DATA-MINING

DIE SUCHE NACH WERTVOLLEN INFORMATIONEN IM DATEN-NIRVANA

EINE UNTERRICHTSEINHEIT VON

KASPAR JOST & PATRICK ASCHWANDEN

Inhaltsverzeichnis

1 EINFÜHRUNG

1.1 BIG-DATA

Willkommen im Big Data Zeitalter! Google und Facebook, die neuen Megareichen vom Silicon Valley sind Meister im Ausnutzen von Webdaten. Dabei verwenden sie Ergebnisse aus Online Suchen, Posts und Meldungen, um Werbungen entsprechend zu platzieren. Im Januar 2012 am World Economic Forum in Davos war Big Data ein zentrales Thema. In einem Report mit der Überschrift „Big Data, Big Impact“[1] werden Daten als neues Wirtschaftsgut, wie Devisen oder Gold erwähnt.

In unterschiedlichen Bereichen, wie z.B. in der Wissenschaft, dem Sport, der Werbung und dem öffentlichen Gesundheitswesen, können wir Ähnliches beobachten. Ein Trend zu datengesteuerten Entdeckungen und Entscheidungsfindungen. So wurden seit Jahrzenten die Werfer (Pitcher) im US-Baseball nach folgender Formel bewertet:

$$E.R.A. = \frac{earned\ runs}{innings\ pitched}\ x\ 9$$

Der E.R.A.-Wert (earned runs avereage) beschreibt, wie viele Läufe (Runs) einem Team gelangen, während dem ein Werfer in einem Spiel insgesamt am Werfen war. Ein Baseballspiel geht über 9 Spielphasen (innings), während denen jedes Team einmal am Werfen ist. Heutzutage werden viel mehr Daten zur Auswertung eines Pitchers gesammelt und komplexere Formeln werden verwendet. Als Beispiel dazu die Siera-Formel (Skill-Interactive Earned Run Average):

$$SIERA = 6.145 - 16.986 * \left(\frac{SO}{PA}\right) + 11.434 * \left(\frac{BB}{PA}\right) - 1.858 * \left(\frac{GB - FB - PU)}{PA}\right) + 7.653 * \left(\frac{SO}{PA}\right)^2 \pm 6.664 * \left(\left(\frac{GB - FB - PU)}{PA}\right)^2\right) + 10.130 * \left(\frac{SO}{PA}\right) * \left(\frac{(GB - FB - PU)}{PA}\right) - 5.195 * \left(\frac{BB}{PA}\right) * \left(\frac{(GB - FB - PU)}{PA}\right)^2$$

Worum geht es eigentlich bei Big Data? Einerseits handelt es sich bei diesem Begriff sicher um einen Marketingbegriff. Andererseits handelt es sich um eine Kurzschrift für einen Technologietrend, welcher die Tür zu einem neuen Approach eines besseren Verständnisses der Welt und zur Entscheidungsfindung darstellt. Gemäss IDC, einem Technologie Marktforschungsunternehmen, verdoppelt sich die Datenmenge jährlich! Es ist nicht nur so, dass es immer mehr Datenströme gibt, es kommen immer gänzlich neue Datenströme dazu. Zum Beispiel sind unzählig viele digitale Sensoren in Industrieanlagen, Fahrzeugen, Stromzählern und Schiffcontainern integriert. Diese messen und kommunizieren die Position, Bewegung, Vibration, Temperatur, Feuchtigkeit und sogar chemische Veränderungen der Luftzusammensetzung.

Werden Kommunikationssensoren mit Computerterminals verbunden, um die Daten auszuwerten, dann bezeichnet man dies als das Internet der Dinge oder auch als industrielles Internet. Der verbesserte Zugang zu Informationen unterstützt zudem den Trend hin zu Big Data. Zum Beispiel wandern Regierungsdaten – Beschäftigungszahlen und andere Informationen – mehr und mehr aufs Internet.

[1] http://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf (8.12.12)
[2] http://www.baseballprospectus.com/article.php?articleid=10027 (8.12.12)

Mit der Eröffnung der Website www.data.gov [3], öffnete Washington im Jahr 2009 die Datentüre noch weiter. Diese stellt eine Unmenge von Regierungsdaten der Öffentlichkeit zur Verfügung.

Daten werden nicht nur verfügbarer, sie werden für Computer und somit auch für Menschen verständlicher. Der Datenschwall von Bildern, Wörtern, Videodokumenten und den oben genannten Datenströmen von Sensoren kommt ungeordnet daher. Man spricht in diesem Fall von unstrukturierten Daten.

Ein spannender Einstieg und Einblick in die Thematik „Big Data“ bietet eine Aufzeichnung von Input, einer Radiosendung von DRS3, die als Podcast erhältlich ist.[4]

1.2 VON DATEN ZUR INFORMATION ZUM WISSEN

Die im obigen Kapitel erwähnten unstrukturierten Daten stellen nicht gezwungenermassen Müll für Datenbanken dar. Anwendungen, welche vom Schatz an unstrukturierten Daten aus dem Internet Profit ziehen, sind enorm im Vormarsch. An vorderster Front sind die rasch voranschreitenden Techniken der Künstlichen Intelligenz, die Spracherkennung, Mustererkennung und das Maschinenlernen.

Werkzeuge der Künstlichen Intelligenz können in vielen Bereichen eingesetzt werden. Als Beispiele können hier die Websuche, das Werbegeschäft und die experimentellen Roboterpersonenwagen von Google herbeigezogen werden, welche tausende von Kilometern auf Kaliforniens Strassen bereits zurücklegten. In beiden Fällen wird ein Bündel von Tricks aus dem Bereich der Künstlichen Intelligenz genutzt. Beides sind gewaltige Big-Data-Anwendungen, die Unmengen von Daten verarbeiten, um unmittelbare Entscheidungen zu treffen.

Bevor in der Folge spezifisch auf die Begriffe Daten und Information eingegangen wird, gilt es zu klären, was ein Informationssystem (IS) und was eine Informationstechnologie (IT) ist. Ein IS ist eine Kombination von *Hardware, Software und Netzwerken, die* den Menschen beim *Sammeln, Kreieren* und Verteilen wichtiger Daten hilft. IT ist ein Werkzeug, das zum *Sammeln, Übertragen, Speichern und Verarbeiten* von Daten verwendet wird. Daten sind „rohe“ Fakten, Abbildungen und Details. Information ist eine organisierte, bedeutungsvolle und brauchbare Interpretation von Daten. Wissen ist ein Verständnis, wie eine Menge von Informationen am besten verwendet werden kann. Anders formuliert: Daten sind aufgezeichnete Fakten. Informationen sind den Daten zugrundeliegende Muster.

Abbildung 1-1: Information vs. Daten

[3] http://www.data.gov/ (9.12.12)
[4] http://www.srf.ch/sendungen/input/big-data-wie-forscher-mit-unseren-daten-die-zukunft-vorhersagen (9.12.12)

1.3 WELCHEN WERT HAT INFORMATION?

Der Mathematiker und Elektrotechniker Claude Shannon (1916 – 2001) gilt als Gründer der Informationstheorie. Er widmete sich im Besonderen den folgenden Fragestellungen:

- Was ist Information?
- Wie können wir Information messen?
- Was ist der Unterschied zwischen Information und Daten?
- Wie können wir Daten speichern und übertragen?

In unserem Alltag begegnen wir einer Unmenge von Signalen. Das kann z.B. ein Stoppsignal auf der Strasse sein. Dieses enthält eine Nachricht, welche wir als Information interpretieren.

Abbildung 1-2: Stoppsignal

Information vermittelt einen Unterschied. Dabei ist sie ausnahmslos, bei jeder Übertragung von Materie, in Form von Energien oder Impulsen ausgeprägt. Sie erreicht den Menschen über die Sinnesorgane bzw. im chemisch-biologischen Sinn über Rezeptoren und Nerven. Bewusst kann sie als Nachricht über einen Kanal (Sehen eines Stoppsignals) oder einen Träger von einem Sender an einen Empfänger übermittelt oder unbewusst durch eine Form und Eigenschaft eines Objektes ausgesandt werden.

Durch die Digitalisierung beliebiger Informationen, kann digitale Information erzeugt werden. Die nachfolgende Grafik zeigt die Aufzeichnung eines analogen Signals. Es zeichnet sich durch einen kontinuierlichen Verlauf aus. Die Nachricht kann in diesem Fall beliebig genau aufgezeichnet werden. Eine Interpretation eines solch genauen Signals, kann sehr aufwändig sein.

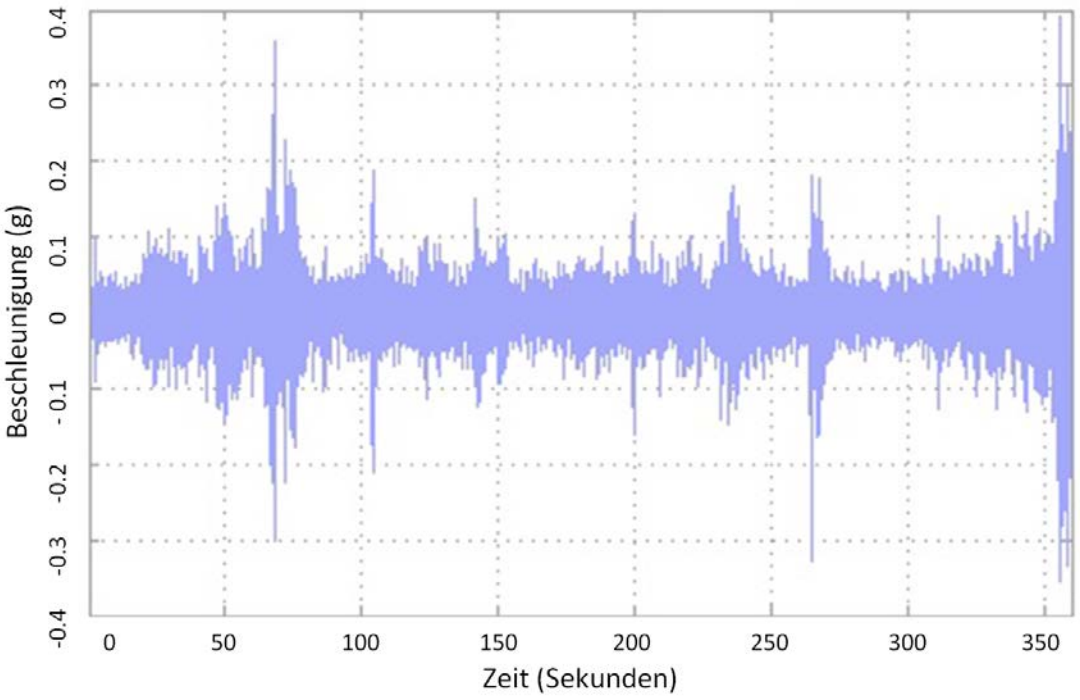


Abbildung 1-3: Beschleunigungsverlauf über die Zeit in analoger Darstellung

In der folgenden Abbildung wird sichtbar wie analoge in digitale Daten umgesetzt werden und umgekehrt. Es ist ersichtlich, dass digitale Signale sich durch einen diskreten Verlauf auszeichnen. Allerdings lassen sie nur eine begrenzte Genauigkeit zu. Andererseits ist die Verarbeitung und Interpretation digitaler Daten mit einem viel geringeren Aufwand verbunden.

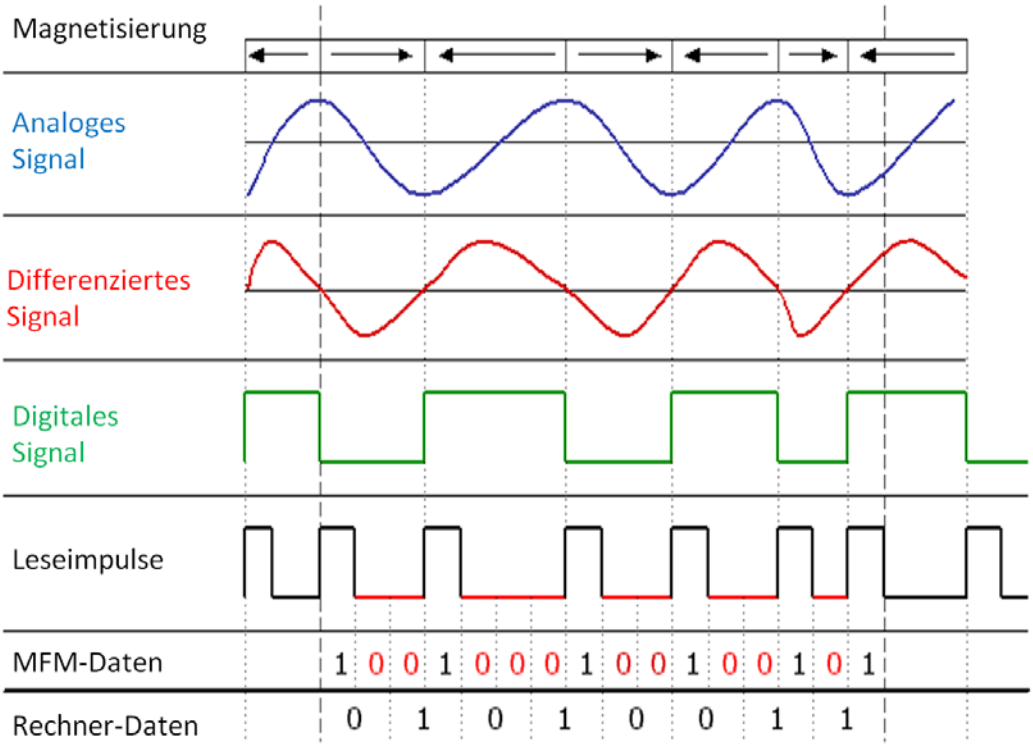


Abbildung 1-4: Digitalisierung eines analogen Signals

Nachrichten werden durch eine Sprache, Syntax und eine Semantik gekennzeichnet.

Definitionen:

- **Syntax** ist die Menge aller Regeln nach denen ein Text aufgebaut ist.
- Die Syntax eines natürlich sprachlichen Textes ist die **Grammatik**.
- Unter **Semantik** versteht man die Bedeutung eines syntaktisch richtig aufgebauten Textes.
- Eine **Sprache** ist durch Syntax und Semantik ihrer Texte definiert.

Beispiele künstlicher Sprachen:

- H_2SO_4
- (x + 7) / (x – 7)
- c3 x d7!
- for (int i = 0; i < n; i ++)

Beispiele für eine Syntax:

- Alphabet (chinesisch, griechisch, etc.)
- Code (Morsecode, Binärcode, etc.)

Binärcodes werden aufgrund ihrer einfachen Darstellung in der Regel für die Verarbeitung digitaler Informationen verwendet. Technisch lassen sie sich sehr einfach abbilden und verarbeiten, z.B. durch Spannungen. Wenn eine Spannung vorhanden ist, dann entspricht das dem Zustand 1 oder logisch wahr, ist keine Spannung vorhanden, entspricht das 0 oder logisch falsch. Diese Informationseinheit aus 0 od. 1 bzw. wahr od. falsch wird in der Informatik als Bit bezeichnet. Höherwertige Informationen lassen sich durch die logische Verknüpfung oder technische Verschaltung mehrerer dieser einfachen Werte realisieren. Die Übertragung von Informationen mittels Binärcodes kann medienunabhängig überall dort durchgeführt werden, wo ein Wechsel zwischen zwei Zuständen erzeugt und gemessen werden kann. Digitale Informationen liessen sich somit, wenn auch mit geringer Datenübertragungsrate per Rauchzeichen übermitteln!

Dezimal	Dualsystem	Aiken-Code
Wertigkeit	8 4 2 1	2 4 2 1
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 0 0 0	1 1 1 0
9	1 0 0 1	1 1 1 1

Tabelle 1-1: Dualsystem vs. Aiken-Code

Um den Entscheidungswertes von IT quantifizieren zu können, müssen wir den Informationswert kennen. Dazu wird die folgende Formel angewendet:

$$Informationswert = Wert\ eines\ Entscheidungsresultates\ mit\ Information - Wert\ des\ Entscheidungsresultates\ ohne\ Information$$

In der Praxis interessiert vor allem, wie viel für eine Information bezahlt werden sollte. Dies ist maximal deren Wert. Bei der Bestimmung des Wertes eines Entscheidungsresultates, stellen sich folgende Probleme:

- Wird der Wert des Entscheidungsresultates ex ante (vorher) oder ex post (nach) der Einschätzung ermittelt?
- Der Wert variiert über die Zeit bzw. über Situationen.
- Es muss auch der Wert von Entscheidungsalternativen eingeschätzt werden, die man nicht genommen hat.

Perfekte Information (PI) spezifiziert genau, welches Ereignis, von einer Menge künftiger Ereignisse eintritt. Dabei handelt es sich beim PI-Wert um ein theoretisches Extrem. PI wird als bestmögliche

Information bezeichnet. Imperfekte Information schränkt einzig die Erwartung möglicher zukünftiger Ereignisse ein. Imperfekte Information ist ausnahmslos weniger wert als perfekte Information. Dabei ist der Wert der perfekten Information die Obergrenze des Informationswertes. Imperfekte Information ist besser als keine Information. Ist sie nun mehr wert als ihre Beschaffungskosten?

Wahrscheinlichkeit und Erwartungswert

Um für diese beiden Begriffe ein Gefühl zu entwickeln, betrachten wir exemplarisch den Durchführungsentscheid für ein Rockkonzert:

- Per Wetterprognose soll bestimmt werden, in welchem Stadion das Konzert durchgeführt werden soll (ex ante Bestimmung des Wertes)
- Wir möchten den Erwartungswert (EW) kennen.
- Stadion 1 kostet Fr. 20'000 und bietet 20'000 überdachte Sitzplätze. Stadion 2 kostet Fr. 15'000 und bietet 25'000 open air Sitzplätze.
- Ein Ticket kostet Fr. 10 und es können beide Stadien gefüllt werden.
- Band und Crew kosten Fr. 50'000, auch wenn das Konzert wegen Regen im Open Air Stadion abgesagt werden muss. Im Falle einer Absage müssen die Ticketpreise zurückerstattet werden!
- Es besteht die Wahrscheinlichkeit von 67% für Regen. Was tun?

