8 Datenvorbereitung

Experience is something you don't get until just after you need it.
Olivier's Law

Bisher haben wir uns um die in der Praxis wohl schwierigste Aufgabe gedrückt: die Datenvorbereitung. Unter Vorbereitung fassen wir die drei ersten Teilprozesse des KDD-Prozesses zusammen, siehe Abbildung 8.1 (vergleiche Abbildung 1.1 auf Seite 5).

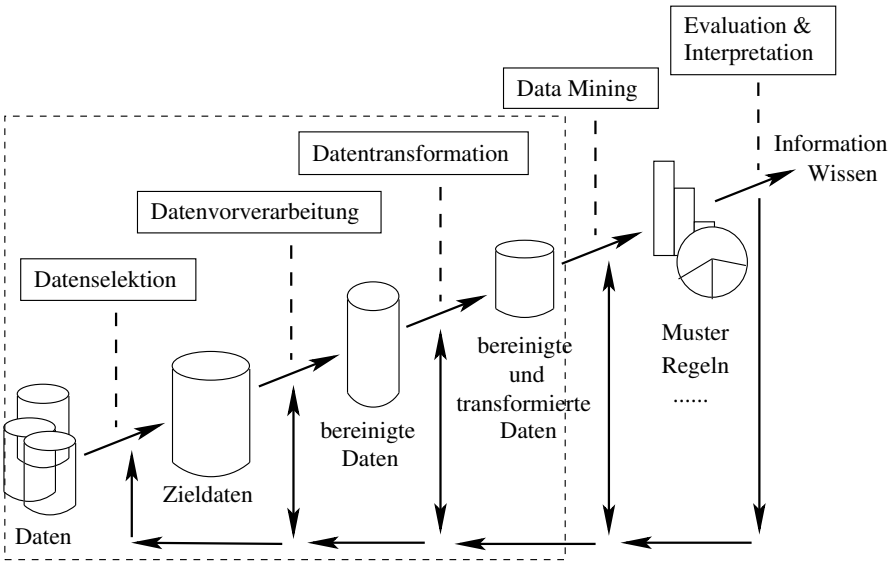


Abb. 8.1: *Datenvorbereitung*

8.1 Motivation

Nicht alle Daten können direkt aus der Datenbank in ein Data-Mining-Verfahren übertragen werden. Sie müssen vielfach erst aufbereitet werden. Dazu ist es nötig, sich mit den konkreten Daten auseinanderzusetzen. Der erste Schritt ist normalerweise das Sammeln aller verfügbaren Daten. Oft liegen die Daten nicht in *einer* Datentabelle vor, sondern müssen aus unterschiedlichen Datenbanken oder anderen Quellen zusammengetragen werden.

Und leider sind die Daten dann möglicherweise auch noch fehlerbehaftet. Welche Probleme können auftreten?

Eines der klassischen Probleme bei der Datenanalyse sind *Ausreißer*. Ausreißer sind Daten, die von den anderen Daten erheblich abweichen oder außerhalb des üblichen Wertebereichs liegen. Es ist daher häufig sinnvoll, metrische Attribute genauer zu untersuchen. Hat man Daten über 30- bis 50-Jährige, so ist ein Datensatz über einen 90-Jährigen genauer zu analysieren. Zum einen kann es sich um einen Ausreißer handeln, andererseits könnte die 90 auch ein falscher Wert sein.

Ob solche Ausreißer für das Data Mining ausgeblendet oder adaptiert werden sollten oder besser doch im Originalzustand zu verwenden sind, hängt vom konkreten Kontext ab.

Eines der Hauptproblemfelder in der Datenvorbereitung ist der Komplex der *fehlenden*, *ungenauen*, *falschen* und *widersprüchlichen* Werte. Verschiedene Faktoren können dazu führen, dass ein Datensatz fehlende, ungenaue oder gar falsche Attributwerte enthält. Die Ursachen können beispielsweise in einer Befragung selbst begründet sein. So ist es denkbar, dass befragte Personen die Beantwortung einzelner Fragen aus persönlichen Gründen verweigern oder gar falsche Angaben machen. Ein weiterer Grund für fehlende Attribute kann die Tatsache sein, dass die Struktur der Datenbank verändert wurde und bereits erfasste Datensätze nicht überarbeitet wurden. Und natürlich muss auch immer mit der Möglichkeit von Flüchtigkeitsfehlern – beispielsweise bei einer manuellen Erfassung der Werte – gerechnet werden.

Wie geht man nun mit solchen Daten um? Beim Umgang mit fehlenden Daten stellt sich die Frage nach der Bedeutung des Fehlens. Ist die Tatsache des Fehlens selbst eine Information, die sich auf das Ergebnis des Data Minings auswirkt? Hat die Ursache des Fehlens vielleicht einen Einfluss auf das Ergebnis? In vielen Fällen hilft es, fehlende Werte durch spezielle Ausprägungen zu ersetzen, die nicht auftreten können (beispielsweise negative Zahlen bei Stückzahlen).

Dies sind typische Probleme, mit denen ein Data-Mining-Projekt zu kämpfen hat, bevor die eigentliche Analyse beginnen kann.

Auch andere Probleme können auftreten. Häufig ist eine *Dimensionsreduktion* erforderlich. Der Grund hierfür kann zum einen in einer begrenzten Rechenkapazität liegen. Meistens liegen in einer Datenbank zu einem Datensatz viele Attribute vor. Dies führt beim Data Mining zu teilweise hochdimensionalen Problemstellungen. Eine hohe Dimensionalität bedeutet auch immer einen hohen Rechenaufwand und somit erhöhte Programmlaufzeiten. Diese sind aber für viele Anwendungsfälle inakzeptabel.

Zum anderen ist die visuelle Wahrnehmungsfähigkeit des Menschen beschränkt. Eine 3-dimensionale Visualisierung, eventuell auch eine 4-dimensionalen Raum-Zeit-Visualisierung stellen uns bereits vor große Herausforderungen. Da Visualisierung nicht nur im Nachgang der Datenanalyse – zur Veranschaulichung der Resultate –, sondern auch *vor* der Datenanalyse genutzt wird, ist eine Dimensionsreduktion bereits hier erforderlich.

Die Reduktion einer zu hohen Dimensionalität kann auf zweierlei Weise erfolgen. Zum einen können einzelne Attribute einfach ausgeblendet werden. Zum anderen kann man versuchen, abhängige Komponenten zu ermitteln und zusammenzufassen.

Ein weiteres Problem in der Datenvorbereitung kann dadurch entstehen, dass Daten in Formaten vorliegen, die für das jeweilige Data-Mining-Verfahren unpassend sind. Beispielsweise können bestimmte Angaben in Millimeter, andere in Kilometer vorliegen. Hier ist eine Datentransformation erforderlich. Ebenso kann eine Skalierung der Daten notwendig sein.

Die Datenvorbereitung spielt beim Data Mining eine entscheidende Rolle, da die Qualität der Daten nicht nur für das Laufzeitverhalten des Data-Mining-Prozesses, sondern auch für die Qualität der Resultate entscheidend ist. Data Mining ist ein typisch iterativer Prozess: Man probiert etwas aus, berechnet ein Resultat und prüft dieses. Häufig ist man nicht gleich im ersten Durchlauf erfolgreich. Nach unserer eigenen Erfahrung liegt der Aufwand in diesem iterativen Prozess mit etwa 80% bei der Datenvorbereitung.

Die Datenvorbereitung ist aus einem ganz einfachen Grund für die Qualität der Resultate essentiell. Wenn man schlechte (beziehungsweise schlecht vorbereitete) Daten hat, so kann man nicht erwarten, dass die Verfahren dies ausbügeln und gute Resultate liefern. Ein altes Prinzip gilt auch hier: GIGO – garbage in, garbage out.

Die Datenvorbereitung, insbesondere die Datenvorverarbeitung befasst sich mit der Qualität der Daten. Ziel der Datenvorbereitung ist es, die Qualität der Daten zu verbessern, um die Chancen auf eine erfolgreiche Datenanalyse zu erhöhen.

Im Folgenden betrachten wir einige typische Probleme der Datenvorbereitung sowie Vorgehensweisen zu deren Lösung. Wir werden dabei nur bestimmte Techniken vorstellen, aber keine Patentrezepte präsentieren können. Jedes Data-Mining-Projekt ist anders. Gerade die Phase des *Data understanding* (vgl. CRISP-Modell, Abbildung 1.2 auf Seite 6) ist von großer Bedeutung, denn eine sinnvolle Datenvorbereitung setzt meistens voraus, dass man die Daten „verstanden“ hat. Fehlt beispielsweise für ein Attribut ein Wert, so muss man wissen, welche Semantik dieses Attribut hat. Die Anforderungen an die Datenvorbereitung sehen folglich immer wieder anders aus.

