaft gleichzeitig zuweisen. Dies wird als **Multiset-Befehl** oder **Block-Set** bezeichnet. Für weitere Informationen siehe Thema Befehl “set” in Kapitel 4 auf S. 35.

In manchen Fällen kann eine strukturierte Eigenschaft äußerst komplex sein. Der umgekehrte Schrägstrich (\) kann als Fortsetzungszeichen verwendet werden, um die Argumente übersichtlich aufzuführen. Ein Beispiel lautet folgendermaßen:

```
mynode:sortnode.keys = [{ 'K' Descending} \
                { 'Alter' Ascending}\
                { 'Na' Descending }]
```

Ein weiterer Vorteil strukturierter Eigenschaften besteht darin, dass mehrere Eigenschaften an einem Knoten festgelegt werden können, bevor der Knoten stabil ist. Standardmäßig legt ein Multiset alle Eigenschaften in einem Block fest, bevor je nach individueller

Eigenschafteneinstellung Vorgänge ausgeführt werden. Beispiel: Wenn die Feldeigenschaften beim Definieren eines Knotes “Datei (fest)” in zwei Schritten festgelegt werden, treten Fehler auf, da der Knoten erst konsistent ist, wenn beide Einstellungen gültig sind. Durch Definieren der Eigenschaften als Multiset wird dieses Problem umgangen, indem beide Eigenschaften festgelegt werden, bevor das Datenmodell aktualisiert wird.

Abkürzungen

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

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

Beispiele für Knoten- und Stream-Eigenschaften

Mit IBM® SPSS® Modeler können Knoten- und Stream-Eigenschaften auf vielfältige Weise verwendet werden. Meistens werden sie in einem Skript, entweder einem **eigenständigen Skript** zur Automatisierung mehrerer Streams oder Operationen oder einem **Stream-Skript** 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 SPSS Modeler verwendet werden. Mit dem Argument -p als Teil des Befehlszeilenaufrufs können Sie eine Stream-Eigenschaft verwenden, um eine Einstellung im Stream zu ändern.

s.max_size	Bezieht sich auf die Eigenschaft max_size des Knotens s.
s:samplenode.max_size	Bezieht sich auf die Eigenschaft max_size des Knotens s, bei dem es sich um einen Stichprobenknoten handeln muss.
:samplenode.max_size	Bezieht sich auf die Eigenschaft max_size des Stichprobenknotens im aktuellen Stream (es darf nur ein Stichprobenknoten vorhanden sein).
s:sample.max_size	Bezieht sich auf die Eigenschaft max_size des Knotens s, bei dem es sich um einen Stichprobenknoten handeln muss.

t.direction.Age	Bezieht sich auf die Rolle des Felds *Alter* im Typknoten t.
:.max_size	*** NICHT ZULÄSSIG *** Sie müssen entweder den Knotennamen oder den Knotentyp angeben.

Das Beispiel s:sample.max_size veranschaulicht, dass Knotentypen nicht vollständig ausgeschrieben werden müssen.

Im Beispiel t.direction.Age wird deutlich, dass einige Slot-Namen selbst strukturiert werden können, und zwar wenn die Attribute eines Knotens komplexer sind als nur individuelle Slots mit individuellen Werten. Solche Slots werden als **strukturierte** oder **komplexe** Eigenschaften bezeichnet.

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 – Zahl, Flag oder Zeichenkette – handeln. In diesem Fall wird für die Einstellungen für die Eigenschaft der richtige Typ erzwungen. Wenn sie nicht erzwungen werden können, wird ein Fehler ausgegeben. Alternativ kann der Eigenschaftsverweis den Bereich zulässiger Werte, wie Discard, PairAndDiscard und IncludeAsText, angeben. In diesem Fall tritt bei Verwendung eines anderen Werts ein Fehler auf. Flag-Eigenschaften sollten anhand der Werte true und false gelesen bzw. festgelegt werden. (Variationen mit Off, OFF, off, No, NO, no, n, N, f, F, false, False, FALSE oder 0 werden beim Festlegen von Werten ebenfalls erkannt, können jedoch beim Lesen von Eigenschaftswerten in manchen Fällen zu Fehlern führen. Alle anderen Werte werden als wahr betrachtet. Durch die durchgängige Verwendung von true und false können Verwechslungen vermieden 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).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
use_custom_name	*Flag*	
name	*string*	Schreibgeschützte Eigenschaft zum Lesen des Namens (entweder automatisch oder benutzerdefiniert) für einen Knoten im Zeichenbereich.
custom_name	*string*	Legt einen benutzerdefinierten Namen für den Knoten fest.
tooltip	*string*	
annotation	*string*	
keywords	*string*	Strukturierter Slot, der eine Liste der mit dem Objekt verknüpften Schlüsselwörter angibt (Beispiel: ["Keyword1" "Keyword2"]).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
cache_enabled	*Flag*	
node_type	source_supernode process_supernode terminal_supernode alle Knotennamen, wie für Skripts 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 userinputnode oder filternode angeben.

Superknotenspezifische Eigenschaften werden wie alle anderen Knoten separat erörtert. Für weitere Informationen siehe Thema Superknoten-Eigenschaften in Kapitel 22 auf S. 326.

Kapitel
10

Stream-Eigenschaften

Verschiedene Stream-Eigenschaften können durch Skripts gesteuert werden. Um Stream-Eigenschaften zu referenzieren, müssen Sie eine bestimmte Stream-Variable verwenden, die durch das Zeichen ^ vor dem Stream gekennzeichnet ist:

```
set ^stream.execute_method = Script
```

Beispiel

Die Knoteneigenschaft dient zur Referenzierung der Knoten im aktuellen Stream. Das folgende Stream-Skript ist ein Beispiel:

```
var listofnodes
var thenode
set listofnodes = ^stream.nodes

set ^stream.annotation = ^stream.annotation >< "\n\nDieser Stream heißt \"" >< ^stream.name >
< "\" und enthält/ die folgenden Knoten\n"

for thenode in listofnodes
set ^stream.annotation = ^stream.annotation >< "\n" >< ^thenode.node_type
endfor
```

Im oben genannten Beispiel wird anhand der Knoteneigenschaft eine Liste aller Knoten im Stream erstellt und diese Liste in die Stream-Anmerkungen geschrieben. Die erzeugte Anmerkung sieht wie folgt aus:

```
Dieser Stream wird als "druglearn" bezeichnet und enthält folgende Knoten:

derivenode
neuralnetworknode
variablefilenode
typenode
c50node
filternode
```

In der folgenden Tabelle werden die Stream-Eigenschaften beschrieben.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
execute_method	Normal Script	
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" TAG MONAT "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	*number*	
date_2digit_baseline	*number*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	*Flag*	
import_datetime_as_string	*Flag*	
decimal_places	*number*	

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
decimal_symbol	Default Period Comma	
angles_in_radians	*Flag*	
use_max_set_size	*Flag*	
max_set_size	*number*	
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	*Flag*	Dient zur automatischen Aktualisierung von Quellenknoten nach Ausführung des Streams.
script	*string*	
annotation	*string*	Beispiel: set ^stream.annotation = "something interesting"
name	*string*	Beispiel: set x = ^stream.name *Hinweis*: Diese Eigenschaft ist schreibgeschützt. Wenn Sie den Namen eines Streams ändern möchten, speichern Sie ihn unter einem anderen Namen.
parameters		Diese Eigenschaft dient zur Aktualisierung von Stream-Parametern innerhalb eines Standalone-Skripts. Beispiel: set ^stream.parameters.height = 23
nodes		Details finden Sie weiter unten.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*boolean*	
stream_rewriting_maximise_sql	*boolean*	
stream_rewriting_optimise_clem_execution	*boolean*	
stream_rewriting_optimise_syntax_execution	*boolean*	
enable_parallelism	*boolean*	
sql_generation	*boolean*	
database_caching	*boolean*	
sql_logging	*boolean*	
sql_generation_logging	*boolean*	
sql_log_native	*boolean*	
sql_log_prettyprint	*boolean*	
record_count_suppress_input	*boolean*	
record_count_feedback_interval	*integer*	

Kapitel 11

Projekteigenschaften

Es steht eine Reihe von Eigenschaften zur Skripterstellung mit Projekten zur Verfügung.

Beispiel

```
load project "C:/clemdata/DrugData.cpj"
set ^project.summary="Erste Modellierungsarbeiten an den aktuellen Medikamentendaten."
set ^project.ordering=NameAddedType
execute_project
```

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
summary	*string*	Die Projektübersicht – eine abgekürzte Version der Anmerkungen.
title	*string*	Der Titel des Berichts.
author	*string*	Der Autor des Berichts.
structure	Phase Class	Bestimmt, wie das Projekt geordnet ist – nach Data-Mining-Phase oder nach Objekttyp (Klasse).
include_mode	IncludedItems ExcludedItems AllItems	Bestimmt, welche Elemente in den Projektbericht aufgenommen werden sollen.
select_mode	AllItems RecentItems OldItems	Bestimmt (nach Alter), welche Elemente in den Projektbericht aufgenommen werden sollen.
recent_item_limit	*integer*	Wird verwendet, wenn select_mode auf RecentItems gesetzt ist.
old_item_limit	*integer*	Wird verwendet, wenn select_mode auf OldItems gesetzt ist.
ordering	TypeNameAdded TypeAddedName NameAddedType AddedNameType	Bestimmt die Reihenfolge, in der die Elemente im Bericht aufgeführt werden.

Kapitel
12

Quellenknoten – Eigenschaften

Allgemeine Eigenschaften von Quellenknoten

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

Beispiel

```
create variablefilenode
set :variablefilenode.full_filename = "$CLEO_DEMOS/DRUG4n"
set :variablefilenode.use_custom_values.Age = True
set :variablefilenode.direction.Age = Input
set :variablefilenode.type.Age = Range
#storage is read only
set :variablefilenode.check.Age = None
set :variablefilenode.values.Age = [1 100]
```

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
direction	Input Target Both None Partition Split Frequency RecordID	Schlüsseleigenschaft für Feldrollen. Format: NODE.direction.FIELDNAME *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 *Specify* (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. Format: NODE.type.FIELDNAME
storage	Unknown String Integer Real Time Date Timestamp	Schreibgeschützte Schlüsseleigenschaft für Feldspeichertyp. Format: NODE.storage.FIELDNAME
check	None Nullify Coerce Discard Warn Abort	Schlüsseleigenschaft für das Überprüfen von Feldtyp und Bereich. Format: NODE.check.FIELDNAME

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
values	[*Wert Wert*]	Bei einem stetigen Feld (Bereich) ist der erste Wert das Minimum und der letzte das Maximum. Geben Sie für nominale (Set-) Felder alle Werte an. Bei Flag-Feldern 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. Format: NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	Bestimmt, wie Werte bei der nächsten Datenübergabe für ein Feld festgelegt werden. Format: NODE.value_mode.FIELDNAME Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf *Specify* (Angeben) festlegen können. Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft values fest.
default_value_mode	Read Pass	Gibt die Standardmethode für das Festlegen von Werten für alle Felder an. Format: NODE.default_value_mode Beispiel: set mynode.default_value_mode = Pass Diese Einstellung kann mithilfe der Eigenschaft value_mode für bestimmte Felder überschrieben werden.
extend_values	*Flag*	Gilt, wenn value_mode auf *Read* (Lesen) festgelegt 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. Format: NODE.extend_values.FIELDNAME
value_labels	*string*	Wird zur Angabe eines Wertelabels verwendet. Beispiel: set :varfilenode.value_labels.Age = [{3 three}{5 five}] Beachten Sie, dass Werte zuerst angegeben werden müssen.
enable_missing	*Flag*	Bei Festlegung auf *T* wird die Verfolgung von fehlenden Werten für das Feld aktiviert. Format: NODE.enable_missing.FIELDNAME
missing_values	[*Wert Wert ...*]	Gibt Datenwerte an, die fehlende Daten kennzeichnen. Format: NODE.missing_values.FIELDNAME
range_missing	*Flag*	Wenn diese Eigenschaft auf *T* gesetzt ist, wird angegeben, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist. Format: NODE.range_missing.FIELDNAME
missing_lower	*string*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an. Format: NODE.missing_lower.FIELDNAME

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
missing_upper	*string*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an. Format: NODE.missing_upper.FIELDNAME
null_missing	*Flag*	Wenn diese Eigenschaft auf *T* gesetzt ist, werden Nullen (undefinierte Werte, die in der Software als $null$ angezeigt werden) als fehlende Werte betrachtet. Format: NODE.null_missing.FIELDNAME
whitespace_missing	*Flag*	Wenn diese Eigenschaft auf *T* festgelegt ist, werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet. Format: NODE.whitespace_missing.FIELDNAME
description	*string*	Dient zur Angabe einer Feldbeschriftung oder Beschreibung.
default_include	*Flag*	Schlüsseleigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden: NODE.default_include Beispiel: set mynode:filternode.default_include = false
include	*Flag*	Schlüsseleigenschaft, mit der bestimmt wird, ob einzelne Felder aufgenommen oder gefiltert werden: NODE.include.FIELDNAME. Beispiel: set mynode:filternode.include.Age = true
new_name	*string*	Beispiel: set mynode:filternode.new_name.'Age' = "years"

Eigenschaften von “cognosimportnode”

Der IBM Cognos BI-Quellenknoten importiert Daten aus Cognos BI-Datenbanken. Für weitere Informationen siehe Thema Importieren von Cognos-Daten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create cognosimportnode
set :cognosimportnode.cognos_connection = {'http://mycogsrv1:9300/p2pd/servlet/dispatch', true, "", "", ""}
set :cognosimportnode.cognos_package_name = '/Public Folders/GOSALES'
set :cognosimportnode.cognos_items = {"[GreatOutdoors].[BRANCH].[BRANCH_CODE]",
```

"[GreatOutdoors].[BRANCH].[COUNTRY_CODE]"}

cognosimportn-ode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
mode	Data Report	Gibt an, ob Cognos BI-Daten (Standard) oder Berichte importiert werden sollen.
cognos_connection	*{"field","field", ... ,"field"}*	Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: {"*Cognos_server_URL*", *login_mode*, "*namespace*", "*username*", "*password*"} Dabei gilt: *Cognos_server_URL* ist die URL des Cognos-Servers, der die Quelle enthält. *login_mode* gibt an, ob eine anonyme Anmeldung verwendet wird, und entspricht entweder true oder false; Wenn true festgelegt ist, sollten die folgenden Felder auf "" festgelegt werden. *namespace* gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. *username* und *password* sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.
cognos_package_name	*string*	Pfad und Name des Cognos-Pakets, von dem Sie Datenobjekte importieren, z. B.: /Public Folders/GOSALES Anmerkung: Nur normale Schrägstriche (/) sind gültig.
cognos_items	*{"field","field", ... ,"field"}*	Der Name eines oder mehrerer Datenobjekte, die importiert werden sollen. Das Format von *field* ist [*namespace*].[*query_subject*].[*query_item*] Beispiel: set :cognosimport.cognos_items = {"[Inventory (query)].[Inventory].[Opening inventory]", "[Inventory (query)].[Inventory].[Quantity shipped]", "[Inventory (query)].[Inventory].[Additions]", "[Inventory (query)].[Inventory].[Unit cost]", "[Inventory (query)].[Inventory].[Closing inventory]", "[Inventory (query)].[Inventory].[Average unit cost]"}
cognos_filters	*Feld*	Der Name eines oder mehrerer Filter, die vor dem Datenimport angewendet werden sollen. Beispiel: set :cognosimport.cognos_filters = {"[Inventory].[Filter].[MyFilter]"}

cognosimportn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
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 Zeichenkette steht in eckigen Klammern. Format: [{"*Param1*", "*Wert*"},…,{"*ParamN*", "*Wert*"}] Beispiel: set :cognosimport.cognos_data_parameters = [{"SexValue","F"},{"a","1"},{"b","1"}]
cognos_report_location	*Feld*	Der Cognos-Pfad eines Ordners bzw. Pakets, aus dem Berichte importiert werden sollen. Beispiel: /Public Folders/GOSALES Anmerkung: Nur normale Schrägstriche (/) sind gültig.
cognos_report_name	*Feld*	Der Pfad und Name innerhalb des Speicherorts eines zu importierenden Berichts. Beispiel: set :cognosimport.cognos_report_name = /Jimmy/Package/Drug4nPackage/3columns
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 Zeichenkette steht in eckigen Klammern. Format: [{"*Param1*", "*Wert*"},…,{"*ParamN*", "*Wert*"}] Beispiel: set :cognosimport.cognos_report_parameters = [{"SexValue","F"},{"a","1"},{"b","1"}]

Eigenschaften von "databasenode"

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. Für weitere Informationen siehe Thema Datenbankquellenknoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create databasenode
set :databasenode.mode = Table
set :databasenode.query = "SELECT * FROM drug4n"
set :databasenode.datasource = "Drug4n_db"
set :databasenode.username = "spss"
set :databasenode.password = "spss"
var test_e
set test_e = :databasenode.epassword
```

set :databasenode.tablename = ".Drug4n"

databasenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
mode	Table 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	*string*	Datenbankname (siehe auch Hinweis unten).
username	*string*	Datenbankverbindungsdetails (siehe auch Hinweis unten).
password	*string*	
epassword	*string*	Gibt ein kodiertes Passwort an, als Alternative zum festen Kodieren eines Passworts in einem Sktipt. Für weitere Informationen siehe Thema Erstellen eines verschlüsselten Passworts in Kapitel 5 auf S. 63.Diese Eigenschaft ist während der Ausführung schreibgeschützt.
tablename	*string*	Name der Tabelle, auf die Sie zugreifen wollen.
strip_spaces	None Left Right Both	Optionen zum Verwerfen von führenden und abschließenden Leerzeichen in Zeichenketten.
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	*string*	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 datasource, username und password eine einzige Datenquelleneigenschaft in folgendem Format:

databasenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
datasource	*string*	Format: {database_name,username,password[,true | false]} Der letzte Parameter ist für die Verwendung bei verschlüsselten Passwörtern gedacht. Wenn er auf true gesetzt ist, wird das Passwort vor der Nutzung verschlüsselt.

Beispiel

```
create databasenode
set :databasenode.mode = Table
set :databasenode.query = "SELECT * FROM drug4n"
set :databasenode.datasource = {"ORA 10gR2", user1, mypsw, true}
```

```
var test_e
set test_e = :databasenode.epassword
set :databasenode.tablename = ".Drug4n"
```

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

Eigenschaften von "datacollectionimportnode"

Der IBM® SPSS® Data Collection Data-Importknoten importiert Umfragedaten auf der Grundlage des von den Data Collection-Marktforschungsprodukten verwendeten IBM Corp. Data Model. Um diesen Knoten verwenden zu können, muss die Data Collection Data Library installiert sein. Für weitere Informationen siehe Thema Data Collection Knoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create datacollectionimportnode
set :datacollectionimportnode.metadata_name="mrQvDsc"
set :datacollectionimportnode.metadata_file="C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd"
set :datacollectionimportnode.casedata_name="mrQvDsc"
set :datacollectionimportnode.casedata_source_type=File
set :datacollectionimportnode.casedata_file="C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd"
set :datacollectionimportnode.import_system_variables = Common
set :datacollectionimportnode.import_multi_response = MultipleFlags
```

datacollectionimportn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
metadata_name	*string*	Der Name des MDSC. Der spezielle Wert DimensionsMDD gibt an, dass das standardmäßige 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	*string*	Name der Datei, in der die Metadaten gespeichert werden.

datacollectionimportn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
casedata_name	*string*	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	*string*	Wenn casedata_source_type den Wert *File* (Datei) aufweist, wird hier die Datei angegeben, die die Falldaten enthält.
casedata_folder	*string*	Wenn casedata_source_type den Wert *Folder* (Ordner) aufweist, wird hier der Ordner angegeben, der die Falldaten enthält.
casedata_udl_string	*string*	Wenn casedata_source_type den Wert *UDL* aufweist, wird hier die OLD-DB-Verbindungszeichenkette für die Datenquelle angegeben, die die Falldaten enthält.
casedata_dsn_string	*string*	Wenn casedata_source_type den Wert *DSN*, aufweist, wird hier die ODBC-Verbindungszeichenkette für die Datenquelle angegeben.
casedata_project	*string*	Beim Lesen von Falldaten aus einer 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	*string*	Wenn version_import_mode den Wert *Specify* (Angeben) aufweist, wird hier die Version der zu importierenden Falldaten festgelegt.
use_language	*string*	Gibt an, ob Beschriftungen einer bestimmten Sprache verwendet werden sollen.

datacollectionimportn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
language	*string*	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	*string*	Gibt an, ob ein bestimmter Kontext importiert werden sollte. Kontexte dienen dazu, die den Antworten zugeordnete Beschreibung zu variieren.
context	*string*	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	*string*	Gibt an, ob ein bestimmter Beschriftungstyp importiert werden sollte.
label_type	*string*	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	*string*	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	*string*	
import_system_variables	Common None All	Gibt an, welche Systemvariablen importiert werden sollen.
import_codes_variables	*Flag*	
import_sourcefile_variables	*Flag*	
import_multi_response	MultipleFlags Single	

Eigenschaften von “excelimportnode”

Der Excel-Importknoten importiert Daten aus einer beliebigen Version von Microsoft Excel. Es ist keine ODBC-Datenquelle erforderlich. Für weitere Informationen siehe Thema Excel-Quellenknoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
#To use a named range:
create excelimportnode
set :excelimportnode.excel_file_type = Excel2007
set :excelimportnode.full_filename = "C:/drug.xls"
```

```
set :excelimportnode.use_named_range = True
set :excelimportnode.named_range = "DRUG"
set :excelimportnode.read_field_names = True

#To use an explicit range:
create excelimportnode
set :excelimportnode.excel_file_type = Excel2007
set :excelimportnode.full_filename = "C:/drug.xls"
set :excelimportnode.worksheet_mode = Name
set :excelimportnode.worksheet_name = "Drug"
set :excelimportnode.explicit_range_start = A1
set :excelimportnode.explicit_range_end = F300
```

excelimportnode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
excel_file_type	Excel2003 Excel2007	
full_filename	*string*	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	*string*	
worksheet_mode	Index Name	Gibt an, ob das Arbeitsblatt durch Index oder durch Namen definiert wird.
worksheet_index	*integer*	Index des zu lesenden Arbeitsblattes, beginnend mit 0 für das erste Arbeitsblatt, 1 für das zweite usw.
worksheet_name	*string*	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	*string*	Wenn data_range_mode den Wert *ExplicitRange* aufweist, wird hier der Startpunkt des zu lesenden Bereichs angegeben.
explicit_range_end	*string*	
read_field_names	*Boolesch*	Gibt an, ob die erste Zeile im angegebenen Bereich als Feldnamen (Spaltennamen) verwendet werden soll.

Eigenschaften von “evimportnode”

Der Enterprise-Ansichts-Knoten erstellt eine Verbindung mit einem IBM SPSS Collaboration and Deployment Services Repository, was es Ihnen ermöglicht, Enterprise-Ansichts-Daten in einen Stream einzulesen und ein Modell in ein Szenario zu packen, auf das andere Benutzer über das Repository zugreifen können. Für weitere Informationen siehe Thema Enterprise-Ansichts-Knoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create evimportnode
set :evimportnode.connection = ['Training data','/Application views/Marketing','LATEST','Analytic',
    '/Data Providers/Marketing']
set :evimportnode.tablename = "cust1"
```

evimportnode -Eigenschaften	Datentyp	Eigenschaftsbeschreibung
connection	*Liste*	Strukturierte Eigenschaft – Liste der Parameter, die eine Enterprise-Ansichtsverbindung ergeben. Format: evimportnode.connection = [description,app_view_path, app_view_version_label,environ-ment,DPD_path]
tablename	*string*	Der Name einer Tabelle in der Application-Ansicht.

Eigenschaften von “fixedfilenode”

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. Für weitere Informationen siehe Thema Knoten “Datei (fest)” in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create fixedfilenode
set :fixedfilenode.full_filename = "$CLEO_DEMOS/DRUG4n"
set :fixedfilenode.record_len = 32
set :fixedfilenode.skip_header = 1
set :fixedfilenode.fields = [{'Age' 1 3} {'Sex' 5 7} {'BP' 9 10} {'Cholesterol' 12 22} {'Na' 24 25} {'K' 27 27} {'Drug' 29 32}]
set :fixedfilenode.decimal_symbol = Period
set :fixedfilenode.lines_to_scan = 30
```

fixedfilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
record_len	*number*	Legt die Zahl der Zeichen in jedem Datensatz fest.
line_oriented	*Flag*	Überspringt am Ende jedes Datensatzes das Zeilenwechselzeichen.

fixedfilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
decimal_symbol	Default Comma Period	Der Typ des in Ihrer Datenquelle verwendeten Dezimaltrennzeichens. Beispiel: set :fixedfilenode.decimal_symbol = Period
skip_header	*number*	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeilen an. Dies ist nützlich, um Spaltenkopfzeilen zu ignorieren.
auto_recognize_datetime	*Flag*	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quelldaten identifiziert werden.
lines_to_scan	*number*	Beispiel: set :fixedfilenode.lines_to_scan = 50.
fields	*Liste*	Strukturierte Eigenschaft. Format: fixedfilenode.fields = [{field start length} {field start length}]
full_filename	*string*	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.
strip_spaces	None Left Right Both	Verwirft beim Importieren führende und abschließende Leerzeichen in Zeichenketten.
invalid_char_mode	Discard Replace	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Kodierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
invalid_char_replacement	*string*	
use_custom_values	*Flag*	Schlüssel-Slot in folgendem Format: set :varfilenode.use_custom_values.Age = true
custom_storage	Unknown String Integer Real Time Date Timestamp	Schlüssel-Slot in folgendem Format: set :varfilenode.custom_storage.'Age' = "Real"
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY"	Diese Eigenschaft ist nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde. Beispiel: set:varfilenode.custom Schlüssel-Slot in folgendem Format: set :varfilenode.custom_date_ format.'LaunchDate' = "DDMMYY"

fixedfilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
	"DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
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. Schlüssel-Slot in folgendem Format: set :varfilenode.custom_time_format. 'Initialize' = "HHMM"
custom_decimal_symbol	*Feld*	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde. Schlüssel-Slot in folgendem Format: set :varfilenode.custom_decimal_ symbol.'Revenue' = "Comma"
encoding	StreamDefault SystemDefault "UTF-8"	Legt die Textkodierungsmethode fest.

Eigenschaften von "sasimportnode"

Der SAS-Importknoten importiert SAS-Daten in IBM® SPSS® Modeler. Für weitere Informationen siehe Thema SAS-Quellenknoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create sasimportnode
set :sasimportnode.format = Windows
set :sasimportnode.full_filename = "C:/data/retail.sas7bdat"
set :sasimportnode.member_name = "Test"
set :sasimportnode.read_formats = False
set :sasimportnode.full_format_filename = "Test"
```

```
set :sasimportnode.import_names = True
```

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

Eigenschaften von “statisticsimportnode”

Der IBM® SPSS® Statistics-Dateiknoten liest Daten aus dem Dateiformat *.sav* ein, das von SPSS Statistics verwendet wird, sowie in IBM® SPSS® Modeler gespeicherte Cache-Dateien, die ebenfalls dasselbe Format verwenden. Für weitere Informationen siehe Thema Statistikdateiknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie unter Eigenschaften von “statisticsimportnode” auf S. 322.

Eigenschaften von “userinputnode”

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 Test-Daten-Set für die Modellierung erstellen möchten. Für weitere Informationen siehe Thema Benutzereingabeknoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create userinputnode
set :userinputnode.data.test1 = "2, 4, 8"
set :userinputnode.names = [test1 test2]
set :userinputnode.custom_storage.test1 = Integer
```

set :userinputnode.data_mode = "Ordered"

userinputnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
data		Schlüsseleigenschaft im Format: set :userinputnode.data.Age = "1 2 3 4" Alternativ kann die Zeichenkette durch Komma getrennte niedrige, hohe und schrittweise Größen angeben. Beispiel: set :userinputnode.data.Age = "10, 70, 5" Die Daten für jedes Feld 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 Zeichenkette (" ") wird das angegebene Feld gelöscht.
names		Strukturierter Slot, der eine vom Knoten erstellte Liste der Feldnamen festlegt oder zurückgibt. Beispiel: ['Field1' 'Field2']
custom_storage	Unknown String Integer Real Time Date Timestamp	Schlüssel-Slot, der den Speichertyp für ein Feld festlegt oder zurückgibt. Beispiel: set :userinputnode.custom_storage.'Age' = "Real"
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.
values		*Diese Eigenschaft wurde zugunsten von* userinputnode.data *verworfen und sollte nicht mehr verwendet werden.*

Eigenschaften von "variablefilenode"

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. Für weitere Informationen siehe Thema Knoten "Datei (var.)" in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create variablefilenode
set :variablefilenode.full_filename = "$CLEO_DEMOS/DRUG4n"
set :variablefilenode.read_field_names = True
set :variablefilenode.delimit_other = True
set :variablefilenode.other = ','
set :variablefilenode.quotes_1 = Discard
set :variablefilenode.decimal_symbol = Comma
set :variablefilenode.invalid_char_mode = "Replace"
set :variablefilenode.invalid_char_replacement = "|"
set :variablefilenode.use_custom_values.Age = True
set :variablefilenode.direction.Age = Input
set :variablefilenode.type.Age = Range
set :variablefilenode.values.Age = [1 100]
```

variablefilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
skip_header	*number*	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeichen an. Format: variablefilenode:skip_header = 3
num_fields_auto	*Flag*	Stellt die Anzahl der Felder in jedem Datensatz automatisch fest. Datensätze müssen mit einem Zeilenwechselzeichen abgeschlossen werden. Format: variablefilenode:num_fields_auto
num_fields	*number*	Legt die Anzahl der Felder in jedem Datensatz manuell fest.
delimit_space	*Flag*	Gibt an, welches Zeichen in der Datei für die Feldbegrenzungen verwendet wird.
delimit_tab	*Flag*	
delimit_new_line	*Flag*	
delimit_non_printing	*Flag*	
delimit_comma	*Flag*	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 das Trennzeichen fest, das die Eigenschaft other verwendet.
delimit_other	*Flag*	Hier können Sie ein benutzerdefiniertes Trennzeichen festlegen, indem Sie die Eigenschaft other verwenden.
other	*string*	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	*Flag*	Behandelt mehrere angrenzende leere Trennzeichen als ein einziges Trennzeichen.

variablefilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
read_field_names	*Flag*	Behandelt die erste Zeile der Datendatei als Beschriftungen für die Spalten.
strip_spaces	None Left Right Both	Verwirft beim Importieren führende und abschließende Leerzeichen in Zeichenketten.
invalid_char_mode	Discard Replace	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Kodierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
invalid_char_replacement	*string*	
lines_to_scan	*number*	Gibt an, wie viele Zeilen nach angegebenen Datentypen durchsucht werden sollen.
auto_recognize_datetime	*Flag*	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quelldaten 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	*string*	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.
use_custom_values	*Flag*	Schlüssel-Slot in folgendem Format: set :varfilenode.use_custom_values.Age = true
custom_storage	Unknown String Integer Real Time Date Timestamp	Schlüssel-Slot in folgendem Format: set :varfilenode.custom_storage.'Age' = "Real"
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY"	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde. Beispiel: set:varfilenode.custom Schlüssel-Slot in folgendem Format: set :varfilenode.custom_date_format. 'LaunchDate' = "DDMMYY"

variablefilenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
	"DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
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. Schlüssel-Slot in folgendem Format: set :varfilenode.custom_time_format. 'Initialize' = "HHMM"
custom_decimal_symbol	*Feld*	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde. Schlüssel-Slot in folgendem Format: set :varfilenode.custom_decimal_ symbol.'Revenue' = "Comma"
encoding	StreamDefault SystemDefault "UTF-8"	Legt die Textkodierungsmethode fest.

xmlimportnode-Eigenschaften

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. Für weitere Informationen siehe Thema XML-Quellenknoten in Kapitel 2 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create xmlimportnode
set :xmlimportnode.full_filename = "c:\import\ebooks.xml"
```

```
set :xmlimportnode.records = "/author/name"
```

xmlimportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
read	single directory	Liest eine einzige Datendatei (Standard) oder alle XML-Dateien in einem Verzeichnis.
recurse	*Flag*	Legt fest, ob zusätzlich XML-Dateien aus allen Unterverzeichnissen des angegebenen Verzeichnisses gelesen werden sollen.
full_filename	*string*	(erforderlich) Vollständiger Pfad und Dateiname der zu importierenden XML-Datei (falls read = single).
directory_name	*string*	(erforderlich) Vollständiger Pfad und Dateiname des Verzeichnisses, in dem sich die zu importierenden XML-Dateien befinden (falls read = directory).
full_schema_filename	*string*	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	*string*	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
13

Datensatzoperationsknoten – Eigenschaften

Eigenschaften von “appendnode”

Der Anhangknoten verkettet Gruppen von Datensätzen miteinander. Er ist insbesondere nützlich für die Kombination von Daten-Sets mit ähnlicher Struktur, aber unterschiedlichen Daten. Für weitere Informationen siehe Thema Anhangknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create appendnode
set :appendnode.match_by = Name
set :appendnode.match_case = True
set :appendnode.include_fields_from = All
set :appendnode.create_tag_field = True
set :appendnode.tag_field_name = "Append_Flag"
```

appendnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
match_by	Position Name	Daten-Sets können Sie auf der Grundlage der Position der Felder in der Hauptdatenquelle oder auf der Grundlage der Namen von Feldern der Eingabe-Daten-Sets anhängen.
match_case	*Flag*	Aktiviert für die Übereinstimmung von Feldnamen die Unterscheidung zwischen Groß- und Kleinschreibung.
include_fields_from	Main All	
create_tag_field	*Flag*	
tag_field_name	*string*	

Eigenschaften von “aggregatenode”

Der Aggregatknoten ersetzt eine Sequenz von Eingabedatensätzen durch zusammengefasste, aggregierte Ausgabedatensätze. Für weitere Informationen siehe Thema Aggregatknoten. in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create aggregatenode
connect :databasenode to :aggregatenode
set :aggregatenode.contiguous = True
set :aggregatenode.keys = ['Drug']
```

```
set :aggregatenode.aggregates.Age = [Sum Mean]
set :aggregatenode.inc_record_count = True
set :aggregatenode.count_field = "index"
set :aggregatenode.extension = "Aggregated_"
set :aggregatenode.add_as = Prefix
```

aggregaten-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
keys	*[Feld Feld ... Feld]*	Listet Felder auf, die als Schlüssel für die Aggregation verwendet werden können. Bei den Schlüsselfeldern Sex und Region beispielsweise erhält jede eindeutige Kombination von M und F mit den Regionen N und S (vier eindeutige Kombinationen) einen aggregierten Datensatz.
contiguous	*Flag*	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.
aggregates		Strukturierte Eigenschaft, die die numerischen Felder auflistet, deren Werte aggregiert werden, sowie die ausgewählten Aggregationsmodi. Beispiel: set :aggregatenode. aggregates.Age = [Sum Mean Min Max SDev Median Count Variance Firstquartile Thirdquartile], wobei die gewünschten Aggregationsmethoden in der Liste aufgeführt sind.
extension	*string*	Geben Sie einen Präfix oder Suffix für doppelt aggregierte Felder an (Beispiel unten).
add_as	Suffix Prefix	
inc_record_count	*Flag*	Erstellt ein zusätzliches Feld, das angibt, wie viele Eingabedatensätze aus den einzelnen Aggregatdatensätzen aggregiert wurden.
count_field	*string*	Gibt den Namen des Felds für die Datensatzanzahl an.

