

# 1 Einführung

*Data you don't need is never lost.*  
Ander's first negative Principle of Computers

## 1.1 Auswertung von Massendaten

Die Menge an verfügbaren Daten verdoppelt sich in immer kürzeren Abständen. Jedes Unternehmen, jede Institution sammelt freiwillig oder aufgrund rechtlicher Bestimmungen Unmengen an Daten. Banken speichern die Transaktionsdaten; Firmen speichern nicht nur Daten über ihre Kunden, beispielsweise über deren Kaufverhalten; Wetterinstitute sammeln Wetterdaten.

Denken Sie einmal nach: Wissen Sie, wer welche Daten über Sie speichert? Wenn Sie berufstätig sind, wie viele Daten werden von Ihrem Unternehmen gesammelt?

Durch leistungsfähige Computer sind wir nun nicht nur in der Lage, diese Daten zu speichern, sondern wir können diese Datenmengen auch analysieren und somit interessante Informationen generieren.

Mit der Veröffentlichung von Datenmengen ist man in den letzten Jahren deutlich vorsichtiger geworden, daher zunächst einige Beispiele aus dem Jahr 2000 [Run00]. Die anfallenden Datenmengen sind seitdem sicher auf ein Vielfaches gestiegen.

### **Industrielle Prozessdaten**

Zur Analyse der Altpapieraufbereitung in einer Papierfabrik stehen an jeder der 8 DeInking-Zellen jeweils 54 Sensoren zur Verfügung: 3.800.000 Messwerte pro Tag.

### **Umsatzdaten**

WalMart führte eine Warenkorb-Analyse durch, 20 Millionen Transaktionen pro Tag erforderten eine Datenbank mit einer Kapazität von 24 Terabyte.

### **Genom-Daten**

In vielen Genom-Projekten wird versucht, aus den Genomen Informationen zu extrahieren. Das menschliche Genom enthält 60.000–80.000 Gene, das sind etwa 3 Milliarden DNA-Basen.

### **Bilder**

Die NASA nimmt mit ihrem *Earth Observing System* Oberflächenbilder der Erde auf: 50 GByte pro Sekunde.

## Textinformationen

Das World Wide Web enthält eine rapide wachsende Anzahl von Seiten und damit verbunden Daten und Informationen.

Die Frage, die sich natürlich sofort ergibt: Was machen wir mit den ganzen Daten, die wir täglich sammeln? Zur Auswertung von umfangreichen Daten reicht „Draufschauchen“ nicht mehr aus. Auch die Statistik stößt häufig an ihre Grenzen. Vielmehr benötigt man weitere *Datenanalyse-Techniken*. Man sucht in den Daten nach Mustern, nach Zusammenhängen, um so beispielsweise Vorhersagen für ein bestimmtes Kundenverhalten treffen zu können.

Diese Suche nach Mustern oder Zusammenhängen in den Daten ist Gegenstand des *Data Minings*. Während man im Bergbau, zum Beispiel beim Coal Mining, die Kohle sucht und abbaut, will man im Data Mining nicht die Daten „abbauen“, sondern man sucht nach den Schätzen, die in den Daten verborgen sind.

Data Mining sucht nach unbekannten Mustern und Abhängigkeiten in den gegebenen Daten. Eines dieser Suchziele ist es, Objekte in Klassen einzuteilen, die vorher bekannt oder auch unbekannt sein können.

Folgende Anwendungsbeispiele verdeutlichen die praktische Relevanz:

- Kredit oder kein Kredit?  
Aus alten Kreditdaten werden Regeln als Entscheidungshilfe für die Bewertung der *Kreditwürdigkeit* eines Kunden abgeleitet.
- Typen von Reisenden:  
Man generiert Muster von typischen *Reisenden*, um auf den jeweiligen Kundentypen zugeschnittene Angebote zusammenzustellen.
- Windeln und Bier:  
Es wird das *Kaufverhalten* von Kunden analysiert, um logische Abhängigkeiten zwischen Produkten zu finden. Wer Windeln kauft, nimmt häufig auch Bier.
- Gen-Analyse:  
Es werden die Gene von *Diabetes-Kranken* mit dem Ziel untersucht, die für die Krankheit vermutlich verantwortlichen oder mitverantwortlichen Gene zu identifizieren.

Data Mining befasst sich mit der Analyse von Massendaten. Das Ziel des Data Minings ist es, aus Massendaten – wie beispielsweise Kunden- oder Unternehmensdaten, Unternehmenskennzahlen und Prozessdaten – nützliches Wissen zu extrahieren. Gelingt dies, so kann daraus ein entscheidender Wettbewerbsvorteil für das Unternehmen im Markt entstehen. Data Mining lässt sich somit in die Gebiete *Wissensmanagement* – es wird Wissen aus Daten extrahiert – und *Business Intelligence* – es werden die verschiedensten Daten aus unterschiedlichen Bereichen analysiert – einordnen.

## 1.2 Data Mining und Business Intelligence

Schaut man sich das Lexikon der Wirtschaftsinformatik an [KBG<sup>+</sup>12] und hier den Beitrag zum Data Mining [Cha12], so stellt man fest, dass das Data Mining als eine *Herangehensweise analytischer Informationssysteme, die wiederum dem Business Intelligence untergeordnet sind*, eingeordnet ist.

Die Begriffshierarchie aus dem Lexikon der Wirtschaftsinformatik:



Wir schließen uns dieser Sichtweise an und sehen in Data Mining eine Sammlung von Techniken, Methoden und Algorithmen für die Analyse von Daten, die somit auch Grundtechniken für neuere und komplexere Ansätze, wie das *Business Intelligence* oder auch *Big Data* darstellen.

*Business Intelligence* (BI) ist ein relativ neuer Begriff. Ausgangspunkt ist die Beobachtung, dass in Zeiten der Globalisierung und des Internets ein effektiver und effizienter Umgang mit dem in einem Unternehmen verfügbaren Wissen nicht nur ein Wettbewerbsvorteil, sondern für das Überleben wichtig ist.

Unter Business Intelligence werden heute Techniken und Architekturen für eine effiziente Verwaltung des Unternehmenswissens zusammengefasst, natürlich einschließlich verschiedener Auswertungsmöglichkeiten. Die Aufgaben von Business Intelligence sind somit:

- Wissensgewinnung,
- Wissensverwaltung und
- Wissensverarbeitung.

Business Intelligence hat – aus Informatik-Sicht – viele Querbezüge zum Informations- und Wissensmanagement, zu Datenbanken und Data Warehouses, zur Künstlichen Intelligenz, sowie natürlich auch zum Data Mining (einschließlich OLAP – Online Analytical Processing, Statistik).

Eine allgemein akzeptierte Definition des Begriffs *Business Intelligence* gibt es bis heute nicht. Hinweise zur Entstehungsgeschichte des Begriffes kann man wieder dem Lexikon der Wirtschaftsinformatik entnehmen [KBG<sup>+</sup>12]. Man findet diese in [Hum12] unter dem entsprechenden Stichwort.

Unter Business Intelligence im engeren Sinn versteht man die Kernapplikationen, die eine Entscheidungsfindung direkt unterstützen. Hierzu zählt man beispielsweise das

Online Analytical Processing (OLAP), die Management Information Systems (MIS) sowie Executive Information Systems (EIS).

Ein etwas weiterer BI-Begriff stellt die Analysen in den Vordergrund. Folglich gehören hierzu alle Anwendungen, bei denen der Nutzer Analysen durchführt oder vorbereitet. Neben den oben genannten Anwendungen zählt man nun auch beispielsweise das Data Mining, das Reporting sowie das analytische Customer Relationship Management dazu.

Business Intelligence im weiten Verständnis umfasst schließlich alle Anwendungen, die im Entscheidungsprozess benutzt werden, also beispielsweise auch Präsentationssysteme sowie die Datenspeicherung und -verwaltung.

Schwerpunkt dieses Buchs sind die Techniken zur Wissensextraktion mittels Data Mining. Wir betrachten folglich nur einen kleinen Ausschnitt aus dem BI-Spektrum.

Für weiterführende Informationen sei auf [TSDK11] verwiesen.

Der Zusammenhang zwischen Data Mining und *Data Warehouses* ist offensichtlich. Data Warehouses haben den Anspruch, integrierte Daten für die Unterstützung von Managemententscheidungen bereitzuhalten, und sollten folgende Eigenschaften aufweisen (vgl. [Pet05, S. 40 ff.]):

- Es sollte sich am Nutzer, dem Entscheidungsträger oder Manager orientieren und so insbesondere den Informationsbedarf des Managements bedienen.
- Es umfasst alle entscheidungsrelevanten Daten in einer konsistenten Form.
- Ein Data Warehouse ist nur die „Sammelstelle“ für Daten aus externen Quellen. Eine Aktualisierung der Daten erfolgt normalerweise nur in fest definierten Abständen. Der Zugriff auf die Daten erfolgt dann im Data Warehouse nur noch lesend.
- Die Daten müssen zeitabhängig verwaltet werden, so dass Trends erkannt werden können.
- Die Daten werden nicht 1:1 aus den Quellen übernommen, sondern bereits kumuliert oder gefiltert. In einem Data Warehouse ist somit Redundanz möglich.

Um diesen Anforderungen gerecht zu werden, sind – neben Komponenten zur effizienten und konsistenten Datenhaltung und für schnelle, flexible Zugriffe auf die Daten – natürlich auch Werkzeuge zur Datenanalyse notwendig. Deshalb verfügen viele Data-Warehouse-Systeme über Komponenten zur Datenanalyse oder zumindest über Schnittstellen zu externen Werkzeugen.

## 1.3 Ablauf einer Datenanalyse

Data Mining ist wie vorher beschrieben eingebettet in die analytischen Informationssysteme und kann allein oder integriert in Business Intelligence oder als Baustein eines Data Warehouses betrieben werden.

Wie läuft nun ein Data-Mining-Prozess selbst ab? Folgende Phasen können unterschieden werden:

**Selektion** – Auswahl der geeigneten Datenmengen:

Zunächst werden die verfügbaren Daten gesichtet und in eine Datenbank, meist sogar in *eine* Datentabelle übertragen.

**Datenvorverarbeitung** – Behandlung fehlender oder problembehafteter Daten:

In dieser Phase werden die Daten bereinigt. Fehler müssen beseitigt, fehlende oder widersprüchliche Werte korrigiert werden.

**Transformation** – Umwandlung in adäquate Datenformate:

Häufig ist in Abhängigkeit vom jeweils verwendeten Verfahren eine Transformation der Daten erforderlich, beispielsweise die Gruppierung von metrischen Werten in Intervalle.

**Data Mining** – Suche nach Mustern:

Hier geschieht das eigentliche Data Mining, die Entwicklung eines *Modells* wie die Erstellung eines Entscheidungsbaums.

**Interpretation und Evaluation** – Interpretation der Ergebnisse und Auswertung:

In der abschließenden Phase müssen die gefundenen Resultate geprüft werden. Sind sie neu, sind sie hilfreich?

In Abbildung 1.1 ist der Ablauf mit den hier vorgestellten Phasen des Data Minings dargestellt.

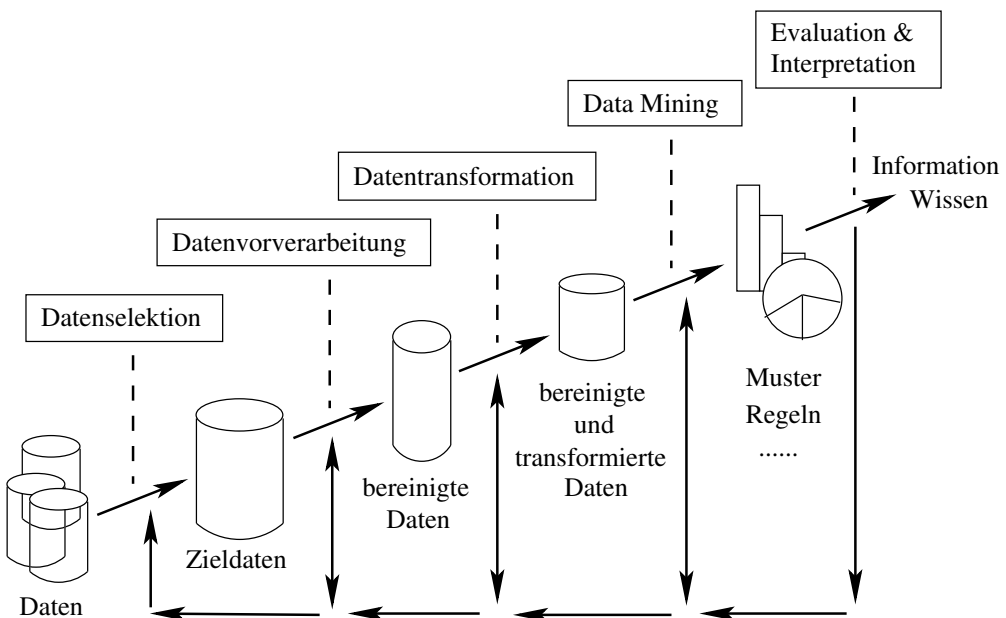


Abb. 1.1: Ablauf eines Data-Mining-Prozesses [FPSS96]

# Das CRISP-Data-Mining-Modell

- NCR Corporation,
- Daimler AG,
- SPSS,
- Teradata und
- OHRA.