Bevor man sich der Datenvorbereitung widmet, sollten im ersten Schritt immer einfache statistische Tests durchgeführt werden. Um einen ersten Eindruck von den Daten zu bekommen, sind Angaben wie die *Zahl der fehlenden Werte* interessant. Bei numerischen Attributen geben die Werte zu *Durchschnitt*, *Median*, *Standardabweichung*, *Maximum* und *Minimum* einen guten Einblick in die Daten. Bei nicht nominalen Daten helfen uns die Angaben über die *möglichen Werte* und deren *Häufigkeit* beim Verstehen der Daten.

8.2 Arten der Datenvorbereitung

Bevor wir auf einzelne Proleme und mögliche Lösungstechniken eingehen, wollen wir zunächst die Phasen der Datenvorbereitung betrachten. Wir unterscheiden:

Datenselektion und -integration
: In diesem Schritt werden geeignete beziehungsweise erforderliche Daten ausgewählt und zusammengefügt. Die Daten – aus unterschiedlichen Quellen – werden zu *einer* Datenbanktabelle vereinigt.

Datensäuberung
Die Daten werden bereinigt.

Datenreduktion
Die Daten werden reduziert, beispielsweise bezüglich der Dimension.

Datentransformation
Zum Schluss werden die Daten umgewandelt, um so adäquate Darstellungsformen – in Abhängigkeit vom jeweiligen Verfahren – für die Daten zu bekommen.

Die Datensäuberung und -reduktion gehören gemäß dem KDD-Ablauf (Abbildung 1.1 auf Seite 5) zum 2. Schritt (Datenvorverarbeitung). Da beide aber vom Charakter her unterschiedlich sind, behandeln wir sie in diesem Kapitel separat.

Die Datentransformation umfasst in der Regel *verfahrensabhängige* Schritte der Datenvorbereitung. Zur Datentransformation gehört beispielsweise das Umwandeln von metrischen Daten in ordinale Daten, falls wir den ID3-Algorithmus anwenden wollen.

Insofern sind die Datensäuberung und die Datenreduktion meistens *verfahrensunabhängig*, die sich anschließende Datentransformation jedoch vom anzuwendenden Verfahren *abhängig*.

8.2.1 Datenselektion und -integration

Der erste Schritt in einem Data-Mining-Projekt ist die Auswahl der nötigen Daten, die *Datenselektion*.

Die ausgewählten Daten stammen möglicherweise aus unterschiedlichen Tabellen oder sogar unterschiedlichen Datenbanken und sind zu *einer* Datentabelle zusammenzuführen. Dies ist die *Datenintegration*. Dabei kann es eine Reihe von Problemen geben:

- Selbst in *einer* Firma kann jede Filiale ihre eigene Datenbank haben.
- Datenbanken können unterschiedliche Strukturen besitzen.
- Es ist eine unterschiedliche Semantik oder Syntax der Attribute möglich.

Aufgabe der Datenintegration ist es, Daten mehrerer Datensätze aus unterschiedlichen Quellen zusammenzuführen. Das Ergebnis ist idealerweise eine konsistente Tabelle mit *schlüssigen Datensätzen*.

Folgende Probleme können bei der Selektion und Integration auftreten:

Entitätenidentifikationsproblem
Welche Merkmale besitzen dieselbe Semantik? Hat man beispielsweise zwei Attribute `Kunden_ID` und `Kundennummer`, so stellt sich die Frage, ob beide Attribute dasselbe bedeuten. Eine Antwort darauf findet man möglicherweise in sogenannten Metadaten – Daten, die die Daten beschreiben.

Redundanzen
Redundanzen können durch Inkonsistenzen in der Nomenklatur von Attributen oder Dimensionen entstehen. Betrachtet man beispielsweise die Attribute `Name` und `name`, so sind beide Attribute syntaktisch unterschiedlich, besitzen aber wohl dieselbe Semantik.

Widersprüche
Beim Zusammenfügen von unterschiedlichen Datenquellen können Widersprüche in den Daten auftreten. Beispielsweise kann es unterschiedliche Adressen für dieselbe Person geben.

Datenwertkonflikte
In den Datensätzen eines Attributs können unterschiedliche Maßeinheiten auftreten, beispielsweise: Entfernungen in Meilen und Kilometern.

Verletzen der referenziellen Integrität
Ein typischer Fehler ist das Verletzen der referenziellen Integrität. Dies kann durch einen fehlerhaften Verweis eines Fremdschlüssels auf einen nicht existierenden Schlüsselwert in einer anderen Tabelle auftreten.

Beispiel 8.1: *Verletzen der referenziellen Integrität*

Seien die folgenden Tabellen gegeben:

Kundendaten

KdNr	Name	Ort
1	Meier	Wismar
2	Schulze	Schwerin
3	Lehmann	Rostock

Bestellungen

Bestell-Nr.	KdNr
1	2
3	1
2	1
4	4

In der Bestellungen-Tabelle wird auf den Kunden 4 verwiesen, den es nicht gibt.

Die obigen Probleme entstehen dadurch, dass *unterschiedliche* Quellen zusammengeführt werden. In den folgenden Abschnitten gehen wir nun davon aus, dass wir bereits alle Daten in *einer* Datentabelle vorliegen haben.

8.2.2 Datensäuberung

Daten in der realen Welt sind mitunter unvollständig, mit Fehlern oder Ausreißern behaftet oder sogar inkonsistent. Dies kann und muss in den Vorbereitungsschritten beseitigt werden. Die adäquate Behandlung dieser Probleme ist – wie oben schon dargestellt – von großer Bedeutung, da unvollständige Daten zu Fehlern im Data-Mining-Prozess führen können, aus denen dann möglicherweise unzuverlässige oder sogar falsche Resultate entstehen.

Beim Bereinigen der Daten ist darauf zu achten, dass möglichst keine neue Information hinzugefügt wird. Eingefügte Werte oder Daten sollten informationsneutral sein, um die vorhandenen Informationen, die aus der realen Welt stammen, nicht zu verzerren oder

zu verfälschen. Dies kann man in der Praxis nicht immer garantieren, man sollte dieses Ziel aber im Blick behalten.

Im Folgenden werden wir uns mit diesen Problemen befassen:

- Fehlende Daten
- Verrauschte Daten
- Falsche Daten
- Inkonsistente Daten

Fehlende Daten

Bei fehlenden Werten klärt man zunächst, ob es Fehler im eigentlichen Sinn sind, oder ob sich aus dem Fehlen eines oder mehrerer Werte eine Information ableiten lässt. Füllt jemand bei einem Kreditantrag bestimmte Felder nicht aus, so kann dies bewusst – man möchte eine Information ungern preisgeben – oder unbewusst geschehen sein. Dies muss man für den jeweiligen Sachverhalt entscheiden. Wir konzentrieren uns hier auf das Beseitigen echter Fehler.

Folgende Möglichkeiten bieten sich an, auf fehlende Daten zu reagieren:

- *Attribut ignorieren*
  Man streicht das komplette Attribut, also ein ganze Spalte aus der Daten-Tabelle heraus.
  Aber seien Sie vorsichtig (siehe oben): Löscht man die gesamte Spalte, gehen eventuell Informationen verloren. Leere Felder können eine Information enthalten.
  Natürlich darf ein Attribut nur dann gelöscht werden, wenn noch genügend *vollständige* Attribute vorhanden sind.

- *Fehlende Werte manuell einfügen*
  Das manuelle Einfügen ist gerade bei großen Datenmengen sehr zeitintensiv und damit unrealistisch. Es ist sogar meistens praktisch undurchführbar. Beispielsweise fehlten in den Datensätzen des Data Mining Cups 2001 [DMC] für ca. 500 beziehungsweise 1000 Kunden – in den Lern- beziehungsweise Testdaten – die Werte für `Kunde_seit`. Hier kann man nicht per Hand Werte nachtragen, sondern muss automatisiert Werte hinzufügen.

- *Globale Konstante*
  Man kann die fehlenden Werte durch eine globale Konstante – wie beispielsweise `unbekannt` oder `minus unendlich` – ersetzen.
  Diese Methode ist sinnvoll, wenn ein leeres Feld als Information angesehen wird oder viele Werte fehlen.

- *Durchschnittswert*
  Bei metrischen Attributen hat man die Möglichkeit, den Durchschnittswert aller Einträge zu verwenden.

Bei Klassifikationsaufgaben kann man dieses Vorgehen verfeinern, indem man für die Durchschnittswertberechnung nur die Datensätze derselben Klasse benutzt. Dies ist eine recht einfache Möglichkeit der Datensäuberung. Sie bietet sich an, falls die (numerischen) Werte der Klasse dicht beieinander liegen und man davon ausgehen kann, dass die fehlenden Werte auch in diesem Bereich liegen. Das Ergänzen von Durchschnittswerten lässt sich schnell umsetzen.
Stehen keine Klassen zur Verfügung, kann man auch den Durchschnittswert der Datensätze nehmen, die dem aktuellen Datensatz ähnlich sind. Dieses Vorgehen folgt der Idee des k-Nearest Neighbours.