Eigenschaften von “balancenode”

Der Balancierungsknoten korrigiert Ungleichgewichte in einem Daten-Set, sodass dieses eine bestimmte Bedingung erfüllt. Die Balancierungsanweisung passt den Anteil der Datensätze, bei denen eine Bedingung wahr ist, um den angegebenen Faktor an. Für weitere Informationen siehe Thema Balancierungsknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create balancenode
set :balancenode.training_data_only = true
set :balancenode.directives = \
 [{1.3 "Age > 60"}{1.5 "Na > 0.5"}]
```

balancenode-Eigenschaften	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	*Flag*	Gibt an, dass nur Trainingsdaten balanciert werden sollen. Wenn im Stream kein Partitionsfeld vorhanden ist, wird diese Option ignoriert.

Beispiel

```
create balancenode
set :balancenode.directives = \
 [{1.3 "Age > 60"}{1.5 "Na > 0.5"}]
```

Diese Knoteneigenschaft besitzt folgendes Format:

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

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

Eigenschaften von "distinctnode"

Der Duplikatknoten entfernt doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Daten-Stream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden. Für weitere Informationen siehe Thema Duplikatknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create distinctnode
set :distinctnode.mode = Include
set :distinctnode.fields = ['Age' 'Sex']
set :distinctnode.keys_pre_sorted = True
```

distinctnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
mode	Include Discard	Duplikatknoten entfernen doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Daten-Stream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden.
fields	[*Feld Feld Feld*]	Listet die Felder auf, die verwendet werden, um zu bestimmen, ob die Datensätze identisch sind.

low_distinct_key_count	*Flag*	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	*Flag*	Gibt an, dass alle Datensätze mit denselben Schlüsselwerten in der Eingabe zusammengefasst werden.

Eigenschaften von "mergenode"

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. Für weitere Informationen siehe Thema Zusammenführungsknoten ("Mergen") in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create mergenode
connect customerdata to :mergenode
connect salesdata to :mergenode
set :mergenode.method = Keys
set :mergenode.key_fields = ['id']
set :mergenode.common_keys = true
set :mergenode.join = PartialOuter
set :mergenode.outer_join_tag.2 = true
set :mergenode.outer_join_tag.4 = true
set :mergenode.single_large_input = true
set :mergenode.single_large_input_tag = '2'
set :mergenode.use_existing_sort_keys = true
```

set :mergenode.existing_sort_keys = [{'id' Ascending}]

mergenode-Eigenschaften	**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	*string*	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	*Flag*	
join	Inner FullOuter PartialOuter Anti	Ein Beispiel lautet folgendermaßen: set :merge.join = FullOuter
outer_join_tag.n	*Flag*	Bei dieser Eigenschaft ist *n* der im Dialogfeld für die Daten-Set-Auswahl angezeigte Tag-Name. Beachten Sie, dass mehrere Tag-Namen angegeben werden können, da jede beliebige Zahl von Daten-Sets unvollständige Datensätze beitragen könnte.
single_large_input	*Flag*	Gibt an, ob die Optimierung verwendet werden soll, wenn eine Eingabe vorhanden ist, die im Vergleich mit den anderen Eingaben relativ groß ist.
single_large_input_tag	*string*	Geben Sie den Tag-Namen an, der im Dialogfeld “Großes Daten-Set auswählen” angezeigt wird. Beachten Sie, dass die Verwendung dieser Eigenschaft leicht von der Eigenschaft outer_join_tag abweicht (Flag gegenüber Zeichenkette), da nur ein einziges Eingabe-Daten-Set angegeben werden kann.
use_existing_sort_keys	*Flag*	Gibt an, ob die Eingaben bereits nach einem oder mehreren Schlüsselfeldern sortiert sind.
existing_sort_keys	[{*string* Ascending} \ {*string* Descending}]	Gibt die bereits sortieren Felder und ihre Sortierrichtung an.

Eigenschaften von “rfmaggregatenode”

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. Für weitere Informationen siehe Thema RFM-Aggregatknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create rfmaggregatenode
connect :fillernode to :rfmaggregatenode
set :rfmaggregatenode.relative_to = Fixed
set :rfmaggregatenode.reference_date = "2007-10-12"
set :rfmaggregatenode.id_field = "CardID"
set :rfmaggregatenode.date_field = "Date"
set :rfmaggregatenode.value_field = "Amount"
set :rfmaggregatenode.only_recent_transactions = True
set :rfmaggregatenode.transaction_date_after = "2000-10-01"
```

rfmaggregaten-ode-Eigenschaften	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 Fixed auf relative_to gesetzt ist.
contiguous	*Flag*	Wenn die Daten vorsortiert sind, sodass alle Datensätze mit derselben ID zusammen im Daten-Stream 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	*string*	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	*Flag*	Aktivieren Sie die Verwendung der Einstellung discard_records_below.

rfmaggregaten-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
discard_records_below	*number*	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	*Flag*	Aktivieren der Verwendung der Einstellung specify_transaction_date oder transaction_within_last.
specify_transaction_date	*Flag*	
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	*number*	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	*Flag*	Zeigt für jeden Kunden das Datum der zweitaktuellsten Transaktion an.
save_r3	*Flag*	Nur verfügbar, wenn save_r2 ausgewählt wurde. Zeigt für jeden Kunden das Datum der drittaktuellsten Transaktion an.

Eigenschaften von "samplenode"

Der Stichprobenknoten wählte eine Teilmenge der Datensätze aus. Es wird eine Vielzahl von Stichprobentypen unterstützt, darunter geschichtete, gruppierte (Klumpenstichproben) und nichtzufällige (strukturierte) Stichproben. Eine Stichprobenziehung kann nützlich zur Verbesserung der Leistungsfähigkeit und zur Auswahl von verwandten Datensätzen bzw. Transaktionen für die Analyse sein. Für weitere Informationen siehe Thema Stichprobenknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
/* Create two Sample nodes to extract
  different samples from the same data */

create variablefilenode
```

```
set :variablefilenode.full_filename = "$CLEO_DEMOS/DRUG1n"

set node = create samplenode at 300 100
rename ^node as 'First 500'
connect :variablefilenode to 'First 500'
set 'First 500':samplenode.method = Simple
set 'First 500':samplenode.mode = Include
set 'First 500':samplenode.sample_type = First
set 'First 500':samplenode.first_n = 500

set node = create samplenode at 300 200
rename ^node as 'Custom Strata'
connect :variablefilenode to 'Custom Strata'
set 'Custom Strata':samplenode.method = Complex
set 'Custom Strata':samplenode.stratify_by = ['Sex' 'Cholesterol']
set 'Custom Strata':samplenode.sample_units = Proportions
set 'Custom Strata':samplenode.sample_size_proportions = Custom
set 'Custom Strata':samplenode.sizes_proportions= \
   [{"M" "High" "Default"}{"M" "Normal" "Default"} \
   {"F" "High" "0.3"}{"F" "Normal" "0.3"}]
```

samplenode-Eigenschaften	**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. Ein Beispiel lautet folgendermaßen: set :samplenode.sample_type = First set :samplenode.first_n = 100
first_n	*integer*	Datensätze bis zum angegebenen Abbruchpunkt werden eingeschlossen oder verworfen.
one_in_n	*number*	Jeden *n*-ten Datensatz einschließen oder verwerfen.
rand_pct	*number*	Geben Sie den Prozentsatz der einzuschließenden oder zu verwerfenden Datensätze an.
use_max_size	*Flag*	Aktivieren Sie die Verwendung der Einstellung maximum_size.
maximum_size	*integer*	Geben Sie die größte Stichprobe an, die in den Daten-Stream eingeschlossen oder verworfen werden soll. Diese Option ist redundant und wird daher deaktiviert, wenn First und Include angegeben werden.
set_random_seed	*Flag*	Aktiviert die Verwendung der Einstellung für den Zufallsstartwert.
random_seed	*integer*	Geben Sie den Wert an, der als Startwert für den Zufallsgenerator verwendet wird.
complex_sample_type	Random Systematic	

samplenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*integer*	
variable_proportions	*Feld*	
variable_counts	*Feld*	
use_min_stratum_size	*Flag*	
minimum_stratum_size	*integer*	Diese Option gilt nur, wenn eine komplexe Stichprobe mit Sample units=Proportions gezogen wird.
use_max_stratum_size	*Flag*	
maximum_stratum_size	*integer*	Diese Option gilt nur, wenn eine komplexe Stichprobe mit Sample units=Proportions gezogen wird.
clusters	*Feld*	
stratify_by	*[field1 ... fieldN]*	
specify_input_weight	*Flag*	
input_weight	*Feld*	
new_output_weight	*string*	
sizes_proportions	[{string*string value*}{string*string value*}…]	Bei sample_units=proportions und sample_size_proportions=Custom zur Angabe eines Werts für jede mögliche Kombination der Werte von Schichtungsfeldern.
default_proportion	*number*	
sizes_counts	[{string*string value*}{string*string value*}…]	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	*number*	

Eigenschaften von "selectnode"

Der Auswahlknoten wählt auf der Grundlage einer bestimmten Bedingung eine Untergruppe von Datensätzen aus einem Daten-Stream aus oder verwirft sie. Sie können beispielsweise die Datensätze auswählen, die zu einer bestimmten Verkaufsregion gehören. Für weitere Informationen siehe Thema Auswahlknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create selectnode
set :selectnode.mode = Include
set :selectnode.condition = "Age < 18"
```

selectnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
mode	Include Discard	Definiert, ob die ausgewählten Datensätze eingeschlossen oder verworfen werden sollen.
condition	*string*	Bedingung für das Einschließen oder Verwerfen von Datensätzen.

Eigenschaften von "sortnode"

Der Sortierknoten sortiert Datensätze anhand der Werte eines oder mehrerer Felder in aufsteigender oder absteigender Reihenfolge. Für weitere Informationen siehe Thema Sortierknoten in Kapitel 3 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create sortnode
set :sortnode.keys = [{'Age' Ascending}{'Sex' Descending}]
set :sortnode.default_ascending = False
set :sortnode.use_existing_keys = True
set :sortnode.existing_keys = [{'Age' Ascending}]
```

sortnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
keys	[{*string* Ascending} \ {*string* Descending}]	Gibt die Felder an, anhand derer sortiert werden soll (Beispiel unten). Wenn keine Richtung angegeben ist, wird der Standard verwendet.
default_ascending	*Flag*	Gibt die Standard-Sortierreihenfolge an.
use_existing_keys	*Flag*	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.

Kapitel 14

Feldoperationsknoten – Eigenschaften

Eigenschaften von anonymizenode

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. Für weitere Informationen siehe Thema Anonymisierungsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create anonymizenode
set: anonymizenode.enable_anonymize = age
set: anonymizenode.use_prefix = true
set: anonymizenode.prefix = "myprefix"
set: anonymizenode.transformation = Random
set: anonymizenode.set_random_seed = true
set: anonymizenode.random_seed = "123"
```

anonymizen-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
enable_anonymize	*Flag*	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	*Flag*	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 der Optionsschaltfläche Benutzerdefiniert im Dialogfeld “Werte ersetzen” für das betreffende Feld.
prefix	*string*	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	*Flag*	Bei Festlegung auf T wird der angegebene Startwert für den Zufallsgenerator verwendet (sofern außerdem “transformation” auf “Random” gesetzt ist).
random_seed	*integer*	Wenn set_random_seed auf T gesetzt ist, wird dieser Wert als Startwert für den Zufallsgenerator verwendet.

anonymizen-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
scale	*number*	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	*number*	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 von “autodataprepnode”

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. Für weitere Informationen siehe Thema Automatisierte Datenaufbereitung in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create autodataprepnode
set: autodataprepnode.objective = Balanced
set: autodataprepnode.excluded_fields = Filter
set: autodataprepnode.prepare_dates_and_times = true
set: autodataprepnode.compute_time_until_date = true
set: autodataprepnode.reference_date = Today
set: autodataprepnode.units_for_date_durations = Automatic
```

autodataprepn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
objective	Balanced Speed Accuracy Custom	
custom_fields	*Flag*	Bei “true” (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei “false” (falsch) werden die aktuellen Einstellungen aus einem weiter oben im Stream gelegenen Typknoten verwendet.
target	*Feld*	Gibt ein einzelnes Zielfeld an.
inputs	[*Feld1 ... FeldN*]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
use_frequency	*Flag*	
frequency_field	*Feld*	

autodataprepn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
use_weight	*Flag*	
weight_field	*Feld*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*Flag*	Zugriff auf alle Datums- und Zeitfelder kontrollieren
compute_time_until_date	*Flag*	
reference_date	Today Fixed	
fixed_date	*Datum*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*Flag*	
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*Flag*	
extract_month_from_date	*Flag*	
extract_day_from_date	*Flag*	
extract_hour_from_time	*Flag*	
extract_minute_from_time	*Flag*	
extract_second_from_time	*Flag*	
exclude_low_quality_inputs	*Flag*	
exclude_too_many_missing	*Flag*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*Flag*	
maximum_number_categories	*number*	
exclude_if_large_category	*Flag*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*Flag*	
adjust_type_inputs	*Flag*	
adjust_type_target	*Flag*	
reorder_nominal_inputs	*Flag*	
reorder_nominal_target	*Flag*	
replace_outliers_inputs	*Flag*	

autodataprepn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
replace_outliers_target	*Flag*	
replace_missing_continuous_in-puts	*Flag*	
replace_missing_continuous_tar-get	*Flag*	
replace_missing_nominal_inputs	*Flag*	
replace_missing_nominal_target	*Flag*	
replace_missing_ordinal_inputs	*Flag*	
replace_missing_ordinal_target	*Flag*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*Flag*	
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*Flag*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*Flag*	
maximize_association_with_tar-get	*Flag*	
p_value_for_merging	*number*	
merge_ordinal_features	*Flag*	
merge_nominal_features	*Flag*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*Flag*	
p_value_for_binning	*number*	
perform_feature_selection	*Flag*	
p_value_for_selection	*number*	
perform_feature_construction	*Flag*	
transformed_target_name_exten-sion	*string*	
transformed_inputs_name_exten-sion	*string*	
constructed_features_root_name	*string*	
years_duration_ name_extension	*string*	
months_duration_ name_extension	*string*	
days_duration_ name_extension	*string*	

autodataprepn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
hours_duration_ name_extension	*string*	
minutes_duration_ name_extension	*string*	
seconds_duration_ name_extension	*string*	
year_cyclical_name_extension	*string*	
month_cyclical_name_extension	*string*	
day_cyclical_name_extension	*string*	
hour_cyclical_name_extension	*string*	
minute_cyclical_name_extension	*string*	
second_cyclical_name_extension	*string*	

Eigenschaften von "binningnode"

Der Klassierknoten erstellt automatisch neue nominale (Set-) Felder 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. Für weitere Informationen siehe Thema Klassierknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create binningnode
set :binningnode.fields = [Na K]
set :binningnode.method = Rank
set :binningnode.fixed_width_name_extension = "_binned"
set :binningnode.fixed_width_add_as = Suffix
set :binningnode.fixed_bin_method = Count
set :binningnode.fixed_bin_count = 10
set :binningnode.fixed_bin_width = 3.5
set :binningnode.tile10 = true
```

binningnode-Eigenschaften	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.

binningnode-Eigenschaften	**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	*string*	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	*integer*	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	*string*	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	*Flag*	Generiert vier Quantil-Klassen, die jeweils 25 % der Fälle enthalten.
tile5	*Flag*	Generiert fünf Qintil-Klassen.
tile10	*Flag*	Generiert 10 Dezil-Klassen.
tile20	*Flag*	Generiert 20 Vingtil-Klassen.
tile100	*Flag*	Generiert 100 Perzentil-Klassen.
use_custom_tile	*Flag*	
custom_tile_name_extension	*string*	Die Standarderweiterung lautet *_TILEN*.
custom_tile_add_as	Suffix Prefix	
custom_tile	*integer*	
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.

binningnode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
rank_order	Ascending Descending	Diese Eigenschaft beinhaltet Ascending (der niedrigste Wert wird mit "1" gekennzeichnet) oder Descending (der höchste Wert wird mit "1" gekennzeichnet).
rank_add_as	Suffix Prefix	Mit dieser Option werden Rang, Bruchzahlrang und Prozentsatzrang angewendet.
rank	*Flag*	
rank_name_extension	*string*	Die Standarderweiterung lautet *_RANK*.
rank_fractional	*Flag*	Weist Fällen Ränge zu, wobei der Wert des neuen Felds gleich dem Rang dividiert durch die Summe der Gewichtungen der nichtfehlenden Fälle ist. Bruchzahlränge fallen in den Bereich 0–1.
rank_fractional_name_ extension	*string*	Die Standarderweiterung lautet *_F_RANK*.
rank_pct	*Flag*	Die einzelnen Ränge werden durch die Anzahl der Datensätze mit gültigen Werten dividiert und mit 100 multipliziert. Als Prozentsatz angegebene Bruchzahlränge fallen in den Bereich 1–100.
rank_pct_name_extension	*string*	Die Standarderweiterung lautet *_P_RANK*.
sdev_name_extension	*string*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*string*	Die Standarderweiterung lautet *_OPTIMAL*.
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	*Feld*	Als Supervisor-Feld ausgewähltes Feld, mit dem die für die Klassierung ausgewählten Felder in Bezug stehen.
optimal_merge_bins	*Flag*	Gibt an, dass alle Klassen mit kleinen Fallzahlen zu einer größeren, benachbarten Klasse hinzugefügt werden.
optimal_small_bin_threshold	*integer*	
optimal_pre_bin	*Flag*	Gibt an, dass eine Vorklassierung des Daten-Sets durchgeführt werden soll.
optimal_max_bins	*integer*	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 von “derivenode”

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”. Für weitere Informationen siehe Thema Ableitungsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
# Create and configure a Flag Derive field node
create derivenode
rename derive:derivenode as "Flag"
set Flag:derivenode.new_name = "DrugX_Flag"
set Flag:derivenode.result_type = Flag
set Flag:derivenode.flag_true = 1
set Flag:derivenode.flag_false = 0
set Flag:derivenode.flag_expr = "Drug = X"
# Create and configure a Conditional Derive field node
create derivenode
rename derive:derivenode as "Conditional"
set Conditional:derivenode.result_type = Conditional
set Conditional:derivenode.cond_if_cond = "@OFFSET(\'Age\', 1) = \'Age\'"
set Conditional:derivenode.cond_then_expr = "(@OFFSET(\'Age\', 1) = \'Age\') >< @INDEX"
set Conditional:derivenode.cond_else_expr = "\'Age\'"
```

derivenode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
new_name	*string*	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	*string*	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	*string*	Ausdruck zum Berechnen eines neuen Feldwerts in einem Ableitungsknoten.
flag_expr	*string*	
flag_true	*string*	
flag_false	*string*	
set_default	*string*	

derivenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
set_value_cond	*string*	Wird zur Bereitstellung der Bedingung, die einem bestimmten Wert zugeordnet ist, strukturiert. Format: set :derivenode. set_value_cond. Retired = 'age > 65'
state_on_val	*string*	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "On" (Ein) erfüllt ist.
state_off_val	*string*	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "Off" (Aus) erfüllt ist.
state_on_expression	*string*	
state_off_expression	*string*	
state_initial	On Off	Weist jedem Datensatz des neuen Feldes einen Anfangswert On oder Off zu. Dieser Wert kann sich ändern, wenn die einzelnen Bedingungen erfüllt werden.
count_initial_val	*string*	
count_inc_condition	*string*	
count_inc_expression	*string*	
count_reset_condition	*string*	
cond_if_cond	*string*	
cond_then_expr	*string*	
cond_else_expr	*string*	

Eigenschaften von "ensemblenode"

Der Ensemble-Knoten kombiniert zwei oder mehr Modell-Nuggets, um genauere Vorhersagen zu erzielen, als aus einem dieser Modelle allein gewonnen werden können. Für weitere Informationen siehe Thema Ensemble-Knoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
create ensemblenode
set :ensemblenode.ensemble_target_field = response
set :ensemblenode.filter_individual_model_output = false
set :ensemblenode.flag_ensemble_method = ConfidenceWeightedVoting
set :ensemblenode.flag_voting_tie_selection = HighestConfidence
```

ensemblenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
ensemble_target_field	*Feld*	Gibt das Zielfeld für alle im Ensemble verwendeten Modelle an.
filter_individual_model_output	*Flag*	Gibt an, ob Scoring-Ergebnisse aus einzelnen Modellen unterdrückt werden sollen.

ensemblenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
flag_ensemble_method	Voting ConfidenceWeight-edVoting RawPropensity-WeightedVoting AdjustedPropensity-WeightedVoting HighestConfidence AverageRawPropen-sity AverageAdjusted-Propensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flag-Feld ist.
set_ensemble_method	Voting ConfidenceWeight-edVoting 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 Flag-Feld 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	*Flag*	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 von "fillernode"

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. Für weitere Informationen siehe Thema Füllerknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

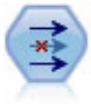
Beispiel

```
create fillernode
set :fillernode.fields = ['Age']
set :fillernode.replace_mode = Always
set :fillernode.condition = "(\'Age\' > 60) and (\'Sex\' = \'M\')"
```

```
set :fillernode.replace_with = "\'old man\'"
```

fillernode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields	[*Feld Feld Feld*]	Felder aus dem Daten-Set, 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	*string*	
replace_with	*string*	

Eigenschaften von "filternode"



Der Filterknoten filtert (verwirft) Felder, benennt Felder um und ordnet Felder von einem Quellenknoten einem anderen zu. Für weitere Informationen siehe Thema Filtern bzw. Umbenennen von Feldern in Kapitel 4 in *IBM SPSS Modeler 15 Quellen-, Prozess- und Ausgabeknoten*.

Beispiel

```
create filternode
set :filternode.default_include = True
set :filternode.new_name.'Drug' = 'Chemical'
set :filternode.include.'Drug' = off
```

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. Hier ein Beispiel: Angenommen, Sie führen folgendes Skript aus:

```
set Filter.default_include=False
# Include only fields in the list
for f in Age Sex
  set Filter.include.^f=True
endfor
```

Dies führt dazu, dass der Knoten die Felder *Age* und *Sex* weitergibt und alle anderen verwirft. Angenommen, Sie führen dasselbe Skript erneut aus, benennen jedoch zwei andere Felder:

```
set Filter.default_include=False
# Include only fields in the list
for f in BP Na
  set Filter.include.^f=True
endfor
```

Dadurch werden zwei weitere Felder zum Filter hinzugefügt, sodass insgesamt vier Felder weitergegeben werden (*Age*, *Sex*, *BP*, *Na*). Anders ausgedrückt, wenn der Wert von default_include auf False (Falsch) gesetzt wird, bedeutet dies nicht, dass automatisch alle Felder zurückgesetzt werden.

Wenn sie stattdessen nun default_include auf True (Wahr) ändern (entweder mithilfe eines Skripts oder im Dialogfeld des Filterknotens, wird das Verhalten umgekehrt, sodass die vier oben aufgeführten Felder nicht aufgenommen, sondern stattdessen verworfen werden. Wenn Sie sich unsicher sind, sollten Sie ein wenig mit den Steuerelementen im Dialogfeld des Filterknotens herumexperimentieren. Dies kann Ihnen beim Verständnis dieser Interaktion helfen.

filternode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
default_include	*Flag*	Schlüsseleigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden: NODE.include.FIELDNAME Ein Beispiel lautet folgendermaßen: set mynode:filternode.default_include = false 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. Weitere Kommentare finden Sie im unten stehenden Beispiel:
include	*Flag*	Schlüsseleigenschaft zum Einbeziehen und Entfernen von Feldern. Format: NODE.include.FIELDNAME Ein Beispiel lautet folgendermaßen: set mynode: filternode.include.Age = false
new_name	*string*	Ein Beispiel lautet folgendermaßen: set mynode: filternode.new_name. Age = "age"

Eigenschaften von “historynode”

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. Für weitere Informationen siehe Thema Verlaufsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create historynode
set :historynode.fields = ['Drug']
set :historynode.offset = 1
set :historynode.span = 3
```

```
set :historynode.unavailable = Discard
set :historynode.fill_with = "undef"
```

historynode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields	[*Feld Feld Feld*]	Felder, für die Sie einen Verlauf wollen.
offset	*number*	Dient zur Angabe des jüngsten Datensatzes (vor dem aktuellen Datensatz), aus dem Verlaufsfeldwerte extrahiert werden sollen.
span	*number*	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 Daten-Set, für die es keine vorangegangenen Datensätze gibt, die als Verlauf dienen könnten.
fill_with	String Number	Gibt einen Wert oder eine Zeichenkette an, die für Datensätze verwendet werden soll, wenn kein Verlaufswert verfügbar ist.

Eigenschaften von "partitionnode"

Der Partitionsknoten erstellt ein Partitionsfeld, das Daten in getrennte Untergruppen für die Trainings-, Test- und Validierungsphase der Modellerstellung aufteilt. Für weitere Informationen siehe Thema Partitionsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create partitionnode
set :partitionnode.create_validation = True
set :partitionnode.training_size = 33
set :partitionnode.testing_size = 33
set :partitionnode.validation_size = 33
set :partitionnode.set_random_seed = True
set :partitionnode.random_seed = "123"
set :partitionnode.value_mode = System
```

partitionnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
new_name	*string*	Der vom Knoten erstellte Name des Partitionsfelds.
create_validation	*Flag*	Gibt an, ob eine Validierungspartition erstellt werden soll.
training_size	*integer*	Prozentsatz der Datensätze (0–100), die der Trainingspartition zugewiesen werden sollen.
testing_size	*integer*	Prozentsatz der Datensätze (0–100), die der Testpartition zugewiesen werden sollen.

partitionnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
validation_size	*integer*	Prozentsatz der Datensätze (0–100), die der Validierungspartition zugewiesen werden sollen. Wird ignoriert, wenn keine Validierungspartition erstellt wird.
training_label	*string*	Beschriftung der Trainingspartition.
testing_label	*string*	Beschriftung der Testpartition.
validation_label	*string*	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 Trainings-Stichprobe 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	*integer*	Ein benutzerdefinierter Startwert für den Zufallsgenerator festlegen. Damit dieser Wert verwendet wird, muss set_random_seed auf True 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 gesetzt sein.