The diagram illustrates the CRISP-DM (Cross-Industry Standard Process for Data Mining) process, which is a cyclical and iterative approach to data mining. The process is represented by a large circle with a thick arrow indicating a clockwise flow. The stages of the process are shown as boxes arranged in a circle, connected by arrows:

- Business Understanding**: The first stage, where the business problem is identified and translated into a data mining problem.
- Data Understanding**: The second stage, where the data is explored and understood.
- Data Preparation**: The third stage, where the data is cleaned, integrated, and transformed into a suitable format for modeling.
- Modelling**: The fourth stage, where various data mining models are built and evaluated.
- Evaluation**: The fifth stage, where the results of the modeling process are evaluated against the business objectives.

Arrows indicate the flow between these stages, showing that the process is iterative and can return to previous stages as needed. A central icon representing data (three cylinders) is labeled "Data".

Das CRISP-Modell geht von einem Lebenszyklus in 6 Etappen aus (vgl. Abbildung 1.2):

### 1. Verstehen der Aufgabe

Ohne ein grundsätzliches Verständnis des Fachgebiets, in dem eine Analyse stattfinden soll, ist ein gutes Resultat selten zu erzielen. In dieser Phase steht das allgemeine Verständnis der Aufgabe im Vordergrund. Es müssen die Ziele festgelegt werden: Ausgehend vom Ist-Zustand wird bestimmt, welche Ergebnisse im Rahmen des Data-Mining-Projekts erreicht werden sollen. Wie ist die Ausgangssituation? Welche Ressourcen stehen zur Verfügung?

Weiterhin sind Erfolgskriterien zu definieren.

Es müssen ebenso die Risiken betrachtet und möglichst quantifiziert werden. Risiken können finanzieller Natur sein. Beispielsweise kann sich das Beschaffen geeigneter Daten als aufwändiger als geplant herausstellen. Ebenso kann es passieren, dass bestimmte Daten aus rechtlichen oder firmenpolitischen Gründen nicht für das Projekt verfügbar sind.

Natürlich müssen die Kosten geplant und der erwartete Nutzen abgeschätzt werden.

Eine Datenanalyse, die noch nicht in die betrieblichen Prozesse integriert ist, besitzt immer Projektcharakter. Ein entsprechendes Projektmanagement ist erforderlich, insbesondere da hier Daten-Analysten, in der Regel Informatiker, mit den Anwendern aus anderen Fachgebieten zusammenarbeiten.

Diese Phase schließt mit einem Projektplan ab.

### 2. Verständnis der Daten

Die zweite Phase befasst sich mit den verfügbaren Daten. In Abhängigkeit vom jeweiligen Ziel der Analyse wird definiert, welche Daten benötigt werden. Parallel dazu wird untersucht, welche Daten verfügbar sind. Für ein erfolgreiches Projekt ist es erforderlich, die Daten und deren Bedeutung genau zu verstehen. Ein Data-Mining-Projekt, in dem man die verfügbaren Daten einfach nutzt, ohne ihre Semantik zu kennen, wird nicht erfolgreich sein.

Die Daten sind also nicht nur zu sammeln, sondern sie sind auch zu beschreiben. Die Daten werden untersucht, nicht nur um sie zu verstehen, sondern auch um die Qualität der zur Verfügung stehenden Daten zu bestimmen. Erste statistische Untersuchungen, wie die Bestimmung statistischer Maßzahlen – beispielsweise Minima, Maxima, Mittelwerte sowie Korrelationskoeffizienten – geben Auskunft über die Daten.

### 3. Datenvorbereitung

Nach dem Verstehen der Aufgabe, der betrieblichen Hintergründe und dem Verstehen der Daten gilt es nun, den eigentlichen Data-Mining-Schritt vorzubereiten. Zunächst werden die aus der Sicht der Aufgabe relevanten Daten selektiert und in eine konsistente Datentabelle überführt. Die Daten müssen gesäubert werden, fehlerhafte und inkonsistente Daten werden korrigiert. Es ist zu überlegen, wie man mit fehlenden Daten umgeht. Gegebenenfalls werden neue Attribute eingeführt oder auch Attribute zusammengeführt. Die Daten werden geeignet transformiert, damit sie durch die Data-Mining-Verfahren verarbeitet werden können. Die Datenvorverarbeitung kann über Erfolg oder Misserfolg einer Datenanalyse mitentscheiden. Deshalb stellen wir die Möglichkeiten der Datenvorverarbeitung in einem separaten Kapitel (Kapitel 8) vor.

#### 4. Data Mining (Modellbildung)

In dieser Phase geschieht die eigentliche Datenanalyse: Es wird ein Modell erstellt, welches beschreibt, wie die Daten einzuordnen oder zu behandeln sind. Modelle können Entscheidungsbäume, verschiedene Formen von Regeln oder Beschreibungen von Clustern sein.

In Abhängigkeit von der jeweiligen Aufgabe wird ein adäquates Verfahren ausgewählt. Die durchzuführenden Experimente werden konzipiert und konfiguriert, Parameter für das Verfahren werden gesetzt. Dann werden die Experimente durchgeführt und gegebenenfalls mit modifizierten Parametern wiederholt, um das Modell zu verfeinern und zu verbessern.

#### 5. Evaluation

Die erzielten Modelle und Resultate müssen geprüft und bewertet werden. Die Ergebnisse werden an den in Phase 1 festgelegten Erfolgskriterien gemessen: Wird zum Beispiel der erwartete wirtschaftliche Nutzen durch die Ergebnisse erzielt? Eine Fehleranalyse kann Möglichkeiten für neue Experimente aufzeigen, und es wird zur Phase 3 – Datenvorbereitung – oder direkt zur Data-Mining-Phase zurückgekehrt.

#### 6. Einsatz im und Konsequenzen für das Unternehmen

In der letzten Phase gilt es, den Einsatz der erzielten Resultate vorzubereiten. Dies ist eine kritische Phase für viele Data-Mining-Projekte. Ohne ein gut geplantes Monitoring und eine ausreichende Motivation und Unterstützung kann die erfolgreiche Umsetzung scheitern.

In der Regel ist ein Data-Mining-Projekt nur *eine* Entwicklungsphase, das Ergebnis ist dann in den Regelbetrieb zu übernehmen und in die laufenden Prozesse zu integrieren. Übliche Forderungen an einen Projektabschluss gelten natürlich auch für Data-Mining-Projekte.

Das CRISP-Modell unterscheidet sich von dem in Abbildung 1.1 auf Seite 5 dargestellten Ablauf nach Fayyad. Die Punkte 1–2 und 6 des CRISP-Modells sind im Fayyad-Modell nicht explizit aufgeführt. Der Schritt 3 des CRISP-Modells enthält die ersten drei Phasen des Fayyad-Modells. Das Fayyad-Modell konzentriert sich auf die eigentliche Datenbereitstellung und die Datenanalyse, während das CRISP-Modell die Sicht der Industrie auf Data-Mining-Projekte widerspiegelt.

Wir werden uns am in Abbildung 1.1 dargestellten Fayyad-Modell orientieren, da die Phasen 1, 2 und 6 des CRISP-Modells sehr stark projektabhängig sind und folglich nicht Bestandteil einer Einführung in das Data Mining sein können.

Wir werden zwischen den Begriffen *Data Mining* und *Knowledge Discovery in Databases (KDD)* nicht unterscheiden und diese synonym verwenden, bevorzugen jedoch den Begriff *Data Mining*.

Es gibt weitere Modelle für den Ablauf eines Data-Mining-Projekts. Das SEMMA-Vorgehensmodell wird von der Firma SAS Institute Inc. in Zusammenhang mit ihrem Produkt Enterprise Miner verwendet, siehe [SAS13]. SEMMA steht für *Sample, Explore, Modify, Model and Assess*. Ein SEMMA-Prozess besteht aus den folgenden Schritten:

1. Die für die Analyse relevanten Daten werden gesammelt (Sample).



2. Die Daten werden – vor der eigentlichen Data-Mining-Modellbildung – untersucht. Das Ziel ist, die Datenqualität zu prüfen sowie ein Datenverständnis zu erreichen. Auch erste Visualisierungen werden vorgenommen (Explore).
3. Die Daten werden modifiziert, um die Datenqualität zu verbessern. Sie werden in ein für das gewählte Verfahren adäquate Form transformiert (Modify).
4. Nun erfolgt die eigentliche Analyse und Modellbildung (Model).
5. Die Resultate werden evaluiert (Assess).

Im Folgenden wenden wir uns nun detaillierter den Teilphasen des Fayyad-Modells (Abbildung 1.1 auf Seite 5) zu, da wir uns an diesem Vorgehensmodell orientieren.

## Datenselektion

In der ersten Phase des KDD-Prozesses sind die Daten, die für die vom Anwender angeforderte Analyse benötigt werden oder für eine Analyse geeignet erscheinen, zu bestimmen und aus den gegebenen Datenquellen zu exportieren. Neben dem Basisdatenbestand können auch externe Daten für die Analyse herangezogen werden. So bieten beispielsweise Adressbroker Informationen an, mit denen potentielle Kunden oder Interessenten besser erkannt werden können. In der Phase der Datenselektion wird geprüft, welche Daten nötig und verfügbar sind, um das gesetzte Ziel zu erreichen.

Können die selektierten Daten aufgrund technischer oder rechtlicher Restriktionen nicht in einen Zieldatenbestand überführt werden, ist die Datenselektion entsprechend zu überdenken und erneut durchzuführen. Technische Restriktionen, welche die Überführung in einen Zieldatenbestand verhindern, sind zum Beispiel Kapazitäts- und Datentyp-Beschränkungen des Zielsystems oder fehlende Zugriffsrechte des Anwenders. Eine Möglichkeit, diese Probleme – zumindest zum Teil – zu umgehen, ist die Beschränkung der Auswahl auf eine repräsentative Teildatenmenge des Gesamtdatenbestands.

## Datenvorverarbeitung

Da die Zieldaten aus den Datenquellen lediglich extrahiert werden, ist im Rahmen der Datenvorverarbeitung die Qualität des Zieldatenbestands zu untersuchen und – sofern nötig – durch den Einsatz geeigneter Verfahren zu verbessern. Aufgrund technischer oder menschlicher Fehler können die Daten operativer Systeme *fehlerhafte Elemente* enthalten. Man rechnet damit, dass bis zu 5% der Felder eines realen Datenbestands falsche Angaben aufweisen. Die Kenntnis der Schwächen der Analysedaten ist für die Qualität der Untersuchungsergebnisse wichtig. Die Anwender der Analysewerkzeuge müssen auf die Zuverlässigkeit und Korrektheit der Daten vertrauen können. Fehlerhafte Daten verfälschen möglicherweise die Resultate, ohne dass der Anwender von diesen Mängeln Kenntnis erlangt.

Fehlende Daten verhindern eventuell die Berechnung von Kennzahlen wie den Umsatz einer Firma. Die zunehmende Durchführung (teil-)automatisierter Datenanalysen hat eine erhöhte Anfälligkeit gegenüber Datenmängeln zur Folge, der durch geeignete Mechanismen zur Erkennung und Beseitigung solcher Schwächen zu begegnen ist. Eine

häufige, leicht zu identifizierende Fehlerart besteht in *fehlenden Werten*. Zur Behandlung von fehlenden Werten stehen unterschiedliche Techniken zur Verfügung, die im Abschnitt 8.2.2 diskutiert werden.

Eine weitere potentielle Fehlerart wird durch *Ausreißer* hervorgerufen. Dabei handelt es sich um Wertausprägungen, die stark vom Niveau der übrigen Werte abweichen. Bei diesen Ausprägungen kann es sich um korrekt erfasste Daten handeln, die damit Eingang in die Analyse finden oder aber um falsche Angaben, die nicht berücksichtigt werden dürfen und daher aus dem Datenbestand zu löschen sind. Die Erkenntnisse, die der Benutzer eines Data-Mining-Systems in dieser Phase über den Datenbestand gewinnt, können Hinweise auf die Verbesserung der Datenqualität der operativen Systeme geben.

## Datentransformation

Die im Unternehmen verfügbaren Rohdatenbestände erweisen sich häufig in ihrer Ursprungsform als nicht für Data-Mining-Analysen geeignet. In der Phase der Datentransformation wird der analyserelevante Zieldatenbestand in ein Datenbankschema transformiert, das von dem verwendeten Data-Mining-System verarbeitet werden kann. Dabei können neue Attribute oder Datensätze generiert beziehungsweise vorhandene Attribute transformiert werden. Dieser Schritt ist nötig, da Analyseverfahren spezifische Anforderungen an die Datenstruktur der Eingangsdaten stellen. Ziel der Transformation ist insbesondere die Gewährleistung invarianter Datendarstellungsformen (beispielsweise durch Übersetzung textueller Informationen in eindeutige Schlüssel oder Codierungen) sowie die Einschränkung von Wertebereichen zur Verringerung der Anzahl zu betrachtender Ausprägungen (Dimensionsreduktion). Letzteres kann durch Verallgemeinerung von Attributwerten auf eine höhere Aggregationsstufe, zum Beispiel durch Nutzung von Taxonomien oder durch Bildung von Wertintervallen geschehen, wodurch sich die Granularität der Daten ändert.