Aufgabe 1-1: Berechnen Sie den Erwartungswert (EW) mit perfekter Information (PI) über den Regen. Wie viel sollten Sie bereit sein, für diese Information (PI) zu bezahlen?

1.4 KREUZTABELLEN (CONTINGENCY TABLES)

Eine Kreuztabelle ist eine Tabellenart in Matrixformat, die die absoluten oder relativen Häufigkeiten von bestimmten Merkmalsausprägungen enthalten. Kontingenz bedeutet in diesem Fall, das zwei Merkmale gemeinsam auftreten. Das heisst, dass Häufigkeiten für mehrere, miteinander durch „und“ oder „sowie“ (Konjunktion) verknüpfte Merkmale, dargestellt werden. Diese Häufigkeiten werden durch deren Randsummen ergänzt, welche die sogenannten Randhäufigkeiten bilden. Ein häufig auftretender Spezialfall einer Kontingenztabelle mit zwei Merkmalen ist eine Konfusionsmatrix.[5]

Das Klassifizieren ist eine wichtige Operation im Data-Mining-Prozess. Für ein Attribut (z.B. den Wohlstand), versucht man den Reichtum künftiger Menschen über Mittelwerte anderer verfügbarer Attribute vorauszusagen. Attribute sind Messwerte einer Instanz. Dabei sind Instanzen individuelle, unabhängige Beispiele eines Konzepts. In der Folge sehen Sie dazu einen Teil der Datensätze, die aus einer im Jahr 1990 in den USA durchgeführten Volksumfrage (US Census), stammen. Diese Daten sind über das UCI Machine Learning Datasets Repository[6] online verfügbar.

age	employment	education	marital	…	job	relation	race	gender	country	wealth
				…						
39	State_gov	Bachelors	Nev-	…	Adm_cleri	Not_in_fam	White	Male	United_Sta	**poor**
51	Self_emp_	Bachelors	Married	…	Exec_man	Husband	White	Male	United_Sta	**poor**
39	Private	HS_grad	Divorced	…	Handlers_	Not_in_fam	White	Male	United_Sta	**poor**
54	Private	11th	Married	…	Handlers_	Husband	Black	Male	United_Sta	**poor**
28	Private	Bachelors	Married	…	Prof_speci	W ife	Black	Female	Cuba	**poor**
38	Private	Masters	Married	…	Exec_man	W ife	White	Female	United_Sta	**poor**
50	Private	9th	Mar-	…	Other_serv	Not_in_fam	Black	Female	Jamaica	**poor**
52	Self_emp_	HS_grad	Married	…	Exec_man	Husband	White	Male	United_Sta	**rich**
31	Private	Masters	Nev-	…	Prof_speci	Not_in_fam	White	Female	United_Sta	**rich**
42	Private	Bachelors	Married	…	Exec_man	Husband	White	Male	United_Sta	**rich**
37	Private	Some_coll	Married	…	Exec_man	Husband	Black	Male	United_Sta	**rich**
30	State_gov	Bachelors	Married	…	Prof_speci	Husband	Asian	Male	India	**rich**
24	Private	Bachelors	Nev-	…	Adm_cleri	Own_child	White	Female	United_Sta	**poor**
33	Private	Assoc_ac	Nev-	…	Sales	Not_in_fam	Black	Male	United_Sta	**poor**
41	Private	Assoc_voc	Married	…	Craft_repai	Husband	Asian	Male	*MissingV	**rich**
34	Private	7th_8th	Married	…	Transport_	Husband	Amer_I	Male	Mexico	**poor**
26	Self_emp_	HS_grad	Nev-	…	Farming_fi	Own_child	White	Male	United_Sta	**poor**
33	Private	HS_grad	Nev-	…	Machine_o	Unmarried	White	Male	United_Sta	**poor**
38	Private	11th	Married	…	Sales	Husband	White	Male	United_Sta	**poor**
44	Self_emp_	Masters	Divorced	…	Exec_man	Unmarried	White	Female	United_Sta	**rich**
41	Private	Doctorate	Married	…	Prof_speci	Husband	White	Male	United_Sta	**rich**
:	:	:	:	:	:	:	:	:	:	:

Tabelle 1-2: 48'842 Datensätze mit 16 Attributen

[5] http://de.wikipedia.org/wiki/Kontingenztafel (4.12.12)
[6] http://archive.ics.uci.edu/ml/ (4.12.12)

Die verwendeten 16 Attribute sind in diesem Fall:

age	edunum	race	hours_worked
employment	marital	gender	country
taxweighting	job	capitalgain	wealth
education	relation	capitolloss	agegroup

Tabelle 1-3: rot = reelle Zahlen, blau = kategorische Werte (symbolische Attribute)

Zur Analyse der Datensätze, können dazugehörige Histogramme betrachtet werden.

Geschlecht	Anzahl
Female	16192
Male	32650

Tabelle 1-4: Geschlecht

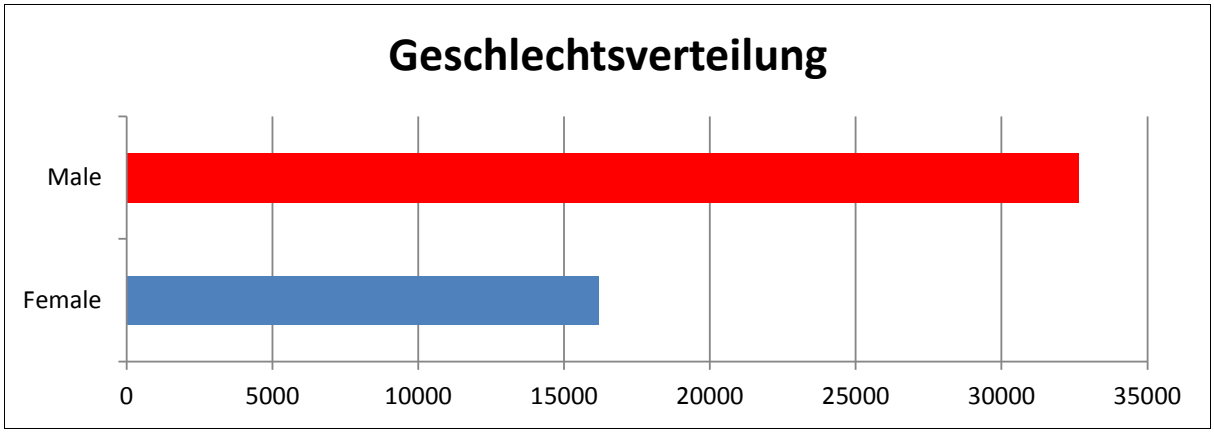


Abbildung 1-5: Histogramm zur Geschlechtsverteilung

Attribut	Anzahl
Divorced	6633
Married_AF_spouse	32650
Married	22379
Married_spouse_absent	628
Never_married	16117
Seperated	1530
Widowed	1518

Tabelle 1-5: Zivilstand

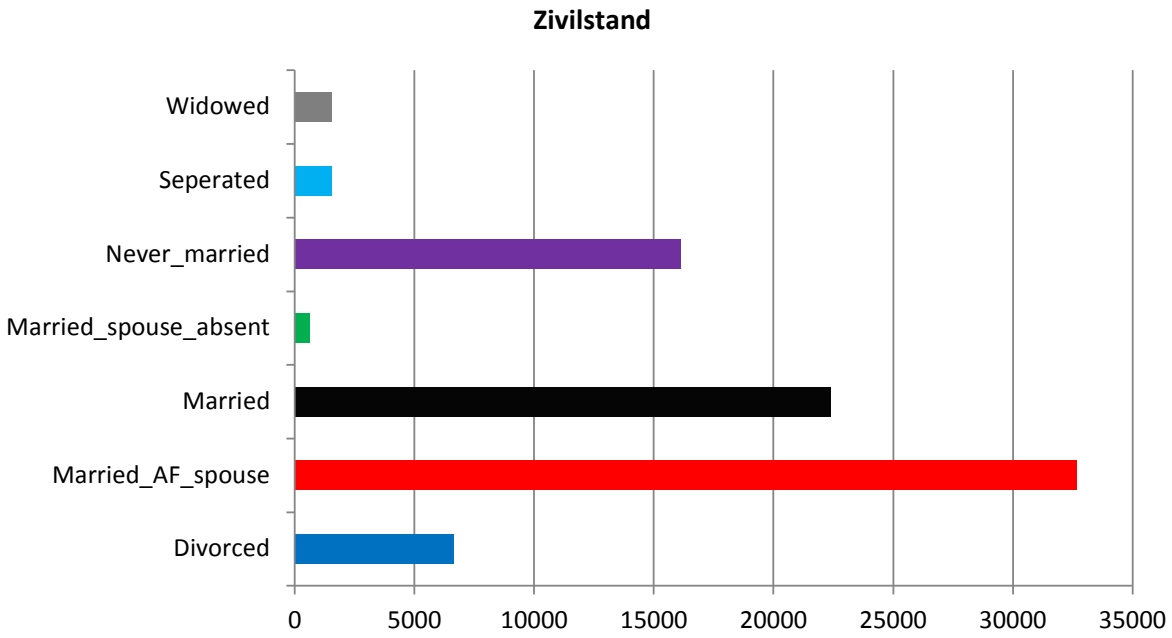


Abbildung 1-6: Histogramm zum Zivilstand

Diese Histogramme werden auch als 1-dimensionale Kreuztabelle (Contingency Table) bezeichnet. Um eine k-dimensionale Kreuztabelle zu erzeugen, wird das folgende Rezept verwendet:

1. Wählen Sie k Attribute aus Ihrem Datensatz aus. Benennen Sie sie a_1, $a_{2, \ldots} a_k$.
2. Für jede mögliche Kombination von Werten, $a_{1'}=x_{1,} a_{2'}=x_{2,}\ldots a_{k'}=x_k$, zeichnen Sie auf, wie häufig die Kombination vorkommt.

Ein Datenbankspezialist würde dies als einen „k-dimensionalen Datenkubus“ bezeichnen.

Nun betrachten wir eine 2-dimensionale Kreuztabelle. Für jedes Wertepaar der Attribute Altersgruppe und Wohlstand sehen wir, wie viele Datensätze in welchem Bereich zu liegen kommen.

wealth values		poor	rich
agegroup	10s	2507	3
	20s	11262	743
	30s	9468	3461
	40s	6738	3986
	50s	4110	2509
	60s	2245	807
	70s	668	147
	80s	115	16
	90s	42	13

Tabelle 1-6: Wohlstandsverteilung nach Altersgruppen

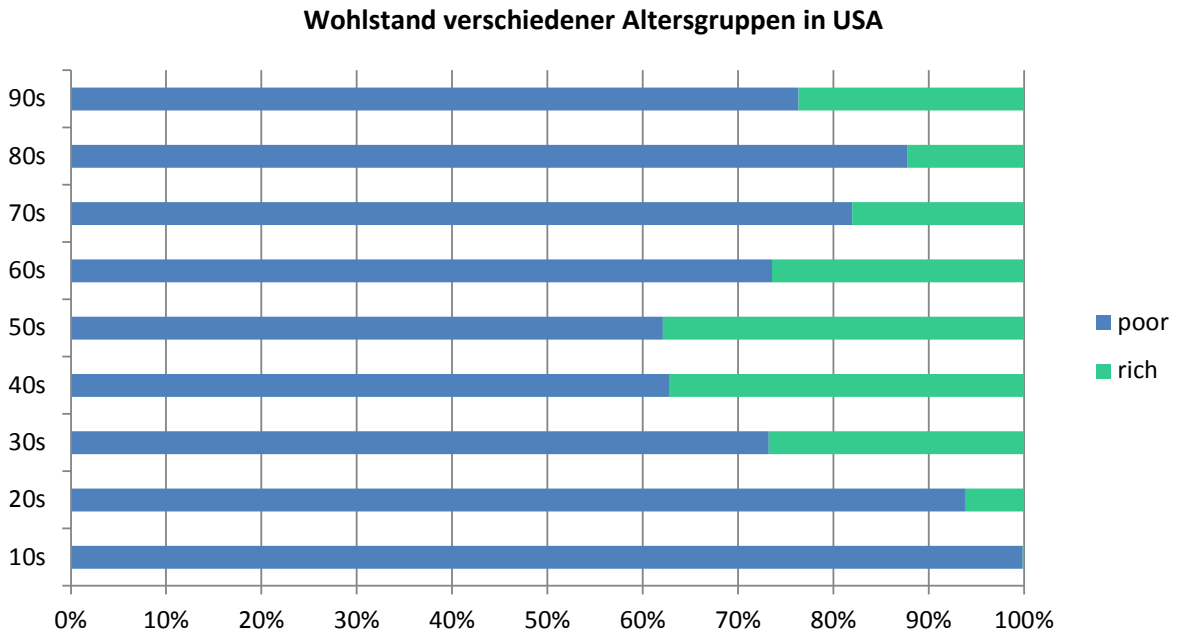


Abbildung 1-7: Histogramm zu Wohlstandsverhältnisse in Abhängigkeit von Altersgruppen

Im Fall dieser Kreuztabelle sehen wir, dass eine entsprechend gewählte Darstellungsform einen Erkenntnisgewinn vereinfachen kann.

Aufgabe 1-2: Betrachten Sie die Tabelle 1-6 bzw. Abbildung 1-7 genau. Welche Schlüsse können für die verschiedenen Altersbereiche gezogen werden? Zur Erinnerung: Die Datensätze stammen aus den USA.

Eine umfassendere 2-dimensionale Kohärenztabelle ist in der folgenden Tabelle dargestellt.

Job:		*MissingValue*	Adm_clerical	Armed_Forces	Craft_repair	Exec_managerial	Farming_fishing	Handlers_cleaners	Machi-ne_op_inspct	Priv_house_serv	Prof_speciality	Protective_serv	Sales	Tech_support	Transport_moving
marital	Divorced	270	1192	0	679	890	90	197	434	46	795	121	664	239	254
	Married_AF_spouse	5	6	0	4	3	1	1	1	0	4	1	5	0	1
	Married	928	1495	7	3818	3600	869	724	1469	27	3182	583	2491	609	1489
	Married_spouse_absent	45	84	0	77	52	35	32	37	9	64	7	55	9	30
	Never_married	1242	2360	8	1301	1260	434	1029	872	99	1849	237	1992	506	486
	Seperated	97	224	0	160	126	23	63	123	21	145	23	146	48	56
	Widowed	222	250	0	73	73	38	26	86	40	133	11	151	35	39

Tabelle 1-7: Grössere 2-dimensionale Kohärenztabelle

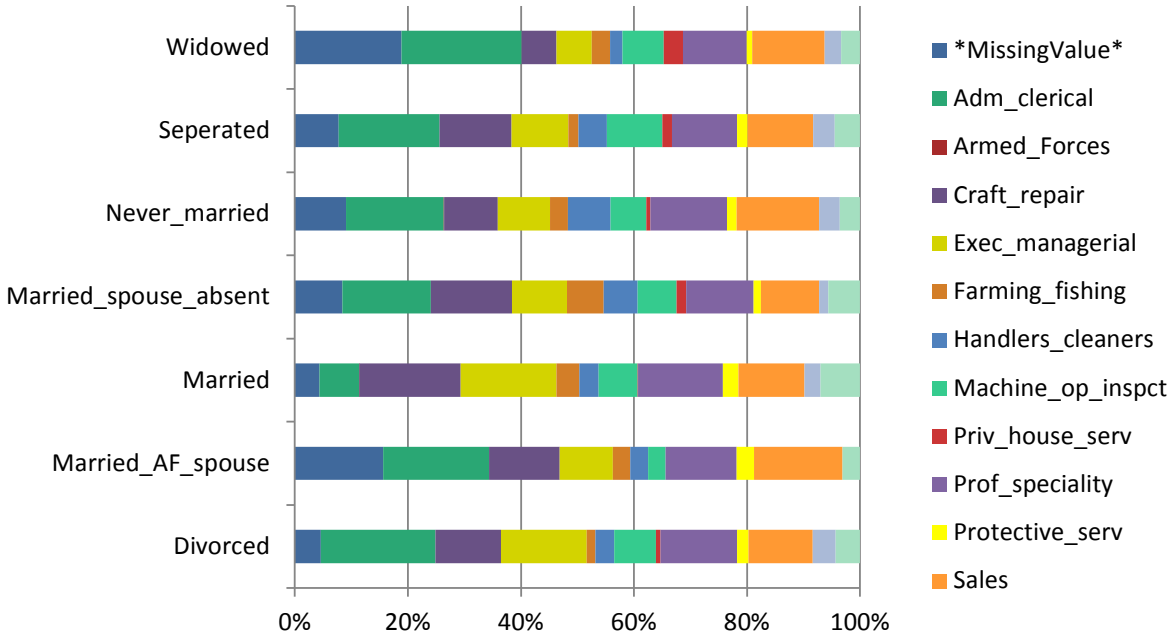


Abbildung 1-8: Histogramm zu Zivilstand und beruflichem Hintergrund

Wenn wir noch einen Schritt weiter gehen, erhalten wir 3-dimensionale Kohärenztabellen. Diese werden zusehends schwieriger zu interpretieren.

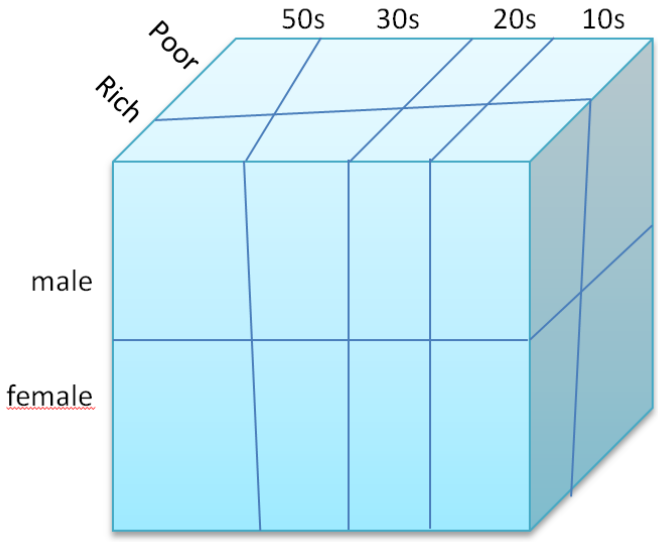


Abbildung 1-9: 3-dimensionale fiktive Kohärenztabelle

Aufgabe 1-3: Wir haben 16 Attribute, a) wie viele 1-dimensionale Kohärenztabellen erhalten wir damit? b) Wie viele 2-dimensionale Kohärenztabellen? c) Wie viele 3-dimensionale Kohärenztabellen? d) Falls wir 100 Attribute hätten, wie viele 3-dimensionale Kohärenztabellen hätten wir? e) Welchen Schluss ziehen Sie in für grössere Kontingenztabellen?