Eigenschaften von "reclassifynode"

Der Umkodierungsknoten transformiert ein Set kategorialer Werte in ein anderes. Die Umkodierung dient zur Reduzierung von Kategorien bzw. Neugruppierung von Daten für die Analyse. Für weitere Informationen siehe Thema Umkodierungsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create reclassifynode
set :reclassifynode.mode = Multiple
set :reclassifynode.replace_field = true
set :reclassifynode.field = "Drug"
set :reclassifynode.new_name = "Chemical"
set :reclassifynode.fields = [Drug, BP]
set :reclassifynode.name_extension = "reclassified"
set :reclassifynode.add_as = Prefix
set :reclassifynode.reclassify.'drugA' = 'Yes'
set :reclassifynode.use_default = True
set :reclassifynode.default = "BrandX"
```

set :reclassifynode.pick_list = [BrandX, Placebo, Generic]

reclassifynode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
mode	Single Multiple	Single kodiert die Kategorien für ein Feld um. Multiple aktiviert Optionen, die die Transformation von mehreren Feldern gleichzeitig erlauben.
replace_field	*Flag*	
field	*string*	Wird nur im Modus “Single” verwendet.
new_name	*string*	Wird nur im Modus “Single” verwendet.
fields	*[Feld1 Feld2 ... Feldn]*	Wird nur im Modus “Multiple” verwendet.
name_extension	*string*	Wird nur im Modus “Multiple” verwendet.
add_as	Suffix Prefix	Wird nur im Modus “Multiple” verwendet.
reclassify	*string*	Strukturierte Eigenschaft für Feldwerte. Format: NODE.reclassify. OLD_VALUE Ein Beispiel lautet folgendermaßen: set :reclassifynode.reclassify.'drugB' = 'Yes'
use_default	*Flag*	Standardwert verwenden.
default	*string*	Standardwert angeben.
pick_list	*[string string ... string]*	Ermöglicht einem Benutzer den Import einer Liste bekannter neuer Werte, um die Dropdown-Liste in der Tabelle zu füllen. Ein Beispiel lautet folgendermaßen: set :reclassify.pick_list = [fruit dairy cereals]

Eigenschaften von “reordernode”

Der Knoten “Felder ordnen” definiert die natürliche Reihenfolge, die bei der Anzeige der weiter unten im Stream liegenden 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 Daten-Sets die relevanten Felder deutlicher hervorzuheben. Für weitere Informationen siehe Thema Knoten “Felder ordnen” in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create reordernode
set :reordernode.mode = Custom
set :reordernode.sort_by = Storage
set :reordernode.ascending = "false"
set :reordernode.start_fields = [Age Cholesterol]
```

set :reordernode.end_fields = [Drug]

reordernode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
mode	Custom Auto	Sie können Werte automatisch sortieren oder eine benutzerdefinierte Reihenfolge angeben.
sort_by	Name Type Storage	
ascending	*Flag*	
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 von "restructurenode"

Der Knoten "Neu strukturieren" wandelt ein nominales Feld oder ein Flag-Feld in eine Gruppe von Feldern um, 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. Für weitere Informationen siehe Thema Neustrukturierungsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create restructurenode
connect :typenode to :restructurenode
set :restructurenode.fields_from.Drug = ["drugA" "drugX"]
set :restructurenode.include_field_name = "True"
set :restructurenode.value_mode = "OtherFields"
set :restructurenode.value_fields = ["Age" "BP"]
```

restructurenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields_from	[*Kategorie Kategorie Kategorie*] all	Beispiel: set :restructurenode.fields_from.Drug = [drugA drugB] erstellt Felder mit dem Namen Drug_drugA und Drug_drugB. Um alle Kategorien des angegebenen Feldes zu verwenden: set :restructurenode.fields_from.Drug = all
include_field_name	*Flag*	Gibt an, ob der Feldname im neu strukturierten Feldnamen verwendet werden soll.

restructurenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
value_mode	OtherFields Flags	Legt den Modus für die Angabe der Werte für die neu strukturierten Felder fest. Bei OtherFields müssen Sie angeben, welche Felder verwendet werden sollen (siehe unten). Bei Flags sind die Werte numerische Flags.
value_fields	[*Feld Feld Feld*]	Erforderlich, wenn value_mode gleich OtherFields. Gibt an, welche Felder als Wertfelder verwendet werden sollen.

Eigenschaften von "rfmanalysisnode"

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)). Für weitere Informationen siehe Thema Knoten "RFM-Analyse" in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create rfmanalysisnode
connect :rfmaggregatenode to :rfmanalysisnode
set :rfmanalysisnode.recency = Recency
set :rfmanalysisnode.frequency = Frequency
set :rfmanalysisnode.monetary = Monetary
set :rfmanalysisnode.tied_values_method = Next
set :rfmanalysisnode.recalculate_bins = IfNecessary
set :rfmanalysisnode.recency_thresholds = [1, 500, 800, 1500, 2000, 2500]
```

rfmanaly-sisnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
recency	*Feld*	Gibt das Feld für "Recency" (Aktualität) an. Dabei kann es sich um ein Datum, einen Zeitstempel 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	*integer*	Dient zur Angabe der Anzahl der zu generierenden Aktualitätsklassen.
recency_weight	*number*	Dient zur Angabe der Gewichtung für die Aktualitätsdaten. Der Standardwert ist 100.
frequency_bins	*integer*	Dient zur Angabe der Anzahl der zu generierenden Häufigkeitsklassen.
frequency_weight	*number*	Dient zur Angabe der Gewichtung für die Häufigkeitsdaten. Der Standardwert ist 10.
monetary_bins	*integer*	Dient zur Angabe der Anzahl der zu generierenden Klassen für den Geldwert.

rfmanaly-sisnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
monetary_weight	*number*	Dient zur Angabe der Gewichtung für die Geldwertdaten. Der Standardwert ist 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	*Flag*	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	
recency_thresholds	*Wert Wert*	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	*Wert Wert*	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.
monetary_thresholds	*Wert Wert*	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.

Eigenschaften von "settoflagnode"

Der Dichotomknoten leitet mehrere Flag-Felder auf der Grundlage der kategorialen Werte ab, die für ein oder mehrere nominale Felder definiert sind. Für weitere Informationen siehe Thema Dichotomknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create settoflagnode
connect :typenode to :settoflag
set :settoflagnode.fields_from.Drug = ["drugA" "drugX"]
set :settoflagnode.true_value = "1"
set :settoflagnode.false_value = "0"
set :settoflagnode.use_extension = "True"
set :settoflagnode.extension = "Drug_Flag"
set :settoflagnode.add_as = Suffix
```

```
set :settoflagnode.aggregate = True
set :settoflagnode.keys = ['Cholesterol']
```

settoflagnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields_from	[*Kategorie Kategorie Kategorie*] all	Beispiel: set :settoflagnode.fields_from.Drug = [drugA drugB] erstellt Flag-Felder mit dem Namen Drug_drugA und Drug_drugB. Um alle Kategorien des angegebenen Feldes zu verwenden: set :settoflagnode.fields_from.Drug = all
true_value	*string*	Gibt den Wahr-Wert an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert ist T.
false_value	*string*	Gibt den Falsch-Wert an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert ist F.
use_extension	*Flag*	Verwenden Sie eine Erweiterung als Suffix oder Präfix für das neue Flag-Feld.
extension	*string*	
add_as	Suffix Prefix	Gibt an, ob die Erweiterung als Suffix oder als Präfix hinzugefügt wird.
aggregate	*Flag*	Fasst Datensätze anhand von Schlüsselfeldern zu Gruppen zusammen. Alle in einer Gruppe vorhandenen Flag-Felder werden aktiviert, wenn einer der Datensätze auf "true" gesetzt wird.
keys	[*Feld Feld Feld*]	Schlüsselfelder.

Eigenschaften von "statisticstransformnode"

Der Statistiktransformationsknoten führt eine Auswahl von IBM® SPSS® Statistics-Syntaxbefehlen an Datenquellen in IBM® SPSS® Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistiktransformationsknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie unter Eigenschaften von "statisticstransformnode" auf S. 322.

Eigenschaften von "timeintervalsnode"

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. Für weitere Informationen siehe Thema Zeitintervallknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create timeintervalsnode
set :timeintervalsnode.interval_type=SecondsPerDay
set :timeintervalsnode.days_per_week=4
set :timeintervalsnode.week_begins_on=Tuesday
set :timeintervalsnode.hours_per_day=10
set :timeintervalsnode.day_begins_hour=7
set :timeintervalsnode.day_begins_minute=5
set :timeintervalsnode.day_begins_second=17
set :timeintervalsnode.mode=Label
set :timeintervalsnode.year_start=2005
set :timeintervalsnode.month_start=January
set :timeintervalsnode.day_start=4
set :timeintervalsnode.pad.AGE=MeanOfRecentPoints
set :timeintervalsnode.agg_mode=Specify
set :timeintervalsnode.agg_set_default=Last
```

timeintervalsnode-Eigenschaften	**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-, Zeitstempel- 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	*integer*	Gibt das Startintervall für Perioden bzw. zyklische Perioden an.
cycle_start	*integer*	Startzyklus für zyklische Perioden.
year_start	*integer*	Bei entsprechenden Intervalltypen das Jahr, in das das erste Intervall fällt.
quarter_start	*integer*	Bei entsprechenden Intervalltypen das Quartal, in das das erste Intervall fällt.

timeintervalsnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
month_start	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	
day_start	*integer*	
hour_start	*integer*	
minute_start	*integer*	
second_start	*integer*	
periods_per_cycle	*integer*	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 Thursday 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	*integer*	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	*integer*	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*).

timeintervalsnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
day_begins_second	*integer*	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	*integer*	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	*integer*	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Anzahl der Stunden pro Tag an.
interval_increment	1 2 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	*string*	
field_name_extension_as_prefix	*Flag*	
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" TAG MONAT "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	

timeintervalsnode-Eigenschaften	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 (z. B. aggregate.AGE=Mean).
pad	Blank MeanOfRecentPoints True False	Gibt die Auffüllmethode für ein Feld an (z. B. pad.AGE=MeanOfRecentPoints).
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	*integer*	Gibt die maximale Anzahl der beim Auffüllen der Reihe zu erstellenden Datensätze an.

timeintervalsnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
estimation_from_beginning	*Flag*	
estimation_to_end	*Flag*	
estimation_start_offset	*integer*	
estimation_num_holdouts	*integer*	
create_future_records	*Flag*	
num_future_records	*integer*	
create_future_field	*Flag*	
future_field_name	*string*	

Eigenschaften von "transposenode"

Der Transponierknoten vertauscht die Daten in Zeilen und Spalten, sodass aus Datensätzen Felder und aus Feldern Datensätze werden. Für weitere Informationen siehe Thema Transponierknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create transposenode
set :transposenode.transposed_names=Read
set :transposenode.read_from_field="TimeLabel"
set :transposenode.max_num_fields="1000"
set :transposenode.id_field_name="ID"
```

transposen-ode-Eigenschaften	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	*string*	
num_new_fields	*integer*	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	*integer*	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 String Custom	Standardmäßig werden nur stetige Felder (numerischer Bereich) transponiert, Sie können jedoch stattdessen auch eine benutzerdefinierte Untermenge numerischer Felder auswählen oder alle Zeichenkettenfelder transponieren.

transposen-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
transpose_fields	[*Feld Feld Feld*]	Gibt die bei Verwendung der Option Custom (Angepasst) zu transponierenden Felder an.
id_field_name	*Feld*	

Eigenschaften von "typenode"

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 Wertelabels angeben oder die Werte für ein Feld angeben. Für weitere Informationen siehe Thema Typknoten in Kapitel 4 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create typenode
connect :variablefilenode to :typenode
set :typenode.check.'Cholesterol' = Coerce
set :typenode.direction.'Drug' = Input
set :typenode.type.K = Range
set :typenode.values.Drug = [drugA drugB drugC drugD drugX drugY drugZ]
set :typenode.null_missing.BP = false
set :typenode.whitespace_missing.BP = "false"
set :typenode.description.BP = "Blood Pressure"
set :typenode.value_labels.BP = [{HIGH 'High Blood Pressure'}{NORMAL 'normal blood pressure'}]
set :typenode.display_places.K = 5
set :typenode.export_places.K = 2
set :typenode.grouping_symbol.Drug = None
set :typenode.column_width.Cholesterol = 25
set :typenode.justify.Cholesterol = Right
```

Beachten Sie: In einigen Fällen müssen Sie möglicherweise den Typknoten 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:

```
create tablenode
connect :typenode to :tablenode
execute :tablenode
```

delete :tablenode

typenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
direction	Input Target Both None Partition Split Frequency RecordID	Schlüsseleigenschaft für Feldrollen. Format: NODE.direction.FIELDNAME *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 type auf Default festgelegt wird, werden alle values-Parametereinstellungen gelöscht, und wenn value_mode den Wert Specify aufweist, wird auf Read zurückgesetzt. Wenn value_mode auf Pass oder Read festgelegt ist, beeinflusst die Einstellung von type den Wert value_mode nicht. Format: NODE.type.FIELDNAME
storage	Unknown String Integer Real Time Date Timestamp	Schreibgeschützte Schlüsseleigenschaft für Feldspeichertyp. Format: NODE.storage.FIELDNAME
check	None Nullify Coerce Discard Warn Abort	Schlüsseleigenschaft für das Überprüfen von Feldtyp und Bereich. Format: NODE.check.FIELDNAME
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 Flag-Feldern 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. Format: NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	Bestimmt, wie Werte festgelegt werden. Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf Specify festlegen können. Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft values fest. Format: NODE.value_mode.FIELDNAME

typenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
extend_values	*Flag*	Gilt, wenn value_mode auf Read festgelegt 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. Format: NODE.extend_values.FIELDNAME
enable_missing	*Flag*	Bei Festlegung auf T wird die Verfolgung von fehlenden Werten für das Feld aktiviert. Format: NODE.enable_missing.FIELDNAME
missing_values	[*Wert Wert ...*]	Gibt Datenwerte an, die fehlende Daten kennzeichnen. Format: NODE.missing_values.FIELDNAME
range_missing	*Flag*	Gibt an, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist.
missing_lower	*string*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an.
missing_upper	*string*	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an.
null_missing	*Flag*	Bei Festlegung auf T werden *Nullen* (undefinierte Werte, die in der Software als $null$ angezeigt werden) als fehlende Werte betrachtet. Format: NODE.null_missing.FIELDNAME
whitespace_missing	*Flag*	Bei Festlegung auf T werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet. Format: NODE.whitespace _missing. FIELDNAME
description	*string*	Gibt die Beschreibung für ein Feld an.
value_labels	*[{Wert Beschriftungsstring} {Wert Beschriftungsstring} ...]*	Gibt Beschriftungen für Wertpaare an. Ein Beispiel lautet folgendermaßen: set :typenode.value_labels.'Drug'=[{drugA label1} {drugB label2}]
display_places	*integer*	Legt die Dezimalstellen für das Feld bei der Anzeige fest (gilt nur für Felder mit dem Speichertyp REAL). Mit dem Wert –1 wird der Stream-Standard verwendet. Format: NODE.display_places.FIELDNAME

typenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
export_places	*integer*	Legt die Dezimalstellen für das Feld beim Export fest (gilt nur für Felder mit dem Speichertyp REAL). Mit dem Wert –1 wird der Stream-Standard verwendet. Format: NODE.export_places.FIELDNAME
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REAL). Format: NODE.decimal_separator.FIELDNAME
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" TAG MONAT "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). Format: NODE.date_format.FIELDNAME Ein Beispiel lautet folgendermaßen: set :tablenode.date_format.'LaunchDate' = "DDMMYY"
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). Format: NODE.time_format.FIELDNAME Ein Beispiel lautet folgendermaßen: set :tablenode.time_format.'BOF_enter' = "HHMMSS"

typenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Legt das Zahlenanzeigeformat für das Feld fest. Format: NODE.number_format.FIELDNAME
standard_places	*integer*	Legt die Dezimalstellen für das Feld für die Anzeige im Standardformat fest. Mit dem Wert –1 wird der Stream-Standard verwendet. Der vorhandene Slot display_places ändert dies zwar auch, wird aber nicht mehr verwendet. Format: NODE.standard_places.FIELDNAME
scientific_places	*integer*	Legt die Dezimalstellen für das Feld für die Anzeige im wissenschaftlichen Format fest. Mit dem Wert –1 wird der Stream-Standard verwendet. Format: NODE.scientific_places.FIELDNAME
currency_places	*integer*	Legt die Dezimalstellen für das Feld für die Anzeige im Währungsformat fest. Mit dem Wert –1 wird der Stream-Standard verwendet. Format: NODE.currency_places.FIELDNAME
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Legt das Symbol für die Zifferngruppierung für das Feld fest. Format: NODE.grouping_symbol.FIELDNAME
column_width	*integer*	Legt die Spaltenbreite für das Feld fest. Mit dem Wert –11 wird die Spaltenbreite auf Auto eingestellt. Format: NODE.column_width.FIELDNAME
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest. Format: NODE.justify.FIELDNAME

Kapitel
15

Diagrammknoten – Eigenschaften

Allgemeine Eigenschaften von Diagrammknoten

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

Allgemeine Eigenschaften von Diagrammknoten	Datentyp	Eigenschaftsbeschreibung
title	*string*	Gibt den Titel an. Beispiel: “Dies ist ein Titel.”
caption	*string*	Gibt die Benennung an. Beispiel: “Dies ist eine Benennung.”
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	*string*	Gibt den Zielpfad und den Dateinamen für die vom Diagrammknoten generierten Ausgabe an.
use_graph_size	*Flag*	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	*number*	Wenn use_graph_size den Wert True aufweist, wird hier die Diagrammhöhe in Pixeln festgelegt.
graph_height	*number*	Wenn use_graph_size den Wert True aufweist, wird hier die Diagrammbreite in Pixeln festgelegt.

Anmerkungen

Deaktivieren optionaler Felder. Optionale Felder, wie Überlagerungsfelder für Plots, können deaktiviert werden, indem der Eigenschaftswert auf " " (leere Zeichenkette) gesetzt wird, wie im folgenden Beispiel gezeigt:

```
set :plotnode.color_field = ""
```

Angeben von Farben. Die Farben für Titel, Benennungen, Hintergründe und Beschriftungen können mit Hexadezimal-Zeichenketten, die mit einem Rautenzeichen (#) beginnen, festgelegt werden. Beispiel: Um den Diagrammhintergrund auf Blau festzulegen, verwenden Sie folgende Anweisung:

```
set mygraph.graph_background="#87CEEB"
```

Die ersten beiden Stellen, 87, geben den roten Inhalt an, die mittleren Stellen, CE, legen den grünen Inhalt fest und die beiden letzten Stellen, EB, 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 Benutzeroberfläche festlegen. Bewegen Sie die Maus über die Farbe, um eine QuickInfo mit den gewünschten Informationen anzuzeigen.

Eigenschaften von "collectionnode"

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. Für weitere Informationen siehe Thema Sammlung – Registerkarte "Plot" in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create collectionnode
position :collectionnode at ^posX ^posY
# "Plot" tab
set :collectionnode.three_D = True
set :collectionnode.collect_field = 'Drug'
set :collectionnode.over_field = 'Age'
set :collectionnode.by_field = 'BP'
set :collectionnode.operation = Sum
# "Overlay" section
set :collectionnode.color_field = 'Drug'
set :collectionnode.panel_field = 'Sex'
set :collectionnode.animation_field = ''
# "Options" tab
set :collectionnode.range_mode = Automatic
set :collectionnode.range_min = 1
set :collectionnode.range_max = 100
set :collectionnode.bins = ByNumber
set :collectionnode.num_bins = 10
set :collectionnode.bin_width = 5
```

collectionnode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
over_field	*Feld*	
over_label_auto	*Flag*	
over_label	*string*	
collect_field	*Feld*	
collect_label_auto	*Flag*	
collect_label	*string*	
three_D	*Flag*	

collectionnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
by_field	*Feld*	
by_label_auto	*Flag*	
by_label	*string*	
operation	Sum Mean Min Max SDev	
color_field	*string*	
panel_field	*string*	
animation_field	*string*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*Flag*	
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 von "distributionnode"

Der Verteilungsknoten zeigt das Auftreten symbolischer (kategorialer) Werte wie beispielsweise Hypothekenart oder Geschlecht. Verteilungsknoten eignen sich insbesondere zum Aufzeigen von Ungleichgewichten in den Daten, die mithilfe eines Balancierungsknotens vor dem Erstellen eines Modells ausgeglichen werden können. Für weitere Informationen siehe Thema Verteilungsknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create distributionnode
# "Plot" tab
set :distributionnode.plot = Flags
set :distributionnode.x_field = 'Age'
set :distributionnode.color_field = 'Drug'
set :distributionnode.normalize = True
set :distributionnode.sort_mode = ByOccurence
```

```
set :distributionnode.use_proportional_scale = True
```

distributionn-ode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
plot	SelectedFields Flags	
x_field	*Feld*	
color_field	*Feld*	Überlagerungsfeld
normalize	*Flag*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*Flag*	

Eigenschaften von "evaluationnode"

Der Evaluationsknoten erleichtert die Evaluation und den Vergleich von Vorhersagemodellen. Das Evaluationsdiagramm zeigt, wie gut Modelle bestimmte Ergebnisse vorhersagen. Die Datensätze werden auf der Grundlage des vorhergesagten Werts und des Konfidenzwerts für die Prognose 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. Für weitere Informationen siehe Thema Evaluationsknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create evaluationnode
position :evaluationnode at ^posX ^posY
# "Plot" tab
set :evaluationnode.chart_type = Gains
set :evaluationnode.cumulative = False
set :evaluationnode.field_detection_method = Name
set :evaluationnode.inc_baseline = True
set :evaluationnode.n_tile = Deciles
set :evaluationnode.style = Point
set :evaluationnode.point_type = Dot
set :evaluationnode.use_fixed_cost = True
set :evaluationnode.cost_value = 5.0
set :evaluationnode.cost_field = 'Na'
set :evaluationnode.use_fixed_revenue = True
set :evaluationnode.revenue_value = 30.0
set :evaluationnode.revenue_field = 'Age'
set :evaluationnode.use_fixed_weight = True
set :evaluationnode.weight_value = 2.0
```

set :evaluationnode.weight_field = 'K'

evaluationnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
chart_type	Gains Response Lift Profit ROI	
inc_baseline	*Flag*	
field_detection_method	Metadata Name	
use_fixed_cost	*Flag*	
cost_value	*number*	
cost_field	*string*	
use_fixed_revenue	*Flag*	
revenue_value	*number*	
revenue_field	*string*	
use_fixed_weight	*Flag*	
weight_value	*number*	
weight_field	*Feld*	
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*Flag*	
style	Line Point	
point_type	Rechteck Punkt Dreieck Hexagon Plus Pentagon Stern BowTie HorizontalDash VerticalDash IronCross Fabrik Haus Kathedrale OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fächer	
export_data	*Flag*	
data_filename	*string*	
delimiter	*string*	

evaluationnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
new_line	*Flag*	
inc_field_names	*Flag*	
inc_best_line	*Flag*	
inc_business_rule	*Flag*	
business_rule_condition	*string*	
plot_score_fields	*Flag*	
score_fields	*[field1 ... fieldN]*	
target_field	*Feld*	
use_hit_condition	*Flag*	
hit_condition	*string*	
use_score_expression	*Flag*	
score_expression	*string*	
caption_auto	*Flag*	

graphboardnode-Eigenschaften

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. Für weitere Informationen siehe Thema Diagrammtafelknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

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

Beispiel

```
create graphboardnode
connect DRUG4n to :graphboardnode
set :graphboardnode.graph_type="Line"
set :graphboardnode.x_field = "K"
set :graphboardnode.y_field = "Na"
execute :graphboardnode
```

graph-board-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap	Identifiziert den Diagrammtyp.

graph-board-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
	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 PointsOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	
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 Hitzekarten verwendet.
size_field	*Feld*	In Blasendiagrammen verwendet.
categories_field	*Feld*	
values_field	*Feld*	
rows_field	*Feld*	
columns_field	*Feld*	
fields	*Feld*	
start_longitude	*Feld*	Bei Pfeilen auf einer Referenzkarte verwendet.
end_longitude	*Feld*	
start_latitude	*Feld*	
end_latitude	*Feld*	

graph-board-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
data_key_field	*Feld*	In verschiedenen Karten verwendet.
panelrow_field	*string*	
panelcol_field	*string*	
animation_field	*string*	
longitude	*Feld*	Bei Koordinaten in Karten verwendet.
latitude	*Feld*	
map_color_field	*Feld*	

Eigenschaften von "histogramnode"

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 Ungleichgewichte in den Daten aufdecken. Für weitere Informationen siehe Thema Histogramm – Registerkarte "Plot" in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create histogramnode
position :histogramnode at ^posX ^posY
# "Plot" tab
set :histogramnode.field = 'Drug'
set :histogramnode.color_field = 'Drug'
set :histogramnode.panel_field = 'Sex'
set :histogramnode.animation_field = ''
# "Options" tab
set :histogramnode.range_mode = Automatic
set :histogramnode.range_min = 1.0
set :histogramnode.range_max = 100.0
set :histogramnode.num_bins = 10
set :histogramnode.bin_width = 10
set :histogramnode.normalize = True
set :histogramnode.separate_bands = False
```

histogramnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
field	*Feld*	
color_field	*Feld*	
panel_field	*Feld*	
animation_field	*Feld*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	

histogramnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
num_bins	*number*	
bin_width	*number*	
normalize	*Flag*	
separate_bands	*Flag*	
x_label_auto	*Flag*	
x_label	*string*	
y_label_auto	*Flag*	
y_label	*string*	
use_grid	*Flag*	
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	*Flag*	Gibt an, ob die Normalverteilungskurve in der Ausgabe angezeigt werden soll.

Eigenschaften von "multiplotnode"

Ein Multidiagramm 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. Multidiagramme sind hilfreich, wenn die Fluktuation mehrerer Variablen im Laufe der Zeit untersucht werden soll. Für weitere Informationen siehe Thema Multidiagrammknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create multiplotnode
# "Plot" tab
set :multiplotnode.x_field = 'Age'
set :multiplotnode.y_fields = ['Drug' 'BP']
set :multiplotnode.panel_field = 'Sex'
# "Overlay" section
set :multiplotnode.animation_field = ''
set :multiplotnode.tooltip = "test"
set :multiplotnode.normalize = True
set :multiplotnode.use_overlay_expr = False
set :multiplotnode.overlay_expression = "test"
set :multiplotnode.records_limit = 500
set :multiplotnode.if_over_limit = PlotSample
```

multiplotnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
x_field	*Feld*	
y_fields	[*Feld Feld Feld*]	
panel_field	*Feld*	
animation_field	*Feld*	
normalize	*Flag*	

multiplotnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
use_overlay_expr	*Flag*	
overlay_expression	*string*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*Flag*	
x_label	*string*	
y_label_auto	*Flag*	
y_label	*string*	
use_grid	*Flag*	
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 von "plotnode"

Der Plotknoten zeigt die Beziehung zwischen numerischen Feldern an. Sie können einen Plot mithilfe von Punkten (Streudiagramm) oder mit Linien erstellen. Für weitere Informationen siehe Thema Plotknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create plotnode
# "Plot" tab
set :plotnode.three_D = True
set :plotnode.x_field = 'BP'
set :plotnode.y_field = 'Cholesterol'
set :plotnode.z_field = 'Drug'
# "Overlay" section
set :plotnode.color_field = 'Drug'
set :plotnode.size_field = 'Age'
set :plotnode.shape_field = ''
set :plotnode.panel_field = 'Sex'
set :plotnode.animation_field = 'BP'
set :plotnode.transp_field = ''
set :plotnode.style = Point
# "Output" tab
set :plotnode.output_mode =
set :plotnode.output_format = JPEG
```

```
set :plotnode.full_filename = "C:/Temp/Graph_Output/plot_output.jpeg"
```

plotnode-Eigenschaften	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	*Flag*	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 Flag-Feld 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 Flag-Feld 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 Linienplots nicht verfügbar.
overlay_type	None Smoother Function	Gibt an, ob eine Überlagerungsfunktion oder ein LOESS-Smoother angezeigt werden soll.
overlay_expression	*string*	Gibt den Ausdruck an, der verwendet wird, wenn overlay_type auf Function gesetzt ist.
style	Point Line	
point_type	Rechteck Punkt Dreieck Hexagon Plus Pentagon Stern BowTie HorizontalDash VerticalDash IronCross Fabrik Haus Kathedrale OnionDome ConcaveTriangle OblateGlobe	

plotnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
	CatEye FourSidedPillow RoundRectangle Fächer	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Automatic UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*Flag*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*Flag*	
x_label	*string*	
y_label_auto	*Flag*	
y_label	*string*	
z_label_auto	*Flag*	
z_label	*string*	
use_grid	*Flag*	
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.
use_overlay_expr	*Flag*	Zugunsten von overlay_type verworfen.

Eigenschaften von "timeplotnode"

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. Für weitere Informationen siehe Thema Zeitdiagrammknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create timeplotnode
set :timeplotnode.y_fields = ['sales' 'men' 'women']
set :timeplotnode.panel = True
set :timeplotnode.normalize = True
set :timeplotnode.line = True
set :timeplotnode.smoother = True
set :timeplotnode.use_records_limit = True
set :timeplotnode.records_limit = 2000
# Appearance settings
set :timeplotnode.symbol_size = 2.0
```

timeplotnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
plot_series	Series Models	
use_custom_x_field	*Flag*	
x_field	*Feld*	
y_fields	[*Feld Feld Feld*]	
panel	*Flag*	
normalize	*Flag*	
line	*Flag*	
points	*Flag*	
point_type	Rechteck Punkt Dreieck Hexagon Plus Pentagon Stern BowTie HorizontalDash VerticalDash IronCross Fabrik Haus Kathedrale OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fächer	
smoother	*Flag*	Sie können nur dann Glättungselemente zum Plot hinzufügen, wenn Sie panel auf True setzen.
use_records_limit	*Flag*	
records_limit	*integer*	

timeplotnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
symbol_size	*number*	Gibt eine Symbolgröße an. Beispiel: set :webnode.symbol_size = 5.5 erstellt eine Symbolgröße, die die Standardeinstellung übersteigt.
panel_layout	Horizontal Vertical	

Eigenschaften von "webnode"

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. Für weitere Informationen siehe Thema Netzdiagrammknoten in Kapitel 5 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create webnode
# "Plot" tab
set :webnode.use_directed_web = True
set :webnode.to_field = 'Drug'
set :webnode.fields = ['BP' 'Cholesterol' 'Sex' 'Drug']
set :webnode.from_fields = ['BP' 'Cholesterol' 'Sex']
set :webnode.true_flags_only = False
set :webnode.line_values = Absolute
set :webnode.strong_links_heavier = True
# "Options" tab
set :webnode.max_num_links = 300
set :webnode.links_above = 10
set :webnode.num_links = ShowAll
set :webnode.discard_links_min = True
set :webnode.links_min_records = 5
set :webnode.discard_links_max = True
set :webnode.weak_below = 10
set :webnode.strong_above = 19
set :webnode.link_size_continuous = True
set :webnode.web_display = Circular
```

webnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
use_directed_web	*Flag*	
fields	[*Feld Feld Feld*]	
to_field	*Feld*	
from_fields	[*Feld Feld Feld*]	
true_flags_only	*Flag*	

webnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*Flag*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*number*	
links_above	*number*	
discard_links_min	*Flag*	
links_min_records	*number*	
discard_links_max	*Flag*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*Flag*	
web_display	Circular Network Directed Grid	
graph_background	*Farbe*	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
symbol_size	*number*	Gibt eine Symbolgröße an. Beispiel: set :webnode.symbol_size = 5.5 erstellt eine Symbolgröße, die die Standardeinstellung übersteigt.