- *Wahrscheinlichster Wert*
  Eine weitere Möglichkeit ist das Ersetzen der fehlenden Werte durch den wahrscheinlichsten Wert dieses Attributs, den man beispielsweise mit statistischen Methoden ermitteln kann.
  Hierfür sollte man sich entscheiden, wenn bei einem Fall oder einigen wenigen Fällen kein Wert vorhanden ist und man genügend Anhaltspunkte für einen sinnvollen oder begründeten Wert hat.

- *Häufigster Wert*
  Handelt es sich um ein nichtnumerisches Attribut, so kann man den häufigsten Wert einsetzen.

- *Relation zwischen Attributen*
  Hat man eine Relation zwischen dem Attribut mit fehlenden Werten und einem anderen Attribut gefunden, so lässt sich dies ausnutzen: Fehlt das Alter einer Person, so kann man es aus dem Attribut *Geburtsjahr* berechnen.
  Findet man mittels Regression einen Zusammenhang zwischen numerischen Attributen, so kann man mit der Regressionsfunktion fehlende Werte berechnen.
  Auch mittels Techniken der Assoziationsanalyse lassen sich Zusammenhänge zwischen Attributen herstellen und fehlende Attributwerte vorhersagen.

- *Datensatz als fehlerhaft kennzeichnen*
  Als Notlösung bleibt uns immer noch, die entsprechenden Datensätze von der Weiterverarbeitung auszuschließen.
  Dies ist sinnvoll, wenn man genug vollständige Daten hat.
  Allerdings ist dies nur für die Trainingsdaten eine Lösung, in der Anwendung muss dann doch auf fehlende Werte reagiert werden.

Knime bietet eine Reihe von Möglichkeiten an, mit fehlenden Werten umzugehen. Für Integer-Werte sind diese im Screenshot in Abbildung 8.2 auf der nächsten Seite dargestellt. Für reelle Zahlen gibt es die gleichen Optionen, mit Ausnahme des häufigsten Wertes. Für Zeichenketten bietet Knime den häufigsten Wert, einen festen Wert und das Löschen des Datensatzes an.

Knime bietet auch die Möglichkeit, individuell für jedes Attribut festzulegen, wie mit fehlenden Werten umgegangen werden soll (Abbildung 8.3).

Mit diesen Techniken werden wir uns notgedrungen von unserem Ziel, unseren Datenbestand *informationsneutral* zu modifizieren, verabschieden müssen. Das Einfügen von

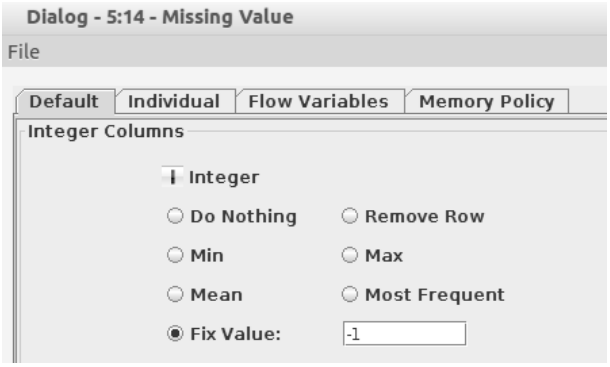


Abb. 8.2: KNIME – *Missing Values*

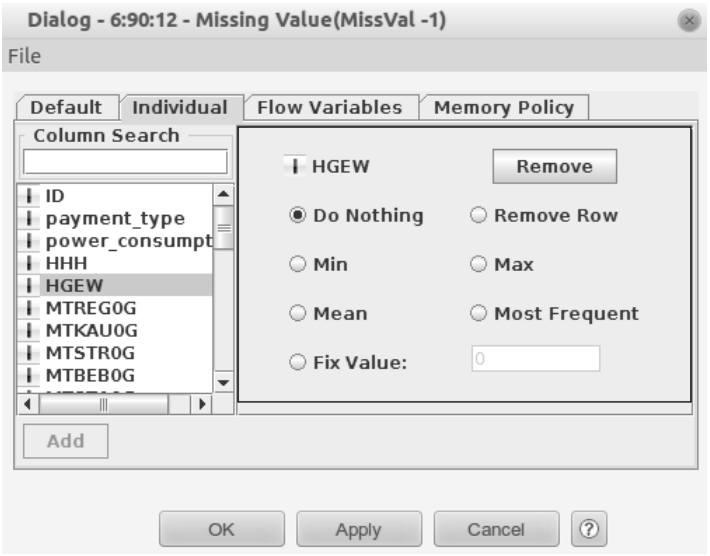


Abb. 8.3: KNIME – *Missing Values – individual*

Werten – nach welcher Methode auch immer – führt zur *Veränderung des Datenbestands* und kann die Qualität der Daten beeinflussen. Ebenso kann die *Semantik der Daten* verfälscht werden. Das Ignorieren oder Weglassen von Attributen oder Datensätzen verfälscht die Daten bezüglich der Wahrscheinlichkeitsverteilung der Attributwerte.

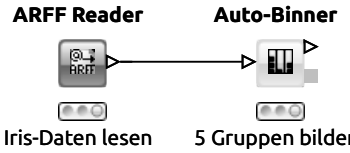
Wir verletzen also unser Ziel, dass jegliche Veränderung unseres Datenbestands informationsneutral sein sollte. Wir haben aber bei fehlenden Daten keine andere Chance.

Alle Veränderungen sind ausführlich und nachvollziehbar zu dokumentieren, nicht zuletzt um deren Einfluss auf das Ergebnis abschätzen zu können.

Verrauschte Daten und Ausreißer

Unter verrauschten Daten verstehen wir Daten, die mit leichten Fehlern behaftet sind. Solche Fehler entstehen häufig durch ungenaue Messwerte oder Schätzungen. Die Daten können auf verschiedene Art und Weise geglättet (angeglichen) werden. Einige der gängigen Verfahren stellen wir hier vor.

- *Klasseneinteilung (binning)*
  Man gruppiert die verrauschten Daten und ersetzt sie durch Mittelwerte oder Grenzwerte.
  KNIME bietet zum Binning einige Methoden an. Der folgende Workflow führt ein Binning für die Iris-Daten (vgl. Abschnitt A.1 und Beispiel 6.1 auf Seite 145) durch.



  Man kann das Binning zum einen KNIME komplett überlassen und gibt nur die Anzahl der Bins vor.

sepal length	sepal width	petal length	petal width	class	sepal length Binned	sepal width Binned	petal length Binned	petal width Binned
5,1	3,5	1,4	0,2	Iris-setosa	Bin 2	Bin 4	Bin 1	Bin 1
4,9	3	1,4	0,2	Iris-setosa	Bin 1	Bin 3	Bin 1	Bin 1
4,7	3,2	1,3	0,2	Iris-setosa	Bin 1	Bin 3	Bin 1	Bin 1
4,6	3,1	1,5	0,2	Iris-setosa	Bin 1	Bin 3	Bin 1	Bin 1
5	3,6	1,4	0,2	Iris-setosa	Bin 1	Bin 4	Bin 1	Bin 1
5,4	3,9	1,7	0,4	Iris-setosa	Bin 2	Bin 4	Bin 1	Bin 1
4,6	3,4	1,4	0,3	Iris-setosa	Bin 1	Bin 3	Bin 1	Bin 1
5	3,4	1,5	0,2	Iris-setosa	Bin 1	Bin 3	Bin 1	Bin 1
4,4	2,9	1,4	0,2	Iris-setosa	Bin 1	Bin 2	Bin 1	Bin 1
...								

  Andererseits kann man die Intervalle aber auch selbst festlegen. In KNIME geschieht dies mit dem *Numeric Binner*.

- *Regression*
  Man beschreibt die Daten durch eine mathematische Funktion. Dann ersetzt man die realen, verrauschten Datenwerte durch die berechneten Funktionswerte der mittels linearer Regression (siehe Seite 62) gefundenen Funktion.

- *Verbundbildung (clustering)*
  Ausreißer lassen sich durch Verbundbildung erkennen. Man bildet Cluster von ähnlichen Werten, beispielsweise mit einem dichtebasierten Verfahren wie DBScan (Abschnitt 6.6). Die Ausreißer liegen dann außerhalb der Cluster.

- *Kombinierte Maschine/Mensch-Untersuchung*
  Der Computer erstellt eine Liste (anscheinend) befremdlicher Werte, danach filtert der Mensch die Ausreißer aufgrund von Erfahrungswerten heraus.