1.5 WAS IST DATA-MINING?

Aufgabe von lernenden Maschinen ist es, Wissen aus Trainingsdaten zu extrahieren. Häufig möchten Programmierer oder Nutzer von lernenden Maschinen das extrahierte Wissen für Menschen zugänglich bzw. verständlich gestalten. Noch interessanter wird es, wenn der Entwickler dieses Wissen sogar verändern kann.

Die Anforderungen aus der Wirtschaftsinformatik und dem Wissensmanagement sind sehr ähnlich. Eine typische Frage aus diesem Bereich ist z.B. in folgender Problematik umschrieben: Ein Betreiber eines Internetshops möchte aus der Nutzungsstatistik seines Shops den Zusammenhang zwischen den Kunden und der für ihn interessanten Klasse von Produkten kennen. Mit diesem Wissen könnte der Anbieter, eine kundenspezifische Werbung anbieten. Ein Paradebeispiel dafür ist der Internetshop von Amazon.

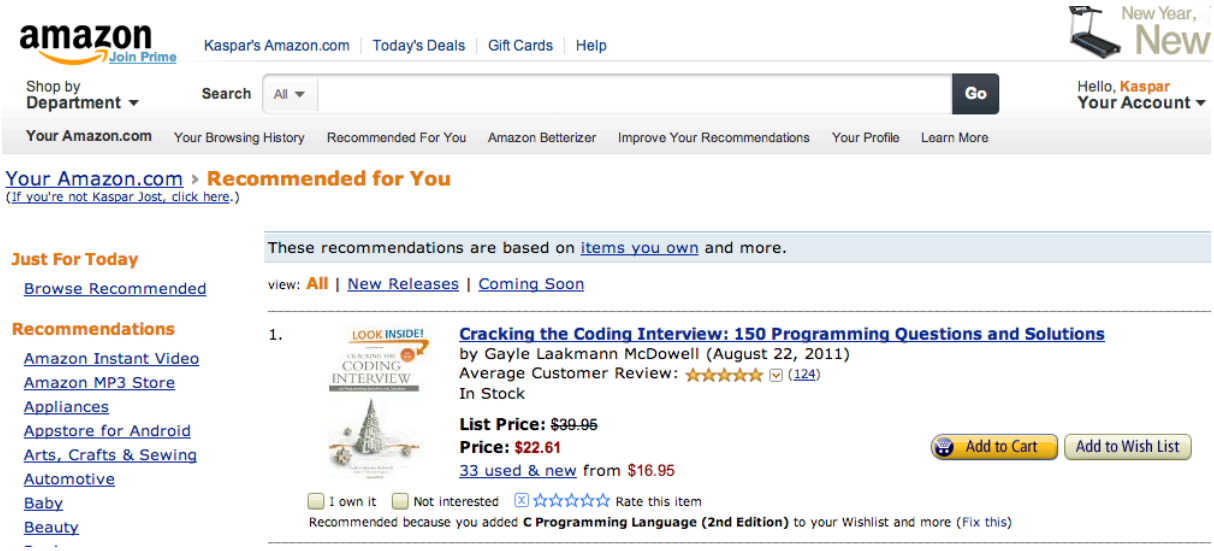


Abbildung 1-10: Personalisierte Kundenwerbung bei Amazon

In Abbildung 1-9 ist erkennbar, wie einem Kunden Produkte vorgeschlagen werden. Diese gleichen jenen Produkten, die er kürzlich auf Amazon betrachtete oder einkaufte. Diverse Bereiche der Werbung, des Marketings und des „Customer Relationship Management (CRM)“ nutzen heutzutage Data Mining. Wann immer grosse Datenmengen zur Verfügung stehen, kann versucht werden, diese zur Analyse der Kundenwünsche zu verwenden, um kundenspezifisch werben zu können.

Der Prozess Wissensgewinns aus Daten sowie dessen Darstellung und Anwendung wird als Data Mining bezeichnet. Die verwendeten Methoden kommen meist aus der Statistik oder der Künstlichen Intelligenz (KI) und sollten auch auf sehr grosse Datenmengen mit vertretbarem Aufwand anwendbar sein. [7]

Bei Internet- bzw. Intranetrecherchen spielt das Text Mining eine immer wichtigere Rolle. Es geht dabei häufig um das Auffinden ähnlicher Texte in Suchmaschinen oder die Klassifikation von Texten, wie sie beispielsweise in Spam-Filtern für Email zum Einsatz kommen.

Die kommerzielle Bedeutung von Data Mining Techniken bringt eine grosse Menge potenter Data Mining-Systeme mit sich. Diese bieten Anwendern diverse Instrumente zur Extraktion des Wissens aus Daten. Ein solches Instrument werden Sie im Rahmen dieser Unterrichtssequenz kennen lernen!

[7] Grundkurs Künstliche Intelligenz (2009), Wolfgang Ertel, S. 185

1.6 DATA MINING IM ÜBERBLICK

Ausgehend von einer Sammlung von Datensätzen wird mittels eines Data Mining Algorithmus versucht, ein Modell zu entwickeln. Falls kein Zielattribut vorkommt, wird unüberwachtes Lernen zum Einsatz kommen. Im Fall, dass ein Zielattribut gegeben ist (es ist bekannt, welches Auto umweltfreundlich ist) kommt das überwachte Lernen zum Einsatz.

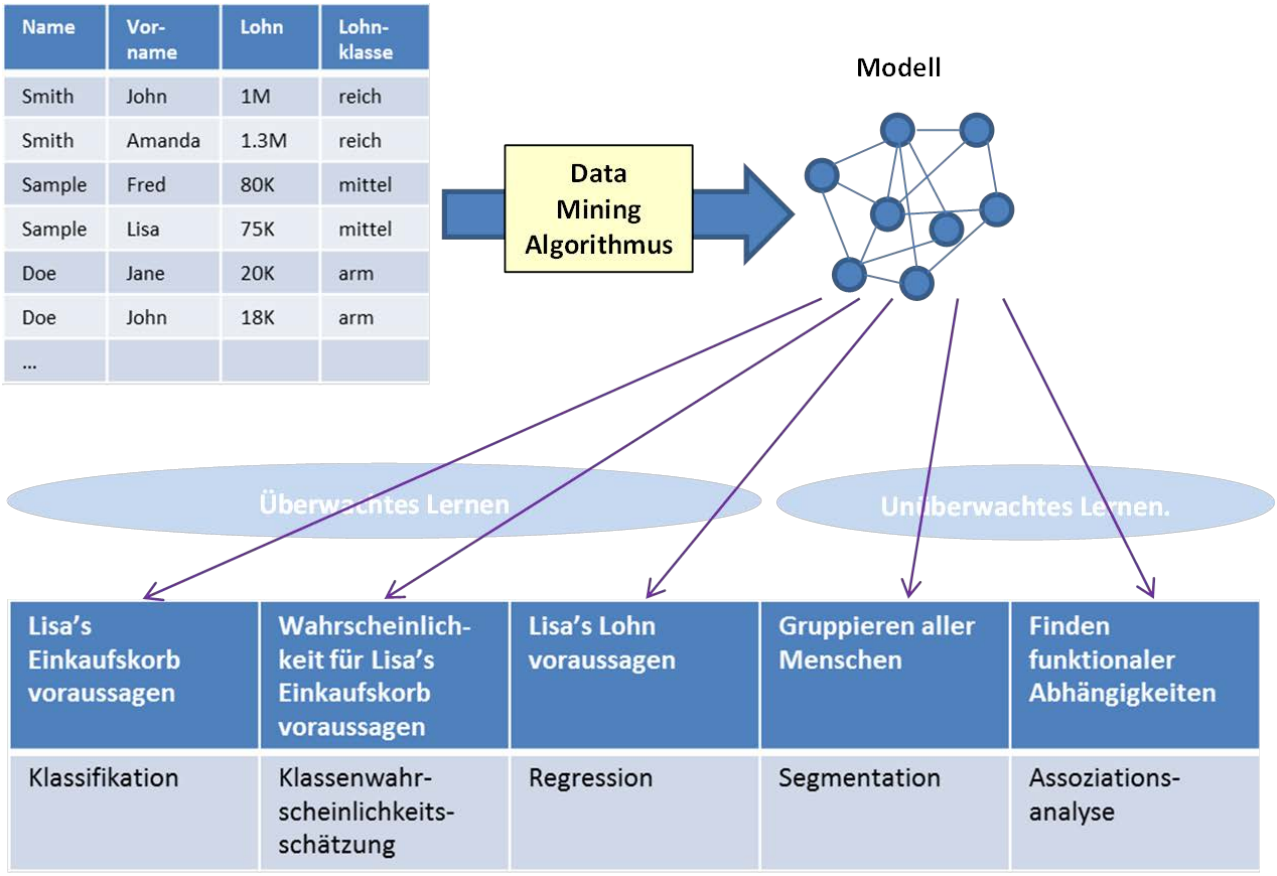


Abbildung 1-11: Data Mining

1.6.1 ÜBERWACHTES LERNEN:

Man gibt ein Zielattribut vor wie, der Kunde ist gut oder schlecht, d.h. wir geben die Qualität vor. Methoden des Überwachten Lernens sind die Klassifikation (Entscheidungsbaum, Bayes Classifier), die Regression und die Klassenwahrscheinlichkeitsschätzung (Bayes).

1.6.2 UNÜBERWACHTES LERNEN:

Es gibt keine Zielattribute. Das heisst, wir benötigen vorhergehende Werte. Methoden des Unüberwachten Lernen sind die Segmentation (Clustering) und die Assoziationsanalyse (Association Rule Mining). Mit der Segmentation werden Zusammenhänge zwischen den Zeilen, d.h. zwischen den Einkaufskörben, gesucht. Bei der Assoziationsanalyse wird nach Zusammenhängen zwischen den Attributen (Kolonnen) gesucht. Das heisst, man möchte z.B. herausfinden, was gemeinsam gekauft wird.

Bei der Datenaufbereitung werden Objekte Gruppen zugeordnet. Diese Gruppen werden als Klassen bezeichnet.

1.6.3 TRAININGS-SET UND TEST-SET

Beim Data Mining werden grosse Datenbestände in ein Training-, ein Validations- und ein Test-Set unterteilt.

- Mit dem Trainings-Set wird versucht ein Model zu entwickeln.
- Das Validations-Set wird verwendet, um zu überprüfen, wie gut das entwickelte Modell funktioniert. Das Validations-Set besteht aus Daten deren Resultate bereits bekannt sind. So können die mit dem Model erhaltenen Resultate mit den bereits bekannten des Validations-Sets verglichen werden.

Ein Test-Set wird verwendet, um herauszufinden, wie gut ein Modell in der Praxis funktioniert, wenn es mit reellen Datenbeständen konfrontiert würde.

Training Set

Id	Attribut1	Attribut2	Attribut3	Klasse
1	ja	gross	125k	nein
2	nein	mittel	100k	nein
3	nein	klein	70k	nein
4	ja	mittel	120k	nein
5	nein	gross	95k	ja
6	nein	mittel	60k	nein
7	ja	gross	220k	nein
8	nein	klein	85k	ja
9	nein	mittel	75k	nein
10	nein	klein	90k	ja

Test Set

Id	Attribut1	Attribut2	Attribut3	Klasse
1	nein	klein	55k	?
2	ja	mittel	80k	?
3	nein	gross	110k	?
4	nein	klein	95k	?
5	nein	gross	67k	?

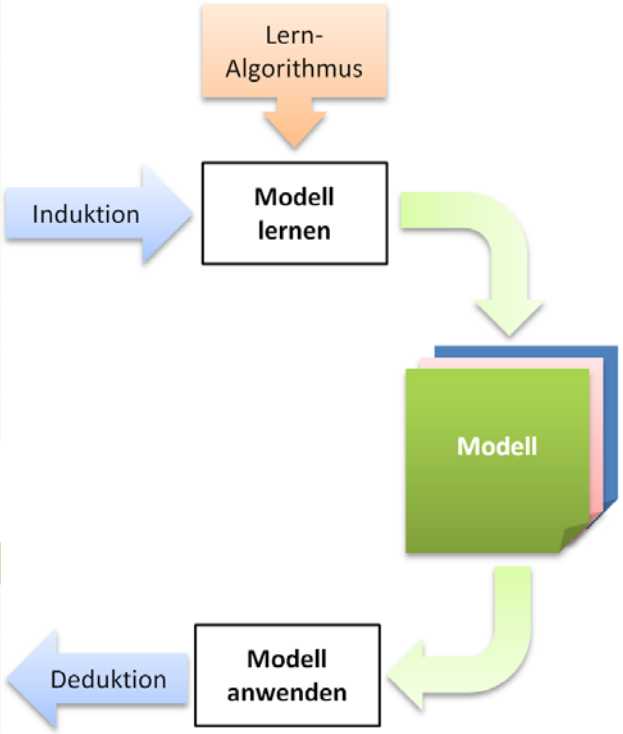


Abbildung 1-12: Training Set und Test Set

2 KLASSIFIZIERUNG – 1R

Eine simple aber effektive Methode um aus Daten Information zu gewinnen stellt das *1-Regel-Verfahren* (engl. *1-Rule*) dar, welches üblicherweise einfach als 1R bezeichnet wird. Trotz der Einfachheit von 1R, liefert die Methode in vielen Fällen gut funktionierende Regeln um die Struktur von Daten zu beschreiben. Es ist übrigens gar nicht so selten, dass die einfachen Methoden beim Data Mining oft erstaunlich gute Resultate liefern. Dieser Umstand beruht wohl darauf, dass die Struktur realer Datensätzen häufig so einfach ist, dass ein einziges Attribut ausreicht, um einzelne Instanzen (= Eintrag im Datensatz) mit guter Genauigkeit einer Klasse zuordnen zu können. Es lohnt sich also auf jeden Fall, das Einfache zuerst zu probieren!

2.1 DAS WETTER-PROBLEM

Die Funktionsweise von 1R soll im Folgenden an einem konkreten Beispiel, dem «Wetter-Problem» erläutert werden. Dabei handelt es sich um einen kleinen Datensatz, welcher Auskunft darüber gibt, ob unter bestimmten Wetterbedingungen ein Anlass durchgeführt werden kann oder nicht.

Da das Wetter-Problem über nur gerade 14 Instanzen (Einträge) verfügt und damit gut überschaubar ist, wird es gerne dazu verwendet, die Funktionsweise verschiedener Data-Mining-Verfahren zu untersuchen und miteinander zu vergleichen. Der Datensatz für das Wetter-Problem hat folgenden Inhalt:

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	hoch	schwach	nein
2	sonnig	heiss	hoch	stark	nein
3	bewoelkt	heiss	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kalt	normal	schwach	ja
6	regnerisch	kalt	normal	stark	nein
7	bewoelkt	kalt	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kalt	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bewoelkt	mild	hoch	stark	ja
13	bewoelkt	heiss	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Tabelle 2-1: Datensatz zum Wetter-Problem

Alle 14 **INSTANZEN** des Wetter-Problems verfügen über die vier **ATTRIBUTE** *Wetter*, *Temperatur*, *Luftfeuchtigkeit* und *Wind*. Die letzte Spalte jeder Instanz ist mit *Anlass* bezeichnet und enthält die Information, ob der Anlass unter den beschriebenen Bedingungen stattfinden kann oder nicht. In dieser Spalte wird also die Zuweisung zu den beiden **KLASSEN** *ja* (der Anlass findet statt) respektive *nein* (der Anlass findet nicht statt) vollzogen.

2.2 ERSTELLEN VON REGEL-SETS MIT 1R

Beim 1R-Verfahren werden die einzelnen Attribute der Reihe nach durchgegangen. Innerhalb eines Attributs wird für jeden Attribut-Wert eine Regel erstellt, bei welcher die am häufigsten vorkommende Klasse dem Attribut-Wert zugewiesen wird. Anschliessend wird für alle Regeln die Fehlerquote bestimmt. Das Regel-Set mit der kleinsten Fehlerquote gewinnt!

Das 1R-Verfahren lässt sich mit Hilfe von Pseudo-Code also wie folgt beschreiben:

```
Für jedes Attribut,
   für jeden Wert dieses Attributs, erstelle eine Regel wie folgt:
      zähle wie oft jede Klasse vorkommt
      finde die häufigste Klasse
      erstelle eine Regel, welche diese Klasse dem aktuellen Attribut-Wert zuordnet.
   Berechne die Fehlerquote aller Regeln.
Wähle das Regel-Set mit der kleinsten Fehlerquote.
```

Angewendet auf den Datensatz zum «Wetter-Problem», würde 1R im ersten Schritt also das Attribut *Wetter* auswählen, welches über die Attribut-Werte *sonnig*, *bewoelkt* und *regnerisch* verfügt. Die erste Regel würde somit für den Attribut-Wert *sonnig* erstellt werden, welcher insgesamt 5 Mal vorkommt und dabei 2 Mal in die Klasse *ja* (der Anlass findet statt) und 3 Mal in die Klasse *nein* (der Anlass findet nicht statt) entfällt. Da der Anlass in der Mehrzahl der Fälle (3 von 5) bei sonnigem Wetter nicht stattfindet, ergibt sich die Regel: *sonnig → nein*. Allerdings gibt es auch Fälle (2 von 5), in denen der Anlass trotz sonnigem Wetter, durchgeführt wird. Die Regel: *sonnig → nein* weist also eine Fehlerquote von 2/5 auf.