## Data Mining

Liegen geeignete Datenbestände in akzeptabler Qualität vor, können die Analysen durchgeführt werden. In dieser Phase erfolgt die Verfahrensauswahl und deren Einsatz zur Identifikation von Mustern auf der Basis des vorbereiteten Datenbestandes. In einem ersten Schritt wird zunächst entschieden, welche grundlegende Data-Mining-Aufgabe (beispielsweise Klassifizierung oder Cluster-Bildung) vorliegt. Daran schließt sich die Auswahl eines geeigneten Data-Mining-Verfahrens an. Nach der Auswahl eines für die konkrete Problemstellung geeigneten Verfahrens wird dieses konfiguriert. Diese Parametrisierung bezieht sich auf die Vorgabe bestimmter methodenspezifischer Werte, wie zum Beispiel die Festlegung minimaler relativer Häufigkeiten für einen Interessantheitsfilter, die Auswahl der bei der Musterbildung oder -beschreibung zu berücksichtigenden Attribute oder die Einstellung von Gewichtungsfaktoren für einzelne Eingabevariablen. Wenn eine zufriedenstellende Konfiguration gefunden wurde, kann mit der Suche nach interessanten Mustern in den Daten begonnen werden. Die Analyse-Verfahren erzeugen ein Modell, welches dann als Grundlage für die Bewertung dieser oder anderer Daten dient.

## Evaluation und Interpretation

In dieser Phase des KDD-Prozesses werden die entdeckten Muster und Beziehungen bewertet und interpretiert. Diese Muster sollen den Anforderungen der *Gültigkeit*, *Neuartigkeit*, *Nützlichkeit* und *Verständlichkeit* genügen, um neues Wissen zu repräsentieren und einer Interpretation zugänglich zu sein. Letztere ist Voraussetzung für die Umsetzung der gewonnenen Erkenntnisse im Rahmen konkreter Handlungsmaßnahmen. Bei weitem nicht alle der aufgedeckten Muster erfüllen diese Kriterien. Die Analyseverfahren fördern häufig viele Regelmäßigkeiten zutage, die irrelevant, trivial, bedeutungslos oder bereits bekannt waren, aus denen dem Unternehmen folglich kein Nutzen erwachsen kann, oder die nicht nachvollziehbar sind. Die Bewertung von Mustern kann anhand des Kriteriums der Interessantheit vollzogen werden. Folgende Dimensionen der *Interessantheit* sind sinnvoll:

- Die *Validität* (Gültigkeit) eines Musters ist ein objektives Maß dafür, mit welcher Sicherheit das gefundene Modell (beispielsweise ein Muster oder eine Assoziationsregel) auch in Bezug auf neue Daten gültig ist.
- Das Kriterium der *Neuartigkeit* erfasst, inwieweit ein Muster das bisherige Wissen ergänzt oder im Widerspruch zu diesem steht.
- Das Kriterium der *Nützlichkeit* eines Musters erfasst den praktischen Nutzen für den Anwender.
- Die *Verständlichkeit* misst, wie gut eine Aussage von einem Anwender verstanden werden kann.

Die korrekte Interpretation von Data-Mining-Ergebnissen erfordert ein hohes Maß an Domänenkenntnissen. Die Interpretation dient dazu, das Domänenwissen des Anwenders effektiv zu verändern. Im Idealfall wird ein Team von Experten aus unterschiedlichen Bereichen gebildet, um sicherzustellen, dass die Bewertung korrekt ist und die gewonnenen Informationen bestmöglich genutzt werden können. Die Interpretationsphase lässt sich durch geeignete Präsentationwerkzeuge sowie durch die Verfügbarkeit zusätzlicher Informationen über die Anwendungsdomäne unterstützen. Typischerweise erfolgt in dieser Phase ein Rücksprung in eine der vorherigen Phasen. So ist meist eine Anpassung der Parameter oder die Auswahl einer anderen Data-Mining-Technik nötig. Es kann auch erforderlich sein, zur Datenselektionsphase zurückzukehren, wenn festgestellt wird, dass sich die gewünschten Ergebnisse nicht mit der benutzten Datenbasis erreichen lassen.

## 1.4 Interdisziplinarität

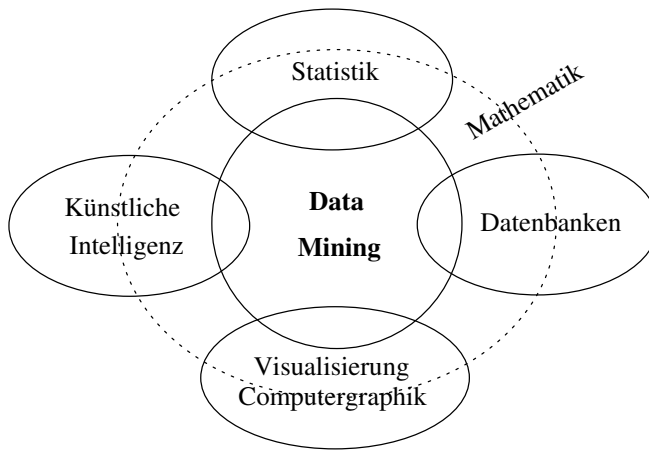
Data Mining ist keine abgeschlossene Nischentechnik, die es erst seit einigen Jahren gibt. Im Gegenteil: Data Mining nutzt bewährte Techniken aus vielen Forschungsgebieten und fügt diesen neue Ansätze hinzu. Letztendlich basieren alle Analyse-Verfahren des Data Minings auf der Mathematik. Insbesondere die Statistik steuert eine Reihe eigener Ansätze für die Datenanalyse bei, wird aber auch für die Datenvorverarbeitung

eingesetzt. Statistik ist zudem Grundlage einiger Verfahren, wie zum Beispiel *Naive Bayes*.

Erst das Gebiet der Datenbanken ermöglicht die Verwaltung großer Datenmengen, und dies wird durch den synonymen Begriff für das Data Mining sogar explizit deutlich: *KDD – Knowledge Discovery in Databases*.

Die Künstliche Intelligenz als die Wissenschaft der Wissensverarbeitung stellt insbesondere Techniken für die Darstellung der Analyseergebnisse bereit: die Repräsentation von Wissen als logische Formeln und insbesondere als Regeln.

Eine andere Form der Ergebnisdarstellung als Grundlage einer Nutzung oder Bewertung ist die graphische Darstellung, die Visualisierung. Computer-Graphik ist somit eine weitere Disziplin, die im engen Kontakt zum Data Mining steht. Abbildung 1.3 illustriert diese Interdisziplinarität des Data Minings.



**Abb. 1.3:** Interdisziplinarität

## Datenbanken und Data Warehouses

*Datenbanken* bilden in vielen Fällen die Grundlage des Data Minings. Häufig wird in bereits existierenden Datenbeständen nach neu zu entdeckenden Zusammenhängen oder Auffälligkeiten gesucht.

Ein *Data Warehouse* setzt sich in der Regel aus mehreren Datenbanken zusammen und enthält unter anderen auch die Daten, die zu analysieren sind. Nach Gluchowski [Glu12] sind die Merkmale eines Data Warehouse die Themenorientierung, die Vereinheitlichung, die Zeitorientierung sowie die Beständigkeit. Technisch ist die Vereinheitlichung hervorzuheben: Daten aus unterschiedlichen Quellen und mit möglicherweise verschiedenen Skalierungen oder Maßeinheiten werden korrekt zusammengeführt. Zudem werden alle Daten zeitbehaftet gespeichert, so dass Zeitreihen entstehen, die für die Auswertung genutzt werden können. Die sogenannte Beständigkeit besteht darin, dass

ein Data Warehouse beständig wächst, die mit ihrem Zeitstempel versehenen Daten werden akkumuliert.

Neben Datenbanken oder einem Data Warehouse können natürlich auch Textdateien oder WWW-Seiten Basis eines Data-Mining-Prozesses sein.

## Expertensysteme

*Expertensysteme* – besser *wissensbasierte Systeme* versuchen, einen oder mehrere qualifizierte menschliche Experten bei der Problemlösung in einem abgegrenzten Anwendungsbereich zu simulieren. Sie enthalten große Wissensmengen über ein eng begrenztes Spezialgebiet. Sie berücksichtigen auch Faustregeln, mit denen Erfahrungen aus den Teilgebieten für spezielle Probleme nutzbar gemacht werden sollen. Gelingt es, in einem Data-Mining-Prozess Wissen aus den Daten zu extrahieren, so kann dieses Wissen dann – zum Beispiel in Form von Regeln – in einem Expertensystem repräsentiert und angewendet werden.

## Maschinelles Lernen

Der Begriff *Lernen* umfasst viele komplexe Aspekte. Nicht jeder davon kann auf einem Rechner nachgebildet werden. Beim *Maschinellen Lernen* (engl. Machine Learning) versucht man, computerbasierte Lernverfahren verfügbar zu machen, so dass das Programm aus Eingabeinformationen *Wissen* generieren kann.

Bei maschinellen Lernsystemen ist – wie auch in der menschlichen Psychologie – die einfachste Lernstrategie das Auswendiglernen. Dabei wird das präsentierte Wissen einfach in einer Liste oder Datenbank abgespeichert. Eine ebenso einfache Form des Maschinellenlernens ist das unmittelbare Einprogrammieren des Wissens in den Sourcecode eines entsprechenden Programms.

Dies ist jedoch nicht das, was in der Künstlichen Intelligenz mit *Maschinellern Lernen* gemeint ist. Hier wird mehr ein Verständnis von Zusammenhängen und Hintergründen (beispielsweise das Erkennen von Mustern oder Abhängigkeiten) angestrebt, um beispielsweise Muster oder Abhängigkeiten erkennen zu können. Beim Induktiven Lernen wird unter anderem versucht, aus Beispielen zu verallgemeinern und so neues Wissen zu erzeugen.

## Statistik

Ohne *Statistik* ist Data Mining nicht denkbar: Seien es die statistischen Maßzahlen, die helfen, die Daten zu verstehen, oder die statistischen Verfahren zum Aufdecken von Zusammenhängen.

Nicht immer ist es möglich oder sinnvoll, ein maschinelles Data-Mining-Verfahren zu entwickeln und anzuwenden. Manchmal bringen auch schon statistische Lösungen einen ausreichenden Erfolg, falls beispielsweise ein Zusammenhang zwischen zwei Merkmalen durch eine Korrelationsanalyse gefunden wird.

Des Weiteren können statistische Verfahren dabei helfen zu erkennen, ob Data Mining überhaupt zu einem gewünschten Ergebnis führen kann.

## Visualisierung

Eine gute *Visualisierung* ist für den Erfolg eines Data-Mining-Projekts unerlässlich. Da Data Mining meistens zur Entscheidungsfindung oder -unterstützung eingesetzt wird und Entscheidungen nicht immer von den Personen getroffen werden, die direkt am Prozess des Data Minings beteiligt sind, müssen die Resultate des Data Minings veranschaulicht werden. Nur wenn es gelingt, gefundenes Wissen anschaulich und nachvollziehbar darzustellen, wird man Vertrauen in die Ergebnisse erzeugen und eine Akzeptanz der Resultate erreichen.

Man kann Visualisierung aber nicht nur zur Darstellung der Resultate einsetzen, sondern auch beim eigentlichen Data Mining. Häufig erkennt man durch eine geschickte Darstellung der Daten erste Zusammenhänge zwischen den Attributen. Man denke hier an Cluster-Bildung oder an besonders einflussreiche Attribute bei einer Klassifizierung.

Visualisierung stellt somit nicht nur die entwickelten Modelle graphisch dar, sondern kann auch als eigene Data-Mining-Technik in der Datenanalyse eingesetzt werden. Da die Ergebnisdarstellung maßgeblich über den Projekterfolg entscheidet, wird der Visualisierung ein eigener Abschnitt gewidmet, siehe Abschnitt 9.6.

### 1.5 Erfolgreiche Beispiele

Ähnlich wie die Verwandtschaftsbeziehungen in fast jeder Einführung in die logische Programmiersprache PROLOG zu finden sind, wird in Data-Mining-Lehrbüchern sehr oft das Wetter-Golf-Spiel-Ja-Nein-Beispiel (siehe Abschnitt A.3) eingesetzt. Auch hier in diesem Buch werden Sie immer wieder auf dieses Beispiel treffen, unter anderem in den Abschnitten 1.6.3 oder 5.2. Gibt es nun auch Beispiele aus der Realität, die zeigen, dass Data Mining erfolgreich ist?

Aus den Anfangsjahren des Data Mining sind erfolgreiche Beispiele überliefert, die mittlerweile zu Klassikern wurden:

- Die amerikanische Handelskette Wall Mart soll herausgefunden haben, dass an bestimmten Tagen Windeln besonders häufig zusammen mit Bier verkauft wurden. Obwohl dieses Beispiel von vielen zitiert wird, gibt es immer wieder Diskussionen, ob dies belegt ist oder in das Reich der Legenden gehört.<sup>1</sup>
- Die Bonitätsprüfung oder die Prüfung der Kreditwürdigkeit von Bankkunden ist eine schon „alte“ Anwendung und wird ebenso häufig angeführt [Han98].
- Die personalisierte Mailing-Aktion als Marketing-Strategie findet sich ebenso in vielen Data-Mining-Einführungen: Aufgrund vorhandener Daten wird ein Modell für die Klassifikation in die Klassen *Anschreiben* sowie *Nichtanschriften* entwickelt und so die Werbe-Information nur an potenzielle Kunden gesendet.