Offen ist die Frage, wie man mit Ausreißern, nachdem man sie erkannt hat, umgeht. Hier bieten sich die gleichen Möglichkeiten wie beim Umgang mit fehlenden Werten an.

Beispiel 8.2: *Binning*

Wir haben folgende Messwerte gegeben: {20, 25, 27, 28, 29, 32, 37, 44, 56, 68, 70, 72} und wollen diese in 3 Gruppen (Bins) einsortieren. Dies kann man folgendermaßen umsetzen:

- *Gleichgroße Bins numerisch*
  - Bin 1: $\{20, 25, 27, 28\}$
  - Bin 2: $\{29, 32, 37, 46\}$
  - Bin 3: $\{56, 68, 70, 70\}$
- *Gleichgroße Bins nominal*
  - *Bin1* für die Werte $\{20, 25, 27, 28\}$
  - *Bin2* für die Werte $\{29, 32, 37, 46\}$
  - *Bin3* für die Werte $\{56, 68, 70, 70\}$
- *Ersetzen durch Mittelwerte*
  - Bin 1: $\{25, 25, 25, 25\}$
  - Bin 2: $\{36, 36, 36, 36\}$
  - Bin 3: $\{66, 66, 66, 66\}$
- *Ersetzen durch Grenzwerte*
  - Bin 1: $\{20, 20, 28, 28\}$
  - Bin 2: $\{29, 29, 46, 46\}$
  - Bin 3: $\{56, 56, 70, 70\}$

Inkonsistente und falsche Daten

Es gibt eine Vielzahl von Fehlermöglichkeiten und Fehlerursachen. Auf einige mögliche Fehler und Fehlerquellen sind wir bereits im vorherigen Abschnitt zur Datenselektion und -integration eingegangen.

Fehler können auf struktureller Ebene – wie bei der Datenintegration dargestellt – auftreten. Fehler können aber auch durch menschliche Fehler wie Fehleinträge oder Schreibfehler entstehen. Auch kryptische Einträge – wie Abkürzungen – können Probleme verursachen.

Häufig hat man es auch mit Werten, die außerhalb ihres Wertebereichs liegen, zu tun. Ebenso können Werte auftreten, die nicht plausibel sind.

Beispiel 8.3: *Fehlerhafte und unzulässige Werte*

- Verletzter Wertebereich: Wenn der Wertebereich auf einstellige natürliche Zahlen beschränkt ist, dann dürfen keine Zahlen $x < 0$ oder $x > 9$ auftreten.
- Verletzte Plausibilitätsbeziehungen: Ein sonst umsatzschwacher Kunde hat plötzlich einen sehr hohen Jahresumsatz.

Widersprüchliche Daten können beispielsweise dadurch entstehen, dass Attributwerte nicht zusammenpassen.

Beispiel 8.4: *Widersprüchliche Daten*

- Das Alter eines Kunden passt nicht zu seinem Geburtsjahr.
- Der Wohnort einer Person und die Postleitzahl widersprechen sich.

Auch das mehrfache Vorkommen von Datensätzen kann Probleme verursachen, da bei vielen Verfahren dieser Datensatz aufgrund des Mehrfachauftretens ein höheres Gewicht bekommt. Dies tritt beispielsweise beim Entscheidungsbaumlernen, beim Trainieren neuronaler Netze und auch beim k-Nearest-Neighbour-Verfahren auf.

Wie kann man mit derartigen Problemen umgehen? Meistens kommt man in diesen Situationen um eine Korrektur per Hand oder durch speziell dafür geschriebene Vorverarbeitungsprogramme nicht herum.

Im Allgemeinen gibt es bei solchen Fehlern – wir gehen davon aus, dass wir bereits nur *eine* Datentabelle haben – nur zwei Kategorien von Korrekturen.

- *Löschen*
  Wir können den Datensatz mit dem falschen Wert löschen. Ebenso können wir Attribute, in denen mehrfach falsche Werte vorkommen, entfernen.
- *Zuhilfenahme anderer Datensätze*
  Wir versuchen, auf der Basis der nicht fehlerbehafteten Datensätze einen plausiblen Wert zu finden.

Die erste Möglichkeit ist offensichtlich, hat aber – analog der Diskussion um fehlende Werte – den Nachteil, dass unser Datenbestand eventuell zu sehr schrumpft, so dass ein sinnvolles Data Mining unmöglich wird. Die zweite Möglichkeit – die Zuhilfenahme anderer Datensätze – ist ebenso nachvollziehbar. Beide Möglichkeiten folgen dem Vorgehen beim Umgang mit fehlenden Werten.

Problematisch wird es, wenn wir widersprüchliche Werte haben und nicht klären können, welcher Wert falsch, welcher richtig ist. Die Möglichkeit des Löschens besteht natürlich immer noch, betrifft dann aber sofort mindestens zwei Datensätze.

8.2.3 Datenreduktion

Bei vielen Data-Mining-Aufgaben sind die Datensätze sehr umfangreich. Dies kann dazu führen, dass Data Mining mit diesen Datenmengen sehr aufwändig beziehungsweise unmöglich ist. Durch Datenreduktion versucht man, dieses Problem zu mindern oder zu lösen. Dabei gibt es zwei Optionen. Zum einen kann man durch das Zusammenfassen von Attributen die Komplexität verringern. Zum anderen kann durch eine geeignete Stichprobe – eine repräsentative Teilmenge aller gegebenen Datensätze – die Problemgröße reduziert werden.

Folgende Techniken können angewendet werden:

1. Aggregation
2. Dimensionsreduktion
3. Datenkompression
4. Numerische Datenreduktion

Aggregation

Unter *Aggregation* – auch Verdichtung – versteht man das *Zusammenfassen* von Fakten zu *einem Fakt* oder das Generalisieren der Daten. So kann man beispielsweise Daten durch ihre Mittelwerte ersetzen oder Teilwerte zu einer Gesamtsumme zusammenfassen (siehe Seite 214). Hat man die Umsätze einer Firma pro Monat vorliegen, so kann man diese Werte zum Jahresumsatz aufsummieren. Statt 12 einzelnen Datensätzen wird dann mit nur einem weitergearbeitet.

Neben dieser zeilenweisen Aggregation, kann auch eine spaltenweise Aggregation erfolgen, die mehrere Attribute aggregiert. Hat man *Tag, Monat, Jahr* als einzelne Attribute, so kann man diese 3 Attribute zu einem Attribut *Datum* zusammenfassen. Ebenso können beispielsweise die Umsätze der einzelnen Unterabteilungen zu einem Attribut – dem Gesamtumsatz – zusammengefasst werden. Damit reduziert man nicht die Zahl der Datensätze, aber die Anzahl der Attribute.

Dimensionsreduktion

Bei der Dimensionsreduktion werden irrelevante Daten – also Attribute – vernachlässigt und relevante Daten (Attribute) einbezogen. Folgende Vorgehensweisen bieten sich an:

- Schrittweise *Vorwärtsauswahl*: Gute Attribute werden schrittweise in die Zielmenge eingegliedert. Die Attributmenge wächst sukzessive.
- Schrittweise *Rückwärtseliminierung*: Ausgehend von der Gesamtmenge werden uninteressante Attribute schrittweise eliminiert.

Beide Vorgehensweisen lassen sich natürlich kombinieren.

Auf metrische Daten können auch Techniken wie die Hauptachsentransformation oder die multidimensionale Skalierung (siehe Seite 243) angewendet werden, um die Dimension zu reduzieren.

Datenkompression

Die Daten werden entweder transformiert oder codiert, um eine Reduktion der Datenmenge und damit eine Reduktion der Komplexität des Data Mining zu erhalten. Hier steht die Verringerung der Anzahl der Attribute im Vordergrund. Beispielsweise fasst man Binärattribute zu einem Byte zusammen oder vereint die separaten Attribute für die Produkte *Löffel, Gabel, Messer ...* zu einem Attribut *Besteck* (vgl. Aggregation, Seite 214).

Numerische Datenreduktion

Eine numerische Datenreduktion – die Auswahl einer repräsentativen Teilmenge der gegebenen Datensätze – kann auf der Basis von Stichproben realisiert werden.

Man nimmt für das Data Mining nicht die gesamte Datenmenge, sondern nur eine – natürlich viel kleinere – Stichprobe. Zu klären ist die Frage, wie man diese Stichprobe bildet.