Die Tabelle 2-2 zeigt sämtliche Regeln und die zugehörigen Fehlerquoten zum «Wetter-Problem» in der Übersicht:

	Attribut	Regel			Fehlerquote	Totale Fehlerquote
1	Wetter	sonnig	→	nein	2/5	4/14
		bewoelkt	→	ja	0/4	
		regnerisch	→	ja	2/5	
2	Temperatur	heiss	→	nein*	2/4	5/14
		mild	→	ja	2/6	
		kalt	→	ja	1/4	
3	Luftfeuchtigkeit	hoch	→	nein	3/7	4/14
		normal	→	ja	1/7	
4	Wind	stark	→	ja	2/8	5/14
		schwach	→	nein*	3/6	

Tabelle 2-2: Alle Regeln zum Wetter-Problem (* = Zufalls-Wahl zwischen zwei gleichwahrscheinlichen Regeln)

Für den Attribut-Wert *heiss* des Attributs *Temperatur* wird gemäss obiger Tabelle die Regel: *sonnig → nein* erstellt und zwar mit einer Fehlerquote von 2/4. Die Auswahl erfolgt in diesem Fall zufällig, da die Regel: *sonnig → ja* ebenfalls die Fehlerquote 2/4 aufweisen würde. Zwei gleichwahrscheinliche Situationen ergeben sich auch für den Attribut-Wert *schwach* des Attributs *Wind*.

Für jedes Attribut erstellt 1R nun ein Regel-Set, indem es die Regeln mit den geringsten Fehlerquote auswählt. Für das Attribut Wetter ergibt sich damit folgendes Regel-Set:

Wetter: *sonnig → nein* *(Totale Fehlerquote = 4/14 | Genauigkeit = 10/14)*
bewoelkt → ja
regnerisch → ja

Wird dieses Regel-Set auf die 14 Instanzen des «Wetter-Problems» angewendet, werden 10 Instanzen korrekt (✓) und deren 4 falsch (✗) klassifiziert. Wird das Regel-Set also beispielsweise auf die Instanz 6 angewendet, müsste der Anlass gemäss der Regel: *regnerisch → ja* eigentlich stattfinden, was in Realität aber nicht zutrifft. Das Regel-Set liefert in diesem Fall also eine falsche Voraussage, wie auch bei den Instanzen 9, 11 und 14. Die Fehlerquote des Regel-Sets beträgt damit 4/14, respektive es weist eine *Genauigkeit* (engl. *accuracy*) von 10/14 auf.

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass	
1	sonnig	heiss	hoch	schwach	nein	✓
2	sonnig	heiss	hoch	stark	nein	✓
3	bewoelkt	heiss	hoch	schwach	ja	✓
4	regnerisch	mild	hoch	schwach	ja	✓
5	regnerisch	kalt	normal	schwach	ja	✓
6	regnerisch	kalt	normal	stark	nein	✗
7	bewoelkt	kalt	normal	stark	ja	✓
8	sonnig	mild	hoch	schwach	nein	✓
9	sonnig	kalt	normal	schwach	ja	✗
10	regnerisch	mild	normal	schwach	ja	✓
11	sonnig	mild	normal	stark	ja	✗
12	bewoelkt	mild	hoch	stark	ja	✓
13	bewoelkt	heiss	normal	schwach	ja	✓
14	regnerisch	mild	hoch	stark	nein	✗

Tabelle 2-3: Überprüfung des Regel-Sets für das Attribut *Wetter*.

Für die drei Attribute *Temperatur*, *Luftfeuchtigkeit* und *Wind* ergeben sich folgende Regel-Sets:

Temperatur: *heiss → nein* *(Totale Fehlerquote = 5/14 | Genauigkeit = 9/14)*
mild → ja
kalt → ja

Luftfeuchtigkeit: *hoch → nein* *(Totale Fehlerquote = 4/14 | Genauigkeit = 10/14)*
normal → ja

Wind: *stark → ja* *(Totale Fehlerquote = 5/14 | Genauigkeit = 9/14)*
schwach → nein

Die 14 Instanzen des «Wetter-Problems» werden mit dem Regel-Set zum Attribut *Wetter*, oder mit jenem zum Attribut *Luftfeuchtigkeit* am genauesten klassifiziert. Beide Regel-Sets sind mit einer Genauigkeit von 10/14 äquivalent. Die beiden Regel-Sets zu den Attributen *Temperatur* und *Luftfeuchtigkeit* weisen mit 9/14 eine geringere Genauigkeit auf und können daher verworfen werden.

2.3 TESTEN DES REGEL-SETS

Im vorangegangenen Kapitel wurden die 14 Instanzen des «Wetter-Problems» als *Trainingsdaten* für 1R verwendet. Dadurch dass für alle 14 Instanzen klar ist, zu welcher Klasse (*ja* oder *nein*) sie gehören, lassen sich mit Hilfe von 1R, vier verschiedene Regel-Sets erstellen und deren Fehlerquote bestimmen. Das Regel-Set mit der grössten Genauigkeit geht als Sieger aus dem Trainingslauf hervor und wird dann für die Einordnung noch nicht klassifizierter Datensätze verwendet. Das Wetter-Problem liefert zufälligerweise zwei gleich gute Regel-Sets, welche beide eine Genauigkeit von 10/14 aufweisen:

Wetter: *sonnig → nein*
bewoelkt → ja
regnerisch → ja

Luftfeuchtigkeit: *hoch → nein*
normal → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	?
2	regnerisch	heiss	hoch	schwach	?

Tabelle 2-4: Nicht klassifizierter Datensatz

Die beiden Regel-Sets werden nun dazu verwendet, um eine Voraussage darüber zu machen, ob unter den gegebenen Wetterbedingungen in Tabelle 2-4, der Anlass stattfinden kann oder nicht.

Aufgabe 2-1: Klassifizieren Sie mit Hilfe der Regel-Sets zu den Attributen «Wetter» und «Luftfeuchtigkeit» den Datensatz aus der Tabelle 3-4. Vergleichen Sie anschliessend die Resultate miteinander. Was fällt dabei auf?

Wetter: *sonnig → nein*
bewoelkt → ja
regnerisch → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	
2	regnerisch	heiss	hoch	schwach	

Luftfeuchtigkeit: *hoch → nein*
normal → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	
2	regnerisch	heiss	hoch	schwach	

Obwohl die beiden Regel-Sets auf den Trainingsdaten dieselbe Fehlerquote aufweisen, klassifizieren Sie den neuen Datensatz völlig verschieden. Es stellt sich unweigerlich die Frage, welchem der beiden Regel-Set nun eher zu trauen ist. Auf jeden Fall scheint es aber keine gute Idee zu sein, die aus den Trainingsdaten gewonnenen Regel-Sets, ungeprüft für die Klassifizierung neuer Daten verwenden zu wollen. Aber wie lässt sich die Qualität des Regel-Sets vor ihrem Einsatz überprüfen?

Ein möglicher Ansatz ist dabei, die Regel-Sets auf einem Testdatensatz zu evaluieren. Dabei wird von einem klassifizierten Datensatz nur ein Teil als *Trainingsdaten* (= *Trainingsset*) und der Rest als *Testdaten* (= *Testset*) verwendet werden. Natürlich ist man jetzt versucht, so viele Daten wie möglich dem Trainingsset zuzuordnen, da das Regel-Set umso besser ausfällt, je mehr Trainingsdaten zur Verfügung stehen. Allerdings wird in einem solchen Fall die Testdatenmenge zu klein, als dass man mit ihr einen aussagekräftigen Testlauf durchführen könnte. Erfahrungsgemäss ist eine Verteilung optimal, bei welcher von der Gesamtdatenmenge etwa 2/3 für die Trainingsdaten und etwa 1/3 für die Testdaten verwendet werden. Bei der Verteilung der Daten auf die Trainings- resp. Testdaten muss darauf geachtet werden, dass die Klassen in den beiden Datenmengen prozentual etwa gleich häufig vorkommen, wie in der Gesamtdatenmenge. Kommt also die Klasse *ja* im gesamten Datensatz zu 60 % vor, dann sollten auch die Trainings- und die Testdaten einen *ja*-Anteil von 60 % aufweisen.

Wenn nur ein Testlauf durchgeführt wird, lässt sich noch keine zuverlässige Aussage darüber machen, wie gut sich das Regel-Set zur Klassifizierung neuer Daten eignen wird. Um diese Unsicherheit zu minimieren, werden üblicherweise mehrere Trainings- und Testläufe durchgeführt. Dazu werden die vorhandenen Daten zufällig in ein erstes Trainings- resp. Testset verteilt. Anschliessend wird aufgrund der Trainingsdaten ein Regel-Set erstellt, welches mit den Trainingsdaten getestet wird. Die Fehlerquote wird notiert, sämtliche Daten werden vereint, um dann wiederum zufällig in ein weiteres Trainings- resp. Testset verteilt zu werden. Diese Vorgänge werden mehrmals wiederholt, um am Schluss aus den erhaltenen Fehlerquoten den Mittelwert zu berechnen. Damit lässt sich in aller Regel gut abschätzen, mit welcher Genauigkeit das Regel-Set neue Daten klassifizieren wird.

Aufgabe 2-2: *Warum ist folgender Vorschlag eine schlechte Idee?*

„Offensichtlich ist es so, dass ein Algorithmus besser lernen kann, je grösser das Trainingsset ist. Damit wäre es doch für das «Wetter-Problem» eine gute Lösung, wenn sämtliche 20 Einträge zum Trainieren verwendet würden. Um das Regel-Set zu testen, könnten dann von denselben 20 Einträgen doch einfach 6 zufällig ausgewählt werden.“

Nehmen wir einmal an, dass für das «Wetter-Problem» ein Datensatz von 20 klassifizierten Instanzen vorhanden wäre. Davon würden dann beispielsweise 14 Instanzen als Trainingsdaten und die restlichen 6 Instanzen als Testdaten zur Überprüfung des gelernten Regel-Sets verwendet werden.

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	hoch	schwach	nein
2	sonnig	heiss	hoch	stark	nein
3	bewoelkt	heiss	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kalt	normal	schwach	ja
6	regnerisch	kalt	normal	stark	nein
7	bewoelkt	kalt	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kalt	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bewoelkt	mild	hoch	stark	ja
13	bewoelkt	heiss	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Tabelle 2-5: Trainingsdaten zum Wetter-Problem

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
15	sonnig	mild	hoch	stark	nein
16	bewoelkt	mild	normal	schwach	nein
17	bewoelkt	kalt	hoch	stark	nein
18	bewoelkt	heiss	normal	stark	ja
19	regnerisch	heiss	normal	stark	ja
20	regnerisch	heiss	hoch	schwach	nein

Tabelle 2-6: Testdaten zum Wetter-Problem

Im ersten Schritt wird 1R ausschliesslich auf den Trainingsdaten trainiert und das erhaltene Regel-Set danach mit den Testdaten getestet. Da letztere ebenfalls bereits klassifiziert sind, kann überprüft werden, ob das Regel-Set das korrekte Ergebnis (✓) liefert oder nicht (✗). Wird beispielsweise das Regel-Set zum Attribut *Wetter* auf die Testdaten angewendet ergibt sich folgendes Resultat.

Wetter: *sonnig → nein*
bewoelkt → ja
regnerisch → ja

(Totale Fehlerquote = 3/6 | Genauigkeit = 3/6)

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass	Wetter Regel-Set	
15	sonnig	mild	hoch	stark	nein	nein	✓
16	bewoelkt	mild	normal	schwach	nein	ja	✗
17	bewoelkt	kalt	hoch	stark	nein	ja	✗
18	bewoelkt	heiss	normal	stark	ja	ja	✓
19	regnerisch	heiss	normal	stark	ja	ja	✓
20	regnerisch	heiss	hoch	schwach	nein	ja	✗

Tabelle 2-7: Überprüfung des Wetter Regel-Sets mit Hilfe der Testdaten

Aufgabe 2-3: Wenden Sie das Regel-Set zum Attribut Luftfeuchtigkeit auf die Testdaten an und vervollständigen Sie untenstehende Tabelle. Bestimmen Sie anschliessend die Fehlerquote resp. Genauigkeit und vergleichen Sie Ihre Resultate mit der Tabelle 2-7.

Luftfeuchtigkeit: hoch → nein (Totale Fehlerquote = ... | Genauigkeit = ...)
normal → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass	Luftf. Regel-Set
15	sonnig	mild	hoch	stark	nein	
16	bewoelkt	mild	normal	schwach	nein	
17	bewoelkt	kalt	hoch	stark	nein	
18	bewoelkt	heiss	normal	stark	ja	
19	regnerisch	heiss	normal	stark	ja	
20	regnerisch	heiss	hoch	schwach	nein	

Tabelle 2-8: Überprüfung des Luftfeuchtigkeit Regel-Sets mit Hilfe der Testdaten

2.4 OVERFITTING (ÜBERANPASSUNG)

Wenn ein gelerntes Regel-Set gut auf den Trainingsdaten, aber schlecht auf den Testdaten abschneidet, ist dies ein Zeichen für *Overfitting* (engl. für *Überanpassung*). Dabei hat sich der Algorithmus beim Lernen zu stark an die Trainingsdaten angepasst, indem er beispielsweise Teile des Datensatzes übermässig gewichtet hat, welche für die korrekte Klassifizierung der einzelnen Daten aber irrelevant sind - der Algorithmus lernt also sozusagen das Hintergrundrauschen.

Im Falle des «Wetter-Problems» kommt es offensichtlich zu einem Overfitting, wenn Trainings- und Testdaten so unausgewogen zusammengestellt werden wie oben (Tabelle 2-5 und 2-6). Die Trainingsdaten sind in diesem Fall nicht repräsentativ, da die prozentuale Verteilung der beiden Klassen *ja* und *nein* innerhalb der Trainings- und der Testdaten nicht derjenigen des Gesamtdatensatzes entspricht. In den Trainingsdaten ist die Klasse *ja* (64 anstatt 55 %) und in den Testdaten die *nein* (67 anstatt 45 %) übervertreten.

	Trainingsdaten	**Testdaten**	**Total**
ja	9 (64 %)	2 (33 %)	11 (55 %)
nein	5 (36 %)	4 (67 %)	9 (45 %)

Tabelle 2-9: Prozentuale Verteilung der beiden Klassen auf die Trainings- und Testdaten

2.5 CROSSVALIDATION (KREUZVALIDIERUNG)

Ein sehr häufig angewendetes Verfahren, um einerseits zuverlässige Regel-Sets zu erstellen und gleichzeitig deren Fehlerquote auf unklassifizierten Datensätzen abschätzen zu können, ist das *Crossvalidation*-Verfahren. Dabei wird ein gegebener Datensatz zufällig in eine bestimmte Anzahl *Folds* (Untereinheiten) unterteilt. Alle Folds müssen in etwa gleich gross sein und eine ausgewogene Verteilung der verschiedenen Klassen aufweisen. Üblicherweise werden 10 Folds verwendet, wobei jeweils 1 Fold für das Testset und 9 Folds für das Trainingsset verwendet wird. Es wird nun der Reihe nach auf 9 Folds trainiert (hellblau) und das Resultat auf dem ausgelassenen Fold (dunkelblau) getestet. Im nächsten Durchgang wird ein anderer Fold für Testzwecke reserviert, auf den restlichen 9 Folds trainiert und erneut die Fehlerquote bestimmt. Nach insgesamt 10 Durchgängen, lässt sich auf diese Weise das beste Regel-Set bestimmen und eine ziemlich gute Voraussage über dessen Fehlerquote auf zukünftigen Datensätzen, machen.

Abbildung 2-1: Vorgehen bei der 10-Fold Crossvalidation (dunkelblau = Testdaten | hellblau = Trainingsdaten)

2.6 NUMERISCHE ATTRIBUTE

Der bisher verwendete Datensatz zum «Wetter-Problem» enthielt noch keine Zahlen, sondern verwendete ausschliesslich Text (Strings) zur Angabe der Temperatur und der Luftfeuchtigkeit. Was macht nun aber 1R, wenn diese beiden Attribute mit Zahlenwerten angegeben werden, wie dies in Tabelle 2-10 der Fall ist?

	Wetter	Temperatur [°C]	Luftfeuchtigkeit [%]	Wind	Anlass
1	sonnig	31	85	schwach	nein
2	sonnig	26	90	stark	nein
3	bewoelkt	29	86	schwach	ja
4	regnerisch	16	96	schwach	ja
5	regnerisch	14	80	schwach	ja
6	regnerisch	11	70	stark	nein
7	bewoelkt	10	65	stark	ja
8	sonnig	18	95	schwach	nein
9	sonnig	15	70	schwach	ja
10	regnerisch	21	80	schwach	ja
11	sonnig	21	70	stark	ja
12	bewoelkt	18	90	stark	ja
13	bewoelkt	27	75	schwach	ja
14	regnerisch	17	91	stark	nein

Tabelle 2-10: Datensatz zum Wetter-Problem mit numerischen Attributen für die Temperatur und die Luftfeuchtigkeit

Das Attribut *Wetter* wird von 1R verarbeitet wie bisher und auch das resultierende Regel-Set entspricht jenem des vorangehenden Kapitels:

Wetter: *sonnig → nein*
bewoelkt → ja
regnerisch → ja

(Totale Fehlerquote = 4/14 | Genauigkeit = 10/14)

Würde 1R bei der Erstellung des Regel-Sets zum Attribut *Temperatur* verfahren wie bisher, so würden für die 12 verschiedenen Temperaturwerte (18 und 21 kommen je zweimal vor) auch 12 verschiedene Regeln erstellt werden. Da für die Temperatur 18 zwei Einträge existieren, welche in verschiedene Klassen entfallen, müssten die Einträge zufällig in eine der beiden Klassen eingeteilt werden, z.B. in die Klasse *ja*. Das Regel-Set zum Attribut Temperatur würde demnach wie folgt aussehen:

Temperatur:	*10*	*11*	*14*	*15*	*16*	*17*	*18*	*21*	*26*	*27*	*29*	*31*
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	ja	*nein*	*ja*	*ja*	*ja*	*nein*	*ja**	*ja*	*nein*	*ja*	*ja*	*nein*

(Totale Fehlerquote = 1/14 | Genauigkeit = 13/14)

Bis auf eine der beiden Instanzen mit der Temperatur 18, welche aufgrund der Zufallswahl von oben gezwungenermassen in die falsche Klasse eingeteilt würde, würden alle anderen Instanzen korrekt klassifiziert und das Regel-Set hätte damit eine Fehlerquote von nur gerade 1/14.