---

<sup>1</sup>Unter [www.kdnuggets.com/news/2000/n13/23i.html](http://www.kdnuggets.com/news/2000/n13/23i.html) wird berichtet, dass dieses Beispiel von Tom Blishok (einem Einzelhandelsberater) ca. 1992 erfunden wurde. Es soll wohl nie eine wirkliche Analyse gegeben haben.

Jede Aufzählung von Beispielen aus der realen Welt ist zum Zeitpunkt des Aufschreibens bereits veraltet. Es ist wohl besser, sich der Frage zu stellen: Wie finde ich erfolgreiche, aktuelle Beispiele? In den Lehrbüchern finden sich in der Regel die Verweise auf die gerade erwähnten Anwendungsfälle. In der Zwischenzeit ist Data Mining den Kinderschuhen entwachsen und hat Eingang in viele Anwendungsfelder – vom Finanzbereich bis zur Medizin, von der Kundenanalyse bis zum E-Learning – gefunden.

Es gibt eine Reihe von Büchern, die erfolgreiche Data-Mining-Anwendungen dokumentieren. Eine gute Auswahl an praktischen Anwendungen findet man in [Gab10]. Diese gehen von räumlichen Analysen in geographischen Systemen, über Anwendungen in der Chemie und Bioinformatik bis zu erfolgreichen Analysen in der Astronomie. In [HKMW01] wird auf eine Vielzahl von erfolgreichen Anwendungen im Marketing eingegangen. Eine typische Anwendung, die Kundensegmentierung – das Finden von Gruppen von ähnlichen Kunden – wird in diesem Buch an mehreren Beispielen vorgestellt. Die dort vorgestellten Projekte stammen aus dem Automobilbereich und der Kundenspezifischen Ansprache. Erfolgreiche Projekte aus der zweiten großen Anwendungsklasse – der Klassifikation – werden ebenso vorgestellt, beispielsweise die Bonitäts- und Kreditwürdigkeitsprüfung von Kunden. Auch Cross selling, wie es von vielen Online-Plattformen genutzt wird, ist dort mit einem erfolgreichen Projekt vertreten. Ähnliche Beispiele – aus dem Bereich E-Business und Finanzen – findet man auch in [SPM<sup>+</sup>08].

Selbst im E-Learning werden Data-Mining-Techniken auf vielfältige Art eingesetzt. In [RV06] und [RVPB11] wird eine Vielzahl von Möglichkeiten vorgestellt, wie Data Mining zur Verbesserung der Lehre eingesetzt wird. Dies geht vom Erkennen von typischen Lerner-Mustern, die eine Nutzer-angepasste Präsentation von Inhalten (sogenannte adaptive Story-Boards) ermöglicht, bis zur automatischen Erkennung von Problemen im Lernprozess.

Es ist in den letzten Jahren der Trend zu beobachten, dass über erfolgreiche Data-Mining-Anwendungen nur noch im akademischen Umfeld detailliert berichtet wird. Dies hat mehrere Gründe. Einerseits ist meistens der wissenschaftliche Neuwert nicht mehr gegeben. Andererseits sind in jeder Data-Mining-Anwendung auch viele Daten enthalten, aus denen erfolgskritisches Wissen abgeleitet wurde. Dieses geben Firmen natürlich ungern preis.

Erfolgreiche Data-Mining-Anwendungen sind für eine andere Gruppe wiederum ein gutes Marketing-Argument: Die Hersteller von Data-Mining-Software beziehungsweise die branchenübergreifenden IT-Service-Unternehmen auf dem Gebiet der Datenanalyse sind auf erfolgreiche Projekte angewiesen, um wieder neue Kunden gewinnen zu können.

Auf den Seiten der von uns in diesem Buch eingesetzten, frei verfügbaren Werkzeuge sind nur auf den Seiten der KNIME-Software einige Anregungen zu finden. Diese tragen aber eher den Charakter von Einführungsbeispielen. Kommerzielle Anbieter von Data-Mining-Software werben dagegen mit ihren Referenzen: Referenz-Kunden oder Referenz-Projekte. Die Firma *Easy.Data.Mining* aus München stellt unter der Überschrift „Data-Mining-Beispiele und Fallstudien aus der Praxis“ [Eas] ihre Projekte vor.

Die Liste der Referenzkunden, die den *Rapid Miner* einsetzen, ist lang und umfasst viele Bereiche von der Elektronik-, Luft- und Automobilbranche über Handel und Marktforschungsunternehmen bis hin zu Banken und Versicherungen, der Pharma- und Biotech-

nologiebranche oder der IT-Branche selbst. Konkrete Anwendungsbeispiele werden zwar nicht aufgeführt, die aufgezählten Branchen und Unternehmen geben aber Hinweise auf den Einsatz von Data-Mining-Lösungen.

Das System *SPSS*<sup>2</sup> wird von IBM eingesetzt, um Data-Mining-Lösungen im Bereich des sogenannten Predictive Analytics zu entwickeln. Einige spektakuläre Anwendungsfälle sind während der Entstehung dieses Buches als Video auf YouTube zu sehen (gewesen)<sup>3</sup>:

- Die Datenanalyse durch die amerikanische Polizei führt dazu, dass man vorhersagen kann, wo und wann demnächst ein Verbrechen stattfinden wird. Diese Prognose erfolgt sicherlich nur mit einer gewissen Wahrscheinlichkeit<sup>4</sup>.
- Der Zusammenhang zwischen dem Wetter und dem Keks-Verkauf in deutschen Bäckereien ist nicht nur unterhaltsam, sondern betriebswirtschaftlich relevant.

Nicht zuletzt können wir auch das Archiv des Data Mining Cups [DMC] durchsehen. Die Daten für die Aufgaben werden von Unternehmen bereitgestellt, die Aufgaben sind somit praxisrelevant.

In den Jahren 2000 und 2001 ging es um die bereits erwähnten Mailingaktionen. Lohnt es sich, einen Kunden anzuschreiben oder nicht? Beide Wettbewerbe wurden – nach Aussagen der Veranstalter – zur großen Zufriedenheit der hinter der Aufgabe stehenden Firmen abgeschlossen. Im Kapitel 10 greifen wir das Thema aus dem Jahre 2002 auf, die Mailing-Aktion eines Energieversorgers.

In mehreren Aufgaben des Data Mining Cups wird das Kundenverhalten analysiert, und es werden Vorhersagen getroffen, sei es für den Einsatz von Gutscheinen oder Rabatt-Coupons, für die Verkaufszahlen von Büchern oder das Verhalten der Kunden in einer Lotterie.

Erfolgreiche Data-Mining-Anwendungen sind in das operative Geschäft vieler Anwendungsgebiete eingebunden und werden tagtäglich genutzt. Suchen Sie selbst – beispielsweise im World Wide Web – und lassen Sie sich von interessanten Anwendungen des Data Minings überraschen.

## 1.6 Werkzeuge

Für die Lösung der in diesem Buch behandelten Aufgaben kann spezielle Data-Mining-Software eingesetzt werden.

Unabhängig davon sind Kenntnisse in der *Tabellenkalkulation* hilfreich. Mit einem Tabellenkalkulationsprogramm können Daten einer ersten Analyse unterzogen werden; es lassen sich Abhängigkeiten zwischen Attributen entdecken, oder die Ergebnisse eines Data-Mining-Modells können analysiert beziehungsweise nachgearbeitet werden.

<sup>2</sup>[www.ibm.com/de/de/](http://www.ibm.com/de/de/), Unterpunkt Lösungen, 03.09.2013

<sup>3</sup>IBM: Advertisements on YouTube:

<http://www.youtube.com/playlist?list=PLAD6EEA3C161A84F1>, 02.09.2013

<sup>4</sup>Siehe auch „Der Spiegel“ 2013/20.



Ein *Text-Editor*, der zudem die Arbeit mit Makros ermöglicht, ist ein weiteres nützliches Werkzeug in der Vor- sowie Nachbereitung einer Datenanalyse.

Nicht zuletzt sei darauf verwiesen, dass hin und wieder die eigene Entwicklung kleiner Programme für die Datenvorverarbeitung sowie die Analyseauswertung notwendig werden kann. Dazu kann eine beliebige Programmiersprache herangezogen werden. Einige Systeme, wie beispielsweise KNIME, ermöglichen den Einbau eigener kleiner Java-Programme (Java Snippets) in den Analyseprozess.

An dieser Stelle geben wir eine kurze Einführung in Data-Mining-Software, die für die Lösung der im Buch behandelten Aufgaben von uns eingesetzt werden:

- Der *Konstanz Information Miner* (KNIME) ist ein System zur Beschreibung ganzer Data-Mining-Prozesse, welcher eine Vielzahl Algorithmen für die verschiedenen Analyse-Phasen bereithält.  
Siehe <http://www.knime.org/>.
- Das *Waikato Environment for Knowledge Analysis* (WEKA) stellt eine Reihe von in Java implementierten Algorithmen bereit, die sowohl interaktiv als auch im Kommandozeilen-Modus ausgeführt werden können.  
Siehe <http://www.cs.waikato.ac.nz/ml/weka/>.  
Die WEKA-Algorithmen können in KNIME als KNIME-Erweiterung integriert werden.  
Siehe hierzu: <http://www.knime.com/downloads/extensions>.
- Speziell auf die Arbeit mit neuronalen Netzen ist der JAVANNS ausgerichtet. JAVANNS ist ein in Java implementiertes Nutzer-Interface für den *Stuttgarter Neuronale Netze Simulator* (SNNS).  
[http://www.ra.cs.uni-tuebingen.de/software/JavanNS/welcome\\_e.html](http://www.ra.cs.uni-tuebingen.de/software/JavanNS/welcome_e.html).

Es gibt viele weitere, leistungsfähige Data-Mining-Tools, beispielsweise den *Rapid Miner* (<http://rapid-i.com/>) und den *IBM SPSS Modeler*. Ebenso sind in vielen Data-Warehouse- und Datenbank-Systemen Data-Mining-Komponenten integriert. In diesem Buch beschränken wir uns aber auf die drei genannten Systeme. Diese sind zum Erlernen und zum Experimentieren sehr gut geeignet.

### 1.6.1 KNIME

KNIME ist ein Data-Mining-Tool, welches ursprünglich an der Universität Konstanz entwickelt wurde. Die Abkürzung KNIME steht für *Konstanz Information Miner*. KNIME läuft unter allen Betriebssystemen, erforderlich ist JAVA.

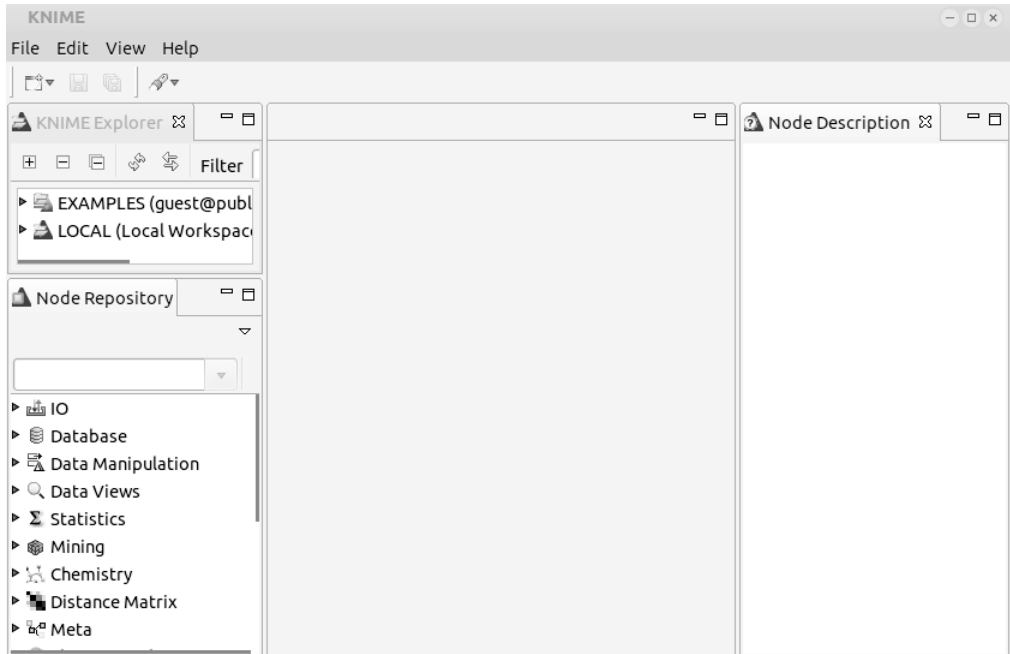
KNIME ist in verschiedenen Versionen verfügbar. Wir verwenden im Buch den *KNIME Desktop*, der auf der KNIME-Seite heruntergeladen werden kann ([www.knime.org](http://www.knime.org)).

KNIME zeichnet sich durch eine sehr einfache Drag&Drop-Bedienung sowie durch eine große Anzahl an verfügbaren Algorithmen und Methoden aus. Der Aufbau ist modular und wird ständig um neue Komponenten erweitert, die leicht intern über das Programm geladen werden können. Neben eigenen Algorithmen lässt sich die Software so um eine Vielzahl weiterer Inhalte erweitern. So sind auch nahezu alle WEKA-Verfahren für

KNIME umgesetzt worden. KNIME ist in seiner Leistungsfähigkeit durchaus vergleichbar mit einer Vielzahl von kommerziellen Data-Mining-Programmen. Der komplette Mining-Prozess vom Datenimport über Datenvorverarbeitung und Datenanalyse bis hin zur Darstellung der Ergebnisse lässt sich mit den bereitgestellten Methoden bewerkstelligen.