- *Zufällige Stichprobe*
  Es werden rein zufällig Datensätze aus der Quelldatenmenge ausgewählt.
- *Repräsentative Stichprobe*
  Es werden zufällig Datensätze aus der Quelldatenmenge ausgewählt, allerdings so, dass die Stichprobe repräsentativ ist. Dazu wird die Auswahl unter Beachtung der Häufigkeitsverteilungen bestimmter Attribute getroffen. Ebenso sollte man absichern, dass jede Attributausprägung – bei nominalen und ordinalen Attributen – vorkommt. Bei Klassifikationsproblemen muss jede Klasse ausreichend vertreten sein.
- *Geschichtete Stichprobe*
  Wie bei der repräsentativen Stichprobe werden Datensätze zufällig ausgewählt. Es wird aber darauf geachtet, dass wichtige Attribute auch einen Wert im Datensatz besitzen.
- *Inkrementelle Stichproben*
  Eine erste Stichprobe wird nach der Datenanalyse Schritt für Schritt erweitert. Dies kann nach einem der obigen Verfahren geschehen.
- *Average Sampling*
  Die Quelldatenmenge wird in Teile gespalten und jeder Teil unabhängig von den anderen einer Analyse unterzogen. Die Ergebnisse werden im Anschluss daran gemittelt und so zu einem Gesamtergebnis vereint.
- *Selektive Stichprobe*
  Aus der Quelldatenmenge werden unergiebige Datensätze herausgefiltert. Anschließend wird eine Stichprobenziehung durchgeführt.
- *Windowing*
  Ähnlich wie bei der inkrementellen Stichprobe wird hier von einer Basisstichprobe ausgegangen. Diese wird nach der Analyse aber nur um ergiebige Datensätze – beispielsweise Datensätze für Klassen, bei denen das Verfahren sich schlecht

verhielt – erweitert. Sprich: Zu Beginn wird nur ein Teil der Trainingsdaten (Datenfenster) verwendet. Dann werden nur noch Trainings-Datensätze in die Trainingsmenge aufgenommen, bei denen sich das erlernte Verfahren – beispielsweise ein Entscheidungsbaum – falsch verhält.

- *Clustergestützte Stichprobe*
  Bei dieser Methode werden ähnliche Datensätze in Clustern zusammengefasst und ein Repräsentant (oder einige) gewählt, der im Anschluss für die Analyse herangezogen wird.

Alternativ kann man eine numerische Datenreduktion auch über lineare Regression erreichen. Dabei ersetzt man die Daten durch die Koeffizienten einer linearen Regressionsfunktion.

KNIME unterstützt die Stichprobenwahl. Die einfachste Variante ist in Abbildung 8.4 dargestellt. Der *Partitioning*-Knoten bietet die Möglichkeit, den gesamten Datenbestand in einem bestimmten Prozentsatz aufzuteilen. Bei Klassifikationsaufgaben sollte man *Stratified Sampling* wählen, um in den Teilmengen die Häufigkeitsverteilung bezüglich des Zielattributs zu erhalten. Dies ist sinnvoll, da ein Verfahren zum Generieren eines Entscheidungsbaums natürlich ausreichend Beispiele für alle möglichen Werte der vorherzusagenden Klasse braucht, um sinnvolle Vorhersagen liefern zu können. Eine Vorhersage für gute und schlechte Kunden wird nicht gelingen, wenn in der Trainingsmenge nur schlechte Kunden auftauchen.

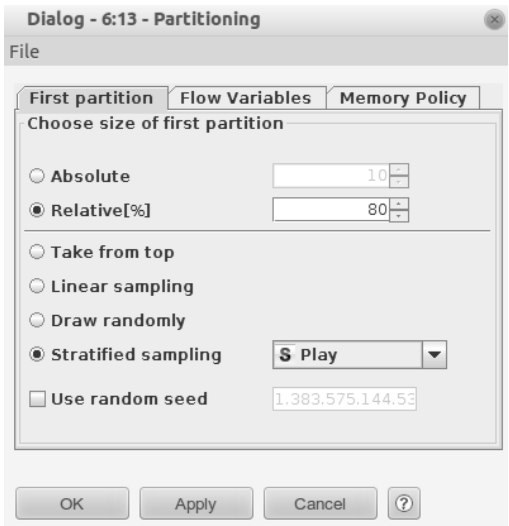


Abb. 8.4: *Stichprobenwahl nach Prozenten*

Oft tritt bei Klassifikationsaufgaben das Problem auf, dass die Zielklassen nicht gleich häufig in der Datenmenge vertreten sind. KNIME bietet auch hierfür Unterstützung, in Form des Knotens *Equal Size Sampling*.

8.2.4 Datentransformation

Bisher haben wir Probleme der Datenvorbereitung diskutiert, die unabhängig vom jeweiligen Verfahren auftreten. In diesem Abschnitt nähern wir uns nun dem eigentlichen Data-Mining-Schritt, der tatsächlichen Datenanalyse. Oftmals sind die Daten in ihrer ursprünglichen Form nicht zum Data Mining geeignet. Die Datentransformation hat die Aufgabe, die Daten in eine Form umzuwandeln, mit der das jeweilige Data-Mining-Verfahren arbeiten kann.

Es gibt eine Reihe von Transformationen, die bei der Datenvorbereitung hilfreich sind. Häufig ist eine Anpassung

1. der *Datentypen*,
2. von *Konvertierungen oder Codierungen*,
3. von *Zeichenketten*,
4. von *Datumsangaben*,
5. von *Maßeinheiten und Skalierungen*

erforderlich.

Weitere Transformationen können sinnvoll sein:

1. *Kombination oder Separierung* von Attributen
2. Berechnung *abgeleiteter Werte*
3. *Datenaggregation*
4. *Datenglättung*

Anpassung der Datentypen

Betrachtet man die 2 als einzelnen Wert, so stellt sich die Frage: Ist diese 2 vom Datentyp `Char` oder `Integer`, und sollte eine Umwandlung stattfinden? Der ID3-Algorithmus (Abschnitt 5.2.3) verlangt ordinale beziehungsweise nominale Attribute und kann mit metrischen Attributen nicht umgehen. Für den k-Nearest-Neighbour-Algorithmus (Abschnitt 5.1) sind metrische Daten aber von Vorteil.

Beispiel 8.5: *Codierungen – Körpergröße*

Wollen wir das Attribut *Körpergröße* bei der Generierung eines Entscheidungsbaums mittels ID3 verwenden, so müssen wir es in ein nominales Attribut umwandeln.

- *Variante 1:* Gibt es nur eine geringe Anzahl von Werten, so können wir einfach den Zahlenwert in einen String umwandeln.
- *Variante 2:* Wir bilden Intervalle, beispielsweise $[1,00-1,60]$, und geben diesen Namen: *klein*. Wir können dies manuell tun oder ein automatisches Verfahren – beispielsweise Cluster-Bildung – anwenden.

Beispiel 8.6: *Codierungen – Leistungsklassen von Autos*

Haben wir ein Attribut *Geschwindigkeit* mit ordinalen Ausprägungen (*sehr schnell*, *schnell*, *mittelschnell*, *langsam*) gegeben und wollen den k-Nearest-Neighbour-Algorithmus mit dem euklidischen Abstand anwenden, so müssen wir das Attribut zunächst in ein numerisches umwandeln. Wollen wir dies normiert tun – also Werte zwischen 0 und 1 haben – so bietet sich folgende Codierung an:

- *sehr schnell* $\rightarrow$ 1
- *schnell* $\rightarrow$ 0,66
- *mittelschnell* $\rightarrow$ 0,33
- *langsam* $\rightarrow$ 0

Die Umwandlung von *sehr schnell* in 1 und *langsam* in 0 ist offensichtlich, da wir ja eine Codierung im Intervall $[0, 1]$ haben wollen. Allerdings könnten wir die beiden anderen Werte durchaus alternativ codieren:

- *sehr schnell* $\rightarrow$ 1
- *schnell* $\rightarrow$ 0,8
- *mittelschnell* $\rightarrow$ 0,5
- *langsam* $\rightarrow$ 0

Wichtig ist nur, dass die Ordnungsrelation

$$\textit{sehr schnell} > \textit{schnell} > \textit{mittelschnell} > \textit{langsam}$$

nicht verletzt wird.
Beide Codierungen sind korrekt; sie erhalten die Ordnung der Werte. Allerdings können sie beim k-Nearest-Neighbour-Algorithmus zu unterschiedlichen Resultaten führen, da die Abstände zwischen den Geschwindigkeitswerten unterschiedlich sind. Bei der ersten Variante ist der Abstand zwischen *sehr schnell* und *mittelschnell* 0,67, bei der zweiten 0,5. Dies kann Auswirkungen auf das Resultat haben.

Anpassung von Konvertierungen oder Codierungen

In Abhängigkeit vom jeweiligen Verfahren kann es nötig sein, die Daten umzuwandeln und sie anders zu codieren.