Die Genauigkeit ist allerdings trügerisch. Es ist zwar so, dass das erhalten Regel-Set die 14 Instanzen des «Wetter-Problems» praktisch fehlerfrei zuordnet, allerdings passt sich der Algorithmus dabei viel zu stark an die Trainingsdaten an. Er würde bei der Klassifizierung von Testdaten (oder auch von neuen Daten) kläglich versagen – ein klassischer Fall von Overfitting.

Sinnvoller wäre es daher, wenn die Werte für die Temperatur in Temperatur-Intervalle unterteilt werden könnten und 1R für diese Temperatur-Intervalle eruieren würde, ob der Anlass stattfindet oder nicht. Dies ist exakt die Vorgehensweise, welche bei 1R zum Zuge kommt und zwar sortiert der Algorithmus dazu die Werte für die Attribute in aufsteigender Reihenfolge und setzt dann überall dort, wo sich die Klasse ändert, eine Intervallgrenze. Für das Attribut *Temperatur* würde diese Unterteilung wie folgt aussehen:

Temperatur:	*10*	*11*	*14*	*15*	*16*	*17*	*18*	*18*	*21*	*21*	*26*	*27*	*29*	*31*
	ja	*nein*	*ja*	*ja*	*ja*	*nein*	*nein*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*

Die Intervallgrenzen würden dabei jeweils in der Mitte, also bei 10.5, 12.5, 16.5, 18.0, 23.5, 26.5 und 30.0 zu liegen kommen. Allerdings ergibt sich in diesem Fall das Problem, dass die beiden Instanzen mit der Temperatur 18 in zwei verschiedene Klassen entfallen. Eine Lösung dafür wäre, die Intervallgrenze bei 18 um eins nach rechts zu schieben. Man würde dadurch die Gruppe 17, 18, 18 erhalten, welche der Klasse *nein* zugewiesen würde. Das weitaus grössere Problem ist aber der Umstand, dass oben sieben Intervallgrenzen identifiziert wurden, welche alle durch eine Regel beschrieben werden müssen. Diese Zahl ist viel zu hoch und führt zu einem Overfitting.

Aus diesem Grund lässt sich 1R beim Lernen ein Parameter (*minBucketSize*) übergeben, welcher das Minimum angibt, mit welchem die häufigste vorkommende Klasse in einem Intervall vertreten sein muss. Wird *minBucketSize* = 3 gewählt, verteilen sich die Intervallgrenzen wie folgt:

Temperatur:	*10*	*11*	*14*	*15*	*16*	*17*	*18*	*18*	*21*	*21*	*26*	*27*	*29*	*31*
	ja	*nein*	*ja*	*ja*	*ja*	*nein*	*nein*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*

In jedem Intervall kommt also mindestens eine Klasse 3 Mal vor, in unserem Fall ist es jeweils die Klasse *ja*. Die beiden Intervallgrenzen machen allerdings wenig Sinn, da sie jeweils Instanzen trennen, welche in dieselbe Klasse entfallen würden. Die Intervallgrenzen werden daher so weit nach rechts verschoben, bis sie auf eine andere Klasse stossen:

Temperatur:	*10*	*11*	*14*	*15*	*16*	*17*	*18*	*18*	*21*	*21*	*26*	*27*	*29*	*31*
	ja	*nein*	*ja*	*ja*	*ja*	*nein*	*nein*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*

Im letzten Schritt werden alle Intervallgrenzen aufgehoben, welche Intervalle mit derselben Hauptklasse trennen. Unter der Hauptklasse versteht man jene Klasse, die in einem Intervall am häufigsten vorkommt. In den beiden ersten Intervallen ist dies die Klasse *ja*, was dazu führt, dass die beiden Intervalle fusioniert werden:

Temperatur:	*10*	*11*	*14*	*15*	*16*	*17*	*18*	*18*	*21*	*21*	*26*	*27*	*29*	*31*
	ja	*nein*	*ja*	*ja*	*ja*	*nein*	*nein*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*

Als Hauptklasse wurde in diesem Fall für das Intervall ganz rechts zufällig *nein* gewählt, ansonsten wäre auch die letzte Intervallgrenze aufgehoben worden. Dies ist ein Zeichen dafür, dass sich das Attribut *Temperatur* eher schlecht für die Klassifizierung dieses Datensatzes eignet.

Aus diesen Trainingsdaten würde 1R mit *minBucketSize* = 3 folgendes Regel-Set für das Attribut *Temperatur* vorschlagen:

Temperatur: *< 23.5 → ja* *(Totale Fehlerquote = 5/14 | Genauigkeit = 9/14)*
≥ 23.5 → nein

Aufgabe 2-4: *Tragen Sie unten schrittweise die Intervallgrenzen ein, welche 1R für das Attribut Luftfeuchtigkeit erstellen würde, wenn minBucketSize = 3 ist. Geben Sie anschliessend das Regel-Set und dessen Fehlerquote an.*

Schritt 1:

Luftfeuchtigkeit:	*65*	*70*	*70*	*70*	*75*	*80*	*80*	*85*	*86*	*90*	*90*	*91*	*95*	*96*
	ja	*nein*	*ja*	*ja*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*	*nein*	*nein*	*ja*

Schritt 2:

Luftfeuchtigkeit:	*65*	*70*	*70*	*70*	*75*	*80*	*80*	*85*	*86*	*90*	*90*	*91*	*95*	*96*
	ja	*nein*	*ja*	*ja*	*ja*	*ja*	*ja*	*nein*	*ja*	*ja*	*nein*	*nein*	*nein*	*ja*

Regelset:

Luftfeuchtigkeit:

Fehlerquote:

2.7 DATA-MINING MIT WEKA

Bisher haben wir sämtliche unserer Überlegungen mit Bleistift und Papier angestellt und das ist auch gut so, wenn man ein Gefühl für die Vorgänge entwickeln will, welche bei der Klassifizierung mit 1R ablaufen. Allerdings haben wir uns mit dem «Wetter-Problem» bisher nur mit einem sehr kleinen Datensatz beschäftig. Viele Datensätze im realen Leben sind jedoch wesentlich grösser und von Hand nicht mehr zu bewältigen. Aus diesem Grund wurden über die letzten Jahre hinweg diverse Software-Tools entwickelt, mit welchen sich typische Data-Mining - Aufgaben automatisieren lassen. Dazu gehört auch **WEKA**, ein Open-Source Programm welches an der Universität Waikato in Neu-Seeland entwickelt wurde: http://www.cs.waikato.ac.nz/ml/weka/

Im Folgenden wird die Version 3.7 von Weka verwendet. Damit Weka ausgeführt werden kann, muss auf dem Computer Java 6 oder höher installiert sein. Sollte dies nicht der Fall sein, lässt sich auch eine Weka-Version herunterladen, welche Java im Bundle mit sich bringt. Die verschiedenen Versionen von Weka lassen sich unter folgenden URLs beziehen:

Windows (inkl. Java):	http://prdownloads.sourceforge.net/weka/weka-3-7-7jre.exe
Windows (ohne Java):	http://prdownloads.sourceforge.net/weka/weka-3-7-7.exe
Mac OS X:	http://prdownloads.sourceforge.net/weka/weka-3-7-7.dmg

Nach der Installation und dem Start des Programms präsentiert sich uns die Schaltzentrale von Weka:

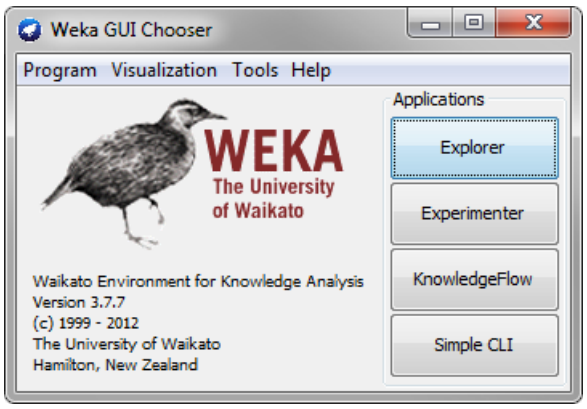


Abbildung 2-2: Die Schaltzentrale von Weka

Mit den vier Schaltflächen auf der rechten Seite lassen sich folgende Anwendungen starten:

- Mit dem **EXPLORER** lassen sich Datensätze untersuchen und verschiedene Data-Mining Algorithmen darauf anwenden.
- Der **EXPERIMENTER** dient dazu statistische Tests und Vergleiche zwischen verschiedenen Data-Mining Verfahren durchzuführen.
- Die **KNOWLEDGEFLOW** Umgebung bietet praktisch dieselbe Funktionalität wie der Explorer an, allerdings lassen sich hier die verschiedenen Data-Mining Schritte mit Hilfe von Drag & Drop sehr einfach miteinander verknüpfen
- **SIMPLE CLI** erlaubt es, alle Funktionen von Weka direkt auf der Kommandozeile zu nutzen.

Im Folgenden soll der Datensatz zum «Wetter-Problem» mit Hilfe von Weka untersucht werden. Dazu wird der **EXPLORER** ausgewählt und anschliessend über **OPEN FILE…** die Datei *WetterProblem.csv* geladen. Standardmässig zeigt Weka bei der Auswahl nur seinen eigenen Datentyp *ARFF* an. Die CSV-Dateien lassen sich aber im Öffnen-Dialog anzeigen, indem bei Dateityp (unten) auf *CSV data files* gewechselt wird. Nachdem der Datensatz geladen wurde, präsentiert sich folgendes Bild:

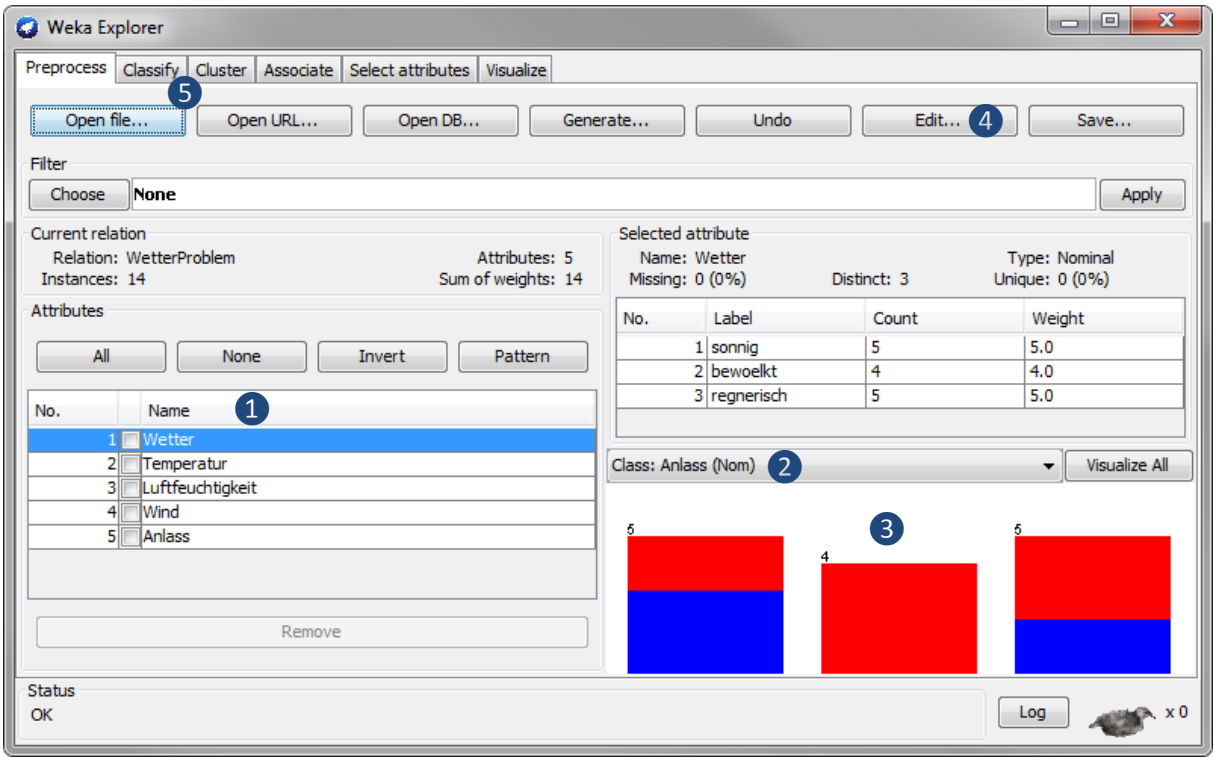


Abbildung 2-3: Weka Explorer mit dem geladenen Datensatz zum «Wetter-Problem»

In dieser Darstellung (**REPROCESS**) ist es möglich, den geladenen Datensatz etwas genauer unter die Lupe zu nehmen. Bei ❶ werden alle Attribute angezeigt und lassen sich anwählen. Bei ❷ kann die Klasse ausgewählt werden, nach welcher die Klassifikation vorgenommen werden soll. In unserem Fall ist dies die Spalte *Anlass*, welche von Weka automatisch ausgewählt werden sollte, da es in der CSV-Datei die letzte Spalte ist und diese üblicherweise die Klasse enthält. Bei ❸ wird jeweils angezeigt, wie sich die Attribut-Werte (in diesem Fall *sonnig*, *bewoelkt* und *regnerisch*) des ausgewählten Attributs (*Wetter*) auf die beiden Klassen (*ja* und *nein*) verteilt. So kommt der Attribut-Wert sonnig beispielsweise insgesamt 5 Mal vor, wovon 2 in die Klasse *ja* (rot) und 3 in die Klasse *nein* (blau) entfallen. Über **EDIT…** ❹ lässt sich der Datensatz in Tabellenform betrachten und gegebenenfalls auch modifizieren, wenn dies gewünscht sein sollte.

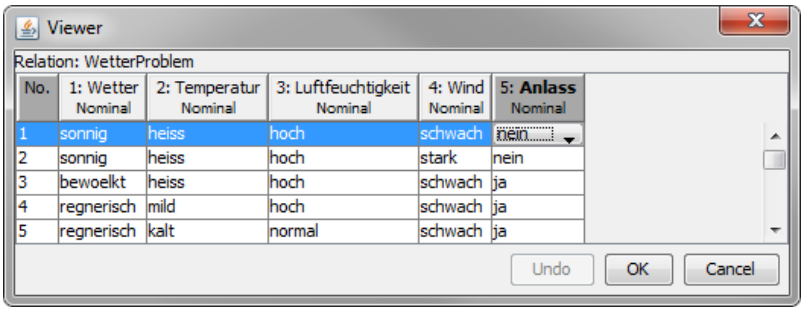


Abbildung 2-4: Betrachter und Editor von Weka

Über den Reiter **CLASSIFY** ❺ lässt sich anschliessend der Algorithmus wählen, welcher die Klassifizierung des Datensatzes lernen soll. Solche Algorithmen werden im Data-Mining als *Classifier* bezeichnet.

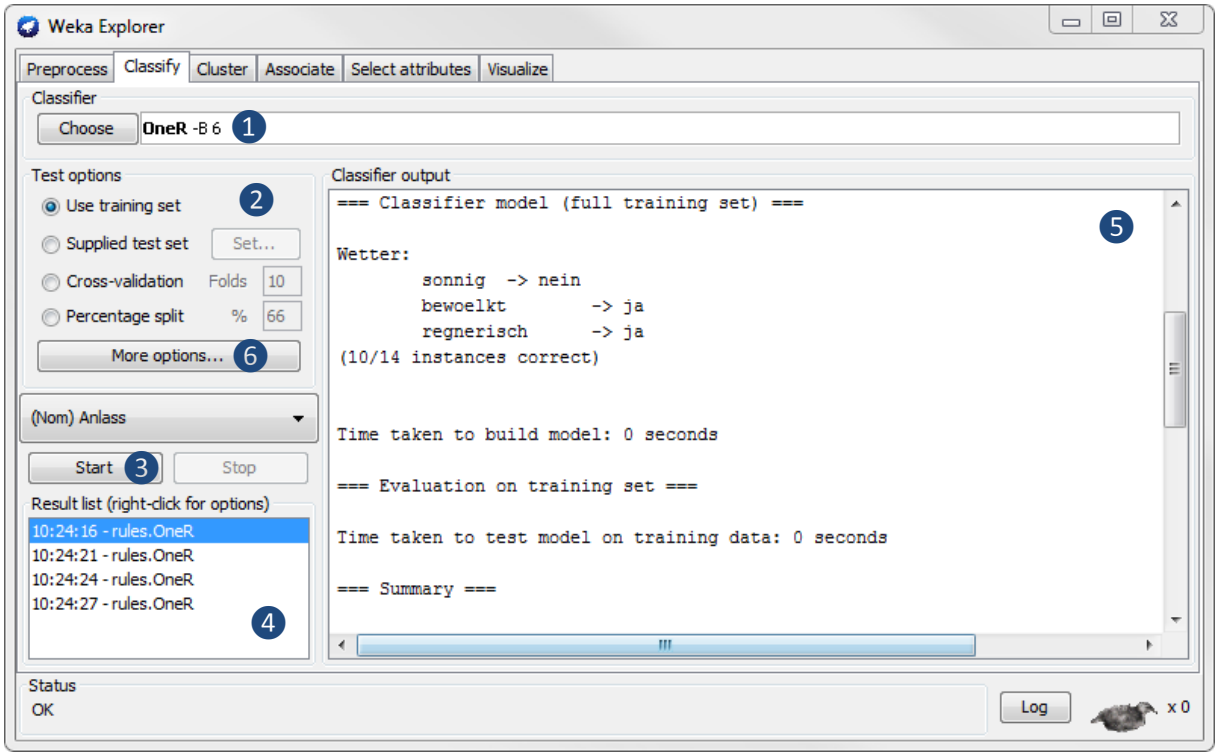


Abbildung 2-5: Das Klassifikations-Fenster von Weka

Im Klassifikations-Fenster lässt sich über **CHOOSE** ❶ der Algorithmus auswählen, mit welchem trainiert werden soll. In diesem Fall wurde *OneR*, die Bezeichnung für 1R in Weka, ausgewählt. OneR findet sich im Auswahlfenster unter *weka » classifiers » rules » OneR*. Neben der Bezeichnung des ausgewählten Algorithmus, werden immer auch allfällige Parameter angezeigt, im Bild oben *-B 6*. Möchte man herausfinden, um was es sich dabei handelt, genügt ein Linksklick auf diese Parameter. In diesem Fall steht *–B 6* offensichtlich für *minBucketSize = 6*, ein Parameter, den Sie bereits aus dem vorherigen Kapitel kennen sollten. Natürlich lässt sich der Wert für *minBucketSize* nach Belieben ändern und damit experimentieren.