Kapitel
16

Modellierungsknoten – Eigenschaften

Modellierungsknoten – Allgemeine Eigenschaften

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

Eigenschaft	Werte	Eigenschaftsbeschreibung
custom_fields	*Flag*	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem weiter oben im Stream gelegenen 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	*Flag*	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
use_split_data	*Flag*	
splits	*[field1 ... fieldN]*	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an. Nur wirksam, wenn use_split_data auf True gesetzt ist.
use_frequency	*Flag*	Gewichts- und Häufigkeitsfelder werden von bestimmten Modellen je nach Angabe für die einzelnen Modelltypen verwendet.
frequency_field	*Feld*	
use_weight	*Flag*	
weight_field	*Feld*	
use_model_name	*Flag*	
model_name	*string*	Benutzerdefinierter Name für neues Modell.
mode	Simple Expert	

Eigenschaften von "anomalydetectionnode"

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. Für weitere Informationen siehe Thema Anomalieerkennungsknoten in Kapitel 4 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create anomalydetectionnode
set :anomalydetectionnode.anomaly_method=PerRecords
set :anomalydetectionnode.percent_records=95
set :anomalydetectionnode.mode=Expert
set :anomalydetectionnode.peer_group_num_auto=true
set :anomalydetectionnode.min_num_peer_groups=3
set :anomalydetectionnode.max_num_peer_groups=10
```

anomalydetectionnode Properties	**Werte**	**Eigenschaftsbeschreibung**
inputs	*[field1 ... fieldN]*	Anomalieerkennungsmodelle führen ein Screening von Datensätzen auf der Grundlage der angegebenen Eingabefelder durch. Sie verwenden kein Zielfeld. Gewichts- und Häufigkeitsfelder werden ebenfalls nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Gibt die Methode für die Bestimmung des Cutoff-Werts zur Kennzeichnung von Datensätzen als anomal an.
index_level	*number*	Gibt den minimalen Cutoff-Wert für die Kennzeichnung von Anomalien an.
percent_records	*number*	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage des Prozentsatzes der Datensätze in den Trainingsdaten fest.
num_records	*number*	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage der Anzahl der Datensätze in den Trainingsdaten fest.
num_fields	*integer*	Die Anzahl der für die einzelnen anomalen Datensätze zu meldenden Felder.
impute_missing_values	*Flag*	

anomalydetectionnode Properties	Werte	Eigenschaftsbeschreibung
adjustment_coeff	*number*	Wert, der zum Balancieren des relativen Gewichts verwendet wird, das den stetigen und den kategorialen Feldern bei der Berechnung der Distanz zugewiesen wird.
peer_group_num_auto	*Flag*	Berechnet automatisch die Anzahl der Vergleichsgruppen.
min_num_peer_groups	*integer*	Gibt an, wie viele Vergleichsgruppen mindestens verwendet werden, wenn peer_group_num_auto auf True gesetzt ist.
max_num_per_groups	*integer*	Gibt die maximale Anzahl an Vergleichsgruppen an.
num_peer_groups	*integer*	Gibt an, wie viele Vergleichsgruppen verwendet werden, wenn peer_group_num_auto auf False gesetzt ist.
noise_level	*number*	Bestimmt, wie Ausreißer bei der Clusterbildung behandelt werden. Geben Sie einen Wert zwischen 0 und 0.5 an.
noise_ratio	*number*	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 von “apriorinode”

Der A-Priori-Knoten extrahiert eine Regelmenge aus den Daten und daraus die Regeln mit dem höchsten Informationsgehalt. A Priori bietet fünf verschiedene Methoden zur Auswahl von Regeln und verwendet ein ausgereiftes Indizierungsschema zur effizienten Verarbeitung großer Daten-Sets. Bei großen Problemen ist A Priori 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 A Priori müssen alle Ein- und Ausgabefelder kategorial sein; dafür bietet es jedoch eine bessere Leistung, da es für diesen Datentyp optimiert ist. Für weitere Informationen siehe Thema A Priori-Knoten in Kapitel 12 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create apriorinode
# "Fields" tab
set :apriorinode.custom_fields = True
set :apriorinode.use_transactional_data = True
set :apriorinode.id_field = 'Age'
set :apriorinode.contiguous = True
set :apriorinode.content_field = 'Drug'
# These seem to have changed, used to be:
#help set :apriorinode.consequents = ['Age']
#help set :apriorinode.antecedents = ['BP' 'Cholesterol' 'Drug']
```

```
# now it seems we have;
#help set :apriorinode.content = ['Age']
set :apriorinode.partition = Test
# "Model" tab
set :apriorinode.use_model_name = False
set :apriorinode.model_name = "Apriori_bp_choles_drug"
set :apriorinode.min_supp = 7.0
set :apriorinode.min_conf = 30.0
set :apriorinode.max_antecedents = 7
set :apriorinode.true_flags = False
set :apriorinode.optimize = Memory
# "Expert" tab
set :apriorinode.mode = Expert
set :apriorinode.evaluation = ConfidenceRatio
set :apriorinode.lower_bound = 7
```

apriorinodeProperties	**Werte**	**Eigenschaftsbeschreibung**
consequents	*Feld*	A Priori-Modelle verwenden anstelle von standardmäßigen Ziel- und Eingabefeldern Sukzedenzien bzw. Antezedenzien. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
antecedents	[*Feld1 ... FeldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*Flag*	
optimize	Speed Memory	
use_transactional_data	*Flag*	
contiguous	*Flag*	
id_field	*string*	
content_field	*string*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften von “autoclassisfiernode”

Mit dem Knoten “Autom. Klassifizierer” 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. Für weitere Informationen siehe Thema Knoten “Automatischer Klassifizierer” in Kapitel 5 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create autoclassifiernode
set :autoclassifiernode.ranking_measure=Accuracy
set :autoclassifiernode.ranking_dataset=Training
set :autoclassifiernode.enable_accuracy_limit=true
set :autoclassifiernode.accuracy_limit=0.9
set :autoclassifiernode.calculate_variable_importance=true
set :autoclassifiernode.use_costs=true
set :autoclassifiernode.svm=false
```

autoclassifiernodeProperties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Für Flag-Ziele verlangt der Knoten “Automatischer Klassifizierer” ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem können Gewichts- und Häufigkeitsfelder angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*integer*	Anzahl der Modelle, die in das Modell-Nugget aufgenommen werden sollen. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
calculate_variable_importance	*Flag*	
enable_accuracy_limit	*Flag*	
accuracy_limit	*integer*	Ganze Zahl zwischen 0 und 100.
enable_ area_under_curve _limit	*Flag*	
area_under_curve_limit	*number*	Reelle Zahl zwischen 0,0 und 1,0.
enable_profit_limit	*Flag*	

autoclassifiernodeProperties	Werte	Eigenschaftsbeschreibung
profit_limit	*number*	Ganze Zahl größer als 0.
enable_lift_limit	*Flag*	
lift_limit	*number*	Reelle Zahl größer 1,0.
enable_number_of_variables_limit	*Flag*	
number_of_variables_limit	*number*	Ganze Zahl größer als 0.
use_fixed_cost	*Flag*	
fixed_cost	*number*	Reelle Zahl größer 0.0.
variable_cost	*Feld*	
use_fixed_revenue	*Flag*	
fixed_revenue	*number*	Reelle Zahl größer 0.0.
variable_revenue	*Feld*	
use_fixed_weight	*Flag*	
fixed_weight	*number*	Reelle Zahl größer als 0,0
variable_weight	*Feld*	
lift_percentile	*number*	Ganze Zahl zwischen 0 und 100.
enable_model_build_time_limit	*Flag*	
model_build_time_limit	*number*	Ganze Zahl für die Anzahl der Minuten, die maximal für die Erstellung jedes einzelnen Modells aufgewendet werden.
enable_stop_after_time_limit	*Flag*	
stop_after_time_limit	*number*	Reelle Zahl für die Anzahl der Stunden, die als Obergrenze für die insgesamt verstrichene Zeit für den Durchlauf eines automatischen Klassifizierers verwendet wird.
enable_stop_after_valid_model_produced	*Flag*	
use_costs	*Flag*	
<algorithm>	*Flag*	Aktiviert oder deaktiviert die Verwendung eines bestimmten Algorithmus, zum Beispiel: set :autoclassifiernode.chaid=true
<algorithm>.<property>	*string*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Für weitere Informationen siehe Thema Festlegen der Algorithmuseigenschaften auf S. 206.

Festlegen der Algorithmuseigenschaften

Für die Knoten vom Typ “Automatischer Klassifizierer”, “Auto-Numerisch” und “Autom. Cluster” können Eigenschaften für bestimmte vom Knoten verwendete Algorithmen mithilfe des folgenden allgemeinen Formats festgelegt werden:

set :autoclassifiernode.<Algorithmus>.<Eigenschaft> = <Wert>

```
set :autonumericnode.<Algorithmus>.<Eigenschaft> = <Wert>

set :autoclusternode.<Algorithmus>.<Eigenschaft> = <Wert>
```

Beispiel:

```
set :autoclassifiernode.neuralnetwork.method = MultilayerPerceptron
```

Die Algorithmusnamen für den Knoten "Automatischer Klassifizierer" 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.

Die Algorithmusnamen für den Knoten "Autom. Cluster" lauten 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, z. B.:

```
set :autoclassifiernode.logreg.tolerance = '1.0E-5'
```

Als Eigenschaft können auch mehrere Werte zugewiesen werden, z. B.:

```
set :autoclassifiernode.decisionlist.search_direction = [Up Down]
```

So können Sie die Verwendung eines bestimmten Algorithmus aktivieren bzw. deaktivieren:

```
set :autoclassifiernode.chaid=true
```

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 "Automatischer Klassifizierer" verfügbar sind oder in denen nur ein einzelner Wert und kein Wertebereich angegeben werden kann, gelten dieselben Einschränkungen bei der Skripterstellung wie beim standardmäßigen Zugriff auf den Knoten.

autoclusternode-Eigenschaften

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 Cluster-Modellen verglichen werden, um ein Maß auf der Basis der Wichtigkeit von bestimmten Feldern zu liefern. Für weitere Informationen siehe Thema Knoten "Autom. Cluster" in Kapitel 5 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create autoclusternode
set :autoclusternode.ranking_measure=Silhouette
set :autoclusternode.ranking_dataset=Training
set :autoclusternode.enable_silhouette_limit=true
set :autoclusternode.silhouette_limit=5
```

autoclusternodeProperties	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	*integer*	Anzahl der im Bericht aufzuführenden Modelle. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
enable_silhouette_limit	*Flag*	
silhouette_limit	*integer*	Ganze Zahl zwischen 0 und 100.
enable_number_less_limit	*Flag*	
number_less_limit	*number*	Reelle Zahl zwischen 0,0 und 1,0.
enable_number_greater_limit	*Flag*	
number_greater_limit	*number*	Ganze Zahl größer als 0.
enable_smallest_cluster_limit	*Flag*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percent-age	*number*	
smallest_cluster_limit_count	*integer*	Ganze Zahl größer als 0.
enable_largest_cluster_limit	*Flag*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percent-age	*number*	
largest_cluster_limit_count	*integer*	
enable_smallest_largest_limit	*Flag*	
smallest_largest_limit	*number*	
enable_importance_limit	*Flag*	
importance_limit_condition	Greater_than Less_than	

autoclusternodeProperties	Werte	Eigenschaftsbeschreibung
importance_limit_greater_than	*number*	Ganze Zahl zwischen 0 und 100.
importance_limit_less_than	*number*	Ganze Zahl zwischen 0 und 100.
<algorithm>	*Flag*	Aktiviert oder deaktiviert die Verwendung eines bestimmten Algorithmus, zum Beispiel: set :autoclusternode.kohonen=true
<algorithm>.<property>	*string*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Für weitere Informationen siehe Thema Festlegen der Algorithmuseigenschaften auf S. 206.

autonumericnode-Eigenschaften

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 "Automatischer Klassifizierer": 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. Für weitere Informationen siehe Thema Knoten "Auto-Numerisch" in Kapitel 5 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create autonumericnode
set :autonumericnode.ranking_measure=Correlation
set :autonumericnode.ranking_dataset=Training
set :autonumericnode.enable_correlation_limit=true
set :autonumericnode.correlation_limit=0.8
set :autonumericnode.calculate_variable_importance=true
set :autonumericnode.neuralnetwork=true
set :autonumericnode.chaid=false
```

autonumericnodeProperties	Werte	Eigenschaftsbeschreibung
custom_fields	*Flag*	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 Gewichts- und Häufigkeitsfelder angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
inputs	*[Feld1 ... Feld2]*	
partition	*Feld*	

autonumericnodeProperties	Werte	Eigenschaftsbeschreibung
use_frequency	*Flag*	
frequency_field	*Feld*	
use_weight	*Flag*	
weight_field	*Feld*	
use_partitioned_data	*Flag*	Wenn ein Partitionsfeld definiert ist, werden nur die Trainingsdaten für die Modellerstellung verwendet.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*integer*	Anzahl der Modelle, die in das Modell-Nugget aufgenommen werden sollen. Geben Sie eine ganze Zahl zwischen 1 und 100 an.
calculate_variable_importance	*Flag*	
enable_correlation_limit	*Flag*	
correlation_limit	*integer*	
enable_number_of_fields_limit	*Flag*	
number_of_fields_limit	*integer*	
enable_relative_error_limit	*Flag*	
relative_error_limit	*integer*	
enable_model_build_time_limit	*Flag*	
model_build_time_limit	*integer*	
enable_stop_after_time_limit	*Flag*	
stop_after_time_limit	*integer*	
stop_if_valid_model	*Flag*	
<algorithm>	*Flag*	Aktiviert oder deaktiviert die Verwendung eines bestimmten Algorithmus, zum Beispiel: set :autonumericnode.chaid=true
<algorithm>.<property>	*string*	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Für weitere Informationen siehe Thema Festlegen der Algorithmuseigenschaften auf S. 206.

Eigenschaften von “bayesnetnode”

Mithilfe des Bayes-Netzwerk-Knotens können Sie ein Wahrscheinlichkeitsmodell erstellen, idem Sie beobachtete und aufgezeichnete Hinweise mit Weltwissen kombinieren, um die Wahrscheinlichkeit ihres Vorkommens zu ermitteln. Der Knoten ist speziell für Netzwerke vom Typ “Tree Augmented Naïve Bayes” (TAN) und “Markov-Decke” gedacht, die in erster Linie zur Klassifizierung verwendet werden. Für weitere Informationen siehe Thema Bayes-Netzwerk-Knoten in Kapitel 7 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create bayesnetnode
set :bayesnetnode.continue_training_existing_model = True
set :bayesnetnode.structure_type = MarkovBlanket
set :bayesnetnode.use_feature_selection = True
# Expert tab
set :bayesnetnode.mode = Expert
set :bayesnetnode.all_probabilities = True
set :bayesnetnode.independence = Pearson
```

bayesnetnode Properties	**Werte**	**Eigenschaftsbeschreibung**
inputs	*[field1 ... fieldN]*	Bayes-Netzwerk-Modelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Stetige Felder werden automatisch klassiert. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue_training_existing_model	*Flag*	
structure_type	TAN MarkovBlanket	Dient zur Auswahl der beim Erstellen des Bayes-Netzwerks zu verwendenden Struktur.
use_feature_selection	*Flag*	
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	*Flag*	
all_probabilities	*Flag*	
independence	Likelihood Pearson	Gibt die Methode an, die zur Einschätzung verwendet wird, ob gepaarte Beobachtungen bei zwei Variablen voneinander unabhängig sind.
significance_level	*number*	Gibt den Trennwert für die Bestimmung der Unabhängigkeit an.
maximal_conditioning_set	*number*	Legt die Maximalzahl der für die Unabhängigkeitstests zu verwendenden Konditionierungsvariablen fest.
inputs_always_selected	*[field1 ... fieldN]*	Gibt an, welche Felder aus dem Daten-Set immer beim Erstellen des Bayes-Netzwerks verwendet werden. *Hinweis*: Das Zielfeld ist immer ausgewählt.

bayesnetnode Properties	Werte	Eigenschaftsbeschreibung
maximum_number_inputs	*number*	Gibt die maximale Anzahl an Eingabefeldern an, die beim Erstellen des Bayes-Netzwerks verwendet werden sollen.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "c50node"

Der C5.0-Knoten erstellt entweder einen Entscheidungsbaum oder ein Regel-Set. 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. Für weitere Informationen siehe Thema C5.0-Knoten in Kapitel 6 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create c50node
# "Model" tab
set :c50node.use_model_name = False
set :c50node.model_name = "C5_Drug"
set :c50node.use_partitioned_data = True
set :c50node.output_type = DecisionTree
set :c50node.use_xval = True
set :c50node.xval_num_folds = 3
set :c50node.mode = Expert
set :c50node.favor = Generality
set :c50node.min_child_records = 3
# "Costs" tab
set :c50node.use_costs = True
set :c50node.costs = [{"drugA" "drugX" 2}]
```

c50nodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	C50-Modelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
output_type	DecisionTree RuleSet	
group_symbolics	*Flag*	
use_boost	*Flag*	
boost_num_trials	*number*	
use_xval	*Flag*	

c50nodeProperties	Werte	Eigenschaftsbeschreibung
xval_num_folds	*number*	
mode	Simple Expert	
favor	Accuracy Generality	Genauigkeit oder Allgemeingültigkeit werden vorselektiert.
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*Flag*	
costs	*strukturiert*	Hierbei handelt es sich um eine strukturierte Eigenschaft.
use_winnowing	*Flag*	
use_global_pruning	*Flag*	Standardmäßig auf True gesetzt.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "carmanode"

Beim CARMA-Modell wird eine Regelmenge aus den Daten extrahiert, ohne dass Sie Eingabe- oder Ziel-Felder angeben müssen. Im Gegensatz zu A Priori bietet der CARMA-Knoten Erstellungseinstellungen für die Regelunterstützung (Unterstützung für Antezedens und Sukzedens) und nicht nur für die Antezedens-Unterstützung. Die erstellten Regeln 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. Für weitere Informationen siehe Thema CARMA-Knoten in Kapitel 12 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create carmanode
# "Fields" tab
set :carmanode.custom_fields = True
set :carmanode.use_transactional_data = True
set :carmanode.inputs = ['BP' 'Cholesterol' 'Drug']
set :carmanode.partition = Test
# "Model" tab
set :carmanode.use_model_name = False
set :carmanode.model_name = "age_bp_drug"
set :carmanode.use_partitioned_data = False
set :carmanode.min_supp = 10.0
set :carmanode.min_conf = 30.0
set :carmanode.max_size = 5
# Expert Options
set :carmanode.mode = Expert
```

```
#help  set :carmanode.exclude_simple = True
set :carmanode.use_pruning = True
set :carmanode.pruning_value = 300
set :carmanode.vary_support = True
set :carmanode.estimated_transactions = 30
set :carmanode.rules_without_antecedents = True
```

carmanodeProperties	Werte	Eigenschaftsbeschreibung
inputs	*[field1 ... fieldn]*	CARMA-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
id_field	*Feld*	Das Feld wird als ID-Feld für die Modellerstellung verwendet.
contiguous	*Flag*	Dient zur Angabe, ob IDs im ID-Feld zusammenhängend sind.
use_transactional_data	*Flag*	
content_field	*Feld*	
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	*number*	Der Standardwert ist 10.
mode	Simple Expert	Der Standardwert ist Simple.
exclude_multiple	*Flag*	Schließt Regeln mit mehreren Sukzedenzien aus. Der Standardwert ist False.
use_pruning	*Flag*	Der Standardwert ist False.
pruning_value	*number*	Der Standardwert ist 500.
vary_support	*Flag*	
estimated_transactions	*integer*	
rules_without_antecedents	*Flag*	

Eigenschaften von “cartnode”

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). Für weitere Informationen siehe Thema C&R-Baumknoten in Kapitel 6 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create cartnode
# "Fields" tab
set :cartnode.custom_fields = True
set :cartnode.target = 'Drug'
set :cartnode.inputs = ['Age' 'BP' 'Cholesterol']
# "Build Options" tab, 'Objective' panel
set :cartnode.model_output_type = InteractiveBuilder
set :cartnode.use_tree_directives = True
set :cartnode.tree_directives = """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4"""
# "Build Options" tab, 'Basics' panel
set :cartnode.prune_tree = False
set :cartnode.use_std_err_rule = True
set :cartnode.std_err_multiplier = 3.0
set :cartnode.max_surrogates = 7
# "Build Options" tab, 'Stopping Rules' panel
set :cartnode.use_percentage = True
set :cartnode.min_parent_records_pc = 5
set :cartnode.min_child_records_pc = 3
# "Build Options" tab, 'Costs & Priors' panel
set :cartnode.use_costs = True
set :cartnode.costs =  [{"drugA" "drugX" 2}]
set :cartnode.priors = Custom
# custom priors must add to 1
set :cartnode.custom_priors = [{"drugA" 0.3}{"drugX" 0.7}]
set :cartnode.adjust_priors = True
# "Build Options" tab, 'Advanced' panel
set :cartnode.min_impurity = 0.0003
set :cartnode.impurity_measure = Twoing
# "Model Options" tab
set :cartnode.use_model_name = False
set :cartnode.model_name = "Cart_Drug"
```

cartnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Modelle vom Typ “C&RT-Baum” verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue_training_exist-ing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm wird für sehr umfangreiche Daten-Sets verwendet und erfordert eine Server-Verbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	

cartnodeProperties	Werte	Eigenschaftsbeschreibung
tree_directives	*string*	Geben Sie Aufbauregeln für die Erweiterung des Baums an. Aufbauregeln können in dreifache Anführungszeichen gesetzt werden, um auf Escape-Zeichen 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 Daten-Sets verallgemeinert werden können. Für weitere Informationen siehe Thema Aufbauregeln für Bäume in Kapitel 6 in *IBM SPSS Modeler 15 Modellierungsknoten*.
use_max_depth	Default Custom	
max_depth	*integer*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	*Flag*	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	*Flag*	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	*number*	Maximale Differenz.
max_surrogates	*number*	Maximale Anzahl Ersatztrenner.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in Klammermn ({}) die tatsächlich prognostizierten Kosten darstellen.
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft im Format: set :cartnode. custom_priors = [{ drugA 0.3 } { drugB 0.6 }]
adjust_priors	*Flag*	
trails	*number*	Anzahl der Komponentenmodelle für Verbesserung oder Verstärkung.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standard-Kombinierungsregel für kategoriale Ziele.

cartnodeProperties	Werte	Eigenschaftsbeschreibung
range_ensemble_method	Mean Median	Standard-Kombinierungsregel für stetige Ziele.
large_boost	*Flag*	Verbesserung auf sehr große Daten-Sets anwenden.
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	
train_pct	*number*	Verhinderung der übermäßigen Anpassung eingestellt.
set_random_seed	*Flag*	Option “Ergebnisse reproduzieren”.
seed	*number*	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von “chaidnode”

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 nichtbinäre 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. Für weitere Informationen siehe Thema CHAID-Knoten in Kapitel 6 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create chaidnode
set :chaidnode.custom_fields = True
set :chaidnode.target = Drug
set :chaidnode.inputs = [Age Na K Cholesterol BP]
set :chaidnode.use_model_name = true
set :chaidnode.model_name = "CHAID"
set :chaidnode.method = Chaid
set :chaidnode.model_output_type = InteractiveBuilder
set :chaidnode.use_tree_directives = True
set :chaidnode.tree_directives = "Test"
set :chaidnode.mode = Expert
set :chaidnode.split_alpha = 0.03
set :chaidnode.merge_alpha = 0.04
set :chaidnode.chi_square = Pearson
set :chaidnode.use_percentage = True
set :chaidnode.min_parent_records_abs = 40
set :chaidnode.min_child_records_abs = 30
set :chaidnode.epsilon = 0.003
```

```
set :chaidnode.max_iterations = 75
set :chaidnode.split_merged_categories = true
set :chaidnode.bonferroni_adjustment = true
```

chaidnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	CHAID-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue_training_existing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm wird für sehr umfangreiche Daten-Sets verwendet und erfordert eine Server-Verbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	
tree_directives	*string*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*integer*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in Klammermn ({}) die tatsächlich prognostizierten Kosten darstellen.
trails	*number*	Anzahl der Komponentenmodelle für Verbesserung oder Verstärkung.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standard-Kombinierungsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standard-Kombinierungsregel für stetige Ziele.
large_boost	*Flag*	Verbesserung auf sehr große Daten-Sets anwenden.
split_alpha	*number*	Signifikanzschwelle für Aufteilung.
merge_alpha	*number*	Signifikanzschwelle für Zusammenführung.

chaidnodeProperties	Werte	Eigenschaftsbeschreibung
bonferroni_adjustment	*Flag*	Signifikanzwerte mit der Bonferroni-Methode anpassen.
split_merged_categories	*Flag*	Erneutes Aufteilen zusammengeführter Kategorien zulassen.
chi_square	Pearson LR	Verwendetes Verfahren für die Berechnung der Chi-Quadrat-Statistik: Pearson oder Likelihood-Quotient
epsilon	*number*	Minimale Änderung in der erwarteten Zellhäufigkeit...
max_iterations	*number*	Maximale Anzahl der Iterationen für Konvergenz.
set_random_seed	*integer*	
seed	*number*	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensi-ties	*Flag*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*integer*	

Eigenschaften von “coxregnode”

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. Für weitere Informationen siehe Thema Cox-Knoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create coxregnode
set :coxregnode.survival_time = tenure
set :coxregnode.method = BackwardsStepwise
# Expert tab
set :coxregnode.mode = Expert
set :coxregnode.removal_criterion = Conditional
```

set :coxregnode.survival = True

coxregnode Properties	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. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
method	Enter Stepwise BackwardsStepwise	
groups	*Feld*	
model_type	MainEffects Custom	
custom_terms	[*"BP*Sex" "BP*Age"*]	Beispiel: set :coxregnode. custom_terms = ["BP*Sex" "BP" "Age"]
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*Flag*	
ci_value	90 95 99	

coxregnode Properties	Werte	Eigenschaftsbeschreibung
correlation	*Flag*	
display_baseline	*Flag*	
survival	*Flag*	
hazard	*Flag*	
log_minus_log	*Flag*	
one_minus_survival	*Flag*	
separate_line	*Feld*	
value	*Zahl* oder *String*	Wenn für ein Feld kein Wert angegeben ist, wird die Standardoption "Mittelwert" für das betreffende Feld verwendet. Syntax für ein numerisches Feld: coxnode.value = [{"age" "35.8"}] Syntax für ein kategoriales Feld: coxnode.value = [{"color" "pink"}]

Eigenschaften von decisionlistnode

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 Geschäftswissen 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. Für weitere Informationen siehe Thema Entscheidungsliste in Kapitel 9 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create decisionlistnode
set :decisionlistnode.search_direction=Down
set :decisionlistnode.target_value=1
set :decisionlistnode.max_rules=4
set :decisionlistnode.min_group_size_pct = 15
```

decisionlistnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	Entscheidungslistenmodelle verwenden ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
model_output_type	Model InteractiveBuilder	

decisionlistnodeProperties	Werte	Eigenschaftsbeschreibung
search_direction	Up Down	Bezieht sich auf das Finden von Segmenten; dabei entspricht “Up” einer hohen Wahrscheinlichkeit und “Down” einer geringen Wahrscheinlichkeit.
target_value	*string*	Wenn dieser Wert nicht angegeben wird, nimmt er für Flags den Wert “True” (Wahr) an.
max_rules	*integer*	Die maximale Anzahl der Segmente ausschließlich des Rests.
min_group_size	*integer*	Mindestsegmentgröße.
min_group_size_pct	*number*	Mindestsegmentgröße als Prozentsatz.
confidence_level	*number*	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	*integer*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*integer*	Suchbreite für Listen.
max_rules_per_cycle	*integer*	Suchbreite für Segmentregeln.
segment_growth	*number*	
include_missing	*Flag*	
final_results_only	*Flag*	
reuse_fields	*Flag*	Ermöglicht die Wiederverwendung von Attributen (Eingabefelder, die in Regeln vorkommen).
max_alternatives	*integer*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von discriminantnode

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. Für weitere Informationen siehe Thema Diskriminanzknoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create discriminantnode
set :discriminantnode.target = custcat
```

set :discriminantnode.use_partitioned_data = False
set :discriminantnode.method = Stepwise

discriminantnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	Diskriminanzmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*Flag*	Statistikoptionen im Dialogfeld “Erweiterte Ausgabe”.
univariate_anovas	*Flag*	
box_m	*Flag*	
within_group_covariance	*Flag*	
within_groups_correlation	*Flag*	
separate_groups_covariance	*Flag*	
total_covariance	*Flag*	
fishers	*Flag*	
unstandardized	*Flag*	
casewise_results	*Flag*	Klassifizierungsoptionen im Dialogfeld “Erweiterte Ausgabe”.
limit_to_first	*number*	Der Standardwert ist 10.
summary_table	*Flag*	
leave_one_classification	*Flag*	
combined_groups	*Flag*	
separate_groups_covariance	*Flag*	Matrizenoption Gruppenspezifische Kovarianzmatrix
territorial_map	*Flag*	
combined_groups	*Flag*	Plotoption Kombinierte Gruppen.
separate_groups	*Flag*	Plotoption Gruppenspezifisch.
summary_of_steps	*Flag*	
F_pairwise	*Flag*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	

discriminantnodeProperties	Werte	Eigenschaftsbeschreibung
F_value_entry	*number*	Der Standardwert ist 3,84.
F_value_removal	*number*	Der Standardwert ist 2,71.
probability_entry	*number*	Der Standardwert ist 0.05.
probability_removal	*number*	Der Standardwert ist 0.10.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "factornode"

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. Für weitere Informationen siehe Thema Faktor/PCA-Knoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create factornode
# "Fields" tab
set :factornode.custom_fields = True
set :factornode.inputs = ['BP' 'Na' 'K']
set :factornode.partition = Test
# "Model" tab
set :factornode.use_model_name = True
set :factornode.model_name = "Factor_Age"
set :factornode.use_partitioned_data = False
set :factornode.method = GLS
# Expert options
set :factornode.mode = Expert
set :factornode.complete_records = true
set :factornode.matrix = Covariance
set :factornode.max_iterations = 30
set :factornode.extract_factors = ByFactors
set :factornode.min_eigenvalue = 3.0
set :factornode.max_factor = 7
set :factornode.sort_values = True
set :factornode.hide_values = True
set :factornode.hide_below = 0.7
# "Rotation" section
set :factornode.rotation = DirectOblimin
set :factornode.delta = 0.3
```

set :factornode.kappa = 7.0

factornodeProperties	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	PCA-/Faktor-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*Flag*	
matrix	Correlation Covariance	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	*number*	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	*number*	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	*Flag*	
hide_values	*Flag*	
hide_below	*number*	

Eigenschaften von "featureselectionnode"