Für einige Beispiele ist die Integration der WEKA-Verfahren in das KNIME-System erforderlich. WEKA-Verfahren können als KNIME-Extension eingebunden werden.

Nach dem Start von KNIME öffnet sich das in Abbildung 1.4 dargestellte Fenster.



**Abb. 1.4:** KNIME – Start-Fenster

Die Oberfläche enthält folgende Komponenten:

1. Workflow-Fenster  
Das mittlere Fenster ist das Hauptfenster, in dem der Data-Mining-Prozess abgebildet wird.
2. Projektfenster  
Das Projektfenster *KNIME Explorer* dient der Verwaltung von Projekten.
3. Node Repository  
Fundament jeglicher KNIME-Anwendung ist die Sammlung der implementierten Algorithmen, aus denen der Nutzer einen Workflow zusammensetzt. Der Fundus an Algorithmen umfasst Komponenten

- für den Datenimport und -export aus Dateien oder Datenbanken,
- zur Datenvorverarbeitung und Datenmanipulation,
- für grundlegende statistische Analysen,
- für die eigentliche Datenanalyse (sowohl KNIME als auch WEKA) und
- zur Visualisierung.

#### 4. Node Description

In diesem Fenster erhält man eine detaillierte Beschreibung der KNIME-Knoten:

- Beschreibung des Algorithmus
- Benötigte Inputformatierungen der Daten
- Art des erzeugten Outputs
- Kurze Beschreibung der Konfigurationsmöglichkeiten

Nun kann man sich die entsprechenden Werkzeuge als Nodes (Knoten) aus dem Repository in das Hauptfenster ziehen und dort durch Pfeile verbinden. In Abbildung 1.5 auf der nächsten Seite ist ein Workflow dargestellt, der aus folgenden Komponenten besteht.

1. Filereader: Hier werden die Daten eingelesen.
2. Partitioning: Die Daten werden in Trainings- und Testmenge aufgeteilt (siehe Abschnitt 9.4).
3. Decision Tree Learner: Aus den Trainingsdaten wird ein Entscheidungsbaum erzeugt.
4. Decision Tree Predictor: Der erlernte Entscheidungsbaum wird auf die Testmenge angewendet.

Das Hauptfenster dient folgenden Aufgaben:

- Es ist das primäre Modellierungsfenster des Miningprozesses.
- Die Knoten können verwaltet und konfiguriert werden (Rechtsklick auf den jeweiligen Knoten).
- Die Knoten können miteinander verknüpft und so zu einem Workflow eines Data-Mining-Prozesses zusammengesetzt werden. Dabei signalisieren die Pfeile den Datenfluss.

Ist ein Workflow zusammengesetzt, muss man die Knoten konfigurieren. Dazu geht man auf den jeweiligen Knoten und aktiviert durch einen rechten Mausklick (oder durch einen Doppelklick mit der linken Maustaste) die entsprechende Auswahl. Beim *Filereader* kann man so das einzulesende File auswählen.

Wenn ein Knoten korrekt konfiguriert ist, wechselt die Ampel unter dem Knoten von rot auf gelb. Nun führt man den Knoten aus, wieder über den rechten Mausklick. Ist

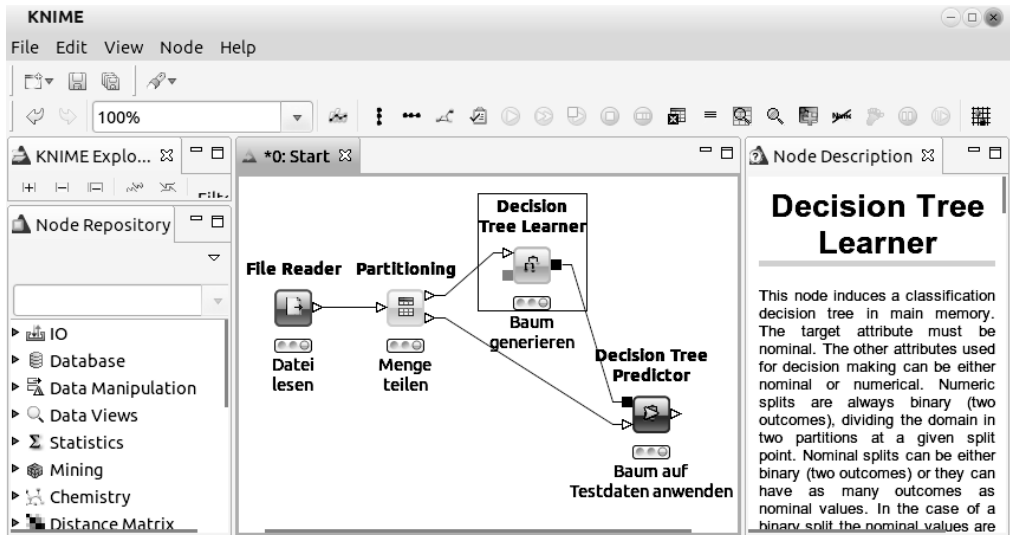


Abb. 1.5: KNIME – Workflow

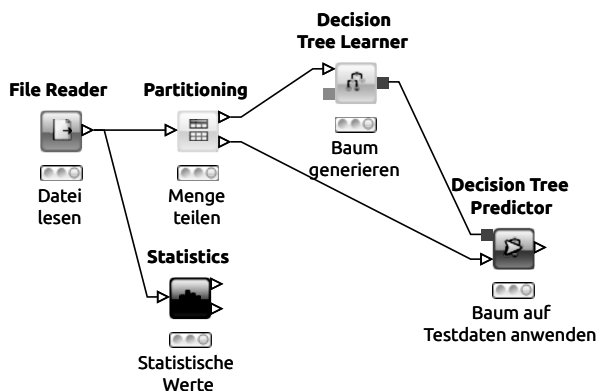


Abb. 1.6: KNIME – Wetterbeispiel

alles korrekt verlaufen, wird die Ampel grün. Dies setzt man bei den Folgeknoten fort. Zeigt sich unter dem Knoten ein Warndreieck, sollte man die hinter diesem Symbol platzierte Fehler- oder Warnmeldung lesen und darauf reagieren. Abbildung 1.6 zeigt den fertigen Workflow für das Wetter-Beispiel (siehe Abschnitt A.3).

Der in den Workflow integrierte Statistik-Knoten liefert die bereits angesprochenen statistischen Maßzahlen und fördert das Verständnis für die zu analysierenden Daten.

Betrachten wir nochmal die wichtigen Schritte im Data-Mining-Workflow.

Zunächst muss die Datei eingelesen werden. Um eine Datei einzulesen, muss in der Knotenauswahl unter *IO* im Abschnitt *Read* der *File reader* ausgewählt und auf die

Arbeitsfläche gezogen werden. KNIME unterstützt verschiedene Formate, unter anderem csv, arff, xls. Das Arff-Format ist das Standard-Format des WEKA-Systems.

KNIME enthält eine Vielzahl von Vorverarbeitungs-Techniken. Diese findet man unter *Data Manipulation*. Den Partitioner erreicht man beispielsweise unter *Row > Transform*.

Die eigentlichen Analyseverfahren sind im Ordner *Mining* enthalten. Die im obigen Workflow verwendeten Knoten *Decision-Tree-Learner* und *Decision-Tree-Predictor* sind im Unterverzeichnis *Decision Tree* platziert. Im *Decision Tree Learner* wählt man aus, welches Attribut das für die Klassifizierung verwendete Zielattribut sein soll (*Class Column*).

Die Konfiguration der Knoten erlaubt häufig viele weitere Einstellungen, auf die wir an dieser Stelle nicht eingehen.

Um den Entscheidungsbaum nun auch bezüglich seiner Trefferrate bei der Klassifikation prüfen zu können, teilt man die gegebene Beispielmenge in Trainings- und Testmenge. Dies macht der *Partitioning*-Knoten. Mit der Trainingsmenge wird ein Modell, in diesem Fall ein Entscheidungsbaum, erstellt. Diesen Entscheidungsbaum wenden wir nun auf die Testmenge an und sehen, wie gut oder schlecht dieser die Klassifikation der ja bekannten Daten vornimmt. Dazu unternimmt man einen Rechtsklick auf den *Decision Tree Predictor* und wählt *Classified Data*. Der Predictor fügt eine zusätzliche Spalte **Prediction (DecTree)** mit der vorhergesagten Klassifikation für den jeweiligen Datensatz ein (Abbildung 1.7).

Properties		Flow Variables				
Table "weather.numeric.csv" - Rows: 5				Spec - Columns: 6		
Row ID	\$ outlook	! tempe...	! humidity	\$ windy	\$ play	\$ Predic...
Row0	sunny	85	85	FALSE	no	yes
Row2	overcast	83	86	FALSE	yes	yes
Row4	rainy	68	80	FALSE	yes	yes
Row7	sunny	72	95	FALSE	no	yes
Row9	rainy	75	80	FALSE	yes	yes

**Abb. 1.7:** KNIME – Resultat

Alternativ kann man den in KNIME implementierten *Scorer* nutzen. Man erhält eine Reihe von Bewertungszahlen (Abbildung 1.9 auf der nächsten Seite), auf die wir im Kapitel 9 eingehen.

Auf zwei angenehme Möglichkeiten, die KNIME neben dem großen Vorrat an vorgefertigten Knoten bietet, wollen wir hinweisen. Zum einen ist es möglich, eigene Knoten zu entwickeln. In Abbildung 1.10 ist ein sogenannter *Java-Snippet*-Knoten dargestellt. Der Java-Snippet-Knoten berechnet die Punkte für die Aufgabe des Data-Mining-Cups 2007 [DMC]. Es geht darum vorherzusagen, ob ein Kunde einen Coupon A beziehungsweise einen Coupon B oder keinen Coupon (N) einlösen wird. Für die korrekte Vorhersage von B gibt es 6 Punkte, für die korrekte Vorhersage von A 3 Punkte, für N keinen Punkt. Eine falsche Vorhersage von A beziehungsweise B wird mit -1 bestraft. Der Java-Snippet-Knoten berechnet für jeden Datensatz, wieviele Punkte es gibt. Dazu

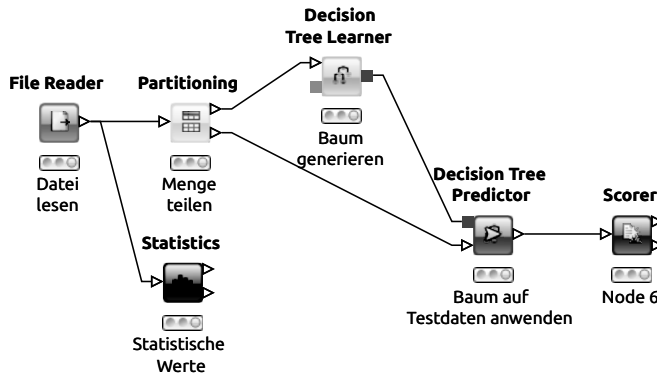


Abb. 1.8: KNIME – Scorer

Accuracy statistics - 0:6 - Scorer

File

Table "default" - Rows: 3   Spec - Columns: 10   Properties   Flow Variables

Row ID	TrueP...	FalseP...	TrueN...	False...	D Recall	D Precisi...	D Sensit...	D Specificity	D F-meas...
yes	3	2	0	0	1	0.6	1	0	0.75
no	0	0	3	2	0	?	0	1	?

Abb. 1.9: KNIME – Resultat

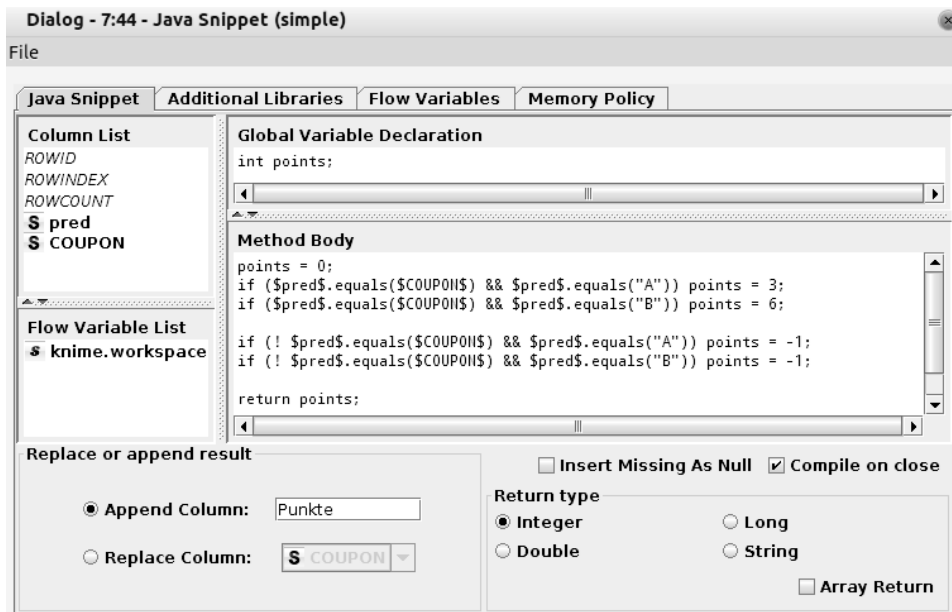


Abb. 1.10: KNIME – Java Snippet für Punktevergabe

wir zunächst die Variable `points` deklariert und auf 0 gesetzt. Sind die Werte für die Attribute `pred` (die Vorhersagespalte `Prediction` (`DecTree`) wurde umbenannt) und `COUPON` (das tatsächliche Verhalten des Kunden) gleich, so werden 3 beziehungsweise 6 Punkte vergeben, falls die Vorhersage A beziehungsweise B ist. Analog werden falsche A/B-Vorhersagen mit  $-1$  bestraft. Es wird eine neue Spalte `Punkte` angefügt. Zu jedem Datensatz gibt es also danach einen neuen Wert `Punkte`.