Unter **TEST OPTIONS** ❷ lässt sich definieren, auf welche Weise das Trainierte Regel-Set getestet werden soll:

- **USE TRAINING SET** verwendet zum Testen des Regel-Sets nur die Trainingsdaten. Im Prinzip wird hier also gar nicht getestet, sondern lediglich evaluiert, wie viele Instanzen vom Regel-Set in die korrekte Klasse eingeteilt werden.
- **SUPPLIED TEST SET** ermöglicht die Verwendung von Testdaten, welche über **SET…** ausgewählt werden können. Anschliessend lässt sich mit **OPEN FILE…** eine ARFF- oder eine CSV-Datei laden wobei über **CLASS** die Klasse eingestellt werden muss, mit welcher getestet werden soll.
- **CROSS-VALIDATION** ist standardmässig ausgewählt führt eine Kreuzvalidierung durch, wie wir sie oben kennengelernt haben. Als Parameter müssen die Anzahl *Folds* übergeben werden.
- Mit **PERCENTAGE SPLIT** lässt sich ein Datensatz zufällig in Trainings- und Testdaten aufspalten. Die angegebene Prozentzahl definiert dabei die Grösse des Trainingsets.

Mit **START** ❸ wird das Training gestartet, das erhaltene Regel-Set getestet und in der **RESULT LIST** ❹ abgelegt. Diese ist besonders nützlich, wenn bei einem Classifier verschiedene Testmethoden ausprobiert werden möchten, da so schnell zwischen den jeweiligen Reporten hin- und hergewechselt werden kann. Die Resultate der einzelnen Trainings- und des Testläufe werden unter **CLASSIFIER OUTPUT** ❺ übersichtlich dargestellt. Der Report enthält unter anderem das trainierte Regel-Set und eine statistische Auswertung des Testlaufes. Für das «Wetter-Problem» ergibt sich folgender Report (gekürzt), wenn als Testverfahren **USE TRAINING SET** ausgewählt wurde:

```
=== Run information ===

Scheme:       weka.classifiers.rules.OneR -B 6
Relation:     WetterProblem
Instances:    14
Attributes:   5
              Wetter
              Temperatur
              Luftfeuchtigkeit
              Wind
              Anlass
Test mode:    evaluate on training data


=== Classifier model (full training set) ===

Wetter:
   sonnig   -> nein
   bewoelkt -> ja
   regnerisch  -> ja
(10/14 instances correct)
Time taken to build model: 0 seconds


=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds


=== Summary ===

Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %
Kappa statistic                          0.3778
Mean absolute error                      0.2857
Root mean squared error                  0.5345
Relative absolute error                 61.5385 %
Root relative squared error            111.4773 %
Coverage of cases (0.95 level)          71.4286 %
Mean rel. region size (0.95 level)      50      %
Total Number of Instances               14


=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
               0.6      0.222    0.6        0.6     0.6        0.378  0.689     0.503     nein
               0.778    0.4      0.778      0.778   0.778      0.378  0.689     0.748     ja
Weighted Avg.  0.714    0.337    0.714      0.714   0.714      0.378  0.689     0.66


=== Confusion Matrix ===

 a b   <-- classified as
 3 2 | a = nein
 2 7 | b = ja
```

Der Report liefert zwar eine ausführliche Statistik über die Leistungsfähigkeit des trainierten Classifiers, gibt aber keine Informationen darüber Preis, wie die einzelnen Instanzen des Trainings- resp. Testsets klassifiziert wurden. Eine solche Übersicht wäre für unsere Zwecke aber oft sehr nützlich und sie lässt sich auch einschalten – gesetzt den Fall man findet sie!

Die Option versteckt sich hinter dem Knopf **MORE OPTIONS …** ❻ , welcher zu folgendem Fenster führt:

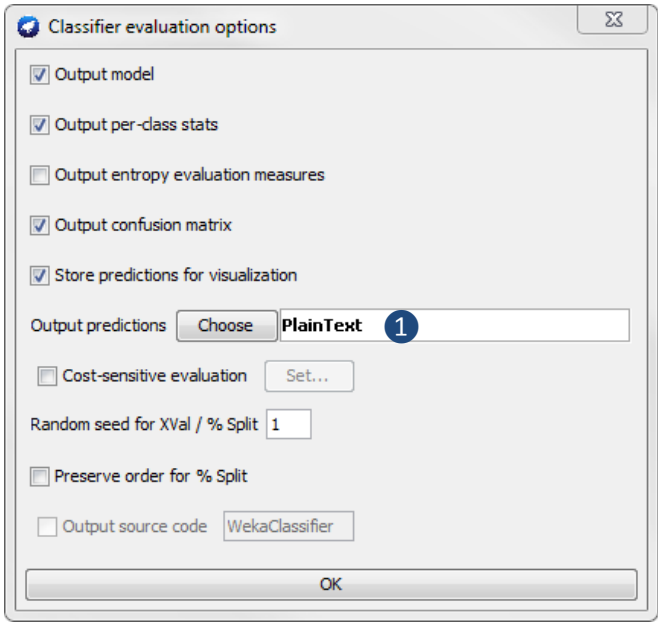


Abbildung 2-6: Betrachter und Editor von Weka

Wird bei **OUTPUT PREDICTIONS** ❶ mit **CHOOSE** die Option *PlainText* angewählt, enthält der generierte Report eine ausführliche Übersicht über die Klassifizierung jeder Instanz des Trainingssets …

```
=== Predictions on training set ===
inst#     actual  predicted error prediction
     1     1:nein     1:nein       1
     2     1:nein     1:nein       1
     3       2:ja       2:ja       1
     4       2:ja       2:ja       1
     5       2:ja       2:ja       1
     6     1:nein       2:ja   +   1
     7       2:ja       2:ja       1
     8     1:nein     1:nein       1
     9       2:ja     1:nein   +   1
    10       2:ja       2:ja       1
    11       2:ja     1:nein   +   1
    12       2:ja       2:ja       1
    13       2:ja       2:ja       1
    14     1:nein       2:ja   +   1
```

… oder des Testsets, wenn mit der entsprechenden Option **SUPPLIED TEST SET** trainiert wurde.

```
=== Predictions on test set ===
 inst#     actual  predicted error prediction
     1     1:nein     1:nein       1
     2     1:nein       2:ja   +   1
     3     1:nein       2:ja   +   1
     4       2:ja       2:ja       1
     5       2:ja       2:ja       1
     6     1:nein       2:ja   +   1
```

2.7.1 WAHRHEITSMATRIX (CONFUSION MATRIX)

Im Report findet sich unter dem Punkt *Classifier model* das Regel-Set, welches vom Algorithmus präferiert wurde und mit welchem sich in diesem Fall 10 der 14 Instanzen des Trainingssets korrekt klassifizieren lassen. Sehr nützlich ist die *Wahrheitsmatrix* (engl. *Confusion Matrix*). Es handelt sich dabei um eine Tabelle, welche in den Spalten auflistet, wie viele Instanzen vom Classifier in welche Klassen eingeteilt wurden. Die Zeilen der Wahrheitsmatrix geben an, wie sich die Instanzen tatsächlich auf die Klassen verteilen müssten.

	Classifier teilt in Klasse *nein* ein	**Classifier teilt in Klasse *ja* ein.**
Gehört in die Klasse *nein*	3	2
Gehört in die Klasse *ja*	2	7

Tabelle 2-11: Wahrheitsmatrix für das «Wetter-Problem»

Der Warheitsmatrix lässt sich also entnehmen, dass das Regel-Set 3 Instanzen korrekt in die Klasse *nein* und 7 Instanzen korrekt in die Klasse *ja* einteilt (grün). Daneben werden vom Classifier aber auch je 2 Instanzen fälschlicherweise in die Klassen *ja* resp. *nein* eingeteilt (rot).

Aufgabe 2-5: *Laden Sie mit Weka den Datensatz zum «Wetter-Problem» und wählen Sie als Classifier **OneR** mit minBucketSize = 6 aus. Probieren Sie nun alle vier Testverfahren durch, welche unter **TEST OPTIONS** ❷ angeboten werden. Als Testset steht Ihnen die Datei WetterProblem-TestSet.csv zur Verfügung.*

Vergleichen und interpretieren Sie die Resultate, insbesondere die Prozentzahlen unter Summary » Correctly Classified Instances resp. Incorrectly Classified Instances und die Wahrheitsmatrizen.

Notieren Sie für alle vier Testverfahren den Wert bei Summary » Kappa statistics. Sie werden diese Zahlen in der nächsten Übung benötigen.

2.7.2 KAPPA STATISTIK (KAPPA STATISTICS)

Unter Summary zeigt der Report an, wie viele Instanzen vom Regel-Set korrekt klassifiziert wurden. In diesem Fall waren es 10 von 14, der Classifier weist also eine Trefferquote von p_C = 71 % auf. Diese Zahl kann aber nicht direkt als Mass für die Erfolgsrate des Classifiers herangezogen werden. Man kann sich nämlich die Frage stellen, welcher Prozentsatz p_Z der Instanzen von einem Zufalls-Classifier ebenfalls korrekt eingeordnet worden wären. Die Differenz zwischen der Trefferquote unseres Classifiers und jener des Zufalls-Classifiers ist ein gutes Mass für die effektive Erfolgsrate des Regel-Sets. Bezeichnet wird diese Grösse als *Cohens Kappa* κ (Kappa statistic) und berechnet sich wie folgt:

$$\kappa = \frac{p_C - p_Z}{1 - p_Z}$$

Um κ zu berechnen, wird die Wahrheitsmatrix benötigt. Zur Veranschaulichung gehen wir von folgendem Beispiel aus:

	Classifier teilt in Klasse *nein* ein	**Classifier teilt in Klasse *ja* ein.**
Gehört in die Klasse *nein*	40	10
Gehört in die Klasse *ja*	20	30

Tabelle 2-12: Wahrheitsmatrix eines fiktiven Problems zur Veranschaulichung von Cohens Kappa κ

BERECHNUNG VON p_C

Der Classifier hat in diesem Problem 40 Instanzen korrekt der Klasse *nein* und 30 Instanzen korrekt der Klasse *ja* zugeteilt. Bezogen auf die 100 Instanzen entspricht dies einer Trefferquote von:

$$p_C = \frac{40 + 30}{100} = 0.7$$

BERECHNUNG VON p_Z

Von den 100 Instanzen entfallen 40+10 = 50 in die Klasse *nein*, was einem Anteil von 0.5 entspricht.

$$\frac{40 + 10}{100} = 0.5$$

Von den 100 Instanzen teilt der Classifier 40+20 = 60 in die Klasse *nein* ein, also ein Anteil von 0.6.

$$\frac{40 + 20}{100} = 0.6$$

Die Wahrscheinlichkeit, dass derselbe Classifier zufällig eine Instanz in die Klasse *nein* einteilt und diese auch wirklich in die Klasse *nein* gehört beträgt:

$$p_{Z(nein)} = 0.6 * 0.5 = 0.3$$

Von den 100 Instanzen entfallen 20+30 = 50 in die Klasse *ja*, was einem Anteil von 0.5 entspricht.

$$\frac{20 + 30}{100} = 0.5$$

Von den 100 Instanzen teilt der Classifier 40+20 = 60 in die Klasse *ja* ein, also ein Anteil von 0.4.

$$\frac{10 + 30}{100} = 0.4$$

Die Wahrscheinlichkeit, dass derselbe Classifier zufällig eine Instanz in die Klasse *ja* einteilt und diese auch wirklich in die Klasse *ja* gehört beträgt:

$$p_{Z(ja)} = 0.4 * 0.5 = 0.2$$

Die Wahrscheinlichkeit p_Z, dass der Zufalls-Classifier die Instanzen also korrekt in die Klassen *ja* und *nein* einteilt, entspricht der Summe der Wahrscheinlichkeiten von «zufällig korrekt in *nein* einteilen» und «zufällig korrekt in *ja* einteilen».

$$p_Z = p_{Z(nein)} + p_{Z(ja)} = 0.3 + 0.2 = 0.5$$

BERECHNUNG VON KAPPA

$$\kappa = \frac{p_C - p_Z}{1 - p_Z} = \frac{0.7 - 0.5}{1 - 0.5} = 0.4 = 40\ \%$$

Der Zufalls-Classifier teilt 50 Instanzen (50%) korrekt ein, währenddessen vom trainierten Classifier 70 Instanzen (70 %) korrekt zugeteilt werden. Letzterer würde also 20 Instanzen korrekt zuweisen, welche vom Zufalls-Classifier falsch klassifiziert würden. Bezogen auf die vom Zufalls-Classifier falsch zugewiesenen 50 Instanzen sind dies 40 %. κ gibt also an, wie viele Prozent der vom Zufalls-Classifier falsch zugewiesenen Instanzen durch den trainierten Classifier korrekt klassifiziert würden.

Über die statistische Signifikanz von κ gehen die Meinungen auseinander. Wir verwenden folgende Faustregel:

- $0.60 \leq \kappa \leq 1.00$: Der Classifier eignet sich gut bis sehr gut.
- $0.40 \leq \kappa < 0.60$: Der Classifier eignet sich mässig.
- $\kappa < 0.40$: Der Classifier ist ungeeignet.

Aufgabe 2-6: Vergleichen Sie die Werte für κ, welche Sie bei der letzten Aufgabe notiert haben und machen Sie eine Aussage über die Brauchbarkeit der verschiedenen Classifier.

Aufgabe 2-7: Berechnen Sie κ für das «Wetter-Problem» von Hand und interpretieren Sie den Wert. Verwenden Sie für Ihre Überlegungen die gegebene Wahrheitsmatrix.

	Classifier teilt in Klasse *nein* ein	**Classifier teilt in Klasse *ja* ein.**
Gehört in die Klasse *nein*	3	2
Gehört in die Klasse *ja*	2	7

2.8 WEITERE ÜBUNGEN ZUM 1R-ALGORITHMUS

Aufgabe 2-8: Berechnen Sie κ für ein Classifier, welcher folgende Wahrheitsmatrix liefert.

	Classifier teilt in Klasse *a* ein	**Classifier teilt in Klasse *b* ein.**	**Classifier teilt in Klasse *c* ein.**
Gehört in die Klasse *a*	88	10	2
Gehört in die Klasse *b*	14	40	6
Gehört in die Klasse *c*	18	10	12

Aufgabe 2-9: Der «Iris-Datensatz» umfasst 150 Instanzen mit den Daten von drei verschiedenen Schwertlilien-Arten (Iris Setosa, Iris Virginica und Iris Versicolor), wobei je 50 Instanzen auf eine der drei Arten entfallen. Der Datensatz enthält vier Attribute: Länge und Breite des Sepalum (Kelchblatt) und Länge und Breite des Petalum (Kronblatt).

Verwenden Sie den 1R-Algorithmus um ein Regel-Set zu finden, welches mit Hilfe einer der vier Attribute, die Instanzen, korrekt klassifiziert. Die Schwertlilien-Art entspricht in diesem Beispiel der Klasse. Verwenden Sie den Datensatz iris.csv für Ihre Arbeit.

Notieren Sie das Regel-Set und machen Sie eine Aussage über dessen Qualität, wenn Sie als Testmethode die Kreuzvalidierung verwenden.

3 KLASSIFIZIERUNG – NAÏVE BAYES

Der 1R-Algorithmus wählt ein einziges Attribut aus, um Datensätze zu klassifizieren. Eine ebenso einfache, aber vom Ansatz her radikal andere Methode, ist die Verwendung aller Attribute. Jedes Attribut trägt dabei einen Teil zur Klassifizierung der einzelnen Instanzen bei. Dabei werden alle Attribute als gleichwertig und voneinander unabhängig (bezüglich der Klasse) betrachtet. Zugegebenermassen ist diese Annahme etwas unrealistisch, würde doch niemand behaupten, dass das Attribut *Wetter (z.B. regnerisch)* nicht einen Einfluss hätte auf das Attribut *Luftfeuchtigkeit (z.B. hoch)*. Allerdings stellt man auch in diesem Fall fest, dass diese Annahmen nicht nur zu einfachen Modellen führen, sondern dass sich diese überraschend gut bei der Klassifizierung realer Datensätze bewähren.

3.1 STATISTISCHE BETRACHTUNG DES «WETTER-PROBLEMS»

Im Folgenden betrachten wir erneut den Datensatz zum «Wetter-Problem»:

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	hoch	schwach	nein
2	sonnig	heiss	hoch	stark	nein
3	bewoelkt	heiss	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kalt	normal	schwach	ja
6	regnerisch	kalt	normal	stark	nein
7	bewoelkt	kalt	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kalt	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bewoelkt	mild	hoch	stark	ja
13	bewoelkt	heiss	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Tabelle 3-1: Datensatz zum Wetter-Problem

In einem ersten Schritt wird ausgewertet, wie sich die einzelnen Attribute auf die beiden Klassen *ja* und *nein* verteilen. Untenstehende Tabelle zählt im oberen Teil, wie oft jeder Attribut-Wert in den beiden Klassen vorkommt. Der untere Teil der Tabelle berechnet dann die prozentuale Verteilung der Attribut-Werte auf die beiden Klassen.