Für die neuronalen Netze und auch die Assoziationsanalyse ist die *Binärcodierung* eine wichtige Codierungsform. Durch Binärcodierung wird aus Attributen mit einer bestimmten Anzahl Merkmalsausprägungen eine Menge binärer Attribute. Jeder Merkmalsausprägung wird ein neues, binäres Attribut zugeordnet, das den Wert 1 annimmt, wenn die Ausprägung in einem einzelnen Datensatz vorkommt und sonst den Wert 0

besitzt (vgl. Beispiel 4.7 auf Seite 77 oder Abschnitt 5.4.1 auf Seite 117). Dieses Verfahren kann beispielsweise das Attribut Kaufverhalten mit den Ausprägungen *Vielkäufer*, *Seltenkäufer* und *Nichtkäufer* so transformieren, dass man das Attribut Kaufverhalten durch 3 Binärattribute `Vielkäufer`, `Seltenkäufer`, `Nichtkäufer` ersetzt. Auf diese Weise wird ein qualitatives Attribut in mehrere, binärcodierte Attribute überführt. Das Binärcodierungsverfahren bereitet nominale und ordinale Attribute für Algorithmen vor, die numerische Attribute erfordern.

Eine abgewandelte binäre Codierung basiert auf der sogenannten unären Codierung, die einfach so viele Einsen enthält, wie die Zahl, die wir darstellen wollen. Betrachten wir hierzu wieder das Attribut *Kaufverhalten* und codieren die Werte gemäß ihrer Ordnung:

$$\textit{Vielkäufer}/2 > \textit{Seltenkäufer}/1 > \textit{Nichtkäufer}/0$$

In der unären Darstellung wird die 2 durch zwei Einsen (1 1), die Eins durch eine 1 und die 0 durch keine Eins dargestellt. Wir füllen die nur aus Einsen bestehende Codierung mit Nullen auf und erhalten eine binäre Codierung, die die Ordnung aufrecht erhält und sogar noch eine Binärstelle weniger als die vorherige Codierung aufweist:

$$\textit{Vielkäufer}/(1\ 1) > \textit{Seltenkäufer}/(1\ 0) > \textit{Nichtkäufer}/(0\ 0)$$

Bei der Anwendung der binären Codierungen ist zu beachten, dass die Performanz der Datenanalyse durch die steigende Attributanzahl beeinträchtigt werden kann.

Diskretisierung wird angewendet, um den Wertebereich von numerischen Attributausprägungen in endlich vielen Teilmengen zusammenzufassen. Eine Reihe von Verfahren kann numerische Attribute nicht verarbeiten und setzt folglich eine Diskretisierung voraus. Die Diskretisierung kann beispielsweise bei der Verallgemeinerung des Alters sinnvoll sein, da auf diese Weise die Altersinformationen zu Altersgruppen {jung, mittel, alt} zusammengefasst werden können und so eine Reduzierung der Attributausprägungen erreicht wird. Mit dem Binning (Seite 203) hatten wir bereits eine weitere Technik kennengelernt.

Anpassung von Zeichenketten

Kann das Data-Mining-Programm mit *Umlauten, Groß- und Kleinschreibung sowie Leerzeichen* in den Datensätzen umgehen, oder muss man die Daten umwandeln? Gegebenenfalls muss dies bereits bei der Datenintegration erfolgen.

Anpassung von Datumsangaben und Maßeinheiten

Datumsangaben sind oft *unterschiedlich codiert* (auf Grund von unterschiedlichen Formaten in verschiedenen Ländern, 12.03.2003; 12-03-03; 03-12-03) und erfordern daher eine Vereinheitlichung. Es können auch Daten aus unterschiedlichen *Zeitzonen* vorhanden sein, die auf jeden Fall angepasst werden müssen, da sonst das Ergebnis verfälscht wird.

Gleiches gilt für die Anpassung von Maßeinheiten, da diese oft nur *nationalen Standards* genügen (beispielsweise Inch, Zentimeter, Yard, Meter).

Die Anpassung von Zeit- oder Datumsangaben sowie die Einführung gleicher Maßeinheiten sollte nach Möglichkeit bereits bei der Datenintegration durchgeführt werden.

Normalisierung und Skalierung

Die *Normalisierung* oder *Normierung* transformiert sämtliche Merkmalsausprägungen eines Attributs auf die Werte einer stetigen, numerischen Skala.

Sollen numerische Werte auf das Intervall $[0, 1]$ normiert werden, so benötigt man dazu nur den minimalen und maximalen Wert $\min(x_i)$, $\max(x_i)$ des Attributs. Der normierte Wert berechnet sich wie folgt:

$$x_{neu} = \frac{x - \min(x_i)}{\max(x_i) - \min(x_i)}$$

Zunächst wird von allen Werten der Minimalwert subtrahiert (dann ist das Minimum 0). Anschließend wird durch die maximale Differenz zweier Werte dividiert.

Eine andere Normalisierungstechnik berechnet den statistischen Mittelwert und die Standardabweichung der Attributwerte, subtrahiert den Mittelwert von jedem Wert und dividiert dann das Ergebnis durch die Standardabweichung.

Normalisierung kann dann angewendet werden, wenn Minimum und Maximum eines Attributes gegeben sind. Die Normalisierung kann beispielsweise zur Codierung des Alters eingesetzt werden. Der Minimalwert hierbei sind 0 Jahre und der Maximalwert beispielsweise 100 Jahre. Ein Alter von 40 Jahren würden dann – auf einer Skala von 0 bis 1 – mit 0,4 codiert werden.

Beispiel 8.7: *Normalisierung*

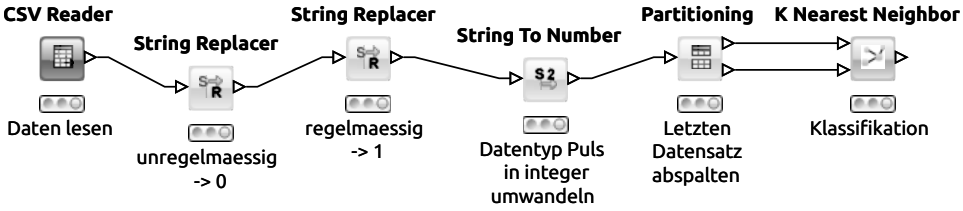
Wir betrachten folgende Daten über den Patientenzustand:

Nr.	Alter	Puls	Blutdruck	Zustand
1	27	unregelmäßig	117	gesund
2	65	unregelmäßig	123	krank
3	33	regelmäßig	127	gesund
4	55	unregelmäßig	150	krank
5	38	unregelmäßig	120	krank
6	36	regelmäßig	121	gesund
7	53	regelmäßig	140	krank
8	26	regelmäßig	107	gesund
9	43	regelmäßig	111	gesund
10	60	regelmäßig	112	gesund

Wir wollen den Algorithmus k-Nearest Neighbour (Abschnitt 5.1) anwenden und codieren folglich – der k-Nearest Neighbour arbeitet mit numerischen Attributen besser – das nominale Attribut `Puls` als numerisches Attribut. Das Zielattribut lassen wir unverändert.

Nr.	Alter	Puls	Blutdruck	Zustand
1	27	0	117	gesund
2	65	0	123	krank
3	33	1	127	gesund
4	55	0	150	krank
5	38	0	120	krank
6	36	1	121	gesund
7	53	1	140	krank
8	26	1	107	gesund
9	43	1	111	gesund
10	60	1	112	gesund

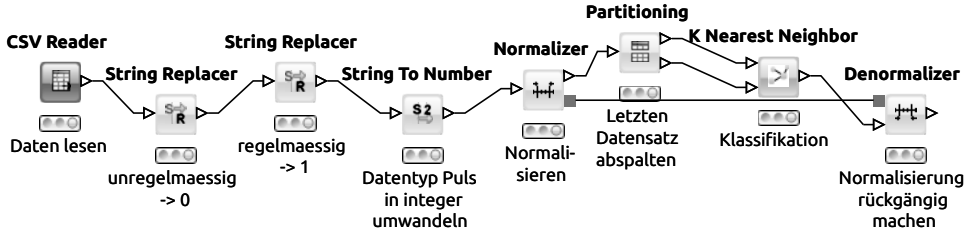
Anhand des letzten Datensatzes wollen wir k-Nearest Neighbour (k=2) testen.



Leider versagt der Algorithmus hier. Für den Patienten 10 wird `krank` vorhergesagt, der Patient ist aber gesund. Können wir uns erklären, wieso hier eine falsche Vorhersage erfolgt? Schauen wir auf die Daten, so erkennen wir, dass die 2 nächsten Nachbarn garantiert unabhängig vom Puls sind. Ausgehend vom euklidischen Abstand liefert der Puls maximal einen Unterschied von 1. Insofern ist klar, dass das Abstandsmaß vom Blutdruck und vom Alter dominiert wird. Dies können wir durch Normalisierung verhindern.

Jedes numerische Attribut wird auf das Intervall $[0, 1]$ normalisiert.