Im nächsten Java Snippet (Abbildung 1.11) werden nun die Punkte aufsummiert. Es entsteht eine neue Spalte `GSumme`, die zu jedem Datensatz die Summe aller Punkte bis zu diesem Datensatz berechnet. Der Eintrag beim letzten Datensatz ist dann die insgesamt erreichte Punktezahl.

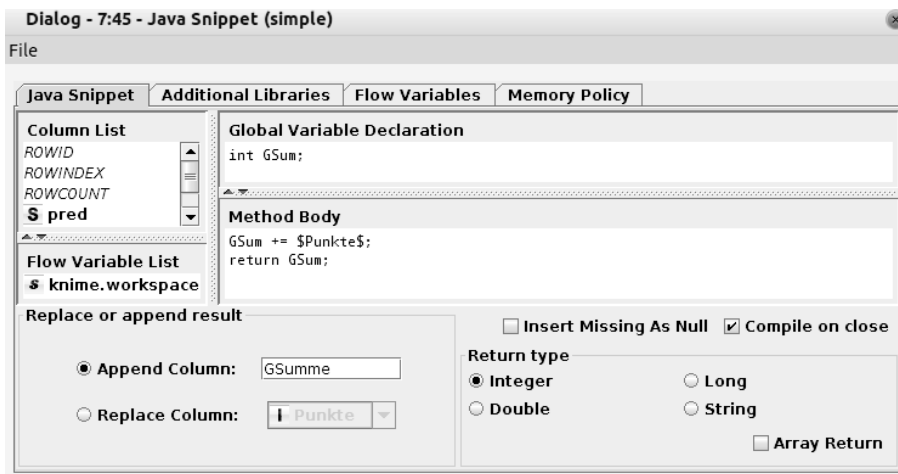


Abb. 1.11: KNIME – Java Snippet zum Summieren

Die zweite, sehr nützliche Möglichkeit, die KNIME bietet, sind sogenannte Schleifen, mit denen ein mehrfaches Durchlaufen eines Workflows definiert werden kann. Wir beginnen zunächst mit einem einfachen Workflow (Abbildung 1.12) zur Klassifikation der Daten aus dem Iris-Beispiel (siehe Abschnitt A.1 und Beispiel 6.1 auf Seite 145) mittels des Verfahrens k-Nearest Neighbours (siehe Abschnitt 5.1). Ziel ist die korrekte Vorhersage des Iris-Typs anhand der gegebenen Maße der jeweiligen Pflanze.

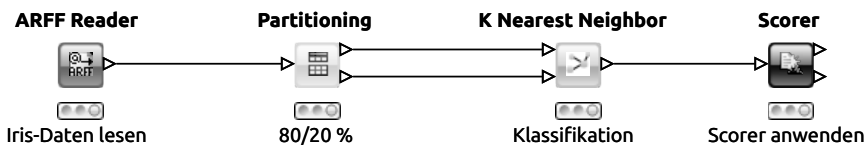


Abb. 1.12: KNIME – Loops 1

Die Daten werden im Verhältnis 80% zu 20% aufgeteilt, und zwar so, dass die Häufigkeitsverteilung der Klassenwerte in den Untermengen gleich der Verteilung in der Gesamtmenge ist (*Stratified Sampling*, vgl. Abschnitt 9.4).

Wir sagen dann die Klasse der Datensätze der 20%-Menge vorher, indem wir die  $k=3$  ähnlichsten Datensätze aus der 80%-Menge wählen und dort die häufigste Klasse bestimmen.

Nun ist die Wahl von  $k=3$  für die nächsten Nachbarn, auf deren Basis wir die Klasse vorhersagen, recht willkürlich. Lieber wäre uns, wenn wir die Berechnungen mit *unterschiedlichen*  $k$  durchführen könnten. Genau dieses Probieren mehrerer Werte für  $k$  wird nun mittels der Schleifen-Knoten realisiert. Zunächst lassen wir uns die Schleifen-Knoten im  $k$ -Nearest-Neighbour-Knoten anzeigen (rechte Maustaste, *Show Flow Variable Ports*). Dann ziehen wir den *Table Creator* und den *TableRow To Variable Loop Start* sowie den Schleifen-Ende-Knoten in unseren Workflow (Abbildung 1.13). Im *Table*

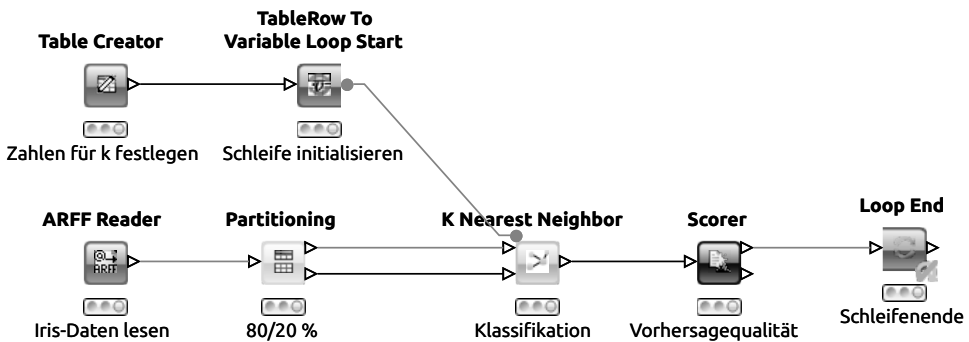


Abb. 1.13: KNIME – Loops 2

*Creator* legen wir die Varianten für  $k$  fest (Abbildung 1.14). Und im Knoten *k-Nearest*

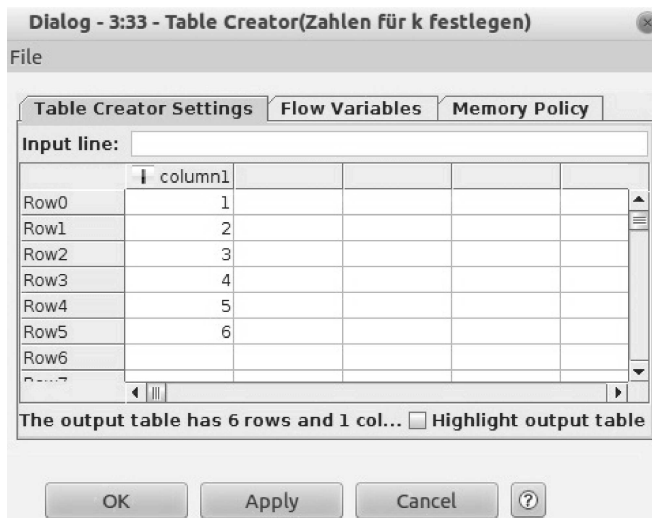
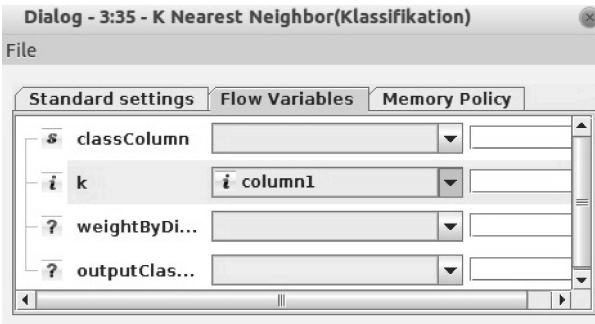


Abb. 1.14: KNIME – Loops 3



*Neighbour* wird nun festgelegt, dass die Zahlen aus dem *Table Creator* für *k* verwendet werden sollen (Abbildung 1.15).



**Abb. 1.15:** KNIME – *Loops 4*

Das Protokoll, das wir uns am Schleifenende anzeigen lassen, ist in Tabelle 1.1 dargestellt. In der Spalte *Iteration* wird dabei nicht das gewählte *k* angezeigt, sondern die Varianten-Nummer, wobei das Zählen bei 0 beginnt. Es fällt auf, das mit allen gewähl-

row ID	Iris-setosa	Iris-versicolor	Iris-virginica	Iteration
Iris-setosa#0	10	0	0	0
Iris-versicolor#0	0	10	0	0
Iris-virginica#0	0	1	9	0
Iris-setosa#1	10	0	0	1
Iris-versicolor#1	0	10	0	1
Iris-virginica#1	0	1	9	1
Iris-setosa#2	10	0	0	2
Iris-versicolor#2	0	10	0	2
Iris-virginica#2	0	1	9	2
Iris-setosa#3	10	0	0	3
Iris-versicolor#3	0	10	0	3
Iris-virginica#3	0	1	9	3
Iris-setosa#4	10	0	0	4
Iris-versicolor#4	0	10	0	4
Iris-virginica#4	0	1	9	4
Iris-setosa#5	10	0	0	5
Iris-versicolor#5	0	10	0	5
Iris-virginica#5	0	1	9	5

**Tabelle 1.1:** *Iris-Daten mit k-Nearest Neighbour (Schleife)*

ten *k* das gleiche Resultat erzielt wird. Die Vorhersage ist immer bis auf einen Datensatz korrekt.

## 1.6.2 WEKA

WEKA ist ein für nicht kommerzielle Anwendungen frei verfügbares Data-Mining-Tool, entwickelt an der University of Waikato in Neuseeland. WEKA ist eine Abkürzung für *Waikato Environment for Knowledge Analysis*. In WEKA sind viele Algorithmen implementiert, siehe hierzu [WFH11]. WEKA ist in Java implementiert und läuft somit unter allen Betriebssystemen. Man findet sowohl WEKA als auch etliche Infos unter [www.cs.waikato.ac.nz/~ml/weka/index.html](http://www.cs.waikato.ac.nz/~ml/weka/index.html). Im Folgenden wird nur ein kurzer Einstieg in den Umgang mit WEKA gegeben.

Man startet WEKA mit `java -jar weka.jar` als Kommandozeile oder mit einem Doppelklick auf die jar-Datei. Es öffnet sich das in Abbildung 1.16 dargestellte Fenster.



**Abb. 1.16:** WEKA – Start-Fenster

Wir werden mit dem Explorer arbeiten. Es kann im WEKA-Startfenster aber auch die Experimentierumgebung (Experimenter) und ein Kommandozeilenfenster geöffnet werden.

In einem neuen Fenster öffnet sich der WEKA-Explorer (Abbildung 1.17 auf der nächsten Seite), welcher eine Art Registertabelle mit den Registern *Preprocess*, *Classify*, *Cluster*, *Associate*, *Select attributes* und *Visualize* enthält.

Im Register *Preprocess* können Einstellungen zu den zu verwendenden Datensätzen vorgenommen werden. Datensätze können aus einer Datei (1), über eine URL (2) oder aus einer Datenbank (3) in das WEKA-System geladen werden (Abbildung 1.17 auf der nächsten Seite).

Dateien werden in WEKA im ARFF-Format erwartet. ARFF-Dateien bestehen aus einem Kopf und einem Körper. Im Kopf stehen die Schlüsselworte, die mit @ beginnen, wie @relation, @attribute. Es werden nur nominale und numerische Daten erkannt. Bei nominalen Attributen folgt die Wertemenge in geschweiften Klammern {...}, bei numerischen Attributen folgt das Schlüsselwort numeric oder real. Zeilen, die mit % beginnen, sind Kommentare. Im Körper (ab @data) stehen die Daten. Ein Datensatz steht in einer Zeile, die einzelnen Attribute werden durch Kommata getrennt, fehlende Werte werden durch ? ersetzt.