Wetter			Temperatur			Luftfeuchtigkeit			Wind			Anlass	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
sonnig	3	2	heiss	2	2	hoch	4	3	schwach	2	6	5	9
bewoelkt	0	4	mild	2	4	normal	1	6	stark	3	3		
regnerisch	2	3	kalt	1	3								
sonnig	3/5	2/9	heiss	2/5	2/9	hoch	4/5	3/9	schwach	2/5	6/9	5/14	9/14
bewoelkt	0/5	4/9	mild	2/5	4/9	normal	1/5	6/9	stark	3/5	3/9		
regnerisch	2/5	3/9	kalt	1/5	3/9								

Tabelle 3-2: Statistische Verteilung der Daten zum Wetter-Problem

So kommt beispielsweise beim Attribut *Wetter* der Wert *sonnig* bei insgesamt 5 Instanzen vor, wobei 3 x *Anlass = nein* und 2 x *Anlass = ja* gilt. Das Attribut Wetter besitzt noch zwei weitere Attribut-Werte *bewoelkt* (0 x *Anlass = nein*, 4 x *Anlass = ja*) und *regnerisch* (2 x *Anlass = nein*, 3 x *Anlass = ja*). Das Attribut Wetter teilt also insgesamt 5 Instanzen in die Klasse *nein* ein, wobei zu 3/5 der Attribut-Wert *Wetter = sonnig* und zu 2/5 *Wetter = regnerisch* vorkommt. Diese Berechnungen werden nun für alle Attribut-Werte durchgeführt und in der Tabelle festgehalten. Von den restlichen Berechnungen abweichend ist die letzte Spalte, welche die Klasse selber enthält. Der Anlass findet 9 x nicht statt und wird 5 x durchgeführt. Von den insgesamt 14 Instanzen entfallen also 5/14 in die Klasse *nein* und 9/14 in die Klasse *ja*.

Sie erhalten nun die Aufgabe, vorauszusagen, ob unter den folgenden Bedingungen der Anlass stattfinden kann. Wie gehen Sie vor?

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	?
2	regnerisch	heiss	hoch	schwach	?

Tabelle 3-3: Nicht klassifizierter Datensatz

Betrachten vorerst nur den ersten Eintrag der Tabelle. Die fünf Werte der Instanz (4 Attribute und 1 Klasse), werden als gleichberechtigt (gleich wichtig) und voneinander unabhängig behandelt. Mit der Tabelle 4-2 lässt sich die Chance berechnen, dass eine Instanz in die Klasse *ja* resp. *nein* eingeteilt wird:

$$\text{Chance für } ja = \underset{\substack{\text{sonnig}\\\text{ja}}}{2/9} \times \underset{\substack{\text{heiss}\\\text{ja}}}{2/9} \times \underset{\substack{\text{normal}\\\text{ja}}}{6/9} \times \underset{\substack{\text{stark}\\\text{ja}}}{3/9} \times \underset{\substack{\text{Klasse}\\\text{ja}}}{9/14} = 0.00705$$

$$\text{Chance für } nein = \underset{\substack{\text{sonnig}\\\text{nein}}}{3/5} \times \underset{\substack{\text{heiss}\\\text{nein}}}{2/5} \times \underset{\substack{\text{normal}\\\text{nein}}}{1/5} \times \underset{\substack{\text{stark}\\\text{nein}}}{3/5} \times \underset{\substack{\text{Klasse}\\\text{nein}}}{5/14} = 0.01029$$

Die oben berechnete «Chancen» für *ja* resp. *nein* stellen noch keine Wahrscheinlichkeiten dar, sondern lediglich die Plausibilität, mit welcher die Instanzen in die Klassen *ja* resp. *nein* eingeteilt werden können. Ausserdem müsste die Summe der beiden Werte 1 ergeben. Die Wahrscheinlichkeit lässt sich nun aber dadurch erhalten, dass die beiden Grössen auf 1 normiert werden:

$$\text{Wahrscheinlichkeit für ja} = P(ja) = \frac{0.00705}{0.00705+0.01029} = 0.407 = 40.7\ \%$$

$$\text{Wahrscheinlichkeit für nein} = P(nein) = \frac{0.01029}{0.00705+0.01029} = 0.593 = 59.3\ \%$$

Gemäss unseren Berechnungen müsste die Instanz also mit einer Wahrscheinlichkeit von ca. 59 % in die Klasse *nein* eingeteilt werden.

Aufgabe 3-1: Ermitteln Sie die Klasse, welcher der zweite Eintrag der Tabelle 3-3 zugeordnet werden müsste.

3.2 DAS BAYES-THEOREM

Das Bayes-Theorem, auch als Satz von Bayes bekannt, beschreibt die Berechnung bedingter Wahrscheinlichkeiten. Mit dem Satz von Bayes ist es möglich, die Wahrscheinlichkeit für ein Ereignis E zu berechnen, welches von der Beobachtung B abhängig ist:

$$P(E|B) = \frac{P(B|E) \cdot P(E)}{P(B)}$$

P(E|B) = Wahrscheinlichkeit, dass das Ereignis E eintritt wenn die Beobachtung B gemacht wurde (wird gelesen als «Wahrscheinlichkeit von E gegeben B»)

P(B|E) = Wahrscheinlichkeit, dass die Beobachtung B gemacht werden kann wenn das Ereignis E eingetreten ist (wird gelesen als «Wahrscheinlichkeit von B gegeben E»)

P(E) = Wahrscheinlichkeit, dass das Ereignis E eintritt

P(B) = Wahrscheinlichkeit, dass die Beobachtung B gemacht wird

Ein Beispiel: In zwei Gefässen ❶ und ❷ sind schwarze und weisse Kugeln wie folgt verteilt:

- *Gefäss ❶ enthält 7 schwarze und 3 weisse Kugeln*
- *Gefäss ❷ enthält 1 schwarze und 9 weisse Kugeln*

Jemand zieht nun für Sie eine Kugel zufällig aus einem der beiden Gefässe. Sie sehen nicht aus welchem, wissen aber, dass die Kugel schwarz ist. Mit welcher Wahrscheinlichkeit stammt sie aus dem Gefäss ❶?

Es wird also die Beobachtung gemacht, dass die Kugel schwarz ist und wir möchten wissen, wie gross die Wahrscheinlichkeit ist dass aufgrund dieser Beobachtung die Kugel aus dem Gefäss ❶ stammt. Damit ergibt sich:

$$P(❶|\text{schwarz}) = \frac{P(\text{schwarz}|❶) \cdot P(❶)}{P(\text{schwarz})} = \frac{7/10 \cdot 1/2}{2/5} = 87.5\%$$

P(❶) = Wahrscheinlichkeit, dass die Kugel aus dem Gefäss ❶ gezogen wurde. Da zwei Gefässe vorhanden sind, aus denen zufällig gezogen wird, beträgt die Wahrscheinlichkeit 1/2.

P(schwarz) = Wahrscheinlichkeit, dass die Kugel schwarz ist. Da von den insgesamt 20 Kugeln 8 schwarz sind, beträgt die Wahrscheinlichkeit 8/20 also 2/5.

P(schwarz|❶) = Wahrscheinlichkeit, dass die Kugel schwarz ist, wenn sie aus dem Gefäss ❶ stammt. Da dieses Gefäss von insgesamt 10 Kugeln 7 schwarze enthält ist die Wahrscheinlichkeit 7/10.

Aufgabe 3-2: Die Wahrscheinlichkeit, dass eine Person eine bestimmte Krankheit in sich trägt ist P(K) = 0.0002. Mit Hilfe eines neuartigen Tests soll ermittelt werden, ob eine Person von dieser Krankheit betroffen ist. Der Test T erkennt die Krankheit mit 99 prozentiger Sicherheit. Von 10'000 untersuchten Personen, wurden 102 positiv getestet. Wie viele von Ihnen sind tatsächlich krank?

Auch die Methode, mit welcher wir oben die Wahrscheinlichkeit berechnet haben, dass eine bestimmte Instanz in die Klasse *ja* resp. *nein* zugeteilt wird, basiert auf dem Satz von Bayes.

$$P(E|B) = \frac{P(B|E) \cdot P(E)}{P(B)}$$

Für das Ereignis E wird eingesetzt, ob der Anlass stattfinden kann oder nicht. E kann also die Werte *ja* oder *nein* annehmen und entspricht damit Klasse. Die Beobachtungen B, welche in diesem Beispiel gemacht werden können, sind jene zu den Wetterbedingungen. Dabei gibt es bei jeder Instanz vier verschiedene Beobachtungen: Wetter, Temperatur, Luftfeuchtigkeit und Wind. Es gibt also nicht nur eine Beobachtung, sondern 4 verschiedene $B_1 - B_4$, welche den einzelnen Attributen entsprechen. Mit der Annahme, dass die 4 Beobachtungen voneinander unabhängig sind, lässt sich mit dem Satz von Bayes die Wahrscheinlichkeit berechnen, dass aufgrund der Beobachtungen B_1-B_4 der Anlass stattfindet resp. nicht stattfindet:

$$P(ja|B) = \frac{P(B_1|ja) \cdot P(B_2|ja) \cdot P(B_3|ja) \cdot P(B_4|ja) \cdot P(ja)}{P(B)}$$

$$P(nein|B) = \frac{P(B_1|nein) \cdot P(B_2|nein) \cdot P(B_3|nein) \cdot P(B_4|nein) \cdot P(nein)}{P(B)}$$

Die Berechnung von P(B) kann man sich dabei schenken, da sich dieser Term bei der Normierung auf 1 sowieso rauskürzen wird. Wir erhalten dadurch dieselben Formeln, wie im vorangegangenen Kapitel. Für den Eintrag 1 der Tabelle 3-3 entspricht B_1 = *sonnig*, B_2 = *heiss*, B_3 = *normal* und B_4 = *stark*.

$$P(ja|B) = P(B_1|ja) \cdot P(B_2|ja) \cdot P(B_3|ja) \cdot P(B_4|ja) \cdot P(ja) = \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} = 0.00705$$

$$P(nein|B) = P(B_1|nein) \cdot P(B_2|nein) \cdot P(B_3|nein) \cdot P(B_4|nein) \cdot P(nein) = \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} = 0.01029$$

Nach erfolgter Normierung auf 1 ergeben sich folgende Wahrscheinlichkeiten:

$$P(ja \mid sonnig, heiss, normal, stark) = \frac{0.00705}{0.00705+0.01029} = 0.407 = 40.7\ \%$$

$$P(nein \mid sonnig, heiss, normal, stark) = \frac{0.01029}{0.00705+0.01029} = 0.593 = 59.3\ \%$$

Die verwendete Methode, die hier verwendet wird, um den Classifier zu trainieren nennt sich Naïve Bayes. Diesen Namen hat sie erhalten, weil sie einerseits auf dem Satz von Bayes basiert und andererseits naiv annimmt, dass die Attribute vollständig unabhängig voneinander sind. Natürlich geht diese Annahme zu weit (die Attribute *Luftfeuchtigkeit* und *Wetter* sind bestimmt nicht völlig unabhängig voneinander), allerdings funktioniert Naïve Bayes auf realen Datensätzen oft erstaunlich gut.

Aufgabe 3-3: Berechnen Sie P(nein | bewoelkt, heiss, hoch, schwach) und erläutern Sie, was von diesem Resultat zu halten ist.

Sobald ein Attribut-Wert 0 Mal vorkommt, führt dies zwingend zu einer Wahrscheinlichkeit von 0 für den ganzen Term (siehe Aufgabe 3-3). Wahrscheinlichkeiten, welche 0 betragen haben also sozusagen ein Veto über die anderen. Dieses Problem lässt sich aber leicht umschiffen, indem bei der Berechnung der Wahrscheinlichkeit, aus der Anzahl auftretender Attribut-Werte, ein Trick angewendet wird. Zur Summe jedes Attribut-Wertes wird eine Konstante μ dazugezählt, der Einfachheit halber wählen wir eine $\mu = 1$. Damit kommt jeder Attribut-Wert mindestens 1 Mal vor. Natürlich ändern sich dadurch auch die Wahrscheinlichkeiten, mit welchen die einzelnen Attribut-Werte in den Klassen vorkommen. Es ergibt sich folgende Tabelle:

Wetter			**Temperatur**			**Luftfeuchtigkeit**			**Wind**			**Anlass**	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
sonnig	3+1	2+1	heiss	2+1	2+1	hoch	4+1	3+1	schwach	2+1	6+1	5+1	9+1
bewoelkt	0+1	4+1	mild	2+1	4+1	normal	1+1	6+1	stark	3+1	3+1		
regnerisch	2+1	3+1	kalt	1+1	3+1								
sonnig	4/8	3/12	heiss	3/8	3/12	hoch	5/7	4/11	schwach	3/7	7/11	6/16	10/16
bewoelkt	1/8	5/12	mild	3/8	5/12	normal	2/7	7/11	stark	4/7	4/11		
regnerisch	3/8	4/12	kalt	2/8	4/12								

Tabelle 3-4: Statistische Verteilung der Daten zum Wetter-Problem korrigiert mit $\mu = 1$

Aufgrund dieses Tricks sind nun alle Wahrscheinlichkeiten grösser als 0. Die Strategie des «1 dazuzählens» ist übrigens eine Standardtechnik und wird nach dem französischen Mathematiker Pierre Laplace, *Laplace Estimator* genannt. Für den Eintrag 1 der Tabelle 3-3 entspricht B_1 = *sonnig*, B_2 = *heiss*, B_3 = *normal* und B_4 = *stark* und es ergeben sich folgende Wahrscheinlichkeiten:

$$P(ja|B) = P(B_1|ja) \cdot P(B_2|ja) \cdot P(B_3|ja) \cdot P(B_4|ja) \cdot P(ja) = \frac{3}{12} \cdot \frac{3}{12} \cdot \frac{7}{11} \cdot \frac{4}{11} \cdot \frac{10}{16} = 0.00904$$

$$P(nein|B) = P(B_1|nein) \cdot P(B_2|nein) \cdot P(B_3|nein) \cdot P(B_4|nein) \cdot P(nein) = \frac{4}{8} \cdot \frac{3}{8} \cdot \frac{2}{7} \cdot \frac{4}{7} \cdot \frac{6}{16} = 0.01148$$

Nach erfolgter Normierung auf 1 ergeben sich folgende Wahrscheinlichkeiten:

$$P(ja \mid sonnig, heiss, normal, stark) = \frac{0.00904}{0.00904+0.01148} = 0.441 = 44.1\ \%$$

$$P(nein \mid sonnig, heiss, normal, stark) = \frac{0.01148}{0.00904+0.01148} = 0.559 = 55.9\ \%$$

Je nachdem, wie gross μ gewählt wird, verändern sich die Prozentzahlen, mit welchen die Zuteilung zu einer Klasse erfolgt. Dies ist allerdings nicht weiter Schlimm, da man schlussendlich nur daran interessiert ist, ob eine Instanz in die Klasse *ja* resp. *nein* zugeteilt wird.

3.3 SPAMERKENNUNG MIT NAÏVE BAYES

Naïve Bayes - Classifier kommen auch bei Spam-Filtern zum Einsatz. Um die Funktionsweise zu untersuchen, betrachten wir ein vereinfachtes Beispiel. Nehmen wir an, wir würden E-Mails erhalten, welche nur die Wörter *ich*, *Gratis*, *Lebenslauf* und *heiss* beinhalten dürften. Für jede Nachricht wird nun überprüft, ob ein bestimmtes Wort vorkommt (*v*) oder nicht (-) und ob die Nachricht Spam ist (*ja*) oder nicht (*nein*). Folgende Tabelle enthält einen Datensatz von 16 E-Mails, welche als Spam resp. nicht Spam klassifiziert wurden und auf deren Wortzusammensetzung hin untersucht wurden:

	ich	Gratis	Lebenslauf	heiss	Spam
1	v	-	-	-	nein
2	v	v	-	-	ja
3	v	-	v	-	nein
4	-	-	-	v	ja
5	v	v	-	-	ja
6	v	-	v	-	nein
7	v	-	-	v	ja
8	v	v	v	-	nein
9	v	-	v	v	ja
10	v	v	-	v	ja
11	v	v	v	-	nein
12	v	-	v	v	nein
13	v	v	-	v	ja
14	-	v	-	v	ja
15	v	-	v	v	nein
16	v	v	v	v	ja

Tabelle 3-5: Spam Datensatz

Die Instanz 3 der Tabelle beschreibt also beispielsweise ein E-Mail, welches aus den Wörtern *ich* und *Lebenslauf* besteht und der Klasse *Spam = nein* zugeordnet wurde. Im nächsten Schritt wird nun eine Tabelle erstellt, welche für jedes Wort auflistet, wie häufig es in den entsprechenden Klassen vorkommt. Natürlich addieren wir auch in diesem Fall immer eine 1, um nicht Gefahr zu laufen, dass die Wahrscheinlichkeit für einzelne Einträge 0 wird, wie z.B. bei P(*ich = -* | *nein*).

ich			Gratis			Lebenslauf			heiss			Spam	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
v	7+1	7+1	v	2+1	6+1	v	6+1	2+1	v	2+1	7+1	7+1	9+1
-	0+1	2+1	-	5+1	3+1	-	1+1	7+1	-	5+1	2+1		
v	8/9	8/11	v	3/9	7/11	v	7/9	3/11	v	3/9	8/11	8/18	10/18
-	1/9	3/11	-	6/9	4/11	-	2/9	8/11	-	6/9	3/11		

Tabelle 3-6: Statistische Verteilung der Daten im Spam-Datensatz

Sie erhalten nun ein neues E-Mail (Tabelle 3-7), von welchem das Mailprogramm entscheiden soll, ob es sich um Spam handelt oder nicht. Dazu wurde der NaiveBayes-Spamfilter des Mailprogramms vorgängig mit dem Datensatz aus Tabelle 3-5 trainiert.

	ich	Gratis	Lebenslauf	heiss	Spam
1	-	v	v	v	?

Tabelle 3-7: Handelt es sich hierbei um Spam?