Nr.	Alter	Puls	Blutdruck	Zustand
1	0,0256	0,0	0,2326	gesund
2	1,0	0,0	0,3721	krank
3	0,1795	1,0	0,4651	gesund
4	0,7436	0,0	1,0	krank
5	0,3089	0,0	0,3023	krank
6	0,2564	1,0	0,3256	gesund
7	0,6923	1,0	0,7674	krank
8	0,0	1,0	0,0	gesund
9	0,4359	1,0	0,0930	gesund
10	0,8718	1,0	0,1163	gesund



Nun gelingt auch die korrekte Vorhersage für Datensatz 10.

Mit der Normalisierung kann man experimentieren. Zunächst sollten wir immer zunächst auf das Intervall $[0, 1]$ normalisieren. Bekommen wir damit keine guten Resultate, kann man durchaus mal ausprobieren, Attribute, die man für wichtig erachtet, eben nicht auf $[0, 1]$ zu normalisieren, sondern auf $[0, X]$ mit unterschiedlich großen X. Je größer das X, desto höher wird auch der Einfluss des Attributs auf das Resultat sein.

Kombination oder Separierung von Attributen

Es kann nötig sein, *verschiedene Attribute zu einem neuen* zusammenzufügen wie beispielsweise Tag, Monat und Jahr zu Datum. Umgekehrt kann es aber auch erforderlich sein, das Datum in seine Bestandteile zu zerlegen, um so beispielsweise den Monatsanfang oder den jeweiligen Wochentag erkennen zu können. Erst durch den Wochentag ist es eventuell möglich, das Kaufverhalten der Kunden besser zu analysieren.

Berechnung abgeleiteter Werte

Durch das Berechnen *abgeleiteter Werte* können ganz neue Attribute aufgenommen werden. So kann das Attribut *Gewinn* durch die Differenz der Attributwerte *Ausgaben* und *Einnahmen* berechnet werden.

Datenaggregation

Datenaggregation kann nicht nur aus Sicht der Datenkompression (Seite 207) erforderlich sein. Vielmehr kann die Aggregation aus inhaltlichen Gründen sinnvoll sein. Oft liegen die Daten in einer zu feinen Aggregationsebene vor. Wird zum Beispiel die Einwohnerzahl von Berlin gesucht und liegen nur die Einwohnerzahlen der einzelnen Stadtteile vor, so kann man durch *Aggregation* – in diesem Fall Summenbildung – die Daten in eine höhere Aggregationsebene überführen. Erst dann sind eventuell sinnvolle Aussagen über Berlin – im Kontext der deutschen Großstädte – möglich.

Datenglättung

Die Hauptidee der Datenglättung (siehe auch verrauschte Daten, Seite 203) ist, dass jeder numerische Wert aus der gegebenen Datenmenge durch idealisierte Werte, die sich beispielsweise durch Regression ergeben, ersetzt wird. Das Ziel der Methode ist, die ursprüngliche Wertemenge durch eine reduzierte Menge zu ersetzen, da man hofft, dass solche Werte zu besseren Lösungen führen. Darüber hinaus wird das Rauschen in den Daten reduziert.

Man kennt *vier Techniken*, mit denen Daten geglättet werden können:

- Eingruppierung (*binning*)
- Clustering
- kombinierte menschliche und maschinelle Kontrolle
- Regression

Diese hatten wir bereits bei der Behandlung von verrauschten Daten und Ausreißern diskutiert (Seite 203ff.).

8.3 Ein Beispiel

Wir erläutern einige Aspekte dieses Kapitels an einem kleinen Beispiel. Dazu nehmen wir an, wir hätten unsere Daten bereits aus den unterschiedlichsten Quellen gesammelt und diese in einer Datentabelle (Tabelle 8.1) vereinigt.

Pers. nr.	Wohnort	Geschlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
1	23966	m	45	32	10	arb	Lehre
2	23966	w	57	35000	25	verw	Bachelor
3	23966	m	52	40	5	manager	Master
4	23966	m	28	27	6	arb	Lehre
5	23966	male	57	45	25	manager	Master
6	23966	fem	26	27	96	arb	Lehre
7	23966	m	39	39	4	manager	Master
8	23966	m	38	32	3	arb	Lehre
9	23966	m	42	31	15	arb	ohne
10	23966	w	37	30	10	verw	Abi
11	23966	m	45	32	8	arb	
12	23966	m	37		5		
13	23966	w	35	30	15	verw	Abi

Tabelle 8.1: *Beispiel – Datenvorbereitung*

Was fällt uns als erstes auf?

Der Wohnort ist bei allen Personen gleich. Dieses Attribut enthält die Information, dass alle in Wismar wohnen. Dies wird uns aber bei Analysen nicht helfen, denn wir benötigen ja gerade unterschiedliche Werte der Attribute, um Unterschiede oder Ähnlichkeiten zwischen den Objekten herstellen zu können. Dieses Attribut kann folglich gelöscht werden. Wir reduzieren damit die Dimension des Problems (vgl. Seite 206).

Das Geschlecht ist sowohl als m/w als auch als male/fem angegeben. In Datensatz 2 ist das Jahresgehalt wohl nicht in Tausend angegeben. Im Datensatz 12 fehlen etliche Werte. Diesen sollten wir folglich weglassen.

Im Datensatz 6 steht bei Betriebszugehörigkeit eine 96. Es gibt 2 Interpretationen: Die 96 könnte als 1996, also als Zugehörigkeit seit 1996 interpretiert werden. Es könnte aber auch sein, dass die Betriebszugehörigkeit in Monaten angegeben wurde. Wir entscheiden uns hier für die *Monate*, da eine Zugehörigkeit seit 1996 bei einem Alter von 26 höchst unwahrscheinlich ist.

In Datensatz 11 fehlt der Bildungsabschluss. Das können wir beispielsweise korrigieren, indem wir dort den häufigsten Wert einsetzen, der für die gleiche Position (arb) vorkommt: Lehre.
Wir müssen uns an dieser Stelle aber dessen bewusst sein, dass wir mit dieser Aktion schon etwas in die Daten hinein interpretieren, und zwar einen möglichen Zusammenhang zwischen Bildungsabschluss und Position. Man könnte alternativ ein Verfahren wie *k-Nearest Neighbour* (vgl. Abschnitt 5.1) einsetzen, um so über den oder die nächsten Nachbarn einen Wert für den Bildungsabschluss zu finden.

Die veränderten Daten sind in Tabelle 8.2 dargestellt.

Pers. nr.	Geschlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
1	m	45	32	10	arb	Lehre
2	w	57	35	25	verw	Bachelor
3	m	52	40	5	manager	Master
4	m	28	27	6	arb	Lehre
5	m	57	45	25	manager	Master
6	w	26	27	8	arb	Lehre
7	m	39	39	4	manager	Master
8	m	38	32	3	arb	Lehre
9	m	42	31	15	arb	ohne
10	w	37	30	10	verw	Abi
11	m	45	32	8	arb	Lehre
13	w	35	30	15	verw	Abi

Tabelle 8.2: *Beispiel – korrigierte Tabelle*

Nun müssen wir das Ziel unserer Analyse in die Datenvorbereitung einbeziehen. Zunächst gehen wir davon aus, dass wir die Daten clustern wollen.

Offensichtlich wird die Personalnummer keine Information enthalten. Informationen wie die Zugehörigkeit zu einer Abteilung sind hier nicht zu erkennen. Die Betriebszugehörigkeit könnte man anhand dieser ID im ordinalen Sinn herauslesen, diese haben wir aber ohnehin explizit gegeben. Die Personalnummer können wir folglich ersatzlos streichen.

Für das Clustern sind metrische Werte günstig, da man mit diesen besser rechnen kann und folglich bessere Abstandsmaße bekommt. Wir können die nominalen Attribute durch Integer-Werte codieren (Tabelle 8.3 auf der nächsten Seite).

Wenn wir mit dieser Tabelle clustern (Knime, 3 Cluster), dann erhalten wir die in Abbildung 8.5 auf der nächsten Seite dargestellten Cluster.

Geschlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
0	45	32	10	0	1
1	57	35	25	1	3
0	52	40	5	2	4
0	28	27	6	0	1
0	57	45	25	2	4
1	26	27	8	0	1
0	39	39	4	2	4
0	38	32	3	0	1
0	42	31	15	0	0
1	37	30	10	1	2
0	45	32	8	0	1
1	35	30	15	1	2

Tabelle 8.3: *Beispiel – numerische Werte*

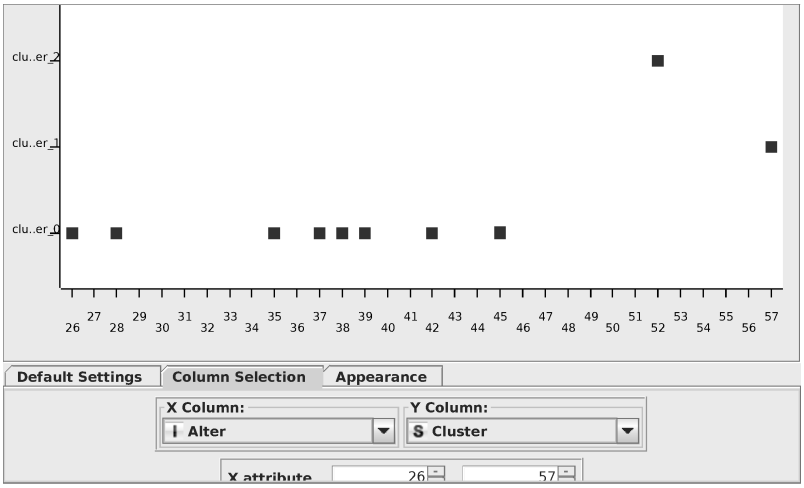


Abb. 8.5: *Clusterversuch 1*

Auffällig ist, dass das Alter offensichtlich ein große Rolle spielt. Woran liegt das? Der Abstand zwischen einer 26jährigen und einer 57jährigen Person ist so gravierend (31), dass die Unterschiede in den anderen Attributen eine zu vernachlässigende Rolle spielen. Nur die Attribute Jahresgehalt und Betriebszugehörigkeit liefern signifikante Abstände. Aber auch diese sind geringer als die Abstände beim Alter. Und die beiden Attribute Position und Bildungsabschluss liefern nur minimale Abstände.

Wie können wir die Dominanz des Alter-Attributs verhindern? Wir normalisieren alle Daten auf das Intervall $[0, 1]$ (Tabelle 8.4 auf der nächsten Seite). Dies hatten wir bereits in Beispiel 8.7 auf Seite 212 diskutiert.

Geschlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
0	0,61	0,28	0,32	0	0,25
1	1	0,44	1	0,5	0,75
0	0,84	0,72	0,09	1	1
0	0,06	0	0,14	0	0,25
0	1	1	1	1	1
1	0	0	0,23	0	0,25
0	0,42	0,67	0,05	1	1
0	0,39	0,28	0	0	0,25
0	0,52	0,22	0,55	0	0
1	0,35	0,17	0,32	0,5	0,5
0	0,61	0,28	0,23	0	0,25
1	0,29	0,17	0,55	0,5	0,5

Tabelle 8.4: *Beispiel – normalisierte Werte*

Nun erhalten wir die in Abbildung 8.6 dargestellten Cluster (wieder nur bezüglich des Alters).

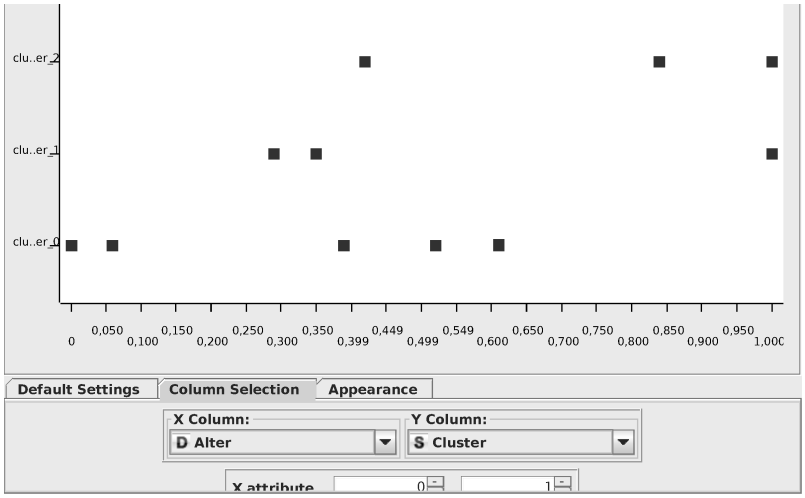


Abb. 8.6: *Clusterversuch 2 – Alter*

Unsere Vermutung beim Betrachten der Ausgangsdaten ist sicherlich, dass der Bildungsabschluss ein wichtiges Attribut sein wird. Diese Vermutung wird durch die nun vorliegende Cluster-Bildung bestätigt, wie wir aus der Darstellung der Cluster bezüglich des Bildungsabschlusses in Abbildung 8.7 erkennen können. Diese Cluster-Bildung stützt unsere These. Durch die Normalisierung der Daten haben wir ein Modell – hier eine Clusterbildung – bekommen, welches uns sinnvoll erscheint.

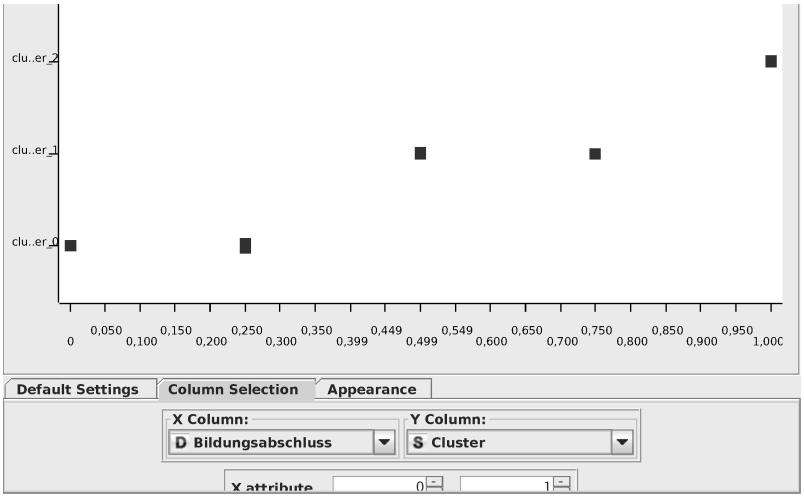


Abb. 8.7: *Clusterversuch 2 – Bildungsabschluss*

Zurück zu den Rohdaten. Nehmen wir an, wir wollen einen Entscheidungsbaum für diese Daten entwickeln, und zwar für das Zielattribut *Jahresgehalt*. Jetzt benötigen wir keine metrischen Attribute, sondern nominale oder ordinale Attribute. Wir müssen daher in unserer Ausgangstabelle die metrischen Attribute in ordinale oder nominale umwandeln (Tabelle 8.5). Dies kann man so bewerkstelligen, dass man auf allen Werten *eines* Attributs Clustering durchführt. Man erhält so zwei Intervalle, die man entsprechend benennt.

Geschlecht	Alter	Jahres-gehalt	Betriebs-zugehörigkeit	Position	Bildungs-abschluss
m	alt	gering	kurz	arb	Lehre
w	alt	viel	lang	verw	Bachelor
m	alt	viel	kurz	manager	Master
m	jung	gering	kurz	arb	Lehre
m	alt	viel	lang	manager	Master
w	jung	gering	kurz	arb	Lehre
m	jung	viel	kurz	manager	Master
m	jung	gering	kurz	arb	Lehre
m	alt	gering	lang	arb	ohne
w	jung	gering	kurz	verw	Abi
m	alt	gering	kurz	arb	Lehre
w	jung	gering	lang	verw	Abi

Tabelle 8.5: *Beispiel – nominale Werte*

Mit diesen Daten kann man nun beispielsweise den ID3-Algorithmus anwenden und erhält mit dem Zielattribut *Gehalt* den in Abbildung 8.8 auf der nächsten Seite darge-

stellten Entscheidungsbaum.

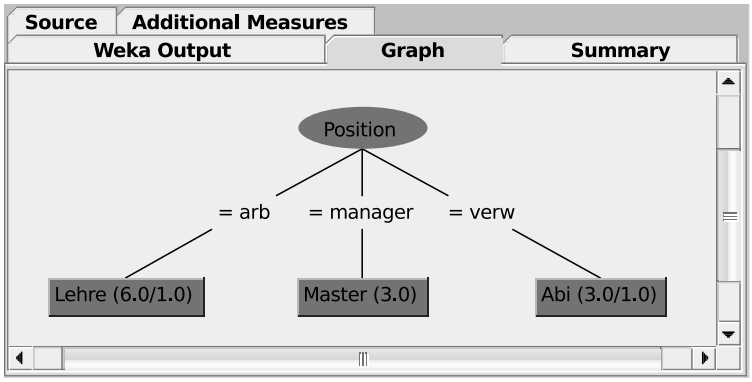


Abb. 8.8: *Entscheidungsbaum mit ID3*

Aufgabe 8.1: *Datenvorbereitung und Buch-Beispiele*

Gehen Sie die Beispiele im Buch durch und untersuchen Sie die Datenvorbereitung. Überlegen Sie sich alternative Transformationen und vergleichen Sie dann die Resultate.