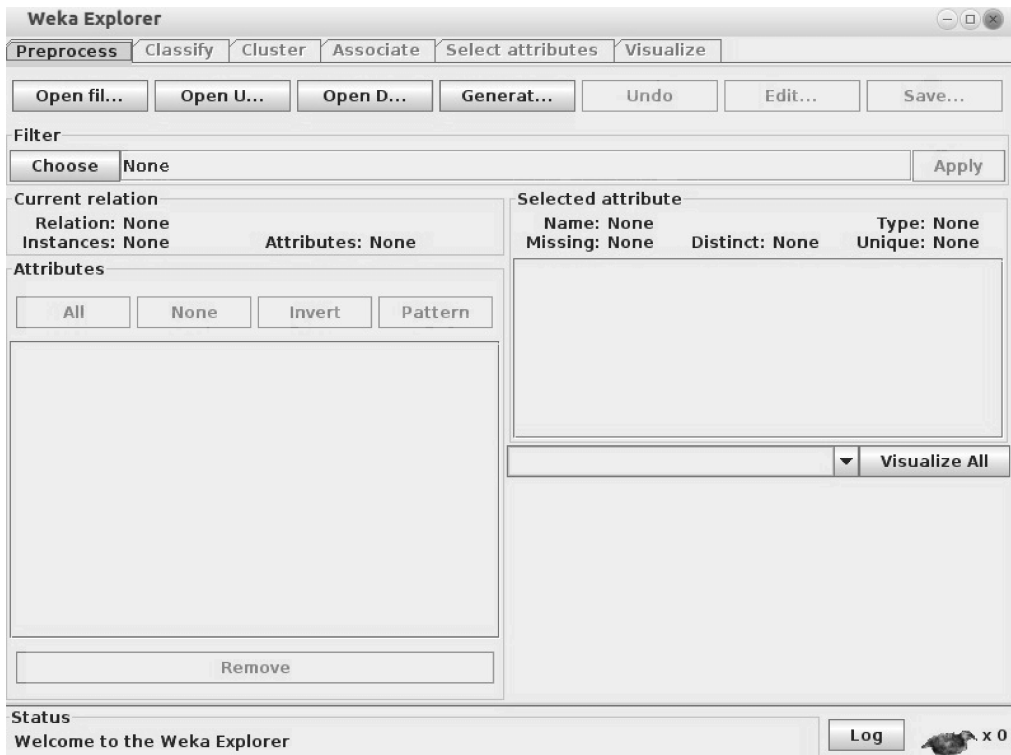


Abb. 1.17: WEKA – Explorer

### Beispiel 1.1: Wetter-Beispiel in WEKA

```
@relation weather.symbolic
```

```
@attribute outlook {sunny, overcast, rainy}
```

```
@attribute temperature {hot, mild, cool}
```

```
@attribute humidity {high, normal}
```

```
@attribute windy {TRUE, FALSE}
```

```
@attribute play {yes, no}
```

```
@data
```

```
sunny, hot, high, FALSE, no
```

```
sunny, hot, high, TRUE, no
```

```
overcast, hot, high, FALSE, yes
```

```
rainy, mild, high, FALSE, yes
```

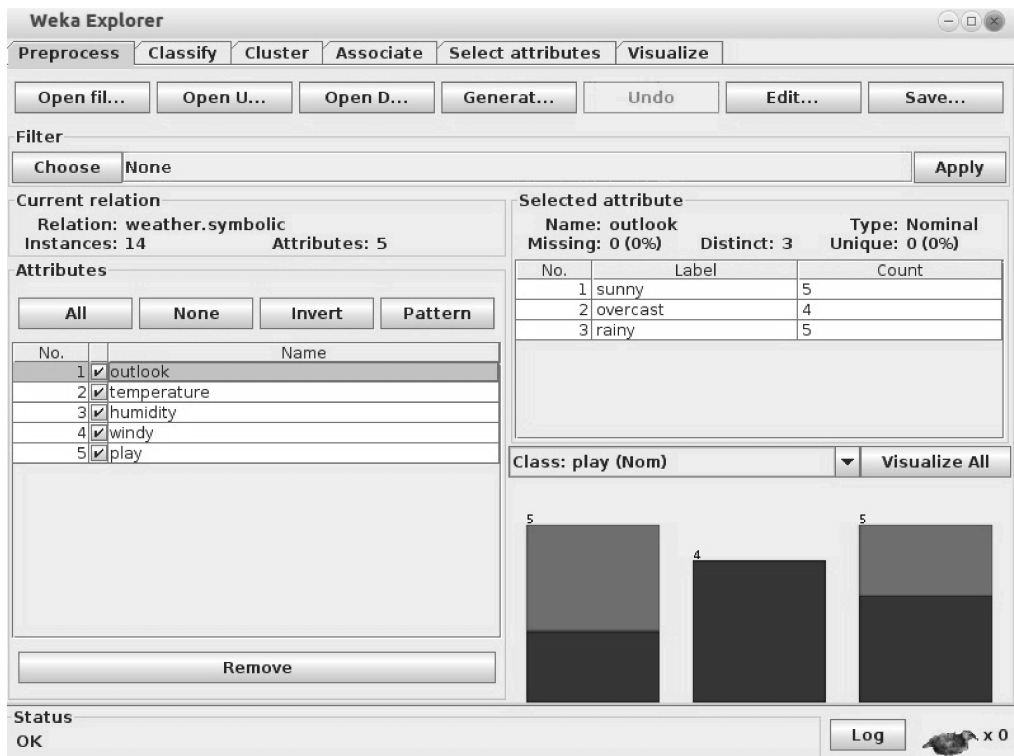
```
rainy, cool, normal, FALSE, yes
```

```
rainy, cool, normal, TRUE, no
```

```
overcast, cool, normal, TRUE, yes
```

sunny, mild, high, FALSE, no  
 sunny, cool, normal, FALSE, yes  
 rainy, mild, normal, FALSE, yes  
 sunny, mild, normal, TRUE, yes  
 overcast, mild, high, TRUE, yes  
 overcast, hot, normal, FALSE, yes  
 rainy, mild, high, TRUE, no

Lädt man diese Datei in WEKA (Abbildung 1.18), erscheinen die Attributnamen. Jetzt kann ausgewählt werden, welche Attribute im weiteren Verlauf verwendet werden. Im rechten, unteren Teil ist die Verteilung des Zielattributs *play* bezüglich des im linken Fensterteil gewählten Attributs dargestellt.



**Weka Explorer**

Preprocess | Classify | Cluster | Associate | **Select attributes** | Visualize

Open fil... | Open U... | Open D... | Generat... | Undo | Edit... | Save...

Filter: Choose | None | Apply

**Current relation**  
 Relation: weather.symbolic  
 Instances: 14 | Attributes: 5

**Attributes**  
 All | None | Invert | Pattern

No.	Name
1	<input checked="" type="checkbox"/> outlook
2	<input checked="" type="checkbox"/> temperature
3	<input checked="" type="checkbox"/> humidity
4	<input checked="" type="checkbox"/> windy
5	<input checked="" type="checkbox"/> play

Remove

**Selected attribute**  
 Name: outlook  
 Missing: 0 (0%) | Distinct: 3 | Type: Nominal  
 Unique: 0 (0%)

No.	Label	Count
1	sunny	5
2	overcast	4
3	rainy	5

Class: play (Nom) | Visualize All

Bar chart showing the distribution of 'play' values for each 'outlook' category:  
 - sunny: 5  
 - overcast: 4  
 - rainy: 5

Status: OK | Log | x 0

**Abb. 1.18:** WEKA – Preprocessing

WEKA bietet eine Reihe von Filtern an, mit denen die Daten vorverarbeitet werden können.

Im Programmteil *Classify* sind etliche Klassifizierer implementiert. Wählt man jetzt beispielsweise den ID3-Algorithmus (unter *Choose > Trees*) und startet diesen, so bekommt man im rechten Teilfenster das Resultat der Berechnung (Abbildung 1.19).

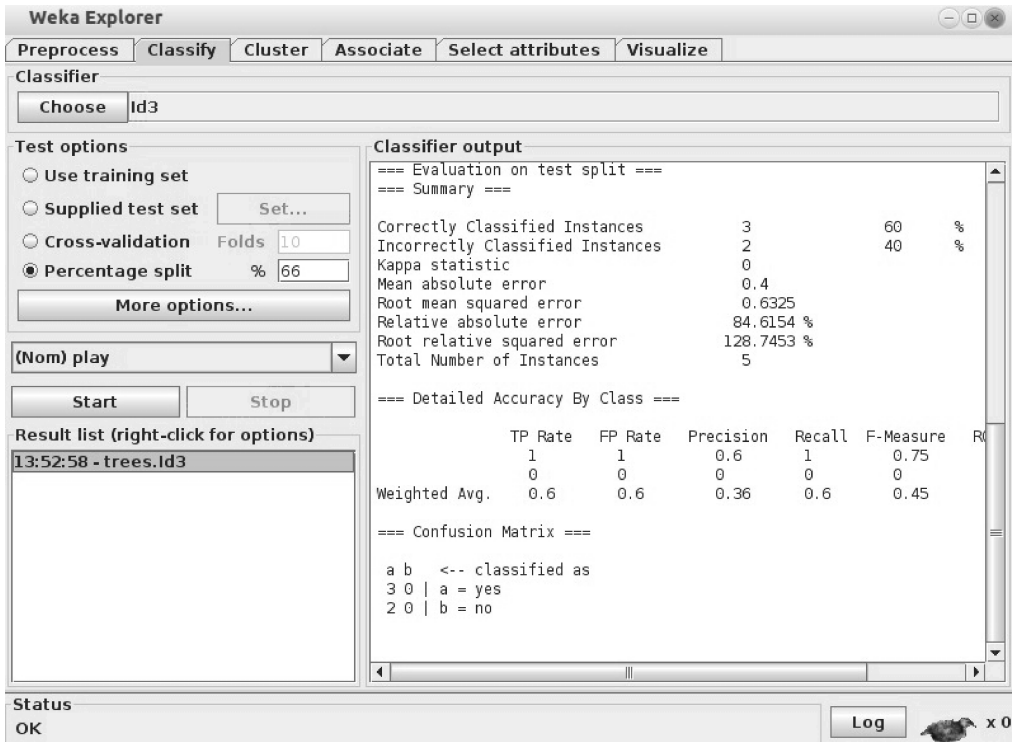


Abb. 1.19: WEKA – ID3

Man kann auswählen, auf welchen Daten gelernt werden soll:

### Use training set

Die gesamte Datenmenge wird zum Lernen benutzt.

### Supplied test set

Eine separate Datei kann zum Lernen angegeben werden.

### Cross validation

Die gegebene Menge wird mittels Kreuzvalidierung gesplittet (Abschnitt 9.4).

### Percentage split

Die gegebene Menge wird prozentual in Trainings- und Testmenge aufgeteilt.

Für den J48-Algorithmus, eine Variante der Erweiterung des ID3-Algorithmus C4.5, ist eine Visualisierung des Entscheidungsbaums (Abbildung 1.20) verfügbar (rechter Mausklick auf das Resultat, *Weka Node View*).

Analog sieht das Vorgehen bei den anderen Anwendungsklassen aus. Um Cluster zu berechnen, wählt man den Reiter *Cluster* aus. Dann selektiert man unter *Choose* beispielsweise den *SimpleKMeans*-Algorithmus (Abschnitt 6.2) aus. Wir clustern unsere

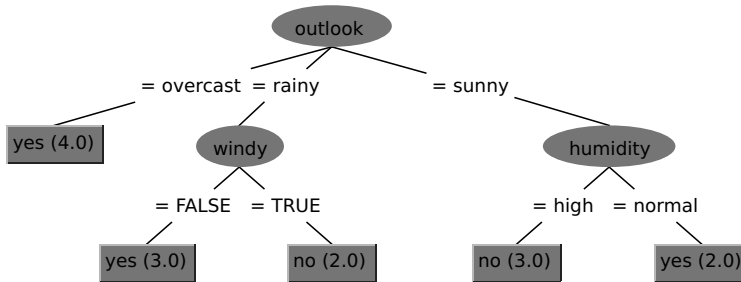


Abb. 1.20: WEKA – Entscheidungsbaum

gesamte Beispielmenge (*Use training set*). Das Resultat ist in Abbildung 1.21 dargestellt.

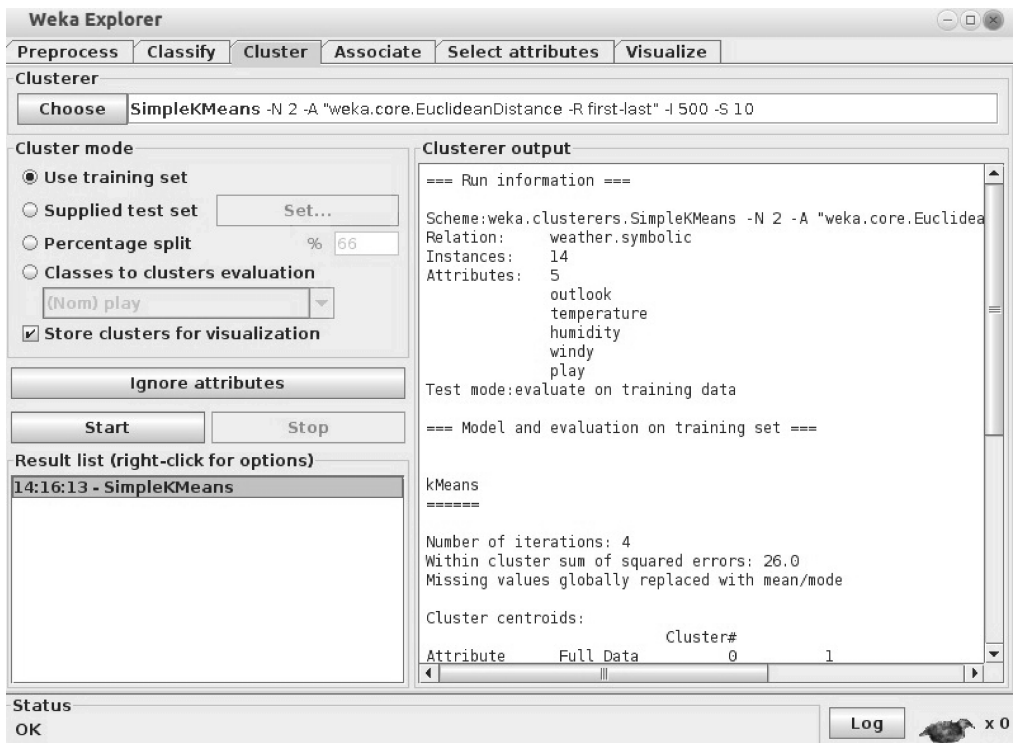


Abb. 1.21: WEKA – SimpleKMeans

### 1.6.3 JavaNNS

In den 90er Jahren wurde an der Universität Stuttgart der *Stuttgarter Neuronale Netze Simulator (SNNS)* entwickelt. Dieser war lange Zeit nur unter Unix-Betriebssystemen einsetzbar. Die Benutzungsoberfläche weicht erheblich von dem allgemein üblichen Aufbau ab, so dass die Handhabung einige Einarbeitung erfordert.