Der Merkmalsauswahlknoten 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 Daten-Set mit Hunderten potenzieller Eingaben. Welche davon sind voraussichtlich für die Modellierung von medizinischen Behandlungsergebnissen von Bedeutung? Für weitere Informationen siehe Thema Merkmalsauswahlknoten in Kapitel 4 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create featureselectionnode
set :featureselectionnode.screen_single_category=true
set :featureselectionnode.max_single_category=95
set :featureselectionnode.screen_missing_values=true
set :featureselectionnode.max_missing_values=80
set :featureselectionnode.criteria = Likelihood
set :featureselectionnode.unimportant_below = 0.8
set :featureselectionnode.important_above = 0.9
set :featureselectionnode.important_label = "Check Me Out!"
set :featureselectionnode.selection_mode = TopN
set :featureselectionnode.top_n = 15
```

Ein detaillierteres Beispiel, mit dem ein Merkmalsauswahlmodell erstellt und angewendet wird, finden Sie unter Beispiel für Standalone-Skript: Generieren eines Merkmalsauswahlmodells in Kapitel 2 auf S. 14.

featureselectionnode Properties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Merkmalsauswahlmodelle teilen Prädiktoren relativ zum angegebenen Ziel in Ränge ein. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
screen_single_category	*Flag*	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	*number*	Gibt an, welcher Schwellenwert verwendet wird, wenn screen_single_category auf True gesetzt ist.
screen_missing_values	*Flag*	Bei True wird ein Screening der Felder durchgeführt, die zu viele fehlende Werte (ausgedrückt als Prozentsatz der Gesamtzahl an Datensätzen) aufweisen.
max_missing_values	*number*	

featureselectionnode Properties	**Werte**	**Eigenschaftsbeschreibung**
screen_num_categories	*Flag*	Bei True wird ein Screening der Felder durchgeführt, die zu viele Kategorien im Verhältnis zur Gesamtzahl der Datensätze aufweisen.
max_num_categories	*number*	
screen_std_dev	*Flag*	Bei True wird ein Screening der Felder durchgeführt, deren Standardabweichung kleiner oder gleich dem angegebenen Mindestwert ist.
min_std_dev	*number*	
screen_coeff_of_var	*Flag*	Bei True wird ein Screening der Felder durchgeführt, deren Varianzkoeffizient kleiner oder gleich dem angegebenen Mindestwert ist.
min_coeff_of_var	*number*	
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	*number*	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	*number*	Zulässig sind Werte von 0,0 bis 1,0.
unimportant_label	*string*	Gibt die Beschriftung für die Rangstufe “unbedeutsam” an.
marginal_label	*string*	
important_label	*string*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.
select_marginal	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
select_unimportant	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob unwichtige Felder ausgewählt werden sollen.

featureselectionnode Properties	Werte	Eigenschaftsbeschreibung
importance_value	*number*	Wenn selection_mode auf ImportanceValue gesetzt ist, wird hier der zu verwendende Cutoff-Wert angegeben. Zulässig sind Werte von 0 bis 100.
top_n	*integer*	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Cutoff-Wert angegeben. Zulässig sind Werte von 0 bis 1000.

Eigenschaften von genlinnode

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. Für weitere Informationen siehe Thema GenLin-Knoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create genlinnode
set :genlinnode.model_type = MainAndAllTwoWayEffects
set :genlinnode.offset_type = Variable
set :genlinnode.offset_field = Claimant
```

genlinnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	Verallgemeinerte lineare Modelle erfordern ein einzelnes Zielfeld, bei dem es sich um ein nominales oder ein Flag-Feld handeln muss, und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
use_weight	*Flag*	
weight_field	*Feld*	Der Feldtyp ist nur stetig.
target_represents_trials	*Flag*	
trials_type	Variable FixedValue	
trials_field	*Feld*	Der Feldttyp ist stetig, Flag oder ordinal.
trials_number	*number*	Der Standardwert ist 10.
model_type	MainEffects MainAndAllTwoWayEffects	

genlinnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
offset_type	Variable FixedValue	
offset_field	*Feld*	Der Feldtyp ist nur stetig.
offset_value	*number*	Muss eine reelle Zahl sein.
base_category	Last First	
include_intercept	*Flag*	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Invers normal. NEGBIN: Negativ binomial.
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	Der Standardwert ist 1.Muss eine nichtnegative reelle Zahl enthalten.
tweedie_parameter	*number*	
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	*number*	Der Wert muss eine reelle Zahl ungleich null sein.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*number*	Der Standardwert ist 1; nur positive ganze Zahlen zulässig.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*number*	Der Standardwert ist 1; muss größer als 0 sein.
covariance_matrix	ModelEstimator RobustEstimator	

genlinnodeProperties	Werte	Eigenschaftsbeschreibung
max_iterations	*number*	Der Standardwert ist 100; nur nichtnegative ganze Zahlen.
max_step_halving	*number*	Der Standardwert ist 5; nur positive ganze Zahlen.
check_separation	*Flag*	
start_iteration	*number*	Der Standardwert ist 20; nur positive ganze Zahlen zulässig.
estimates_change	*Flag*	
estimates_change_min	*number*	Der Standardwert ist 1E-006; nur positive Zahlen zulässig.
estimates_change_type	Absolute Relative	
loglikelihood_change	*Flag*	
loglikelihood_change_min	*number*	Nur positive Zahlen zulässig.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*Flag*	
hessian_convergence_min	*number*	Nur positive Zahlen zulässig.
hessian_convergence_type	Absolute Relative	
case_summary	*Flag*	
contrast_matrices	*Flag*	
descriptive_statistics	*Flag*	
estimable_functions	*Flag*	
model_info	*Flag*	
iteration_history	*Flag*	
goodness_of_fit	*Flag*	
print_interval	*number*	Der Standardwert ist 1; muss eine positive ganze Zahl sein.
model_summary	*Flag*	
lagrange_multiplier	*Flag*	
parameter_estimates	*Flag*	
include_exponential	*Flag*	
covariance_estimates	*Flag*	
correlation_estimates	*Flag*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*number*	Der Standardwert ist 0,0001.
confidence_interval	*number*	Der Standardwert ist 95.
loglikelihood_function	Full Kernel	

genlinnodeProperties	Werte	Eigenschaftsbeschreibung
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "glmmnode"

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. Für weitere Informationen siehe Thema GLMM-Knoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

glmmnodeProperties	Werte	Eigenschaftsbeschreibung
residual_subject_spec	*strukturiert*	Die Wertekombination der angegebenen kategorialen Felder, die Subjekte innerhalb des Daten-Sets 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	*Flag*	Gibt an, ob das im übergeordneten 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.

glmmnodeProperties	Werte	Eigenschaftsbeschreibung
use_trials	*Flag*	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 lautet false.
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	*integer*	Wert zur Angabe der Anzahl an Tests. Wenn angegeben, ist der Minimalwert 1.
use_custom_target_reference	*Flag*	Gibt an, ob eine benutzerdefinierte Referenzkategorie für ein kategoriales Ziel verwendet werden soll. Die Standardeinstellung lautet false.
target_reference_value	*string*	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.
link_function_param	*number*	Für die Verknüpfungsfunktion zu verwendender Parameterwert. Wird nur verwendet, wenn link_function auf POWER gesetzt ist.

glmmnodeProperties	Werte	Eigenschaftsbeschreibung
use_predefined_inputs	*Flag*	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 lautet false.
fixed_effects_list	*strukturiert*	Wenn use_predefined_inputs auf false gesetzt ist, werden die Eingabefelder als Felder für feste Effekte verwendet.
use_intercept	*Flag*	Wenn true gesetzt ist (Standard), 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. Bei dem Wert None wird kein Offset verwendet.
offset_value	*number*	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. Bei dem Wert Data wird die Sortierreihenfolge der Daten verwendet. Die Standardeinstellung lautet Ascending.
inputs_category_order	Ascending Descending Data	Sortierreihenfolge für kategoriale Prädiktoren. Bei dem Wert Data wird die Sortierreihenfolge der Daten verwendet. Die Standardeinstellung lautet Ascending.
max_iterations	*integer*	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden. Eine nichtnegative ganze Zahl. Der Standardwert ist 100.
confidence_level	*integer*	Konfidenzniveau für die Berechnung von Intervallschätzungen der Modellkoeffizienten. Eine nichtnegative 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.

glmmnodeProperties	Werte	Eigenschaftsbeschreibung
use_model_name	*Flag*	Gibt an, ob ein benutzerdefinierter Name (true) oder ein vom System erzeugter Name (false) für das Modell verwendet werden soll. Die Standardeinstellung lautet false.
model_name	*string*	Wenn use_model_name auf true gesetzt ist, wird hier der Modellname angegeben.
confidence	highest difference	Grundlage für die Berechnung des Konfidenzwerts für das Scoring: höchste vorhergesagte Wahrscheinlichkeit oder Differenz zwischen der höchsten und zweithöchsten vorhergesagten Wahrscheinlichkeit.
score_category_probabilities	*Flag*	Wenn auf true gesetzt, werden die vorhergesagten Wahrscheinlichkeiten für kategoriale Ziele generiert. Die Standardeinstellung lautet false.
max_categories	*integer*	Wenn score_category_probabilities auf true gesetzt ist, wird hier die maximale Anzahl der zu speichernden Kategorien festgelegt.
score_propensity	*Flag*	Wenn auf true gesetzt, werden Neigungs-Scores für Flag-Zielfelder 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.
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 von “kmeansnode”

Der K-Means-Knoten teilt das Daten-Set in unterschiedliche Gruppen (oder Cluster) auf. Bei diesem Verfahren wird eine festgelegte Anzahl von Clustern definiert, den Clustern werden iterativ Datensätze zugewiesen und die Cluster-Zentren 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. Für weitere Informationen siehe Thema K-Means-Knoten in Kapitel 11 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create kmeansnode
# "Fields" tab
set :kmeansnode.custom_fields = True
set :kmeansnode.inputs = ['Cholesterol' 'BP' 'Drug' 'Na' 'K' 'Age']
# "Model" tab
set :kmeansnode.use_model_name = False
set :kmeansnode.model_name = "Kmeans_allinputs"
set :kmeansnode.num_clusters = 9
set :kmeansnode.gen_distance = True
set :kmeansnode.cluster_label = "Number"
set :kmeansnode.label_prefix = "Kmeans_"
set :kmeansnode.optimize = Speed
# "Expert" tab
set :kmeansnode.mode = Expert
set :kmeansnode.stop_on = Custom
set :kmeansnode.max_iterations = 10
set :kmeansnode.tolerance = 3.0
set :kmeansnode.encoding_value = 0.3
```

kmeansnodeProperties	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	K-Means-Modelle führen eine Clusteranalyse an einer Menge von Eingabefeldern durch, verwenden jedoch kein Zielfeld. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
num_clusters	*number*	
gen_distance	*Flag*	
cluster_label	String Number	
label_prefix	*string*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	

kmeansnodeProperties	Werte	Eigenschaftsbeschreibung
encoding_value	*number*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften von "knnnode"

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 Ganzzahl ist. Ähnliche Fälle liegen nah beieinander und Fälle mit geringer Ähnlichkeit sind weit voneinander entfernt. Für weitere Informationen siehe Thema KNN-Knoten in Kapitel 16 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create knnnode
# Objectives tab
set: knnnode.objective = Custom
# Settings tab - Neighbors panel
set: knnnode.automatic_k_selection = false
set: knnnode.fixed_k = 2
set: knnnode.weight_by_importance = True
# Settings tab - Analyze panel
set: knnnode.save_distances = True
```

knnnodeProperties	Werte	Eigenschaftsbeschreibung
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*Flag*	
use_case_labels	*Flag*	Kontrollkästchen markieren, um nächste Option zu aktivieren.
case_labels_field	*Feld*	
identify_focal_cases	*Flag*	Kontrollkästchen markieren, um nächste Option zu aktivieren.
focal_cases_field	*Feld*	
automatic_k_selection	*Flag*	
fixed_k	*integer*	Nur aktiviert, wenn automatic_k_selectio auf False gesetzt ist.
minimum_k	*integer*	Nur aktiviert, wenn automatic_k_selectio auf True gesetzt ist.
maximum_k	*integer*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*Flag*	

knnnodeProperties	Werte	Eigenschaftsbeschreibung
range_predictions	Mean Median	
perform_feature_selection	*Flag*	
forced_entry_inputs	[*Feld1 ... FeldN*]	
stop_on_error_ratio	*Flag*	
number_to_select	*integer*	
minimum_change	*number*	
validation_fold_assign_by_field	*Flag*	
number_of_folds	*integer*	Nur aktiviert, wenn validation_fold_assign_by_field auf False gesetzt ist.
set_random_seed	*Flag*	
random_seed	*number*	
folds_field	*Feld*	Nur aktiviert, wenn validation_fold_assign_by_field auf True gesetzt ist.
all_probabilities	*Flag*	
save_distances	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "kohonennode"

Der Kohonen-Knoten erstellt eine Art von neuronalem Netzwerk, das verwendet werden kann, um ein Clustering der Datenmenge 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 Modell-Nugget erfassten Beobachtungen gibt Aufschluss über die starken Einheiten. Dadurch wird ein Eindruck von der ungefähren Zahl der Cluster vermittelt. Für weitere Informationen siehe Thema Kohonen-Knoten in Kapitel 11 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create kohonennode
# "Model" tab
set :kohonennode.use_model_name = False
set :kohonennode.model_name = "Symbolic Cluster"
set :kohonennode.stop_on = Time
set :kohonennode.time = 1
set :kohonennode.set_random_seed = True
set :kohonennode.random_seed = 12345
set :kohonennode.optimize = Speed
# "Expert" tab
set :kohonennode.mode = Expert
set :kohonennode.width = 3
set :kohonennode.length = 3
set :kohonennode.decay_style = Exponential
```

```
set :kohonennode.phase1_neighborhood = 3
set :kohonennode.phase1_eta = 0.5
set :kohonennode.phase1_cycles = 10
set :kohonennode.phase2_neighborhood = 1
set :kohonennode.phase2_eta = 0.2
set :kohonennode.phase2_cycles = 75
```

kohonennodeProperties	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	Kohonen-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Häufigkeits- und Gewichtsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue	*Flag*	
show_feedback	*Flag*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
cluster_label	*Flag*	
mode	Simple Expert	
width	*number*	
length	*number*	
decay_style	Linear Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

Eigenschaften von "linearnode"

Bei linearen Regressionsmodellen wird ein stetiges Ziel auf der Basis linearer Beziehungen zwischen dem Ziel und einem oder mehreren Prädiktoren vorhergesagt. Für weitere Informationen siehe Thema Lineare Modelle in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create linearnode
# Build Options tab - Objectives panel
set: linearnode.objective = Standard
```

```
# Build Options tab - Model Selection panel
set: linearnode.model_selection = BestSubsets
set: linearnode.criteria_best_subsets = ASE
# Build Options tab - Ensembles panel
set: linearnode.combining_rule_categorical = HighestMeanProbability
```

Eigenschaften von linearnode	Werte	Eigenschaftsbeschreibung
Ziel	*Feld*	Gibt ein einzelnes Zielfeld an.
inputs	[*Feld1 ... FeldN*]	Im Modell verwendete Prädiktorfelder.
continue_training_existing_model	*Flag*	
objective	Standard Bagging Boosting PSM	PSM wird für sehr umfangreiche Daten-Sets verwendet und erfordert eine Server-Verbindung.
use_auto_data_preparation	*Flag*	
confidence_level	*number*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*Flag*	
max_effects	*number*	
use_max_steps	*Flag*	
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mittelwert Median	
component_models_n	*number*	
use_random_seed	*Flag*	
random_seed	*number*	
use_custom_model_name	*Flag*	
custom_model_name	*string*	
use_custom_name	*Flag*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

Eigenschaften von “logregnode”

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. Für weitere Informationen siehe Thema Logistikknoten in Kapitel 10 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel für ein multinomiales Modell

```
create logregnode
# "Fields" tab
set :logregnode.custom_fields = True
set :logregnode.target = 'Drug'
set :logregnode.inputs = ['BP' 'Cholesterol' 'Age']
set :logregnode.partition = Test
# "Model" tab
set :logregnode.use_model_name = False
set :logregnode.model_name = "Log_reg Drug"
set :logregnode.use_partitioned_data = True
set :logregnode.method = Stepwise
set :logregnode.logistic_procedure = Multinomial
set :logregnode.multinomial_base_category = BP
set :logregnode.model_type = FullFactorial
set :logregnode.custom_terms = [{BP Sex}{Age}{Na K}]
set :logregnode.include_constant = False
# "Expert" tab
set :logregnode.mode = Expert
set :logregnode.scale = Pearson
set :logregnode.scale_value = 3.0
set :logregnode.all_probabilities = True
set :logregnode.tolerance = "1.0E-7"
# "Convergence..." section
set :logregnode.max_iterations = 50
set :logregnode.max_steps = 3
set :logregnode.l_converge = "1.0E-3"
set :logregnode.p_converge = "1.0E-7"
set :logregnode.delta = 0.03
# "Output..." section
set :logregnode.summary = True
set :logregnode.likelihood_ratio = True
set :logregnode.asymptotic_correlation = True
set :logregnode.goodness_fit = True
set :logregnode.iteration_history = True
set :logregnode.history_steps = 3
set :logregnode.parameters = True
set :logregnode.confidence_interval = 90
set :logregnode.asymptotic_covariance = True
set :logregnode.classification_table = True
# "Stepping" options
set :logregnode.min_terms = 7
set :logregnode.use_max_terms = true
```

```
set :logregnode.max_terms = 10
set :logregnode.probability_entry = 3
set :logregnode.probability_removal = 5
set :logregnode.requirements = Containment
```

Beispiel für ein binomiales Modell

```
create logregnode
# "Fields" tab
set :logregnode.custom_fields = True
set :logregnode.target = 'Cholesterol'
set :logregnode.inputs = ['BP' 'Drug' 'Age']
set :logregnode.partition = Test
# "Model" tab
set :logregnode.use_model_name = False
set :logregnode.model_name = "Log_reg Cholesterol"
set :logregnode.multinomial_base_category = BP
set :logregnode.use_partitioned_data = True
set :logregnode.binomial_method = Forwards
set :logregnode.logistic_procedure = Binomial
set :logregnode.binomial_categorical_input = Sex
set :logregnode.binomial_input_contrast.Sex = Simple
set :logregnode.binomial_input_category.Sex = Last
set :logregnode.include_constant = False
# "Expert" tab
set :logregnode.mode = Expert
set :logregnode.scale = Pearson
set :logregnode.scale_value = 3.0
set :logregnode.all_probabilities = True
set :logregnode.tolerance = "1.0E-7"
# "Convergence..." section
set :logregnode.max_iterations = 50
set :logregnode.l_converge = "1.0E-3"
set :logregnode.p_converge = "1.0E-7"
# "Output..." section
set :logregnode.binomial_output_display = at_each_step
set :logregnode.binomial_goodness_fit = True
set :logregnode.binomial_iteration_history = True
set :logregnode.binomial_parameters = True
set :logregnode.binomial_ci_enable = True
set :logregnode.binomial_ci = 85
# "Stepping" options
set :logregnode.binomial_removal_criterion = LR
```

set :logregnode.binomial_probability_removal = 0.2

logregnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Logistische Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Häufigkeits- und Gewichtsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
logistic_procedure	Binomial Multinomial	
include_constant	*Flag*	
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 (Einschluss) verwendet. Wenn der Modelltyp auf Custom gesetzt ist, jedoch keine benutzerdefinierten Felder angegeben wurden, wird ein Haupteffektmodell erstellt.
custom_terms	[*{BP Geschlecht}{BP}{Alter}*]	Beispiel: set :logregnode. custom_terms = [{Na} {K} {Na K}]
multinomial_base_category	*string*	Gibt an, wie die Referenzkategorie bestimmt wird.
binomial_categorical_input	*string*	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Schlüsseleigenschaft für kategoriale Eingaben, die angibt, wie der Kontrast bestimmt wird. Format: NODE.binomial_input_contrast.FIELD-NAME
binomial_input_category	First Last	Schlüsseleigenschaft für kategoriale Eingaben, die angibt, wie die Referenzkategorie bestimmt wird. Format: NODE.binomial_input_category.FIELD-NAME

logregnodeProperties	Werte	Eigenschaftsbeschreibung
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*Flag*	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	*Flag*	
max_terms	*number*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*number*	
max_steps	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	*Flag*	
history_steps	*number*	
summary	*Flag*	
likelihood_ratio	*Flag*	
asymptotic_correlation	*Flag*	
goodness_fit	*Flag*	
parameters	*Flag*	

logregnodeProperties	Werte	Eigenschaftsbeschreibung
confidence_interval	*number*	
asymptotic_covariance	*Flag*	
classification_table	*Flag*	
stepwise_summary	*Flag*	
info_criteria	*Flag*	
monotonicity_measures	*Flag*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*Flag*	
binomial_parameters	*Flag*	
binomial_iteration_history	*Flag*	
binomial_classification_plots	*Flag*	
binomial_ci_enable	*Flag*	
binomial_ci	*number*	
binomial_residual	outliers all	
binomial_residual_enable	*Flag*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	

Eigenschaften von "neuralnetnode"

Vorsicht: In dieser Version ist eine neuere Fassung des Netzwerk-Modellierungsknotens 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 Skripts zur Verwendung der neuen Version. Details der vorherigen Version werden hier aus Referenzgründen aufbewahrt.

Beispiel

```
create neuralnetnode
# "Fields" tab
set :neuralnetnode.custom_fields = True
set :neuralnetnode.targets = ['Drug']
set :neuralnetnode.inputs = ['Age' 'Na' 'K' 'Cholesterol' 'BP']
# "Model" tab
set :neuralnetnode.use_partitioned_data = True
set :neuralnetnode.method = Dynamic
set :neuralnetnode.train_pct = 30
set :neuralnetnode.set_random_seed = True
set :neuralnetnode.random_seed = 12345
set :neuralnetnode.stop_on = Time
```

```
set :neuralnetnode.accuracy = 95
set :neuralnetnode.cycles = 200
set :neuralnetnode.time = 3
set :neuralnetnode.optimize = Speed
# "Multiple Method Expert Options" section
set :neuralnetnode.m_topologies = "5 30 5; 2 20 3, 1 10 1"
set :neuralnetnode.m_non_pyramids = False
set :neuralnetnode.m_persistence = 100
```

neuralnetnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
targets	[*Feld1 ... FeldN*]	Der Netzwerkknoten erwartet mindestens ein Zielfeld und mindestens ein Eingabefeld. Häufigkeits- und Gewichtsfelder werden ignoriert. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*Flag*	
train_pct	*number*	
set_random_seed	*Flag*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Time	Stoppmodus.
accuracy	*number*	Stoppgenauigkeit.
cycles	*number*	Zu trainierende Zyklen.
time	*number*	Dauer der Trainingsphase (Minuten)
continue	*Flag*	
show_feedback	*Flag*	
binary_encode	*Flag*	
use_last_model	*Flag*	
gen_logfile	*Flag*	
logfile_name	*string*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	

neuralnetnodeProperties	Werte	Eigenschaftsbeschreibung
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*string*	
m_non_pyramids	*Flag*	
m_persistence	*number*	
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*Flag*	
r_alpha	*number*	
r_eta	*number*	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
calculate_variable_importance	*Flag*	Anmerkung: Die in früheren Versionen verwendete Eigenschaft sensitivity_analysis wurde zugunsten dieser Eigenschaft verworfen. Die alte Eigenschaft wird weiterhin unterstützt, es wird jedoch calculate_variable_importance empfohlen.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "neuralnetworknode"

Der Netzwerkknoten 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 Mehrzweck-Schätzer, für deren Training und Anwendung nur sehr geringe statistische oder mathematische Kenntnisse erforderlich sind.

Beispiel

```
create neuralnetworknode
# Registerkarte "Erstellungsoptionen" - Panel "Ziel"
set: neuralnetworknode.objective = Standard
# Build Options tab - Stopping Rules panel
set: neuralnetworknode.model_selection = BestSubsets
set: neuralnetworknode.criteria_best_subsets = ASE
# Build Options tab - Ensembles panel
set: neuralnetworknode.combining_rule_categorical = HighestMeanProbability
```

Eigenschaften von neuralnetworknode	**Werte**	**Eigenschaftsbeschreibung**
targets	[*Feld1 ... FeldN*]	Gibt die Zielfelder an.
inputs	[*Feld1 ... FeldN*]	Im Modell verwendete Prädiktorfelder.
Aufteilungen	*[field1 ... fieldN*	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an.
use_partition	*Flag*	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
continue	*Flag*	Training des bestehenden Modells fortsetzen.
objective	Standard Bagging Boosting PSM	PSM wird für sehr umfangreiche Daten-Sets verwendet und erfordert eine Server-Verbindung.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*Flag*	
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*Flag*	
max_time	*number*	
use_max_cycles	*Flag*	
max_cycles	*number*	
use_min_accuracy	*Flag*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	

Eigenschaften von neuralnetworknode	Werte	Eigenschaftsbeschreibung
combining_rule_continuous	Mittelwert Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*Flag*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_custom_model_name	*Flag*	
custom_model_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*Flag*	
max_categories	*number*	
score_propensity	*Flag*	
use_custom_name	*Flag*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

Eigenschaften von "questnode"

Der QUEST-Knoten bietet eine binäre Klassifizierungsmethode zum Erstellen von Entscheidungsbäumen, die dazu dient, die für große C&R-Baum-Analysen 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. Für weitere Informationen siehe Thema QUEST-Knoten in Kapitel 6 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create questnode
set :questnode.custom_fields = True
set :questnode.target = Drug
set :questnode.inputs = [Age Na K Cholesterol BP]
set :questnode.model_output_type = InteractiveBuilder
set :questnode.use_tree_directives = True
set :questnode.mode = Expert
set :questnode.max_surrogates = 5
set :questnode.split_alpha = 0.03
set :questnode.use_percentage = False
set :questnode.min_parent_records_abs = 40
set :questnode.min_child_records_abs = 30
set :questnode.prune_tree = True
```

```
set :questnode.use_std_err = True
set :questnode.std_err_multiplier = 3
set :questnode.priors = Custom
set :questnode.custom_priors = [{drugA 0.3}{drugB 0.4}]
set :questnode.adjust_priors = true
```

questnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	QUEST-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue_training_existing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm wird für sehr umfangreiche Daten-Sets verwendet und erfordert eine Server-Verbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	
tree_directives	*string*	
use_max_depth	Default Custom	
max_depth	*integer*	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	*Flag*	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	*Flag*	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	*number*	Maximale Differenz.
max_surrogates	*number*	Maximale Anzahl Ersatztrenner.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in Klammermn ({}) die tatsächlich prognostizierten Kosten darstellen.
priors	Data Equal Custom	

questnodeProperties	Werte	Eigenschaftsbeschreibung
custom_priors	*strukturiert*	Strukturierte Eigenschaft im Format: set :cartnode. custom_priors = [{ drugA 0.3 } { drugB 0.6 }]
adjust_priors	*Flag*	
trails	*number*	Anzahl der Komponentenmodelle für Verbesserung oder Verstärkung.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standard-Kombinierungsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standard-Kombinierungsregel für stetige Ziele.
large_boost	*Flag*	Verbesserung auf sehr große Daten-Sets anwenden.
split_alpha	*number*	Signifikanzschwelle für Aufteilung.
train_pct	*number*	Verhinderung der übermäßigen Anpassung eingestellt.
set_random_seed	*Flag*	Option “Ergebnisse reproduzieren”.
seed	*number*	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von “regressionnode”

Die lineare Regression ist ein statistisches Verfahren zur Zusammenfassung von Daten und die Erstellung von Prognosen 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.

Beispiel

```
create regressionnode
# "Fields" tab
set :regressionnode.custom_fields = True
set :regressionnode.target = 'Age'
set :regressionnode.inputs = ['Na' 'K']
set :regressionnode.partition = Test
set :regressionnode.use_weight = True
set :regressionnode.weight_field = 'Drug'
# "Model" tab
set :regressionnode.use_model_name = False
set :regressionnode.model_name = "Regression Age"
set :regressionnode.use_partitioned_data = True
```

```
set :regressionnode.method = Stepwise
set :regressionnode.include_constant = False
# "Expert" tab
set :regressionnode.mode = Expert
set :regressionnode.complete_records = False
set :regressionnode.tolerance = "1.0E-3"
# "Stepping..." section
set :regressionnode.stepping_method = Probability
set :regressionnode.probability_entry = 0.77
set :regressionnode.probability_removal = 0.88
set :regressionnode.F_value_entry = 7.0
set :regressionnode.F_value_removal = 8.0
# "Output..." section
set :regressionnode.model_fit = True
set :regressionnode.r_squared_change = True
set :regressionnode.selection_criteria = True
set :regressionnode.descriptives = True
set :regressionnode.p_correlations = True
set :regressionnode.collinearity_diagnostics = True
set :regressionnode.confidence_interval = True
set :regressionnode.covariance_matrix = True
set :regressionnode.durbin_watson = True
```

regressionnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
target	*Feld*	Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
method	Enter Stepwise Backwards Forwards	
include_constant	*Flag*	
use_weight	*Flag*	
weight_field	*Feld*	
mode	Simple Expert	
complete_records	*Flag*	
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.

regressionnodeProperties	Werte	Eigenschaftsbeschreibung
stepping_method	useP useF	useP : F-Wahrscheinlichkeit verwenden useF: F-Wert verwenden
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*Flag*	
confidence_interval	*Flag*	
covariance_matrix	*Flag*	
collinearity_diagnostics	*Flag*	
regression_coefficients	*Flag*	
exclude_fields	*Flag*	
durbin_watson	*Flag*	
model_fit	*Flag*	
r_squared_change	*Flag*	
p_correlations	*Flag*	
descriptives	*Flag*	
calculate_variable_importance	*Flag*	

Eigenschaften von "sequencenode"

Der Sequenzknoten erkennt Assoziationsregeln in sequenziellen oder zeitorientierten Daten. Eine Sequenz ist eine Liste mit Element-Sets, 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. Für weitere Informationen siehe Thema Sequenzknoten in Kapitel 12 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create sequencenode
connect :databasenode to :sequencenode
# "Fields" tab
set :sequencenode.id_field = 'Age'
set :sequencenode.contiguous = True
set :sequencenode.use_time_field = True
set :sequencenode.time_field = 'Date1'
set :sequencenode.content_fields = ['Drug' 'BP']
set :sequencenode.partition = Test
# "Model" tab
set :sequencenode.use_model_name = True
set :sequencenode.model_name = "Sequence_test"
set :sequencenode.use_partitioned_data = False
set :sequencenode.min_supp = 15.0
set :sequencenode.min_conf = 14.0
set :sequencenode.max_size = 7
```

```
set :sequencenode.max_predictions = 5
# "Expert" tab
set :sequencenode.mode = Expert
set :sequencenode.use_max_duration = True
set :sequencenode.max_duration = 3.0
set :sequencenode.use_pruning = True
set :sequencenode.pruning_value = 4.0
set :sequencenode.set_mem_sequences = True
set :sequencenode.mem_sequences = 5.0
set :sequencenode.use_gaps = True
set :sequencenode.min_item_gap = 20.0
set :sequencenode.max_item_gap = 30.0
```

sequencenodeProperties	**Werte**	**Eigenschaftsbeschreibung**
id_field	*Feld*	Um ein Sequenzmodell zu erstellen, müssen Sie ein ID-Feld, ein optionales Zeitfeld und mindestens ein Inhaltsfeld angeben. Gewichts- und Häufigkeitsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
time_field	*Feld*	
use_time_field	*Flag*	
content_fields	[*Feld1 ... Feldn*]	
contiguous	*Flag*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*Flag*	
max_duration	*number*	
use_gaps	*Flag*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*Flag*	
pruning_value	*number*	
set_mem_sequences	*Flag*	
mem_sequences	*integer*	

Eigenschaften von slrmnode

Mithilfe des Knotens für das Selbstlern-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. Für weitere Informationen siehe Thema SLRM-Knoten in Kapitel 14 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create slrmnode
set :slrmnode.target = Offer
set :slrmnode.target_response = Response
set :slrmnode.inputs = ['Cust_ID' 'Age' 'Ave_Bal']
```

slrmnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	Beim Zielfeld muss es sich um ein nominales oder ein Flag-Feld handeln. Außerdem kann ein Häufigkeitsfeld angegeben werden. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
target_response	*Feld*	Der Typ muss “Flag” sein.
continue_training_existing_model	*Flag*	
target_field_values	*Flag*	Alle verwenden: Alle Werte aus der Quelle verwenden. Angeben: Erforderliche Werte auswählen.
target_field_values_specify	*[field1 ... fieldN]*	
include_model_assessment	*Flag*	
model_assessment_random_seed	*number*	Muss eine reelle Zahl sein.
model_assessment_sample_size	*number*	Muss eine reelle Zahl sein.
model_assessment_iterations	*number*	Anzahl der Iterationen.
display_model_evaluation	*Flag*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Ascending Descending	Gibt an, ob die Angebote mit den höchsten oder die mit den niedrigsten Scores zuerst angezeigt werden.
model_reliability	*Flag*	
calculate_variable_importance	*Flag*	

Eigenschaften von “statisticsmodelnode”

Mithilfe des Knotens “Statistikmodell” 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 SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistikmodellknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie unter Eigenschaften von “statisticsmodelnode” auf S. 323.

Eigenschaften von “svmnode”

Der Knoten “Support Vector Machine” (SVM) ermöglicht die Klassifizierung von Daten in eine von zwei Gruppen ohne Überanpassung. SVM eignet sich gut für umfangreiche Daten-Sets, beispielsweise solche mit einer großen Anzahl an Eingabefeldern. Für weitere Informationen siehe Thema SVM-Knoten in Kapitel 15 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create svmnode
# Expert tab
set :svmnode.mode=Expert
set :svmnode.all_probabilities=True
set :svmnode.kernel=Polynomial
set :svmnode.gamma=1.5
```

svmnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
all_probabilities	*Flag*	
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	*number*	Auch als C-Parameter bekannt.
precision	*number*	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 Kernel-Funktion.
rbf_gamma	*number*	Wird nur verwendet, wenn kernel auf RBF gesetzt ist.
gamma	*number*	Wird nur verwendet, wenn kernel auf Polynomial oder Sigmoid gesetzt ist.
bias	*number*	
degree	*number*	Wird nur verwendet, wenn kernel auf Polynomial gesetzt ist.

svmnodeProperties	Werte	Eigenschaftsbeschreibung
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Eigenschaften von timeseriesnode

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. Für weitere Informationen siehe Thema Zeitreihen – Modellierungsknoten in Kapitel 13 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create timeseriesnode
set :timeseriesnode.method = Exsmooth
set :timeseriesnode.exsmooth_model_type = HoltsLinearTrend
set :timeseriesnode.exsmooth_transformation_type = None
```

timeseriesnodeProperties	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 Gewichtsfelder werden nicht verwendet. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
continue	*Flag*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*Flag*	
consider_seasonal	*Flag*	
detect_outliers	*Flag*	

timeseriesnodeProperties	Werte	Eigenschaftsbeschreibung
expert_outlier_additive	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_innovational	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_transient	*Flag*	
expert_outlier_seasonal_additive	*Flag*	
expert_outlier_local_trend	*Flag*	
expert_outlier_additive_patch	*Flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	
arima_d	*integer*	
arima_q	*integer*	
arima_sp	*integer*	
arima_sd	*integer*	
arima_sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*Flag*	
tf_arima_p.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_d.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_q.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_sp.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_sd.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_sq.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_delay.*fieldname*	*integer*	Für Transferfunktionen.
tf_arima_transformation_type.*fieldname*	None SquareRoot NaturalLog	Für Transferfunktionen.
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*Flag*	
arima_outlier_level_shift	*Flag*	
arima_outlier_innovational	*Flag*	
arima_outlier_transient	*Flag*	
arima_outlier_seasonal_additive	*Flag*	
arima_outlier_local_trend	*Flag*	
arima_outlier_additive_patch	*Flag*	

timeseriesnodeProperties	Werte	Eigenschaftsbeschreibung
conf_limit_pct	*real*	
max_lags	*integer*	
events	*Felder*	
scoring_model_only	*Flag*	Für Modelle mit sehr großen Zahlen (Zehntausende) von Zeitreihen.

Eigenschaften von "twostepnode"

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 Unterclustern komprimiert werden. Im zweiten Schritt werden die Untercluster 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 Daten-Sets effizient verarbeitet werden. Für weitere Informationen siehe Thema TwoStep-Cluster-Knoten in Kapitel 11 in *IBM SPSS Modeler 15 Modellierungsknoten*.

Beispiel

```
create twostep
set :twostep.custom_fields = True
set :twostep.inputs = ['Age' 'K' 'Na' 'BP']
set :twostep.partition = Test
set :twostep.use_model_name = False
set :twostep.model_name = "TwoStep_Drug"
set :twostep.use_partitioned_data = True
set :twostep.exclude_outliers = True
set :twostep.cluster_label = "String"
set :twostep.label_prefix = "TwoStep_"
set :twostep.cluster_num_auto = False
set :twostep.max_num_clusters = 9
set :twostep.min_num_clusters = 3
set :twostep.num_clusters = 7
```

twostepnodeProperties	Werte	Eigenschaftsbeschreibung
inputs	[*Feld1 ... FeldN*]	TwoStep-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichts- und Häufigkeitsfelder werden nicht erkannt. Für weitere Informationen siehe Thema Modellierungsknoten – Allgemeine Eigenschaften auf S. 201.
standardize	*Flag*	
exclude_outliers	*Flag*	
percentage	*number*	
cluster_num_auto	*Flag*	

twostepnodeProperties	Werte	Eigenschaftsbeschreibung
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String Number	
label_prefix	*string*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Kapitel
17

Modell-Nugget-Knoten – Eigenschaften

Modell-Nugget-Knoten besitzen dieselben allgemeinen Eigenschaften wie andere Knoten. Für weitere Informationen siehe Thema Allgemeine Knoteneigenschaften in Kapitel 9 auf S. 122.

Eigenschaften von "applyanomalydetectionnode"

Mithilfe von Modellierungsknoten vom Typ "Anomalieerkennung" kann ein Modell-Nugget vom Typ "Anomalieerkennung" generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyanomalydetectionnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von "anomalydetectionnode" in Kapitel 16 auf S. 202.

Eigenschaften von applyanomalyde-tectionnode	Werte	Eigenschaftsbeschreibung
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Bestimmt, welche Ausgaben für das Scoring erstellt werden.
num_fields	*integer*	Zu meldende Felder.
discard_records	*Flag*	Gibt an, ob Datensätze aus der Ausgabe verworfen werden sollen oder nicht.
discard_anomalous_records	*Flag*	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 von "applyapriorinode"

Mithilfe von A Priori-Modellierungsknoten kann ein Modell-Nugget vom Typ "A Priori" generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyapriorinode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von "apriorinode" in Kapitel 16 auf S. 203.

Eigenschaften von applyapriorinode	Werte	Eigenschaftsbeschreibung
max_predictions	*Zahl (Ganzzahl)*	
ignore_unmatached	*Flag*	
allow_repeats	*Flag*	

Eigenschaften von applyapriorinode	Werte	Eigenschaftsbeschreibung
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Deployability	

applyautoclassifiernode-Eigenschaften

Mithilfe von Modellierungsknoten des Typs “Automatischer Klassifizierer” kann ein Modell-Nugget vom Typ “Automatischer Klassifizierer” generiert werden. Der Skriptname dieses Modell-Nuggets ist *applyautoclassifiernode*.Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “autoclassisfiernode” in Kapitel 16 auf S. 205.

applyautoclassifiern-ode-Eigenschaften	Werte	Eigenschaftsbeschreibung
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVot-ing HighestConfidence AverageRawPropensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flag-Feld 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 Flag-Feld 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 Set-Feld 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.

applyautoclusternode-Eigenschaften

Mithilfe von Modellierungsknoten des Typs “Autom. Cluster” kann ein Modell-Nugget vom Typ “Autom. Cluster” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyautoclusternode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter autoclusternode-Eigenschaften in Kapitel 16 auf S. 207.

applyautonumericnode-Eigenschaften

Mithilfe von Modellierungsknoten des Typs “Auto-Numerisch” kann ein Modell-Nugget vom Typ “Auto-Numerisch” generiert werden. Der Skriptname dieses Modell-Nuggets ist *applyautonumericnode*.Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter autonumericnode-Eigenschaften in Kapitel 16 auf S. 209.

applyautonumericn-ode-Eigenschaften	Werte	Eigenschaftsbeschreibung
calculate_standard_error	*Flag*	

Eigenschaften von “applybayesnetnode”

Mithilfe von Bayes-Netzwerk-Modellierungsknoten kann ein Modell-Nugget vom Typ “Bayes-Netzwerk” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applybayesnetnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “bayesnetnode” in Kapitel 16 auf S. 210.

Eigenschaften von applybayesnetnode	Werte	Eigenschaftsbeschreibung
all_probabilities	*Flag*	
raw_propensity	*Flag*	
adjusted_propensity	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applyc50node”

Mithilfe von C5.0-Modellierungsknoten kann ein C5.0-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyc50node*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “c50node” in Kapitel 16 auf S. 212.

Eigenschaften von applyc50node	Werte	Eigenschaftsbeschreibung
sql_generate	Never NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen der Regelmenge.
calculate_conf	*Flag*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applycarmanode”

Mithilfe von CARMA-Modellierungsknoten kann ein CARMA-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applycarmanode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “carmanode” in Kapitel 16 auf S. 213.

Eigenschaften von “applycartnode”

Mithilfe von Modellierungsknoten vom Typ “C&RT-Baum” kann ein Modell-Nugget vom Typ “C&RT-Baum” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applycartnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “cartnode” in Kapitel 16 auf S. 214.

Eigenschaften von applycartnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen der Regelmenge.
calculate_conf	*Flag*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
display_rule_id	*Flag*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applychaidnode”

Mithilfe von CHAID-Modellierungsknoten kann ein CHAID-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applychaidnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “chaidnode” in Kapitel 16 auf S. 217.

Eigenschaften von applychaidnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues	
calculate_conf	*Flag*	
display_rule_id	*Flag*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von "applycoxregnode"

Mithilfe von Cox-Modellierungsknoten kann ein Cox-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applycoxregnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von "coxregnode" in Kapitel 16 auf S. 219.

Eigenschaften von applycoxregnode	Werte	Eigenschaftsbeschreibung
future_time_as	Intervals Fields	
time_interval	*Zahl*	
num_future_times	*integer*	
time_field	*Feld*	
past_survival_time	*Feld*	
all_probabilities	*Flag*	
cumulative_hazard	*Flag*	

Eigenschaften von "applydecisionlistnode"

Mithilfe von Entscheidungslisten-Modellierungsknoten kann ein Modell-Nugget vom Typ "Entscheidungsliste" generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applydecisionlistnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von decisionlistnode in Kapitel 16 auf S. 221.

Eigenschaften von applydecisionlistnode	Werte	Eigenschaftsbeschreibung
enable_sql_generation	*Flag*	Wenn dieser Wert wahr ist, versucht IBM® SPSS® Modeler, das Entscheidungslistenmodell über einen Pushback-Vorgang an SQL zurückzuführen.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von "applydiscriminantnode"

Mithilfe von Diskriminanz-Modellierungsknoten kann ein Diskriminanz-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applydiscriminantnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von discriminantnode in Kapitel 16 auf S. 222.

Eigenschaften von applydiscriminantnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applyfactornode”

Mithilfe von Faktor-Modellierungsknoten kann ein Faktor-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyfactornode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “factornode” in Kapitel 16 auf S. 224.

Eigenschaften von “applyfeatureselectionnode”

Mithilfe von Modellierungsknoten vom Typ “Merkmalsauswahl” kann ein Modell-Nugget vom Typ “Merkmalsauswahl” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyfeatureselectionnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “featureselectionnode” in Kapitel 16 auf S. 226.

Eigenschaften von applyfeatureselectionnode	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 von “applygeneralizedlinearnode”

Mithilfe von Modellierungsknoten vom Typ “Verallgemeinert linear (genlin)” kann ein Modell-Nugget vom Typ “Verallgemeinert linear” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applygeneralizedlinearnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von genlinnode in Kapitel 16 auf S. 228.

Eigenschaften von applygeneralizedlinearnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applykmeansnode”

Mithilfe von K-Means-Modellierungsknoten kann ein Faktor-Modell-K-Means generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applykmeansnode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “kmeansnode” in Kapitel 16 auf S. 235.

applyknnnode, Eigenschaften

Mithilfe von KNN-Modellierungsknoten kann ein KNN-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyknnnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “knnnode” in Kapitel 16 auf S. 236.

applyknnnode, Eigenschaften	Werte	Eigenschaftsbeschreibung
all_probabilities	*Flag*	
save_distances	*Flag*	

Eigenschaften von “applykohonennode”

Mithilfe von Kohonen-Modellierungsknoten kann ein Kohonen-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applykohonennode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “c50node” in Kapitel 16 auf S. 212.

Eigenschaften von “applylinearnode”

Mithilfe von Cox-Modellierungsknoten kann ein Nugget für ein lineares Modell generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applylinearnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “linearnode” in Kapitel 16 auf S. 238.

linear Eigenschaften	Werte	Eigenschaftsbeschreibung
use_custom_name	*Flag*	
custom_name	*string*	
enable_sql_generation	*Flag*	

Eigenschaften von “applylogregnode”

Mithilfe von Modellierungsknoten vom Typ “Logistische Regression” kann ein Modell-Nugget vom Typ “Logistische Regression” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applylogregnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “logregnode” in Kapitel 16 auf S. 240.

Eigenschaften von applylogregnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	*Flag*	

Eigenschaften von "applyneuralnetnode"

Mithilfe von Netzwerk-Modellierungsknoten kann ein Netzwerk-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyneuralnetnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von "neuralnetnode" in Kapitel 16 auf S. 244.

Vorsicht: In dieser Version ist eine neuere Fassung des Netzwerk-Nuggets mit erweiterten Funktionen verfügbar, die im nächsten Abschnitt beschrieben wird (*applyneuralnetwork*). Die Vorgängerversion ist zwar weiterhin verfügbar, wir empfehlen jedoch die Aktualisierung Ihrer Skripts zur Verwendung der neuen Version. Zur Referenz sind hier Details zur Vorgängerversion enthalten, doch wird die Unterstützung dafür in zukünftigen Versionen wegfallen.

Eigenschaften von applyneuralnetnode	Werte	Eigenschaftsbeschreibung
calculate_conf	*Flag*	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
enable_sql_generation	*Flag*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

applyneuralnetworknode Eigenschaften

Mithilfe von Netzwerk-Modellierungsknoten kann ein Netzwerk-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyneuralnetworknode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von "neuralnetworknode" in Kapitel 16 auf S. 247.

applyneuralnetworknode-Eigenschaften	Werte	Eigenschaftsbeschreibung
use_custom_name	*Flag*	
custom_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*Flag*	
max_categories	*Zahl*	
score_propensity	*Flag*	

Eigenschaften von “applyquestnode”

Mithilfe von QUEST-Modellierungsknoten kann ein QUEST-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyquestnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “questnode” in Kapitel 16 auf S. 248.

Eigenschaften von applyquestnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	*Flag*	
display_rule_id	*Flag*	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applyregressionnode”

Mithilfe von Modellierungsknoten vom Typ “Lineare Regression” kann ein Modell-Nugget vom Typ “Lineare Regression” generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyregressionnode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “regressionnode” in Kapitel 16 auf S. 250.

Eigenschaften von “applyselflearningnode”

Mithilfe von Modellierungsknoten vom Typ (Self-Learning Response Model (SLRM) kann ein SLRM-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applyselflearningnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von slrmnode in Kapitel 16 auf S. 254.

Eigenschaften von applyselflearningnode	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	*Flag*	Berücksichtigt die Option für die Reliabilität auf der Registerkarte “Einstellungen”.

Eigenschaften von “applysequencenode”

Mithilfe von Sequenz-Modellierungsknoten kann ein Sequenzmodell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applysequencenode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “sequencenode” in Kapitel 16 auf S. 252.

Eigenschaften von “applysvmnode”

Mithilfe von SVM-Modellierungsknoten kann ein SVM-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applysvmnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “svmnode” in Kapitel 16 auf S. 255.

Eigenschaften von applysvmnode	Werte	Eigenschaftsbeschreibung
all_probabilities	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Eigenschaften von “applytimeseriesnode”

Mithilfe von Zeitreihen-Modellierungsknoten kann ein Zeitreihenmodell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applytimeseriesnode*. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von timeseriesnode in Kapitel 16 auf S. 256.

Eigenschaften von applytimeseriesnode	Werte	Eigenschaftsbeschreibung
calculate_conf	*Flag*	
calculate_residuals	*Flag*	

Eigenschaften von “applytwostepnode”

Mithilfe von TwoStep-Modellierungsknoten kann ein TwoStep-Modell-Nugget generiert werden. Der Skriptname dieses Modell-Nuggets lautet *applytwostepnode*. Für dieses Modell-Nugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Skripts für den Modellierungsknoten selbst finden Sie unter Eigenschaften von “twostepnode” in Kapitel 16 auf S. 258.

Kapitel
18

Datenbankmodellierungsknoten – Eigenschaften

IBM® SPSS® Modeler unterstützt die Integration mit Data Mining-Tools und Daten-Modellierungstools von Datenbankherstellern, z. B. Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM® DB2® InfoSphere Warehouse und IBM® Netezza® Analytics. Für weitere Informationen siehe Thema Übersicht über die Datenbank-Modellierung in Kapitel 2 in *IBM SPSS Modeler 15 In-Database Mining-Handbuch*. Sie können mithilfe von datenbankeigenen Algorithmen Modelle erstellen und scoren – ohne dazu die SPSS Modeler-Anwendung verlassen zu müssen. Datenbankmodelle können außerdem mithilfe von Skripterstellung unter Verwendung der in diesem Abschnitt beschriebenen Eigenschaften erstellt und bearbeitet werden.

Das folgende Skript-Exzerpt veranschaulicht z. B. die Erstellung eines Microsoft Decision Trees-Modells über das SPSS Modeler-Skript-Interface:

```
create mstreenode
rename :mstreenode as msbuilder
set msbuilder.analysis_server_name = 'localhost'
set msbuilder.analysis_database_name = 'TESTDB'
set msbuilder.mode = 'Expert'
set msbuilder.datasource = 'LocalServer'
set msbuilder.target = 'Drug'
set msbuilder.inputs = ['Age' 'Sex']
set msbuilder.unique_field = 'IDX'
set msbuilder.custom_fields = true
set msbuilder.model_name = 'MSDRUG'

connect :typenode to msbuilder
execute msbuilder

insert model MSDRUG connected between :typenode and :tablenode
set MSDRUG.sql_generate = true
execute :tablenode
```

Knoteneigenschaften für Microsoft-Modellierung

Microsoft-Modellierungsknoten – Eigenschaften

Allgemeine Eigenschaften

Folgende Eigenschaften haben alle Microsoft-Datenbank-Modellierungsknoten gemeinsam.

Allgemeine Eigenschaften von Microsoft-Knoten	Werte	Eigenschaftsbeschreibung
analysis_database_name	*string*	Name der Analysis Services-Datenbank.
analysis_server_name	*string*	Name des Analysis Services-Hosts.
use_transactional_data	*Flag*	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. Für weitere Informationen siehe Thema Algorithmusparameter auf S. 272.
with_drillthrough	*Flag*	Mit Drillthrough-Option.

MS-Entscheidungsbaum

Für Knoten vom Typ mstreenode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Clustering

Für Knoten vom Typ msclusternode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Assoziationsregeln

Für Knoten vom Typ msassocnode sind folgende Eigenschaften verfügbar.

msassocnodeProperties	Werte	Eigenschaftsbeschreibung
id_field	*Feld*	Identifiziert jede Transaktion in den Daten.
trans_inputs	*[Feld Feld Feld]*	Eingabefelder für Transaktionsdaten.
transactional_target	*Feld*	Prädiktorfeld (Transaktionsdaten).

MS Naive Bayes

Für Knoten vom Typ msbayesnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS – Lineare Regression

Für Knoten vom Typ msregressionnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS – Neuronales Netzwerk

Für Knoten vom Typ msneuralnetworknode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS – Logistische Regression

Für Knoten vom Typ mslogisticnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS Time Series

Für Knoten vom Typ mstimeseriesnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS Sequenz-Clustering

Für Knoten vom Typ mssequenceclusternode sind folgende Eigenschaften verfügbar.

mssequenceclustern-odeProperties	**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. Beispiel:

```
set :msregressionnode.msas_parameters =
[{"MAXIMUM_INPUT_ATTRIBUTES" 255}{"MAXIMUM_OUTPUT_ATTRIBUTES" 255}]
```

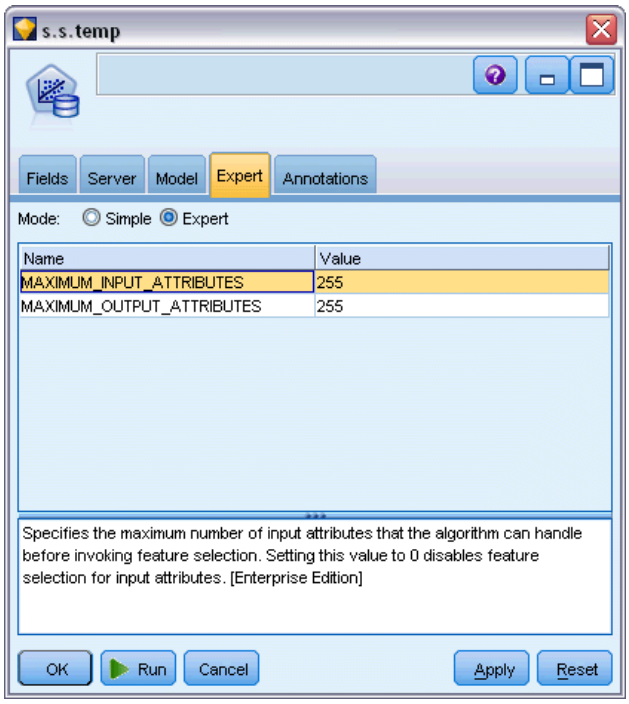
Diese Parameter werden vom SQL-Server abgeleitet. Gehen Sie wie folgt vor, um die relevanten Parameter für die einzelnen Knoten anzuzeigen:

- Platzieren Sie einen Datenbankquellenknoten im Zeichenbereich.
- Öffnen Sie den Datenbankquellenknoten.

- Wählen Sie eine gültige Quelle in der Dropdown-Liste Datenquelle aus.
- Wählen Sie eine gültige Tabelle in der Liste Tabellenname aus.
- Klicken Sie auf OK, um den Datenbankquellenknoten zu schließen.
- Fügen Sie den Microsoft-Datenbankmodellierungsknoten ein, dessen Eigenschaften aufgelistet werden sollen.
- Öffnen Sie den Datenbankmodellierungsknoten.
- Wählen Sie die Registerkarte Experten.

Die verfügbaren msas_parameters-Eigenschaften für diesen Knoten werden angezeigt.

Abbildung 18-1
Beispiel für Algorithmusparameteranzeige



Microsoft-Modell-Nugget – Eigenschaften

Folgende Eigenschaften gelten für die Modell-Nuggets, die mithilfe der Microsoft-Datenbankmodellierungsknoten erstellt wurden.

MS-Entscheidungsbaum

applymstreenodeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.
datasource	*string*	Name der SQL Server ODBC-Datenquelle (DSN).
sql_generate	*Flag*	Aktiviert die SQL-Erzeugung.

MS – Lineare Regression

applymsregressionnodeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.

MS – Neuronales Netzwerk

applymsneuralnetworknodeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.

MS – Logistische Regression

applymslogisticnodeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.

MS Time Series

applymstime-seriesnodeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.

applymstime-seriesnodeProperties	Werte	Beschreibung
start_from	new_prediction historical_pre-diction	Gibt an, ob Zukunfts- oder historische Vorhersagen getroffen werden.
new_step	*number*	Definiert die Startzeit für Zukunftsvorhersagen.
historical_step	*number*	Definiert die Startzeit für historische Vorhersagen.
end_step	*number*	Definiert die Endzeit für Vorhersagen.

MS Sequenz-Clustering

applymssequenceclustern-odeProperties	Werte	Beschreibung
analysis_database_name	*string*	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	*string*	Name des Analyseserver-Hosts.

Knoteneigenschaften für Oracle-Modellierung

Oracle-Modellierungsknoten – Eigenschaften

Folgende Eigenschaften haben alle Oracle-Datenbank-Modellierungsknoten gemeinsam.

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	*Flag*	
model_name	*string*	Benutzerdefinierter Name für neues Modell.
use_partitioned_data	*Flag*	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	*Flag*	Aktiviert oder deaktiviert die automatische Datenvorbereitungsfunktion von Oracle (nur 11g-Datenbanken).

Allgemeine Eigenschaften von Oracle-Knoten	Werte	Eigenschaftsbeschreibung
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5}{drugA drugC 2.1}], wobei die Argumente in {} die tatsächlich prognostizierten Kosten darstellen.
mode	Simple Expert	Wenn diese Einstellung auf Simple (Einfach) gesetzt ist, werden bestimmte Eigenschaften ignoriert (vgl. die Eigenschaften der einzelnen Knoten).
use_prediction_probability	*Flag*	
prediction_probability	*string*	
use_prediction_set	*Flag*	

Oracle Naive Bayes

Für Knoten vom Typ oranbnode sind folgende Eigenschaften verfügbar.

oranbnodeProperties	Werte	Eigenschaftsbeschreibung
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft im Format: set :oranbnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Adaptive Bayes

Für Knoten vom Typ oraabnnode sind folgende Eigenschaften verfügbar.

oraabnnodeProperties	Werte	Eigenschaftsbeschreibung
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*Flag*	*
execution_time_limit	*integer*	Der Wert muss größer als 0 sein.*
max_naive_bayes_predictors	*integer*	Der Wert muss größer als 0 sein.*
max_predictors	*integer*	Der Wert muss größer als 0 sein.*
priors	Data Equal Custom	
custom_priors	*strukturiert*	Strukturierte Eigenschaft im Format: set :oraabnnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Support Vector Machines

Für Knoten vom Typ orasvmnode sind folgende Eigenschaften verfügbar.

orasvmnodeProperties	Werte	Eigenschaftsbeschreibung
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*integer*	Nur Gauss'scher Kern. Der Wert muss größer als 0 sein.*
convergence_tolerance	*number*	Der Wert muss größer als 0 sein.*
use_standard_deviation	*Flag*	Nur Gauss'scher Kern.*
standard_deviation	*number*	Der Wert muss größer als 0 sein.*
use_epsilon	*Flag*	Nur Regressionsmodelle.*
epsilon	*number*	Der Wert muss größer als 0 sein.*
use_complexity_factor	*Flag*	*
complexity_factor	*number*	*
use_outlier_rate	*Flag*	Nur Ein-Klassen-Variante.*
outlier_rate	*number*	Nur Ein-Klassen-Variante. 0.0–1.0.*
weights	Data Equal Custom	
custom_weights	*strukturiert*	Strukturierte Eigenschaft im Format: set :orasvmnode.custom_weights = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Verallgemeinerte lineare Modelle von Oracle

Für Knoten vom Typ oraglmnode sind folgende Eigenschaften verfügbar.

oraglmnodeProperties	Werte	Eigenschaftsbeschreibung
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWith-Mean UseCompleteRe-cords	
use_row_weights	*Flag*	*
row_weights_field	*Feld*	*
save_row_diagnostics	*Flag*	*
row_diagnostics_table	*string*	*
coefficient_confidence	*number*	*

oraglmnodeProperties	Werte	Eigenschaftsbeschreibung
use_reference_category	*Flag*	*
reference_category	*string*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*Flag*	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Decision Tree

Für Knoten vom Typ oradecisiontreenode sind folgende Eigenschaften verfügbar.

oradecisiontreenodeProperties	Werte	Eigenschaftsbeschreibung
use_costs	*Flag*	
impurity_metric	Entropy Gini	
term_max_depth	*integer*	2–20.*
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*
term_minrec_node	*integer*	Der Wert muss größer als 0 sein.*
term_minrec_split	*integer*	Der Wert muss größer als 0 sein.*
display_rule_ids	*Flag*	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle O-Cluster

Für Knoten vom Typ oraoclusternode sind folgende Eigenschaften verfügbar.

oraoclusternodeProperties	Werte	Eigenschaftsbeschreibung
max_num_clusters	*integer*	Der Wert muss größer als 0 sein.
max_buffer	*integer*	Der Wert muss größer als 0 sein.*
sensitivity	*number*	0.0–1.0.*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle KMeans

Für Knoten vom Typ orakmeansnode sind folgende Eigenschaften verfügbar.

orakmeansnodeProperties	Werte	Eigenschaftsbeschreibung
num_clusters	*integer*	Der Wert muss größer als 0 sein.
normalization_method	zscore minmax none	

orakmeansnodeProperties	Werte	Eigenschaftsbeschreibung
distance_function	Euclidean Cosine	
iterations	*integer*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	Die Standardeinstellung ist “Variance”.
num_bins	*integer*	Der Wert muss größer als 0 sein.*
block_growth	*integer*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle NMF

Für Knoten vom Typ oranmfnode sind folgende Eigenschaften verfügbar.

oranmfnodeProperties	Werte	Eigenschaftsbeschreibung
normalization_method	minmax none	
use_num_features	*Flag*	*
num_features	*integer*	0–1. Der Standardwert wird vom Algorithmus durch Schätzung aus den Daten ermittelt.*
random_seed	*number*	*
num_iterations	*integer*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*Flag*	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Apriori

Für Knoten vom Typ oraapriorinode sind folgende Eigenschaften verfügbar.

oraapriorinodeProperties	Werte	Eigenschaftsbeschreibung
content_field	*Feld*	
id_field	*Feld*	
max_rule_length	*integer*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*Flag*	

Oracle Minimum Description Length (MDL)

Für Knoten vom Typ oramdlnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Oracle-Eigenschaften am Anfang dieses Abschnitts.

Oracle Attribute Importance (AI)

Für Knoten vom Typ oraainode sind folgende Eigenschaften verfügbar.

oraainodeProperties	Werte	Eigenschaftsbeschreibung
custom_fields	*Flag*	Bei “true” (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei “false” (falsch) werden die aktuellen Einstellungen aus einem weiter oben im Stream gelegenen Typknoten verwendet.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.
important_label	*string*	Gibt die Beschriftung für die Rangstufe “bedeutsam” an.
select_marginal	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
marginal_label	*string*	Gibt die Beschriftung für die Rangstufe “marginal” an.
important_above	*number*	0.0–1.0.
select_unimportant	*Flag*	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob unwichtige Felder ausgewählt werden sollen.
unimportant_label	*string*	Gibt die Beschriftung für die Rangstufe “unbedeutsam” an.
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	Wenn selection_mode auf ImportanceValue gesetzt ist, wird hier der zu verwendende Cutoff-Wert angegeben. Zulässig sind Werte von 0 bis 100.
top_n	*number*	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Cutoff-Wert angegeben. Zulässig sind Werte von 0 bis 1000.

Oracle-Modell-Nugget – Eigenschaften

Folgende Eigenschaften gelten für die Modell-Nuggets, die mithilfe der Oracle-Modelle erstellt wurden.

Oracle Naive Bayes

Für Knoten vom Typ applyoranbnode sind keine speziellen Eigenschaften definiert.

Oracle Adaptive Bayes

Für Knoten vom Typ applyoraabnnode sind keine speziellen Eigenschaften definiert.

Oracle Support Vector Machines

Für Knoten vom Typ applyorasvmnode sind keine speziellen Eigenschaften definiert.

Oracle Decision Tree

Für Knoten vom Typ applyoradecisiontreenode sind folgende Eigenschaften verfügbar.

applyoradecisiontreenodeProperties	Werte	Eigenschaftsbeschreibung
use_costs	*Flag*	
display_rule_ids	*Flag*	

Oracle O-Cluster

Für Knoten vom Typ applyoraoclusternode sind keine speziellen Eigenschaften definiert.

Oracle KMeans

Für Knoten vom Typ applyorakmeansnode sind keine speziellen Eigenschaften definiert.

Oracle NMF

Für Knoten vom Typ applyoranmfnode ist die folgende Eigenschaft verfügbar.

applyoranmfnodeProperties	Werte	Eigenschaftsbeschreibung
display_all_features	*Flag*	

Oracle Apriori

Dieses Modell-Nugget kann nicht in Skripts verwendet werden.

Oracle MDL

Dieses Modell-Nugget kann nicht in Skripts verwendet werden.

Knoteneigenschaften für IBM DB2-Modellierung

IBM DB2-Modellierungsknoten – Eigenschaften

Folgende Eigenschaften haben alle IBM InfoSphere Warehouse-(ISW-)Datenbank-Modellierungsknoten gemeinsam:

Allgemeine Eigenschaften von ISW-Knoten	Werte	Eigenschaftsbeschreibung
inputs	*Liste mit Feldern*	
datasource		
username		
password		
epassword		
enable_power_options	*Flag*	
power_options_max_memory	*integer*	Der Wert muss größer als 32 sein.
power_options_cmdline	*string*	
mining_data_custom_sql	*string*	
logical_data_custom_sql	*string*	
mining_settings_custom_sql		

ISW-Entscheidungsbaum

Für Knoten vom Typ db2imtreenode sind folgende Eigenschaften verfügbar.

db2imtreenodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	
perform_test_run	*Flag*	
use_max_tree_depth	*Flag*	
max_tree_depth	*integer*	Der Wert muss größer als 0 sein.
use_maximum_purity	*Flag*	
maximum_purity	*number*	Eine Zahl zwischen 0 und 100.
use_minimum_internal_cases	*Flag*	
minimum_internal_cases	*integer*	Der Wert muss größer als 1 sein.
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in {} die tatsächlich prognostizierten Kosten darstellen.

ISW-Assoziation

Für Knoten vom Typ db2imassocnode sind folgende Eigenschaften verfügbar.

db2imassocnodeProperties	Werte	Eigenschaftsbeschreibung
use_transactional_data	*Flag*	
id_field	*Feld*	
content_field	*Feld*	
data_table_layout	basic limited_length	
max_rule_size	*integer*	Der Wert muss größer als 2 sein.
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*Flag*	
item_constraints_type	Include Exclude	
use_taxonomy	*Flag*	
taxonomy_table_name	*string*	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	*string*	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	*string*	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	*Flag*	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

Für Knoten vom Typ db2imsequencenode sind folgende Eigenschaften verfügbar.

db2imsequencenodeProperties	Werte	Eigenschaftsbeschreibung
id_field	*Feld*	
group_field	*Feld*	
content_field	*Feld*	
max_rule_size	*integer*	Der Wert muss größer als 2 sein.
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*Flag*	
item_constraints_type	Include Exclude	
use_taxonomy	*Flag*	

db2imsequencenodeProperties	Werte	Eigenschaftsbeschreibung
taxonomy_table_name	*string*	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	*string*	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	*string*	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	*Flag*	Steuert, ob in 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

Für Knoten vom Typ db2imregnode sind folgende Eigenschaften verfügbar.

db2imregnodeProperties	Werte	Eigenschaftsbeschreibung
target	*Feld*	
regression_method	transform linear polynomial rbf	In der nächsten Tabelle finden Sie Eigenschaften, die nur gelten, wenn regression_method aufrbf gesetzt ist.
perform_test_run	*Feld*	
limit_rsquared_value	*Flag*	
max_rsquared_value	*number*	Der Wert muss zwischen 0,0 und 1,0 liegen.
use_execution_time_limit	*Flag*	
execution_time_limit_mins	*integer*	Der Wert muss größer als 0 sein.
use_max_degree_polynomial	*Flag*	
max_degree_polynomial	*integer*	
use_intercept	*Flag*	
use_auto_feature_selec-tion_method	*Flag*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*Flag*	
min_significance_level	*number*	
use_min_significance_level	*Flag*	

Die folgenden Eigenschaften gelten nur, wenn regression_method auf rbf gesetzt ist.

db2imregnodeProperties	Werte	Eigenschaftsbeschreibung
use_output_sample_size	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.

output_sample_size	*integer*	Die Standardeinstellung ist 2. Minimum ist 1.
use_input_sample_size	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.
input_sample_size	*integer*	Die Standardeinstellung ist 2. Minimum ist 1.
use_max_num_centers	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.
max_num_centers	*integer*	Die Standardeinstellung ist 20. Minimum ist 1.
use_min_region_size	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.
min_region_size	*integer*	Die Standardeinstellung ist 15. Minimum ist 1.
use_max_data_passes	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.
max_data_passes	*integer*	Die Standardeinstellung ist 5. Minimum ist 2.
use_min_data_passes	*Flag*	Wenn wahr, den Wert automatisch auf Standard setzen.
min_data_passes	*integer*	Die Standardeinstellung ist 5. Minimum ist 2.

ISW Clustering

Für Knoten vom Typ db2imclusternode sind folgende Eigenschaften verfügbar.

db2imclusternodeProperties	**Werte**	**Eigenschaftsbeschreibung**
cluster_method	demographic kohonen birch	
kohonen_num_rows	*integer*	
kohonen_num_columns	*integer*	
kohonen_passes	*integer*	
use_num_passes_limit	*Flag*	
use_num_clusters_limit	*Flag*	
max_num_clusters	*integer*	Der Wert muss größer als 1 sein.
birch_dist_measure	log_likelihood euclidean	Die Standardeinstellung lautet log_likelihood.
birch_num_cfleaves	*integer*	Die Standardeinstellung ist 1000.
birch_num_refine_passes	*integer*	Die Standardeinstellung lautet 3; Minimum ist 1.
use_execution_time_limit	*Flag*	
execution_time_limit_mins	*integer*	Der Wert muss größer als 0 sein.
min_data_percentage	*number*	0–100%
use_similarity_threshold	*Flag*	
similarity_threshold	*number*	Der Wert muss zwischen 0,0 und 1,0 liegen.

ISW Naive Bayes

Für Knoten vom Typ db2imnbsnode sind folgende Eigenschaften verfügbar.

db2imnbnodeProperties	Werte	Eigenschaftsbeschreibung
perform_test_run	*Flag*	
probability_threshold	*number*	Die Standardeinstellung ist 0,001. Minimalwert ist 0; Maximalwert ist 1.000
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in {} die tatsächlich prognostizierten Kosten darstellen.

ISW Logistische Regression

Für Knoten vom Typ db2imlognode sind folgende Eigenschaften verfügbar.

db2imlognodeProperties	Werte	Eigenschaftsbeschreibung
perform_test_run	*Flag*	
use_costs	*Flag*	
costs	*strukturiert*	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in {} die tatsächlich prognostizierten Kosten darstellen.

ISW Time Series

Hinweis: Der Eingabefeldparameter wird für diesen Knoten nicht verwendet. Wenn der Eingabefeldparameter in dem Skript gefunden wird, erscheint eine Warnung, dass der Knoten als eingehende Felder *time* und *targets*, aber keine Eingabefelder hat.

Für Knoten vom Typ db2imtimeseriesnode sind folgende Eigenschaften verfügbar.

db2imtime-seriesnodeProperties	Werte	Eigenschaftsbeschreibung
time	*Feld*	"Integer", "time" oder "date" sind zulässig.
targets	*Liste mit Feldern*	
forecasting_algorithm	arima exponen-tial_smoothing sea-sonal_trend_de-composition	
forecasting_end_time	auto integer date time	
use_records_all	*boolean*	Wenn falsch, müssen use_records_start und use_records_end festgelegt werden.

db2imtime-seriesnodeProperties	Werte	Eigenschaftsbeschreibung
use_records_start	*integer / time / date*	Abhängig vom Zeitfeldtyp
use_records_end	*integer / time / date*	Abhängig vom Zeitfeldtyp
interpolation_method	none linear exponen-tial_splines cubic_splines	

IBM DB2-Modelli-Nugget – Eigenschaften

Folgende Eigenschaften gelten für die Modell-Nuggets, die mithilfe der IBM DB2 ISW-Modelle erstellt wurden.

ISW-Entscheidungsbaum

Für Knoten vom Typ applydb2imtreenode sind keine speziellen Eigenschaften definiert.

ISW-Assoziation

Dieses Modell-Nugget kann nicht in Skripts verwendet werden.

ISW-Sequenz

Dieses Modell-Nugget kann nicht in Skripts verwendet werden.

ISW-Regression

Für Knoten vom Typ applydb2imregnode sind keine speziellen Eigenschaften definiert.

ISW Clustering

Für Knoten vom Typ applydb2imclusternode sind keine speziellen Eigenschaften definiert.

ISW Naive Bayes

Für Knoten vom Typ applydb2imnbnode sind keine speziellen Eigenschaften definiert.

ISW Logistische Regression

Für Knoten vom Typ applydb2imlognode sind keine speziellen Eigenschaften definiert.

ISW Time Series

Dieses Modell-Nugget kann nicht in Skripts verwendet werden.

Knoteneigenschaften für IBM Netezza Analytics-Modellierung

Netezza-Modellierungsknoten – Eigenschaften

Folgende Eigenschaften haben alle IBM Netezza-Datenbank-Modellierungsknoten gemeinsam.

Allgemeine Eigenschaften von Netezza-Knoten	Werte	Eigenschaftsbeschreibung
custom_fields	*Flag*	Bei “true” (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei “false” (falsch) werden die aktuellen Einstellungen aus einem weiter oben im Stream gelegenen Typknoten verwendet.
inputs	*[field1 ... fieldN]*	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
target	*Feld*	Zielfeld (stetig oder kategorial).
record_id	*Feld*	Das als eindeutiger Bezeichner für einen Datensatz zu verwendende Feld.
use_upstream_connection	*Flag*	Falls “True” (Standard), die in einem oberhalb gelegenen Knoten angegebenen Verbindungsdetails. Wird bei Angabe von move_data_to_connection nicht verwendet.
move_data_connection	*Flag*	Falls “True”, wird der Wert in die durch connection angegebene Datenbank verschoben. Wird bei Angabe von use_upstream_connection nicht verwendet.
connection	*strukturiert*	Die Verbindungszeichenkette für die Netezza-Datenbank, in der das Modell gespeichert ist. Strukturierte Eigenschaft im Format: ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' {true|false}] Dabei gilt: <dsn> ist der Datenquellenname <username> und <psw> sind der Benutzername und das Passwort für die Datenbank. <catname> ist der Katalogname. <conn_attribs> sind die Verbindungsattribute. true | false gibt an, ob das Passwort erforderlich ist.
table_name	*string*	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.
use_model_name	*Flag*	Falls “True”, wird der von model_name angegebene Name als Name des Modells verwendet. Andernfalls wird der Modellname vom System erstellt.
model_name	*string*	Benutzerdefinierter Name für neues Modell.
include_input_fields	*Flag*	Falls “True”, werden alle Eingabefelder nach unten weitergegeben. Andernfalls werden nur record_id und vom Modell erstellte Felder weitergegeben.

Netezza-Entscheidungsbaum

Für Knoten vom Typ netezzadectreenode sind folgende Eigenschaften verfügbar.

netezzadectreenodeProperties	**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	*integer*	Maximale Anzahl der Ebenen, auf die der Baum erweitert werden kann. Der Standardwert ist 63 (größter zulässiger Wert).
min_improvement_splits	*number*	Mindestverbesserung in Unreinheit, damit eine Aufteilung stattfinden kann. Die Standardeinstellung ist 0.01.
min_instances_split	*integer*	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 im Format: set :netezza_dectree.weights = [{drugA 0.3}{drugB 0.6}] Standardgewicht ist für alle Klassen 1.
pruning_measure	Acc wAcc	Die Standardeinstellung ist Acc (Genauigkeit). Bei der alternativen Einstellung wAcc (gewichtete Genauigkeit) werden Kassengewichtungen in die Reduzierung/Beschneidung mit einbezogen.
prune_tree_options	allTrainingData partitionTrain-ingData useOtherTable	Die Standardvorgabe besteht in der Verwendung von allTrainingData zur Schätzung der Modellgenauigkeit. Verwenden Sie partitionTrainingData um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdaten-Set aus einer angegebenen Datenbanktabelle zu verwenden.
perc_training_data	*number*	Wenn prune_tree_options auf partitionTrainingData gesetzt ist, wird hier der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	*integer*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wennprune_tree_options auf partitionTrainingData gesetzt ist; Standardwert ist 1.
pruning_table	*string*	Tabellenname eines separaten Daten-Sets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	*Flag*	Wenn “True”; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.

Netezza-K-Means

Für Knoten vom Typ netezzakmeansnode sind folgende Eigenschaften verfügbar.

netezzakmeansnodeProperties	Werte	Eigenschaftsbeschreibung
distance_measure	Euclidean Manhattan Canberra maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
num_clusters	*integer*	Anzahl der zu erstellenden Cluster; Standardwert ist 3.
max_iterations	*integer*	Anzahl der Algorithmusiterationen, nach der das Modelltrainig beendet werden soll; Standardwert ist 5.
rand_seed	*integer*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert; Standardwert ist 12345.

Netezza-Bayes-Netz

Für Knoten vom Typ netezzabayesnode sind folgende Eigenschaften verfügbar.

netezzabayesnodeProperties	Werte	Eigenschaftsbeschreibung
base_index	*integer*	Die numerische Kennung, die zur internen Verwaltung dem ersten Eingabefeld zugewiesen wird; Standardwert ist 777.
sample_size	*integer*	Umfang der zu ziehenden Stichprobe, wenn die Anzahl der Attribute sehr groß ist; Standardwert ist 10.000.
display_additional_information	*Flag*	Wenn “True”, werden weitere Fortschrittsinformationen in einem Meldungsdialogfeld angezeigt.
type_of_prediction	best neighbors nn-neighbors	Typ des zu verwendenden Vorhersagealgorithmus: Beste (Nachbar mit höchster Korrelation), Nachbarn (gewichtete Vorhersage von Nachbarn) oder NN-Nachbarn (Nicht-NULL-Nachbarn).

Netezza – Naive Bayes

Für Knoten vom Typ netezzanaivebayesnode sind folgende Eigenschaften verfügbar.

netezzanaive-bayesnodeProperties	Werte	Eigenschaftsbeschreibung
compute_probabilities	*Flag*	Wenn “True”; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.
use_m_estimation	*Flag*	Wenn “True”, wird das m-Schätzverfahren zur Vermeidung der Wahrscheinlichkeit null während der Schätzung verwendet.

Netezza-KNN

Für Knoten vom Typ netezzaknnnode sind folgende Eigenschaften verfügbar.

netezzaknnnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
weights	*strukturiert*	Strukturierte Eigenschaft, die zur Zuweisung von Gewichten für die einzelnen Klassen verwendet wird. Beispiel: set :netezzaknnnode.weights = [{drugA 0.3}{drugB 0.6}]
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
num_nearest_neighbors	*integer*	Anzahl der nächsten Nachbarn für einen bestimmten Fall; Standardwert ist 3
standardize_measurements	*Flag*	Wenn "True", werden vor der Berechnung der Abstandwerte die Messungen für stetige Eingabefelder standardisiert.
use_coresets	*Flag*	Wenn "True", wird Stichprobennahme mit Core-Sets verwendet, um die Berechnung bei großen Daten-Sets zu beschleunigen.

Netezza – Divisives Clustering

Für Knoten vom Typ netezzadivclusternode sind folgende Eigenschaften verfügbar.

netezzadivclustern-odeProperties	**Werte**	**Eigenschaftsbeschreibung**
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
max_iterations	*integer*	Maximale Anzahl an Algorithmusiterationen, die durchgeführt werden sollen, bevor das Modelltraining beendet wird; Standardwert ist 5.
max_tree_depth	*integer*	Maximale Anzahl an Ebenen, in die das Daten-Set unterteilt werden kann; Standardwert ist 3.
rand_seed	*integer*	Für die Reproduktion von Analysen verwendeter Zufallsstartwert; Standardwert ist 12345.
min_instances_split	*integer*	Mindestanzahl von Datensätzen, die aufgeteilt werden können; Standardwert ist 5.
level	*integer*	Hierarchieebene, auf der die Datensätze gescort werden; Standardwert ist -1.

Netezza-PCA

Für Knoten vom Typ netezzapcanode sind folgende Eigenschaften verfügbar.

netezzapcanodeProperties	Werte	Eigenschaftsbeschreibung
center_data	*Flag*	Wenn "True" (Standard), wird vor der Analyse Datenzentrierung (auch als Mittelwertsubtraktion bezeichnet) durchgeführt.
perform_data_scaling	*Flag*	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	*Flag*	Wenn "True", wird eine weniger genaue, jedoch schnellere Methode zur Ermittlung der Hauptkomponenten verwendet.
pc_number	*integer*	Anzahl an Hauptkomponenten, auf die das Daten-Set reduziert werden soll; Standardwert ist 1.

Netezza-Regressionsbaum

Für Knoten vom Typ netezzaregtreenode sind folgende Eigenschaften verfügbar.

netezzaregtreenodeProperties	Werte	Eigenschaftsbeschreibung
max_tree_depth	*integer*	Maximale Anzahl an Ebenen, auf die ein Baum unterhalb des Stammknotens erweitert werden kann; Standardwert ist 10.
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	*number*	Mindestwert der Unreinheitsreduzierung, bevor eine neue Aufteilung des Baums erfolgt.
min_instances_split	*integer*	Mindestanzahl an Datensätzen, die aufgeteilt werden kann.
pruning_measure	mse r2 pearson spearman	Für die Reduzierung zu verwendende Methode.
prune_tree_options	allTrainingData partitionTrain- ingData useOtherTable	Die Standardvorgabe besteht in der Verwendung von allTrainingData zur Schätzung der Modellgenauigkeit. Verwenden Sie partitionTrainingData um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdaten-Set aus einer angegebenen Datenbanktabelle zu verwenden.
perc_training_data	*number*	Wenn prune_tree_options auf PercTrainingData gesetzt ist, wird hier der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	*integer*	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wennprune_tree_options auf PercTrainingData gesetzt ist; Standardwert ist 1.

netezzaregtreenodeProperties	Werte	Eigenschaftsbeschreibung
pruning_table	*string*	Tabellenname eines separaten Daten-Sets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	*Flag*	Wenn "True", wird angegeben, ob die Varianz zugewiesener Klassen in die Ausgabe aufgenommen werden soll.

Netezza – Lineare Regression

Für Knoten vom Typ netezzalineregressionnode sind folgende Eigenschaften verfügbar.

netezzalineregressionnodeProperties	Werte	Eigenschaftsbeschreibung
use_svd	*Flag*	Wenn "True", wird anstelle der ursprünglichen Matrix die Matrix zur Einzelwertzerlegung verwendet, um eine höhere Geschwindigkeit und numerische Genauigkeit zu erreichen.
include_intercept	*Flag*	Wenn "True" (Standard), wird die Gesamtgenauigkeit der Lösung erhöht.
calculate_model_diagnostics	*Flag*	Wenn "True", werden Diagnosedaten für das Modell berechnet.

Netezza-Zeitreihe

Für Knoten vom Typ netezzatimeseriesnode sind folgende Eigenschaften verfügbar.

netezzatimeseriesnodeProperties	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 SeasonalTrendDecomposition oder std	Für die Modellierung von Zeitreihen zu verwendender Algorithmus.
trend_name	*string*	Trendtyp für exponentielles Glätten.

netezzatime-seriesnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
seasonality_type	*string*	Saisonalitätstyp für exponentielles Glätten.
interpolation_method	linear cubicspline exponential-spline	Zu verwendende Interpolationsmethode.
earliest_time	*Feld*	Startzeit bei Verwendung eines Teilbereichs der Zeitreihe.
latest_time	*Feld*	Endzeit bei Verwendung eines Teilbereichs der Zeitreihe.
p	*integer*	ARIMA – nichtsaisonale Autokorrelationsmaße.
q	*integer*	ARIMA – nichtsaisonaler Ableitungswert.
d	*integer*	ARIMA – nichtsaisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.
sp	*integer*	ARIMA – saisonale Autokorrelationsmaße.
sq	*integer*	ARIMA – saisonaler Ableitungswert.
sd	*integer*	ARIMA – saisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.
period	*integer*	Länge des saisonalen Zyklus, die in Verbindung mit units_period angegeben wird. Wird nicht für Spektralanalyse verwendet.
units_period	Milliseconds Seconds Minutes Hours Days Weeks Quarters Years	Einheiten für period. Eine wöchentliche Zeitreihe hat zum Beispiel den Wert 1 für period und Weeks für units_period.
include_history	*Flag*	Gibt an, ob historische Werte bei der Ausgabe berücksichtigt werden sollen.
include_interpolated_values	*Flag*	Gibt an, ob interpolierte Werte bei der Ausgabe berücksichtigt werden sollen. Wird nicht verwendet, wenn include_history auf false gesetzt ist.

Netezza Allgemeines lineares Modell

Für Knoten vom Typ netezzaglmnode sind folgende Eigenschaften verfügbar.

netezzaglmnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
response_variable_distribution	bernoulli gaussian poisson binomial negativebino-mial wald gamma	Verteilungstyp. Der Standardwert ist bernoulli.
distribution_parameter	*number*	Zu verwendender Wert für Verteilungsparameter. Wird nur verwendet, wenn distribution auf Negativebinomial gesetzt ist.

netezzaglmnodeProperties	**Werte**	**Eigenschaftsbeschreibung**
trials	*integer*	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 Netezza allgemeine lineare Modell gespeichert werden soll.
max_iterations	*integer*	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden sollen. Der Standardwert ist 20.
max_error	*number*	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.
error_threshold	*number*	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.
link_function	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Zu verwendende Verknüpfungsfunktion. Der Standardwert ist logit.
link_function_parameter	*number*	Zu verwendender Wert für Verknüpfungsfunktionsparameter. Wird nur verwendet, wenn link_function auf power oder oddspower gesetzt ist.
intercept	*Flag*	Wenn auf true gesetzt, wird konstanter Term in Modell einbezogen.

Netezza-Modell-Nugget – Eigenschaften

Folgende Eigenschaften haben alle Modell-Nuggets von Netezza-Datenbanken gemeinsam.

Allgemeine Eigenschaften von Netezza-Modell-Nuggets	**Werte**	**Eigenschaftsbeschreibung**
connection	*string*	Die Verbindungszeichenkette für die Netezza-Datenbank, in der das Modell gespeichert ist.
table_name	*string*	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.

Die anderen Eigenschaften des Modell-Nuggets stimmen mit denen für den zugehörigen Modellierungsknoten überein.

Die Skriptnamen des Modell-Nuggets lauten wie folgt.

Modell-Nugget	**Skriptname**
Entscheidungsbaum	applynetezzadectreenode
K-Means	applynetezzakmeansnode
Bayes-Netz	applynetezzabayesnode
Naive Bayes	applynetezzanaivebayesnode
KNN	applynetezzaknnnode
Divisives Clustering	applynetezzadivclusternode
PCA	applynetezzapcanode
Regressionsbaum	applynetezzaregtreenode
Lineare Regression	applynetezzalineregressionnode
Zeitreihen	applynetezzatimeseriesnode
Verallgemeinert Linear (GenLin)	applynetezzaglmnode

Kapitel
19

Ausgabeknoten – Eigenschaften

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 Stream-Parameter festgelegt wird.

In diesem Abschnitt werden die für Ausgabeknoten verfügbaren Skript-Eigenschaften beschrieben.

Eigenschaften von "analysisnode"

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 Modell-Nuggets angestellt. Sie können außerdem Vorhersagemodelle miteinander vergleichen. Für weitere Informationen siehe Thema Analyseknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create analysisnode
# "Analysis" tab
set :analysisnode.coincidence = True
set :analysisnode.performance = True
set :analysisnode.confidence = True
set :analysisnode.threshold = 75
set :analysisnode.improve_accuracy = 3
set :analysisnode.inc_user_measure = True
# "Define User Measure..."
set :analysisnode.user_if = "@TARGET = @PREDICTED"
set :analysisnode.user_then = "101"
set :analysisnode.user_else = "1"
set :analysisnode.user_compute = [Mean Sum]
set :analysisnode.by_fields = ['Drug']
# "Output" tab
set :analysisnode.output_format = HTML
set :analysisnode.full_filename = "C:/output/analysis_out.html"
```

analysisnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.

analysisnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
output_name	*string*	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	*string*	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
coincidence	*Flag*	
performance	*Flag*	
confidence	*Flag*	
threshold	*number*	
improve_accuracy	*number*	
inc_user_measure	*Flag*	
user_if	*Ausdr*	
user_then	*Ausdr*	
user_else	*Ausdr*	
user_compute	[Mean Sum Min Max SDev]	

Eigenschaften von "dataauditnode"

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. Für weitere Informationen siehe Thema Data Audit-Knoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create dataauditnode
connect :variablefilenode to :dataauditnode
set :dataauditnode.custom_fields = True
set :dataauditnode.fields = [Age Na K]
set :dataauditnode.display_graphs = True
set :dataauditnode.basic_stats = True
set :dataauditnode.advanced_stats = True
set :dataauditnode.median_stats = False
set :dataauditnode.calculate = [Count Breakdown]
set :dataauditnode.outlier_detection_method = std
set :dataauditnode.outlier_detection_std_outlier = 1.0
set :dataauditnode.outlier_detection_std_extreme = 3.0
```

set :dataauditnode.output_mode = Screen

dataauditnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
custom_fields	*Flag*	
fields	*[Feld1 ... FeldN]*	
overlay	*Feld*	
display_graphs	*Flag*	Dient zur Aktivierung bzw. Deaktivierung der Anzeige von Diagrammen in der Ausgabematrix.
basic_stats	*Flag*	
advanced_stats	*Flag*	
median_stats	*Flag*	
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	*number*	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	*number*	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	*number*	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	*number*	Wenn für outlier_detection_method die Option iqr verwendet wird, wird die Zahl angegeben, die für die Definition der Extremwerte verwendet werden soll.
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.

dataauditnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
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	*Flag*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	*number*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	*string*	

Eigenschaften von "matrixnode"

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 Flag-Feldern oder numerischen Feldern herangezogen werden. Für weitere Informationen siehe Thema Matrixknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create matrixnode
# "Settings" tab
set :matrixnode.fields = Numerics
set :matrixnode.row = 'K'
set :matrixnode.column = 'Na'
set :matrixnode.cell_contents = Function
set :matrixnode.function_field = 'Age'
set :matrixnode.function = Sum
# "Appearance" tab
set :matrixnode.sort_mode = Ascending
set :matrixnode.highlight_top = 1
set :matrixnode.highlight_bottom = 5
set :matrixnode.display = [Counts Expected Residuals]
set :matrixnode.include_totals = True
# "Output" tab
set :matrixnode.full_filename = "C:/output/matrix_output.html"
set :matrixnode.output_format = HTML
set :matrixnode.paginate_output = true
```

set :matrixnode.lines_per_page = 50

matrixnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields	Selected Flags Numerics	
row	*Feld*	
column	*Feld*	
include_missing_values	*Flag*	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	*string*	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Ascending Descending	
highlight_top	*number*	Wenn ungleich 0, dann ist die Eigenschaft wahr.
highlight_bottom	*number*	Wenn ungleich 0, dann ist die Eigenschaft wahr.
display	[Counts Expected Residuals RowPct ColumnPct TotalPct]	
include_totals	*Flag*	
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	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.

matrixnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
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 Modifizierfaktor transposed verwendet werden, der die Zeilen und Spalten in der Tabelle transponiert. Beispiel: NODE.output_format=transposed Delimited
paginate_output	*Flag*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	*number*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	*string*	

Eigenschaften von "meansnode"

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. Für weitere Informationen siehe Thema Mittelwertknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create meansnode
set :meansnode.means_mode = BetweenFields
set :meansnode.paired_fields = [{'OPEN_BAL' 'CURR_BAL'}]
set :meansnode.label_correlations = true
set :meansnode.output_view = Advanced
set :meansnode.output_mode = File
set :meansnode.output_format = HTML
set :meansnode.full_filename = "C:/output/means_output.html"
```

meansnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
means_mode	BetweenGroups BetweenFields	Gibt den Typ der Mittelwertstatistik an, die für die Daten ausgeführt werden soll.
test_fields	[field1 ... fieldn]	Gibt das Testfeld an, wenn means_mode auf BetweenGroups gesetzt ist.
grouping_field	*Feld*	Gibt das Gruppierungsfeld an.

meansnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
paired_fields	[{field1 field2} {field3 field4} ...]	Gibt die zu verwendenden Feldpaare an, wenn means_mode auf BetweenFields gesetzt ist.
label_correlations	*Flag*	Gibt an, ob Korrelationsbeschriftungen in der Ausgabe angezeigt werden sollen. Diese Einstellung greift nur, wenn means_mode auf BetweenFields festgelegt ist.
correlation_mode	Probability Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
weak_label	*string*	
medium_label	*string*	
strong_label	*string*	
weak_below_probability	*number*	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Cutoff-Wert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	*number*	Cutoff-Wert für starke Korrelationen.
weak_below_absolute	*number*	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Cutoff-Wert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	*number*	Cutoff-Wert für starke Korrelationen.
unimportant_label	*string*	
marginal_label	*string*	
important_label	*string*	
unimportant_below	*number*	Cutoff-Wert für niedrige Feldwichtigkeit. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
important_above	*number*	
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	Zu verwendender Name.
output_mode	Screen File	Gibt den Zielort für die vom Ausgabeknoten erstellte Ausgabe an.

meansnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Gibt den Ausgabetyp an.
full_filename	*string*	
output_view	Simple Advanced	Gibt an, ob die einfache oder die erweiterte Ansicht in der Ausgabe angezeigt werden soll.

Eigenschaften von "reportnode"

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. Für weitere Informationen siehe Thema Berichtknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create reportnode
set :reportnode.output_format = HTML
set :reportnode.full_filename = "C:/report_output.html"
set :reportnode.lines_per_page = 50
set :reportnode.title = "Report node created by a script"
set :reportnode.highlights = False
```

reportnode-Eigenschaften	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	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
text	*string*	
full_filename	*string*	
highlights	*Flag*	
title	*string*	
lines_per_page	*number*	

Eigenschaften von "setglobalsnode"

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. Für weitere Informationen siehe Thema Globalwerteknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create setglobalsnode
connect :typenode to :setglobalsnode
set :setglobalsnode.globals.Na = [Max Sum Mean]
set :setglobalsnode.globals.K = [Max Sum Mean]
set :setglobalsnode.globals.Age = [Max Sum Mean SDev]
set :setglobalsnode.clear_first = False
set :setglobalsnode.show_preview = True
```

setglobalsnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
globals	[Sum Mean Min Max SDev]	Strukturierte Eigenschaft, bei der festzulegende Felder mit folgender Syntax referenziert werden müssen: set :setglobalsnode.globals.Age = [Sum Mean Min Max SDev]
clear_first	*Flag*	
show_preview	*Flag*	

Eigenschaften von "statisticsnode"

Der Statistikknoten liefert grundlegende Übersichtsdaten zu numerischen Feldern. Er berechnet Übersichtsstatistiken für einzelne Felder und für die Korrelationen zwischen den Feldern. Für weitere Informationen siehe Thema Statistikknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create statisticsnode
# "Settings" tab
set :statisticsnode.examine = ['Age' 'BP' 'Drug']
set :statisticsnode.statistics = [Mean Sum SDev]
set :statisticsnode.correlate = ['BP' 'Drug']
# "Correlation Labels..." section
set :statisticsnode.label_correlations = True
set :statisticsnode.weak_below_absolute = 0.25
set :statisticsnode.weak_label = "lower quartile"
set :statisticsnode.strong_above_absolute = 0.75
set :statisticsnode.medium_label = "middle quartiles"
set :statisticsnode.strong_label = "upper quartile"
```

```
# "Output" tab
set :statisticsnode.full_filename = "c:/output/statistics_output.html"
set :statisticsnode.output_format = HTML
```

statisticsnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	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	*string*	
examine	[*Feld Feld Feld*]	
correlate	[*Feld Feld Feld*]	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Probability Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
label_correlations	*Flag*	
weak_label	*string*	
medium_label	*string*	
strong_label	*string*	
weak_below_probability	*number*	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Cutoff-Wert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	*number*	Cutoff-Wert für starke Korrelationen.
weak_below_absolute	*number*	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Cutoff-Wert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	*number*	Cutoff-Wert für starke Korrelationen.

Eigenschaften von “statisticsoutputnode”

Mit dem Statistikausgabeknoten können Sie eineIBM® SPSS® Statistics-Prozedur aufrufen, um Ihre IBM® SPSS® Modeler-Daten zu analysieren. Es stehen zahlreiche SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistikausgabeknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie unter Eigenschaften von “statisticsoutputnode” auf S. 324.