Mit der Entwicklung einer in Java implementierten Oberfläche für den SNNS ist die Handhabung erheblich erleichtert worden, und die Nutzung sowohl unter Linux- oder allgemein Unix-Systemen als auch unter Windows-Betriebssystemen ist möglich. Die Kombination aus SNNS und des in Java implementierten Nutzer-Interface ist nun unter dem Namen JAVANNS bekannt:

<http://www.ra.cs.uni-tuebingen.de/software/JavanNS/> (Zugriff 2013-04-26)

Mit dem JAVANNS können verschiedene künstliche neuronale Netze entwickelt werden. Anhand der Entwicklung eines vorwärtsgerichteten künstlichen neuronalen Netzes für eine Klassifikationsaufgabe wird die Arbeit mit dem JAVANNS erläutert. Als Beispiel dient das Wetter-Problem, siehe Anhang A.3: Wird gespielt? Die Daten sind der Tabelle 1.2 zu entnehmen.

Tag	outlook	temperature	humidity	windy	play
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

**Tabelle 1.2:** Wetter-Daten

Für die Arbeit mit einem neuronalen Netz sind die ordinalen Daten in eine binäre Darstellung zu transformieren. Der neuronale Netze-Baustein in KNIME wandelt derartige Daten automatisch um, behandelt dabei ordinale Attribute genauso wie nominale. Die Werte werden in 0-1-Folgen codiert, die so viele Binärwerte aufweisen, wie das Attribut unterschiedliche Werte aufweist. So werden zum Beispiel die Werte des Attributs *outlook* wie folgt codiert:

rainy      = (1 0 0),  
 overcast = (0 1 0),  
 sunny     = (0 0 1).

Da dabei aber die Ordnung verloren geht – *sunny* > *overcast* > *rainy* – setzen wir für die Arbeit mit dem JAVANNNS die folgende Codierung ein, die die Ordnung berücksichtigt.

```

outlook:    rainy    = (0,0);    overcast = (0,1); sunny = (1,1);
temperature: cold    = (0,0);    mild      = (0,1); hot   = (1,1);
humidity:   normal  = 0;         high     = 1;
windy:      false   = 0;         true      = 1;
play:       no      = 0;         yes       = 1;

```

Für das Trainieren des Netzes ist eine Muster-Datei *.pat* zu erstellen, die alle Muster in der codierten Form enthält und folgenden Aufbau besitzt:

```

SNNS pattern definition file V3.2
generated at Mon Apr 28 18:08:50 2013

```

```

No. of patterns      : 14
No. of input units  : 6
No. of output units : 1

```

```

# Input pattern 1:
1 1  1 1  1  0
# Output pattern 1:
0
# Input pattern 2:
1 1  1 1  1  1
# Output pattern 2:
0
# Input pattern 3:
1 1  0 1  1  0
# Output pattern 3:
0
...

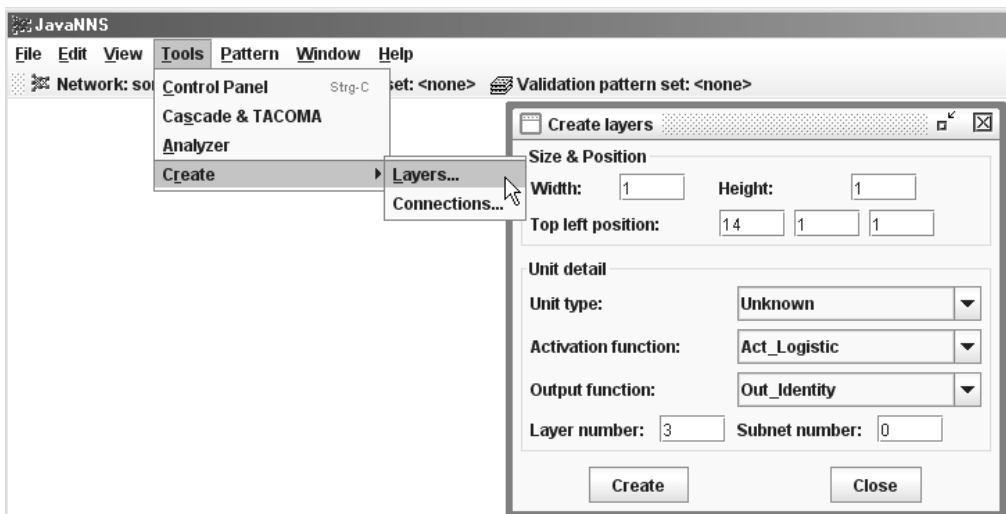
```

Mit der Codierung zu beginnen hat somit den Vorteil, dass sich hieraus sowohl die Größe der Eingabe-Schicht als auch die der Ausgabe-Schicht ergibt: Für das Beispiel wird ein Netz mit sechs Eingabe-Neuronen und einem Ausgabe-Neuron benötigt. Wir werden eine kleine Zwischenschicht aus zwei Neuronen hinzufügen. Dazu sind nach dem Starten von JAVANNNS die folgenden Schritte durchzuführen:

- Anlegen der drei Neuronen-Schichten mittels *Tools > Create > Layers ...*
  - Die erste Schicht von 6 Neuronen kann als 2×3- oder 1×6-Schicht, erzeugt im *Create-Layer*-Fenster, angelegt werden. Der Typ der Neuronen (*Unit Type*) muss auf *Input* gesetzt werden.
  - Die Zwischenschicht wird als 1×2-Schicht ausgelegt und der Typ mit *Hidden* festgelegt. Verändern Sie die *Top-Left-Position* der Schicht, in dem Sie den empfohlenen Wert um 1 erhöhen, damit in der Darstellung ein sichtbarer Abstand zwischen den erzeugten Neuronen-Schichten entsteht.



- Die Ausgabe-Schicht besteht nur aus einem Neuron ( $1 \times 1$ -Schicht). Verändern Sie wieder die *Top-Left-Position* wie vorher angegeben, und setzen Sie den Typ des Neurons auf *Output*.
- Stellen Sie die vorwärtsgerichtete Vernetzung zwischen den Schichten her:  
*Tools > Create > Connection*  
 Wählen Sie *Connect feed-forward* und vergessen Sie nicht, mittels *Connect* die Verbindungen zu erzeugen.
- Speichern Sie dieses Netz.
- Laden Sie die Muster-Datei. Die Datei ist vom Typ .pat und muss den oben dargestellten Inhalt aufweisen.



**Abb. 1.22:** JAVANNS: Erzeugen einer Neuronen-Schicht

Das Netz kann nun noch „verschönert“ werden, indem die Ein- sowie Ausgabe-Neuronen angemessene Bezeichnungen erhalten: Dazu markiere man ein Neuron und editiere es entsprechend. Nun kann das künstliche neuronale Netz trainiert und so ein Klassifikator erzeugt werden. Hierzu wird das *control*-Fenster geöffnet (*Tools > Control*), mit dem der Trainingsprozess gesteuert werden kann. Für die Kontrolle des Lernerfolgs ist die Anzeige des Netzfehlers sinnvoll: *View > Error graph*.

Die Trainingsumgebung ist in Abbildung 1.23 zu sehen. Nachdem das Netz initialisiert wurde, kann es unter Nutzung der Standardwerte (*Backpropagation-Lernverfahren*, *Lernparameter 0,2*) trainiert werden. Dazu wird die Zyklenzahl festgelegt und das Training mittels *Learn All* angestoßen. Die Fehlerkurve vermittelt einen Eindruck vom Lernerfolg.

Das Ergebnis kann dann mittels *Save Data > Auswahl Result files .res* gespeichert werden (Abbildung 1.24 auf der nächsten Seite).

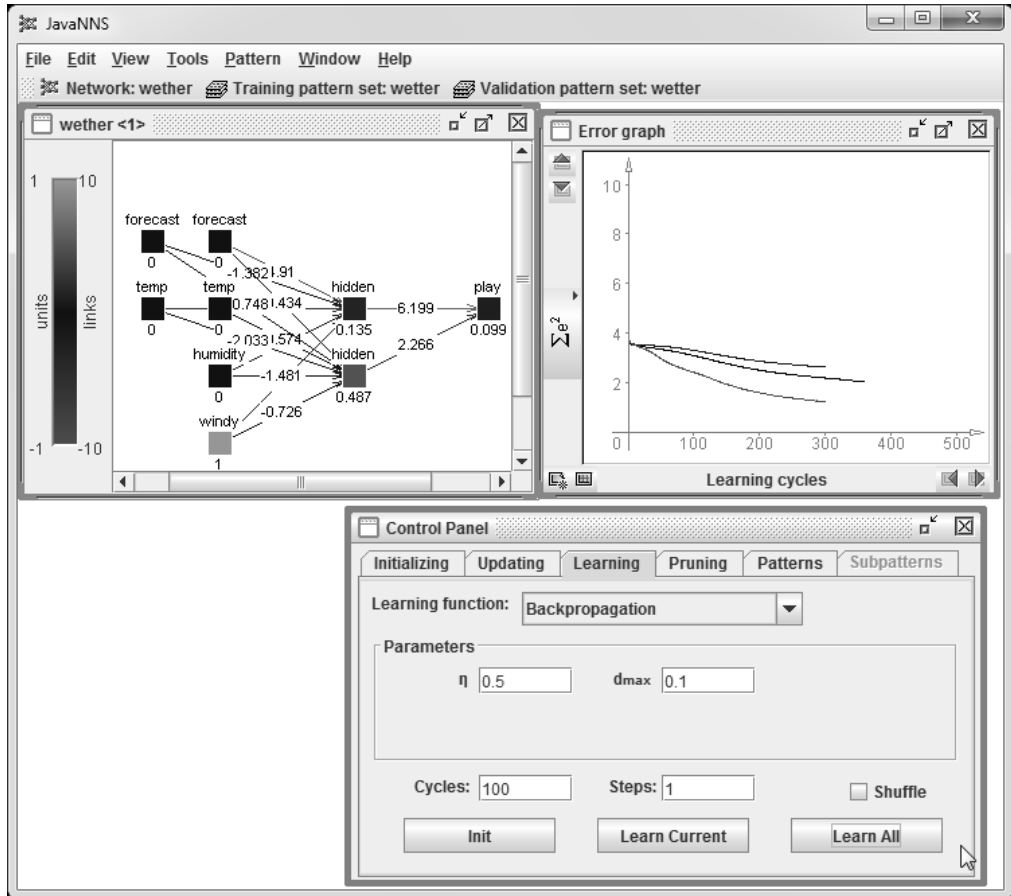


Abb. 1.23: JAVANNS: Trainieren eines Netzes

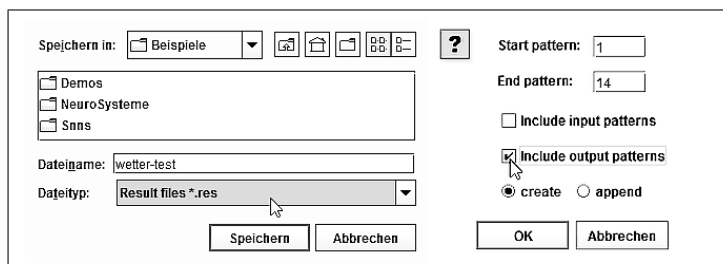


Abb. 1.24: JAVANNS: Speichern des Ergebnisses

Üblicherweise werden die Trainingsausgaben (siehe Abbildung 1.24) mit in die Ergebnisdatei aufgenommen. Diese ist eine Text-Datei, die für jedes Muster sowohl den Trainingswert als auch den vom Netz berechneten Wert enthält, so dass eine weitere Auswertung, zum Beispiel mittels Tabellenkalkulation vorgenommen werden kann. Das folgende Listing zeigt den Inhalt einer .res-Datei.

```
SNNS result file V1.4-3D
generated at Sun May 05 16:43:13 2013
```

```
No. of patterns      : 14
No. of input units   : 6
No. of output units  : 1
startpattern         : 1
endpattern           : 14
teaching output included
#1.1
0
0.20123
#2.1
0
0.01936
#3.1
0
0.9002
#4.1
1
0.78917
#5.1
1
0.94537
```

Der JAVANNS kann für viele Typen künstlicher neuronaler Netze eingesetzt werden, vorzugsweise jedoch für Klassifikationsaufgaben mittels vorwärtsgerichteter neuronaler Netze. Da der JAVANNS den Stuttgarter Neuronale Netze Simulator (SNNS) benutzt, kann ein entwickeltes Netz auch mittels Kommandozeilen-Steuerung in andere Anwendungen eingebettet werden.

### **Aufgabe 1.1:** *Data-Mining-Werkzeuge*

Machen Sie sich mit den vorgestellten Systemen vertraut, indem Sie die vorgestellten Beispiele nachvollziehen.