Eigenschaften von “tablenode”

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. Für weitere Informationen siehe Thema Tabellenknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create tablenode
set :tablenode.highlight_expr = "Age > 30"
set :tablenode.output_format = HTML
set :tablenode.transpose_data = true
set :tablenode.full_filename = "C:/output/table_output.htm"
set :tablenode.paginate_output = true
set :tablenode.lines_per_page = 50
```

tablenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
full_filename	*string*	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	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.
transpose_data	*Flag*	Transponiert die Daten vor dem Export, sodass die Zeilen Felder und die Spalten Datensätze darstellen.
paginate_output	*Flag*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.

tablenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
lines_per_page	*number*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
highlight_expr	*string*	
output	*string*	Eine schreibgeschützte Eigenschaft, die eine Referenz zur letzten vom Knoten erstellten Tabelle enthält.
value_labels	*[{Wert Beschrif-tungsstring} {Wert Beschriftungsstring} ...]*	Gibt Beschriftungen für Wertpaare an. Beispiel: set :typenode.value_labels. 'Drug'=[{drugA label1} {drugB label2}]
display_places	*integer*	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 Stream-Standard verwendet. Format: NODE.display_places. FIELDNAME
export_places	*integer*	Legt die Dezimalstellen für das Feld beim Exportieren fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Mit dem Wert –1 wird der Stream-Standard verwendet. Format: NODE.export_places.FIELDNAME
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Format: NODE.decimal_separator. FIELDNAME
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" TAG MONAT "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"	Legt das Datumsformat für das Feld fest (gilt nur für Felder mit dem Speichertyp DATE oder TIMESTAMP). Format: NODE.date_format.FIELDNAME Beispiel: set :tablenode.date_format. 'LaunchDate' = "DDMMYY"

tablenode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
	"TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	
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). Format: NODE.time_format.FIELDNAME Beispiel: set :tablenode.time_format. set 'BOF_enter' = "HHMMSS"
column_width	*integer*	Legt die Spaltenbreite für das Feld fest. Mit dem Wert –1 wird die Spaltenbreite auf Auto eingestellt. Format: NODE.column_width.FIELDNAME
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest. Format: NODE.justify.FIELDNAME

Eigenschaften von "transformnode"

Mit dem Transformationsknoten können Sie die Ergebnisse von Transformationen auswählen und in einer Vorschau anzeigen, bevor Sie sie auf ausgewählte Felder anwenden. Für weitere Informationen siehe Thema Transformationsknoten in Kapitel 6 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create transformnode
set :transformnode.fields = [AGE INCOME]
set :transformnode.formula = Select
set :transformnode.formula_log_n = true
set :transformnode.formula_log_n_offset = 1
```

transformnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
fields	[*Feld1… Feldn*]	Die bei der Transformation zu verwendenden Felder.
formula	All Select	Gibt an, ob alle oder nur die ausgewählten Transformationen berechnet werden sollen.

transformnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
formula_inverse	*Flag*	Gibt an, ob die Kehrwerttransformation verwendet werden soll.
formula_inverse_offset	*number*	Gibt einen Daten-Offset an, der für die Formel verwendet werden soll. Standardmäßig auf 0 gesetzt, sofern vom Benutzer nicht anders angegeben.
formula_log_n	*Flag*	Gibt an, ob die $\log_n$-Transformation verwendet werden soll.
formula_log_n_offset	*number*	
formula_log_10	*Flag*	Gibt an, ob die $\log_{10}$-Transformation verwendet werden soll.
formula_log_10_offset	*number*	
formula_exponential	*Flag*	Gibt an, ob die exponentielle Transformation (e^x) verwendet werden soll.
formula_square_root	*Flag*	Gibt an, ob die Quadratwurzeltransformation verwendet werden soll.
use_output_name	*Flag*	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	*string*	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	HTML (*.html*) Output (*.cou*)	Dient zur Angabe des Ausgabetyps.
paginate_output	*Flag*	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	*number*	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	*string*	Gibt den für die Dateiausgabe zu verwendenden Dateinamen an.

Kapitel
20

Exportknoten – Eigenschaften

Exportknoten – Allgemeine Eigenschaften

Folgende Eigenschaften haben alle Exportknoten gemeinsam:

Eigenschaft	Werte	Eigenschaftsbeschreibung
publish_path	*string*	Geben Sie den Stammnamen für die veröffentlichten Image- und Parameterdateien an.
publish_metadata	*Flag*	Gibt an, ob eine Metadatendatei erzeugt wird, welche die Ein- und Ausgaben des Bilds und der zugehörigen Datenmodelle beschreibt.
publish_use_parameters	*Flag*	Gibt an, ob Stream-Parameter in der *.par-Datei enthalten sind.
publish_parameters	*Stringliste*	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 von "cognosexportnode"

Der IBM Cognos BI-Exportknoten exportiert Daten in einem Format, das von Cognos BI-Datenbanken gelesen werden kann. Für weitere Informationen siehe Thema IBM Cognos BI-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Anmerkung: 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.

cognosexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
cognos_connection	*{"field","field", ... ,"field"}*	Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: *{"Cognos_server_URL", login_mode, "namespace", "username", "password"}* Dabei gilt: *Cognos_server_URL* ist die URL des Cognos-Servers, auf den Sie exportieren.

cognosexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
		login_mode gibt an, ob eine anonyme Anmeldung verwendet wird, und entspricht entweder true oder false; Wenn true festgelegt ist, sollten die folgenden Felder auf "" festgelegt werden. *namespace* gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. *username* und *password* sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.
cognos_package_name	*string*	Pfad und Name des Cognos-Pakets, an die Sie Daten exportieren, z. B.: /Public Folders/MyPackage
cognos_datasource	*string*	
cognos_export_mode	Publish ExportFile	
cognos_filename	*string*	

ODBC-Verbindung

Die Eigenschaften für die ODBC-Verbindung sind identisch mit denen, die im nächsten Bereich für databaseexportnode aufgelistet sind, mit der Ausnahme, dass die Eigenschaft der datasource nicht gültig ist.

Eigenschaften von "databaseexportnode"

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. Für weitere Informationen siehe Thema Datenbankexportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
/*
Use this sample with fraud.str from demo folder
Assumes a datasource named "MyDatasource" has been configured
*/
create databaseexport
connect claimvalue:applyneuralnetwork to :databaseexport
 # Export tab
set :databaseexport.username = "user"
set :databaseexport.datasource = "MyDatasource"
set :databaseexport.password = "password"
set :databaseexport.table_name = "predictions"
set :databaseexport.write_mode = Create
set :databaseexport.generate_import = true
set :databaseexport.drop_existing_table = true
set :databaseexport.delete_existing_rows = true
set :databaseexport.default_string_size = 32
```

```
# Schema dialog
set :databaseexport.type.region = "VARCHAR(10)"
set :databaseexport.export_db_primarykey.id = true
set :databaseexportnode.use_custom_create_table_command = true
set :databaseexportnode.custom_create_table_command = "My SQL Code"

# Indexes dialog
set :databaseexport.use_custom_create_index_command = true
set :databaseexport.custom_create_index_command = \
  "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"
set :databaseexport.indexes.MYINDEX.fields = [id region]
```

databaseexportnode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
datasource	*string*	
username	*string*	
password	*string*	
epassword	*string*	Dieser Slot ist während der Ausführung schreibgeschützt. Um ein kodiertes Passwort zu erstellen, verwenden Sie das Passwort-Tool im Menü “Extras”. Für weitere Informationen siehe Thema Erstellen eines verschlüsselten Passworts in Kapitel 5 auf S. 63.
table_name	*string*	
write_mode	Create Append Merge	
map	*string*	Ordnet einen Stream-Feldnamen zu einer Datenbankspalte zu (nur gültig, wenn write_mode auf Merge eingestellt ist). Beispiel: set :databaseexportn-ode.map.streamBP = 'databaseBP' Je nach Feldposition werden Mehrfachzuordnungen unterstützt, zum Beispiel: set :databaseexportn-ode.map=[{streamfield1 field1}{streamfield2 field2}{streamfield3 field3}] 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.

databaseexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
key_fields	*[Feld Feld ... Feld]*	Gibt an, dass das Stream-Feld für "key" verwendet wird. Die map-Eigenschaft zeigt, welche Entsprechung in der Datenbank vorhanden ist.
join	Database Add	Beispiel: set : databaseexportnode.join = Database
drop_existing_table	*Flag*	
delete_existing_rows	*Flag*	
default_string_size	*integer*	
type		Strukturierte Eigenschaft, mit der das Schema festgelegt wird. Format: set :databaseexportnode.type.BP = 'VARCHAR(10)'
generate_import	*Flag*	
use_custom_create_table_command	*Flag*	Mit dem Slot *custom_create_table* ändern Sie den SQL-Standardbefehl CREATE TABLE.
custom_create_table_command	*string*	Gibt eine Befehlszeichenkette an, die statt des SQL-Standardbefehls CREATE TABLE verwendet werden soll.
use_batch	*Flag*	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 deaktiviert.
batch_size	*number*	Gibt die 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.
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	*string*	

databaseexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
specify_data_file	*Flag*	Mit dem Flag "wahr" 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	*string*	
specify_loader_program	*Flag*	Mit dem Flag "wahr" wird die unten genannte Eigenschaft loader_program aktiviert, mit der Sie den Namen und den Speicherort eines externen Ladeskripts oder Ladeprogramms angeben können.
loader_program	*string*	
gen_logfile	*Flag*	Mit dem Flag "wahr" 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	*string*	
check_table_size	*Flag*	Mit dem Flag "wahr" 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	*string*	Legen Sie für das Ladeprogramm zusätzliche Argumente fest, z. B. -comment und -specialdir.
export_db_primarykey	*Flag*	Gibt an, ob es sich bei einem bestimmten Feld um einen Primärschlüssel handelt.
use_custom_create_index_command	*Flag*	Bei true wird hiermit benutzerdefinierte SQL für alle Indizes aktiviert.
custom_create_index_command	*string*	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.

databaseexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
indexes.INDEXNAME.use_custom_create_index_command	*Flag*	Wird zum Aktivieren bzw. Deaktivieren 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	*Flag*	Bei true wird hiermit der angegebene Index aus der Menge der Indizes entfernt.
table_space	*string*	Gibt den zu erstellenden Tabellenbereich an.
use_partition	*Flag*	Gibt an, dass das Verteilungs-Hash-Feld verwendet werden soll.
partition_field	*string*	Gibt den Inhalt des Verteilungs-Hash-Felds 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.

databaseexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
use_compression	*Boolesch*	Wenn diese Option auf true gesetzt ist, 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.

Beispiel – SQL Server

```
var DBSource
set DBSource = get node TestCompressionSQL
set ^DBSource.use_compression = true
set ^DBSource.compression_mode = Page

execute DBSource
```

Beispiel – Oracle 11gR1

```
var DBSource
set DBSource = get node TestCompressionOracle11gR1
set ^DBSource.use_compression = true
set ^DBSource.compression_mode = Direct_Load_Operations

execute DBSource
```

Beispiel – Oracle 11gR2

```
var DBSource
set DBSource = get node TestCompressionOracle11gR2
set ^DBSource.use_compression = true
set ^DBSource.compression_mode = Basic

execute DBSource
```

Eigenschaften von "datacollectionexportnode"

Der IBM® SPSS® Data Collection-Exportknoten gibt Daten in dem von der Marktforschungssoftware Data Collection verwendeten Format aus. Um diesen Knoten verwenden zu können, muss die Data Collection Data Library installiert sein. Für weitere Informationen siehe Thema IBM SPSS Data Collection-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create datacollectionexportnode
set :datacollectionexportnode.metadata_file = "c:\museums.mdd"
set :datacollectionexportnode.merge_metadata = Overwrite
set :datacollectionexportnode.casedata_file = "c:\museumdata.sav"
set :datacollectionexportnode.generate_import = true
set :datacollectionexportnode.enable_system_variables = true
```

datacollectionexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
metadata_file	*string*	Name der zu exportierenden Metadatendatei.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*Flag*	Gibt an, ob die exportierte *.mdd*-Datei Data Collection-Systemvariablen enthalten soll.
casedata_file	*string*	Der Name der *.sav*-Datei, in die die Falldaten exportiert werden.
generate_import	*Flag*	

Eigenschaften von “excelexportnode”

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. Für weitere Informationen siehe Thema Excel-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create excelexportnode
set :excelexportnode.full_filename = "C:/output/myexport.xls"
set :excelexportnode.excel_file_type = Excel2007
set :excelexportnode.inc_field_names = True
set :excelexportnode.inc_labels_as_cell_notes = False
set :excelexportnode.launch_application = True
set :excelexportnode.generate_import = True
```

excelexportnode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
full_filename	*string*	
excel_file_type	Excel2003 Excel2007	
export_mode	Create Append	
inc_field_names	*Flag*	Gibt an, ob Feldnamen in die erste Zeile des Arbeitsblattes eingefügt werden sollen.
start_cell	*string*	Gibt die Startzelle für den Export an.
worksheet_name	*string*	Name des zu schreibenden Arbeitsblattes.
launch_application	*Flag*	Gibt an, ob Excel für die resultierende Datei aufgerufen werden soll. Beachten Sie, dass der Pfad für den Start von Excel im Dialogfeld “Hilfsprogramme” (Menü “Extras”, “Hilfsprogramme”) angegeben werden muss.
generate_import	*Flag*	Gibt an, ob ein Excel-Importknoten generiert werden soll, der die exportierte Datendatei liest.

Eigenschaften von “outputfilenode”

Der Textdatei-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. Für weitere Informationen siehe Thema Textdatei-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create outputfile
set :outputfile.full_filename = "c:/output/flatfile_output.txt"
set :outputfile.write_mode = Append
set :outputfile.inc_field_names = False
set :outputfile.use_newline_after_records = False
set :outputfile.delimit_mode = Tab
set :outputfile.other_delimiter = ","
set :outputfile.quote_mode = Double
set :outputfile.other_quote = "*"
set :outputfile.decimal_symbol = Period
set :outputfile.generate_import = True
```

outputfilenode-Eigenschaften	**Datentyp**	**Eigenschaftsbeschreibung**
full_filename	*string*	Name der Ausgabedatei.
write_mode	Overwrite Append	
inc_field_names	*Flag*	
use_newline_after_records	*Flag*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*Element*	
quote_mode	None Single Double Other	
other_quote	*Flag*	
generate_import	*Flag*	
encoding	StreamDefault SystemDefault "UTF-8"	

Eigenschaften von "sasexportnode"

Mit dem SAS-Exportknoten werden Daten in das SAS-Format ausgegeben, die dann in SAS oder in SAS-kompatible Softwarepakete eingelesen werden können. Es stehen drei SAS-Dateiformate zur Verfügung: SAS für Windows/OS2, SAS für UNIX sowie SAS Version 7/8 Für weitere Informationen siehe Thema SAS-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create sasexportnode
set :sasexportnode.full_filename = "c:/output/SAS_output.sas7bdat"
set :sasexportnode.format = SAS8
set :sasexportnode.export_names = NamesAndLabels
```

set :sasexportnode.generate_import = True

sasexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
format	Windows UNIX SAS7 SAS8	Beschriftungsfelder für die Eigenschaft “Variante”.
full_filename	*string*	
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	*Flag*	

Eigenschaften von “statisticsexportnode

Der Statistikexportknoten gibt Daten im Format IBM® SPSS® Statistics.*sav* aus. Die *.sav*-Dateien können von SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cache-Dateien in IBM® SPSS® Modeler verwendet. Für weitere Informationen siehe Thema Statistikexportknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie unter Eigenschaften von “statisticsexportnode auf S. 324.

xmlexportnode Eigenschaften

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 den Stream einzulesen. Für weitere Informationen siehe Thema XML-Exportknoten in Kapitel 7 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create xmlexportnode
set :xmlexportnode.full_filename = "c:\export\data.xml"
set :xmlexportnode.map = [{"/catalog/book/genre" genre}{"/catalog/book/title" title}]
```

xmlexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
full_filename	*string*	(erforderlich) Vollständiger Pfad und Dateiname der XML-Exportdatei.
use_xml_schema	*Flag*	Legt fest, ob ein XML-Schema (XSD- oder DTD-Datei) für die Steuerung der Struktur der exportierten Daten verwendet wird.
full_schema_filename	*string*	Vollständiger Pfad und Dateiname der zu verwendenden XSD- oder DTD-Datei. Erforderlich, wenn use_xml_schema auf “wahr” gesetzt ist.

xmlexportnode-Eigenschaften	Datentyp	Eigenschaftsbeschreibung
generate_import	*Flag*	Generiert einen XML-Quellenknoten, der die exportierten Daten wieder in den Stream einliest.
records	*string*	XPath-Ausdruck, der die Datensatzgrenze angibt.
map	*string*	Ordnet den Feldnamen der XML-Struktur zu. Beispiel: set :xmlexportnode.map = [{"/top/node1" field1}{"/top/node2" field2}] Dadurch wird das Stream-Feld field1 dem XML-Element /top/node1 zugeordnet, usw.

Kapitel
21

IBM SPSS Statistics-Knoteneigenschaften

Eigenschaften von "statisticsimportnode"

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. Für weitere Informationen siehe Thema Statistikdateiknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create statisticsimportnode
set :statisticsimportnode.full_filename = "C:/data/drug1n.sav"
set :statisticsimportnode.import_names = true
set :statisticsimportnode.import_data = true
```

Eigenschaften von statisticsimportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	*string*	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 SPSS Statistics-Feldformatinformationen beim Import verwendet werden.

Eigenschaften von "statisticstransformnode"

Der Statistiktransformationsknoten führt eine Auswahl von IBM® SPSS® Statistics-Syntaxbefehlen an Datenquellen in IBM® SPSS® Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistiktransformationsknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create statisticstransformnode
set :statisticstransformnode.syntax = "COMPUTE NewVar = Na + K."
set :statisticstransformnode.new_name.NewVar = "Mixed Drugs"
```

set :statisticstransformnode.check_before_saving = true

Eigenschaften von statisticstransformnode	Datentyp	Eigenschaftsbeschreibung
Syntax	*string*	
check_before_saving	*Flag*	Überprüft die eingegebene Syntax vor dem Speichern der Einträge. Zeigt eine Fehlermeldung an, wenn die Syntax ungültig ist.
default_include	*Flag*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.
include	*Flag*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.
new_name	*string*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.

Eigenschaften von “statisticsmodelnode”

Mithilfe des Knotens “Statistikmodell” 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 SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistikmodellknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

create statisticsmodelnode
set :statisticsmodelnode.syntax = "COMPUTE NewVar = Na + K."
set :statisticsmodelnode.new_name.NewVar = "Mixed Drugs"

Eigenschaften von statisticsmodelnode	Datentyp	Eigenschaftsbeschreibung
Syntax	*string*	
default_include	*Flag*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.
include	*Flag*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.
new_name	*string*	Für weitere Informationen siehe Thema Eigenschaften von “filternode” in Kapitel 14 auf S. 167.

Eigenschaften von “statisticsoutputnode”

Mit dem Statistikausgabeknoten können Sie eineIBM® SPSS® Statistics-Prozedur aufrufen, um Ihre IBM® SPSS® Modeler-Daten zu analysieren. Es stehen zahlreiche SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von SPSS Statistics erforderlich. Für weitere Informationen siehe Thema Statistikausgabeknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create statisticsoutputnode
set :statisticsoutputnode.syntax = "SORT CASES BY Age(A) Sex(A) BP(A) Cholesterol(A)"
set :statisticsoutputnode.use_output_name = False
set :statisticsoutputnode.output_mode = File
set :statisticsoutputnode.full_filename = "Cases by Age, Sex and Medical History"
set :statisticsoutputnode.file_type = HTML
```

Eigenschaften von statisticsoutputnode	Datentyp	Eigenschaftsbeschreibung
mode	Dialog Syntax	Wählt die Option “SPSS Statistics-Dialogfeld” oder Syntaxeditor aus
Syntax	*string*	
use_output_name	*Flag*	
output_name	*string*	
output_mode	Screen Datei	
full_filename	*string*	
file_type	HTML SPV SPW	

Eigenschaften von “statisticsexportnode

Der Statistikexportknoten gibt Daten im Format IBM® SPSS® Statistics.*sav* aus. Die *.sav*-Dateien können von SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cache-Dateien in IBM® SPSS® Modeler verwendet. Für weitere Informationen siehe Thema Statistikexportknoten in Kapitel 8 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Beispiel

```
create statisticsexportnode
set :statisticsexportnode.full_filename = "c:/output/SPSS_Statistics_out.sav"
set :statisticsexportnode.field_names = Names
set :statisticsexportnode.launch_application = True
set :statisticsexportnode.generate_import = True
```

Eigenschaften von statisticsexportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	*string*	
launch_application	*Flag*	

Eigenschaften von statisticsexportnode	Datentyp	Eigenschaftsbeschreibung
export_names	NamesAndLabels NamesAsLabels	Dient der Zuordnung von Feldnamen von SPSS Modeler zu SPSS Statistics- oder SAS-Variablennamen nach dem Export.
generate_import	*Flag*	

Kapitel
22

Superknoten-Eigenschaften

In den folgenden Tabellen werden die für Superknoten spezifischen Eigenschaften beschrieben. Beachten Sie, dass allgemeine Knoteneigenschaften auch für Superknoten gelten.

Tabelle 22-1
source_supernode

Eigenschaftsname	Eigenschaftstyp/Liste der Werte	Eigenschaftsbeschreibung
parameters	*beliebig*	Verwenden Sie diese Eigenschaft zum Erstellen und Zugreifen auf Parameter, die in der Parametertabelle eines Superknotens angegeben sind. Details finden Sie weiter unten.

Tabelle 22-2
process_supernode

Eigenschaftsname	Eigenschaftstyp/Liste der Werte	Eigenschaftsbeschreibung
parameters	*beliebig*	Verwenden Sie diese Eigenschaft zum Erstellen und Zugreifen auf Parameter, die in der Parametertabelle eines Superknotens angegeben sind. Details finden Sie weiter unten.

Tabelle 22-3
terminal_supernode

Eigenschaftsname	Eigenschaftstyp/Liste der Werte	Eigenschaftsbeschreibung
parameters	*beliebig*	Verwenden Sie diese Eigenschaft zum Erstellen und Zugreifen auf Parameter, die in der Parametertabelle eines Superknotens angegeben sind. Details finden Sie weiter unten.
execute_method	Script Normal	
script	*String*	

Superknoten-Parameter

Mithilfe von Skripts können Sie SuperNode-Parameter mit allgemeinem Format erstellen bzw. festlegen:

```
set mySuperNode.parameters.minvalue = 30
```

Alternativ können Sie den Typ des Superknotens zusätzlich zum Namen (oder statt des Namens) angeben:

```
set :process_supernode.parameters.minvalue = 30

set mySuperNode:process_supernode.parameters.minvalue = 30
```

Außerdem können Sie den Parameterwert mit einem CLEM-Ausdruck festlegen:

```
set :process_supernode.parameters.minvalue = "<Ausdruck>"
```

Festlegen von Eigenschaften für verkapselte Knoten

Sie können Eigenschaften für einzelne, in einem Superknoten verkapselte Knoten festlegen, indem Sie einen Superknoten-Parameter erstellen, der mit dem Literalnamen des Knotens und der festzulegenden Eigenschaft übereinstimmt. Nehmen Sie beispielsweise an, dass Sie einen Quellen-Superknoten mit einem verkapselten Knoten vom Typ “Variable Datei” zum Einlesen der Daten haben. Sie können den Namen der zu lesenden Datei (mithilfe der Eigenschaft full_filename angegeben) wie folgt weitergeben:

```
set :source_supernode.parameters.':variablefilenode.full_filename' = "c:/eigene-daten.txt"
```

Dadurch wird ein Superknoten-Parameter mit der Bezeichnung :variablefilenode.full_filename und dem Wert c:/eigene-daten.txt erstellt. Angenommen, ein Knoten des angegebenen Typs ist im Superknoten vorhanden; in diesem Fall wird der Wert für die benannte Eigenschaft entsprechend festgelegt. Beachten Sie, dass dies im Stream-Skript geschieht – also dem Skript für den Stream, der den Superknoten *einschließt* – und nicht im Superknoten-Skript. Vergessen sie nicht, den Parameternamen in einfache Anführungsstriche zu setzen.

Dieser Ansatz kann mit jedem verkapselten Knoten verwendet werden, solange sich daraus eine gültige Knoten- und Eigenschaftsreferenz ergibt. Um beispielsweise die Eigenschaft rand_pct für einen verkapselten Stichprobenknoten festzulegen, können folgende Befehle verwendet werden:

```
set mySuperNode.parameters.':samplenode.rand_pct' = 50
```

ODER

```
set mySuperNode.parameters.'Sample.rand_pct'= 50
```

ODER

```
set mySuperNode.parameters.'Sample:samplenode.rand_pct'= 50
```

Bei der erstgenannten Referenz wird davon ausgegangen, dass nur ein einziger Stichprobenknoten im Stream vorliegt; bei der zweiten, dass nur ein Knoten mit dem Namen “Sample” vorhanden ist, unabhängig vom Knotentyp. Die dritte Referenz ist am ausführlichsten; sie gibt sowohl den Namen als auch den Typ für den Knoten an.

Für weitere Informationen siehe Thema Superknoten-Parameter in Kapitel 9 in *IBM SPSS Modeler 15 Quellen- , Prozess- und Ausgabeknoten*.

Einschränkungen bei Superknoten-Skripts. Superknoten können andere Streams nicht bearbeiten und den aktuellen Stream nicht ändern. Daher können Befehle, die sich auf Streams beziehen, wie beispielsweise `open stream`, `get stream`, `execute_script` usw., nicht in Superknoten-Skripts verwendet werden.

Anhang

A

Hinweise

Diese Informationen wurden für weltweit angebotene Produkte und Dienstleistungen erarbeitet.

IBM bietet die in diesem Dokument behandelten Produkte, Dienstleistungen oder Merkmale möglicherweise nicht in anderen Ländern an. Informationen zu den derzeit in Ihrem Land erhältlichen Produkten und Dienstleistungen erhalten Sie bei Ihrem zuständigen IBM-Mitarbeiter vor Ort. Mit etwaigen Verweisen auf Produkte, Programme oder Dienste von IBM soll nicht behauptet oder impliziert werden, dass nur das betreffende Produkt oder Programm bzw. der betreffende Dienst von IBM verwendet werden kann. Stattdessen können alle funktional gleichwertigen Produkte, Programme oder Dienste verwendet werden, die keine geistigen Eigentumsrechte von IBM verletzen. Es obliegt jedoch der Verantwortung des Benutzers, die Funktionsweise von Produkten, Programmen oder Diensten von Drittanbietern zu bewerten und zu überprüfen.

IBM verfügt möglicherweise über Patente oder hat Patentanträge gestellt, die sich auf in diesem Dokument beschriebene Inhalte beziehen. Durch die Bereitstellung dieses Dokuments werden Ihnen keinerlei Lizenzen an diesen Patenten gewährt. Lizenzanfragen können schriftlich an folgende Adresse gesendet werden:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Bei Lizenzanfragen in Bezug auf DBCS-Daten (Double-Byte Character Set) wenden Sie sich an die für geistiges Eigentum zuständige Abteilung von IBM in Ihrem Land. Schriftliche Anfragen können Sie auch an folgende Adresse senden:

Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.

Der folgende Abschnitt findet in Großbritannien und anderen Ländern keine Anwendung, in denen solche Bestimmungen nicht mit der örtlichen Gesetzgebung vereinbar sind: INTERNATIONAL BUSINESS MACHINES STELLT DIESE VERÖFFENTLICHUNG IN DER VERFÜGBAREN FORM OHNE GARANTIEN BEREIT, SEIEN ES AUSDRÜCKLICHE ODER STILLSCHWEIGENDE, EINSCHLIESSLICH JEDOCH NICHT NUR DER GARANTIEN BEZÜGLICH DER NICHT-RECHTSVERLETZUNG, DER GÜTE UND DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Manche Rechtsprechungen lassen den Ausschluss ausdrücklicher oder implizierter Garantien bei bestimmten Transaktionen nicht zu, sodass die oben genannte Ausschlussklausel möglicherweise nicht für Sie relevant ist.

Diese Informationen können technische Ungenauigkeiten oder typografische Fehler aufweisen. An den hierin enthaltenen Informationen werden regelmäßig Änderungen vorgenommen. Diese Änderungen werden in neuen Ausgaben der Veröffentlichung aufgenommen. IBM kann jederzeit und ohne vorherige Ankündigung Optimierungen und/oder Änderungen an den Produkten und/oder Programmen vornehmen, die in dieser Veröffentlichung beschrieben werden.

Jegliche Verweise auf Drittanbieter-Websites in dieser Information werden nur der Vollständigkeit halber bereitgestellt und dienen nicht als Befürwortung dieser. Das Material auf diesen Websites ist kein Bestandteil des Materials zu diesem IBM-Produkt und die Verwendung erfolgt auf eigene Gefahr.

IBM kann die von Ihnen angegebenen Informationen verwenden oder weitergeben, wie dies angemessen erscheint, ohne Ihnen gegenüber eine Verpflichtung einzugehen.

Lizenznehmer dieses Programms, die Informationen dazu benötigen, wie (i) der Austausch von Informationen zwischen unabhängig erstellten Programmen und anderen Programmen und (ii) die gegenseitige Verwendung dieser ausgetauschten Informationen ermöglicht wird, wenden sich an:

IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.

Derartige Informationen stehen ggf. in Abhängigkeit von den jeweiligen Geschäftsbedingungen sowie in einigen Fällen der Zahlung einer Gebühr zur Verfügung.

Das in diesem Dokument beschriebene lizenzierte Programm und sämtliche dafür verfügbaren lizenzierten Materialien werden von IBM gemäß dem IBM-Kundenvertrag, den Internationalen Nutzungsbedingungen für Programmpakete der IBM oder einer anderen zwischen uns getroffenen Vereinbarung bereitgestellt.

Jegliche hier enthaltene Daten zur Leistung wurden in einer überwachten Umgebung ermittelt. Aus diesem Grund können in anderen Betriebsumgebungen gewonnene Ergebnisse stark davon abweichen. Einige Messungen wurden unter Umständen auf Systemen im Entwicklungsstadium durchgeführt und es kann nicht garantiert werden, dass diese Messungen auf allgemein verfügbaren Systemen zum gleichen Ergebnis führen. Darüber hinaus wurden einige Messungen unter Umständen durch Extrapolation bestimmt. Die tatsächlichen Ergebnisse können hiervon abweichen. Die Benutzer dieses Dokuments sollten die entsprechenden Daten für die jeweils vorliegende Umgebung prüfen.

Informationen zu Produkten von Drittanbietern wurden von den Anbietern des jeweiligen Produkts, aus deren veröffentlichten Ankündigungen oder anderen, öffentlich verfügbaren Quellen bezogen. IBM hat diese Produkte nicht getestet und kann die Genauigkeit bezüglich Leistung, Kompatibilität oder anderen Behauptungen nicht bestätigen, die sich auf Drittanbieter-Produkte beziehen. Fragen bezüglich der Funktionen von Drittanbieter-Produkten sollten an die Anbieter der jeweiligen Produkte gerichtet werden.

Alle Aussagen bezüglich der zukünftigen Ausrichtung von IBM oder der Absichten des Unternehmens können ohne vorherige Ankündigung geändert oder zurückgenommen werden und stellen lediglich Ziele und Vorgaben dar.

Diese Informationen enthalten Beispiele zu Daten und Berichten, die im täglichen Geschäftsbetrieb Verwendung finden. Um diese so vollständig wie möglich zu illustrieren, umfassen die Beispiele Namen von Personen, Unternehmen, Marken und Produkten. Alle diese Namen sind fiktiv und jegliche Ähnlichkeit mit Namen und Adressen realer Unternehmen ist rein zufällig.

Unter Umständen werden Fotografien und farbige Abbildungen nicht angezeigt, wenn Sie diese Informationen nicht in gedruckter Form verwenden.

Marken

IBM, das IBM-Logo, ibm.com und SPSS sind Marken der IBM Corporation und in vielen Ländern weltweit registriert. Eine aktuelle Liste der IBM-Marken finden Sie im Internet unter *http://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, anderen Ländern oder beidem.

UNIX ist eine eingetragene Marke der The Open Group in den USA und anderen Ländern.

Java und alle Java-basierten Marken sowie Logos sind Marken von Sun Microsystems, Inc. in den USA, anderen Ländern oder beidem.

Andere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Index