Mit Hilfe des Satzes von Bayes lässt sich die Wahrscheinlichkeit bestimmen, mit welchem ein E-Mail, welches die Wörter *Gratis*, *Lebenslauf* und *heiss* enthält, in die Klasse *Spam = ja* gehört.

$$P(E|B) = \frac{P(B|E) \cdot P(E)}{P(B)}$$

Die Beobachtung B entspricht einem E-Mail, welches aus einer Kombination bestimmter Wörter B_1 - B_n besteht. Im obigen Fall ist B_1: *ich = -*, B_2: *Gratis = v*, B_3: *Lebenslauf = v* und B_4: *heiss = v*. Die Wahrscheinlichkeit P(B) entspricht derjenigen, mit welcher ein bestimmtes E-Mail (eine bestimmte Wortkombination) auftreten würde und lässt sich nicht direkt berechnen. Glücklicherweise kürzt sich dieser Term bei der Normierung sowieso, so dass wir auf dessen Bestimmung verzichten können. Mit Hilfe der Wahrscheinlichkeiten aus Tabelle 3-6 erhalten wir:

$$P(ja|B) = P(B_1|ja) \cdot P(B_2|ja) \cdot P(B_3|ja) \cdot P(B_4|ja) \cdot P(ja) = \frac{3}{11} \cdot \frac{7}{11} \cdot \frac{3}{11} \cdot \frac{8}{11} \cdot \frac{10}{18} = 0.0191$$

$$P(nein|B) = P(B_1|nein) \cdot P(B_2|nein) \cdot P(B_3|nein) \cdot P(B_4|nein) \cdot P(nein) = \frac{1}{9} \cdot \frac{3}{9} \cdot \frac{7}{9} \cdot \frac{3}{9} \cdot \frac{8}{18} = 0.0043$$

Nach erfolgter Normierung auf 1 ergeben sich folgende Wahrscheinlichkeiten:

$$P(ja \mid ich = –, Gratis = v, Lebenslauf = v, heiss = v) = \frac{0.0191}{0.0191+0.0043} = 0.818 = 81.1\ \%$$

$$P(nein \mid ich = –, Gratis = v, Lebenslauf = v, heiss = v) = \frac{0.0043}{0.0191+0.0043} = 0.182 = 18.2\ \%$$

Der NaiveBayes-Classifier würde dieses E-Mail also ziemlich deutlich als Spam aussortieren. Ausschlaggebend dafür sind hauptsächlich die Worte *Gratis* und *heiss*, welche gemäss dem Datensatz in Tabelle 3-5 häufig in Spam-Mails anzutreffen sind und in der Berechnung von P(*ja*|B) mit 7/11 resp. 8/11 zu Buche schlagen. Dies lässt sich auch gut erkennen mit einem Klick auf **VISUALIZE ALL** ❶.

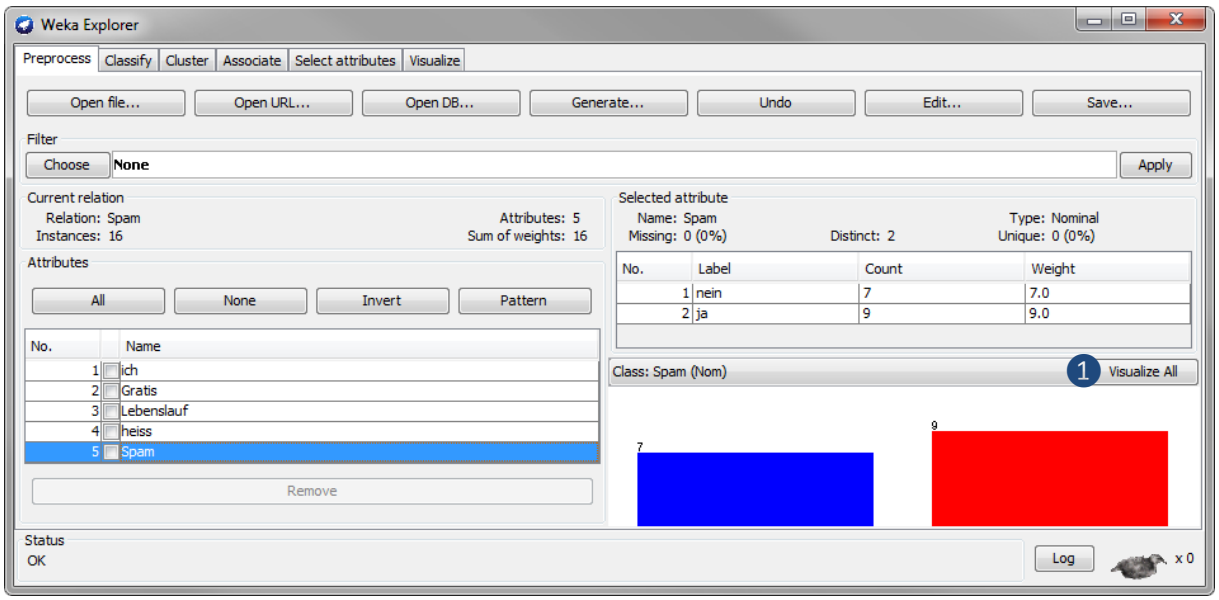


Abbildung 3-1: Balkendiagramme via VISUALIZE ALL

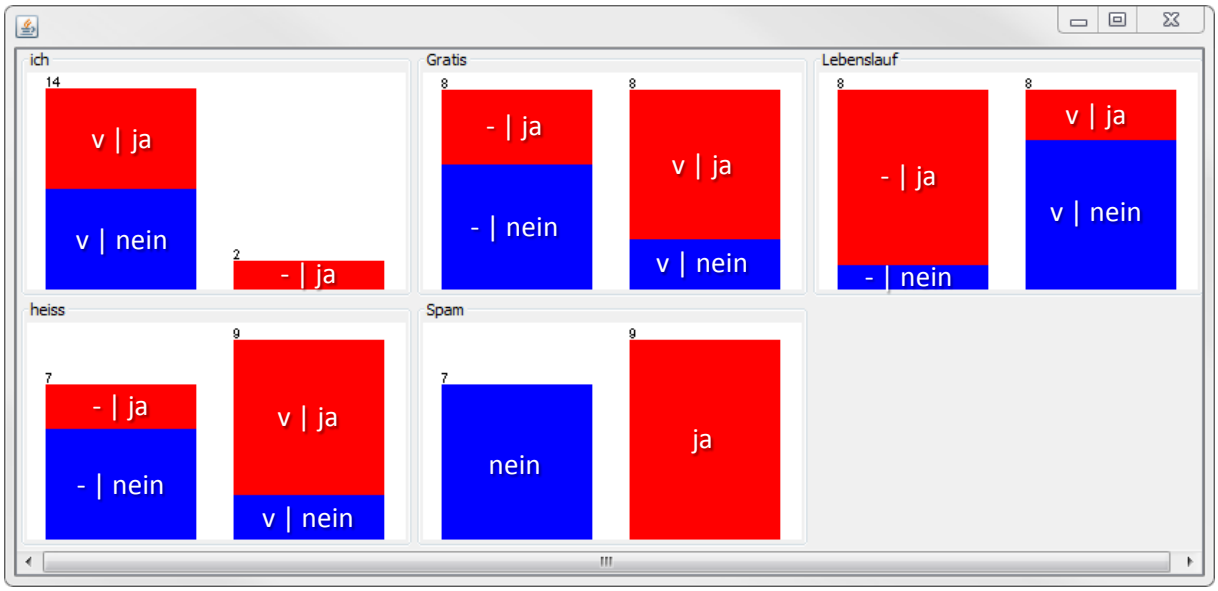


Abbildung 3-2: Alle Balkendiagramme zum Spam-Datensatz in der Übersicht

Dadurch werden Balkendiagramme angezeigt, welche die Verteilung der Attribut-Werte auf die Klassen (in diesem Fall *Spam = ja* resp. *Spam = nein*) visualisieren. Abbildung 3-2 zeigt beispielsweise für das Attribut *heiss*, dass es 9 Mal vorkommt (*heiss = v*; rechts) und dabei sehr viel häufiger in Spam-Mails (rot) auftritt, als in E-Mails, die nicht als Spam klassifiziert wurden (blau). Im Unterschied dazu, werden E-Mails, in denen *heiss* nicht vorkommt, auch nicht als Spam markiert (links, blau). In diesem Beispiel landen also Attribute, welche links einen grossen Blau-Anteil und rechts einen grossen Rot-Anteil aufweisen, mit grosser Wahrscheinlichkeit im Spamfilter. Dazu gehören *Gratis* und *heiss*. Speziell ist das Attribut *ich*. Da es in praktisch jedem E-Mail vorkommt, lässt sich mit ihm nicht wirklich eine Voraussage machen. Leider zeigt Weka die Attributwerte nicht für alle Attribute gleich an, so ist für das Attribut *ich* der Wert *v* links, für das Attribut *heiss* dagegen rechts. Fährt man mit der Maus aber über die Balken, erscheint ein Tooltip, welches den Wert anzeigt.

Aufgabe 3-4: *Laden Sie mit Weka den Datensatz Spam-Basis.arff, welcher 4'600 E-Mails enthält, die aufgrund von 55 verschiedenen Attributen (Worte und Zeichen) als Spam resp. nicht Spam klassifiziert wurden.*

1) *Trainieren Sie aufgrund dieses Datensatzes einen NaiveBayes-Classifier. Wählen Sie als Testmethode «Cross-Validation». Wie gut schneidet der Classifier ab?*

2) *Berechnen Sie die bedingte Wahrscheinlichkeit dafür dass ein E-Mail, welches das Wort «3d» enthält Spam ist P(3d|spam). Die Zahlen, welche Sie dafür benötigen, können Sie dem Report zum Classifier entnehmen.*

3) *Wählen Sie nun als Testmethode «Supplied test set» aus und laden Sie als Testset die Datei Spam-Test.arff. Das Test-Set enthält zwei Instanzen (E-Mails), wobei eine als Spam, die andere als nicht Spam klassifiziert wird. Erstaunlich ist, dass die beiden E-Mails sich nur in einem Wort unterscheiden. Finden Sie heraus, welches Wort dies ist und erklären Sie, weshalb es für die Klassifizierung einen solch grossen Unterschied macht.*

3.4 NUMERISCHE WERTE

Beim Umgang mit numerischen Werten geht man von der Annahme aus, dass diese einer Normalverteilung (Gauss-Verteilung) entsprechen. Betrachten wir erneut das «Wetter-Problem», bei welchem sowohl für die Temperatur, als auch für die Luftfeuchtigkeit, Zahlenwerte gegeben sind.

	Wetter	Temperatur [°C]	Luftfeuchtigkeit [%]	Wind	Anlass
1	sonnig	31	85	schwach	nein
2	sonnig	26	90	stark	nein
3	bewoelkt	29	86	schwach	ja
4	regnerisch	16	96	schwach	ja
5	regnerisch	14	80	schwach	ja
6	regnerisch	11	70	stark	nein
7	bewoelkt	10	65	stark	ja
8	sonnig	18	95	schwach	nein
9	sonnig	15	70	schwach	ja
10	regnerisch	21	80	schwach	ja
11	sonnig	21	70	stark	ja
12	bewoelkt	18	90	stark	ja
13	bewoelkt	27	75	schwach	ja
14	regnerisch	17	91	stark	nein

Tabelle 3-8: Datensatz zum Wetter-Problem mit numerischen Attributen für die Temperatur und die Luftfeuchtigkeit

Auch die numerischen Werte werden der Klasse ja resp. nein zugeordnet. Anschliessen wird der Durchschnitt (∅) und die Standardabweichung (Stabw) der Attribut-Werte innerhalb der Klasse berechnet. Es ergibt sich folgende Tabelle:

Wetter			Temperatur			Luftfeuchtigkeit			Wind			Anlass	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
sonnig	3+1	2+1		11	10		70	65	schwach	2+1	6+1	5+1	9+1
bewoelkt	0+1	4+1		17	14		85	70	stark	3+1	3+1		
regnerisch	2+1	3+1		18	15		90	70					
				26	16		91	75					
				31	18		95	80					
					21			80					
					21			86					
					27			90					
					29			96					
sonnig	4/8	3/12	∅	20.6	19.0	∅	86.2	79.1	schwach	3/7	7/11	6/16	10/16
bewoelkt	1/8	5/12	Stabw	7.9	6.2	Stabw	9.7	10.2	stark	4/7	4/11		
regnerisch	3/8	4/12											

Tabelle 3-9: Statistische Verteilung der Daten zum Wetter-Problem mit numerischen Werten

Die *Wahrscheinlichkeitsdichtefunktion* (oder einfach *Dichtefunktion*) für eine Normalverteilung mit Mittelwert μ und Standardabweichung σ wird durch folgende Formel beschrieben:

$$f(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Keine Angst – das Ganze ist nicht so kompliziert, wie es aussieht. Möchte man beispielsweise wissen, wie gross der Wert der Dichtefunktion bei der Temperatur 30 °C für den Fall Anlass = *ja* ist, setzt man x = 30, μ = 19.0 und σ = 6.2.

$$f(\text{Temperatur} = 30 \mid \text{ja}) = \frac{1}{\sqrt{2\pi} \cdot 6.2} \cdot e^{-\frac{(30-19.0)^2}{2 \cdot (6.2)^2}} = 0.0132$$

Analog dazu lässt sich auch der Wert der Dichtefunktion für Anlass = *nein* berechnen, wenn μ = 20.6 und σ = 7.9 gesetzt wird.

$$f(\text{Temperatur} = 30 \mid \text{nein}) = \frac{1}{\sqrt{2\pi} \cdot 7.9} \cdot e^{-\frac{(30-20.6)^2}{2 \cdot (7.9)^2}} = 0.0249$$

Berechnet man auch noch die Werte der Dichtefunktion für eine Luftfeuchtigkeit von 70 % …

$$f(\text{Luftfeuchtigkeit} = 70 \mid \text{ja}) = \frac{1}{\sqrt{2\pi} \cdot 10.2} \cdot e^{-\frac{(70-79.1)^2}{2 \cdot (10.2)^2}} = 0.0262$$

$$f(\text{Luftfeuchtigkeit} = 70 \mid \text{nein}) = \frac{1}{\sqrt{2\pi} \cdot 9.7} \cdot e^{-\frac{(70-86.2)^2}{2 \cdot (9.7)^2}} = 0.0103$$

… lassen sich die Wahrscheinlichkeiten folgender Klassifizierung berechnen:

$$P(\text{ja} \mid \text{sonnig}, 30\,°C, 70\,\%, \text{stark}) = \frac{3}{12} \cdot 0.0132 \cdot 0.0262 \cdot \frac{4}{11} \cdot \frac{10}{16} = 1.963 \cdot 10^{-5}$$

$$P(\text{nein} \mid \text{sonnig}, 30\,°C, 70\,\%, \text{stark}) = \frac{4}{8} \cdot 0.0249 \cdot 0.0103 \cdot \frac{4}{7} \cdot \frac{6}{16} = 2.733 \cdot 10^{-5}$$

$$P(\text{ja} \mid \text{sonnig}, 30\,°C, 70\,\%, \text{stark}) = \frac{1.963 \cdot 10^{-5}}{1.963 \cdot 10^{-5} + 2.733 \cdot 10^{-5}} = 0.418 = 41.8\,\%$$

$$P(\text{nein} \mid \text{sonnig}, 30\,°C, 70\,\%, \text{stark}) = \frac{2.733 \cdot 10^{-5}}{1.963 \cdot 10^{-5} + 2.733 \cdot 10^{-5}} = 0.582 = 58.2\,\%$$

Die Werte entsprechen ziemlich genau jenen, die wir oben für die Berechnung von P(*ja* | *sonnig*, *heiss*, *normal*, *stark*) und P(*nein* | *sonnig*, *heiss*, *normal*, *stark*) erhalten haben. Der Naïve Bayes Classifier wendet für die Berechnung der Wahrscheinlichkeiten noch ein genaueres Verfahren an, auf das hier aber nicht weiter eingegangen werden soll. Nur so viel: Die Wahrscheinlichkeitsdichtefunktion für ein Ereignis entspricht nicht exakt dessen Wahrscheinlichkeit. Oder anders formuliert: Die Wahrscheinlichkeit, dass die Temperatur exakt 30 °C beträgt, ist 0. Was man tatsächlich berechnet, ist die Wahrscheinlichkeit, dass sich die Temperatur in einem bestimmten Bereich ε befindet, z.B. 30 ± ε/2.

3.5 WEITERE ÜBUNGEN ZU NAÏVE BAYES

Aufgabe 3-5: *Laden Sie mit Weka den Datensatz WetterProblem.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes (findet sich unter bayes » NaiveBayes). Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert (More options … » Output predictions » Choose » PlainText).*

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

Aufgabe 3-6: *Laden Sie mit Weka den Datensatz WetterProblem-Numerisch.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes. Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert.*

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

Aufgabe 3-7: *Laden Sie mit Weka den Datensatz Iris.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes. Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert.*

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

4 KLASSIFIZIERUNG – DER ENTSCHEIDUNGSBAUM

Wenn es darum geht, aus einem Datensatz Wissen zu gewinnen, dann stellt das Lernen von *Entscheidungsbäumen* (engl. *Decision Trees*) ein mächtiges Verfahren dar, welches sowohl durch seine Effizienz, als auch durch seine Einfachheit glänzt. Ein grosser Vorteil der Methode ist der Umstand, dass sich die erhaltenen Resultate sehr übersichtlich als Entscheidungsbäume darstellen lassen, welche vom Menschen einfach gelesen, überprüft und interpretiert werden können. Im Gegensatz dazu liegen bei den meisten anderen Data-Mining-Verfahren die gewonnenen Erkenntnisse als gedanklich schwer fassbare Funktionen (Blackbox) vor, so z.B. auch bei einem Naïve Bayes-Classifier.

Im Folgenden werden wir (einmal mehr!) den Datensatz zum «Wetter-Problem» betrachten:

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	hoch	schwach	nein
2	sonnig	heiss	hoch	stark	nein
3	bewoelkt	heiss	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kalt	normal	schwach	ja
6	regnerisch	kalt	normal	stark	nein
7	bewoelkt	kalt	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kalt	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bewoelkt	mild	hoch	stark	ja
13	bewoelkt	heiss	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Tabelle 4-1: Datensatz zum Wetter-Problem

Ein Entscheidungsbaum wird nun so erstellt, dass im ersten Schritt ein Attribut (z.B. *Wetter*) ausgesucht wird, welches den sogenannten *Wurzelknoten* (engl. *root node*) bildet. Durch die Attribut-Werte (z.B. *sonnig*, *bewoelkt* und *regnerisch*), wird der Datensatz dann in Untereinheiten aufgespalten. Für das Attribut *Wetter* ergibt sich beispielsweise folgende Aufteilung:

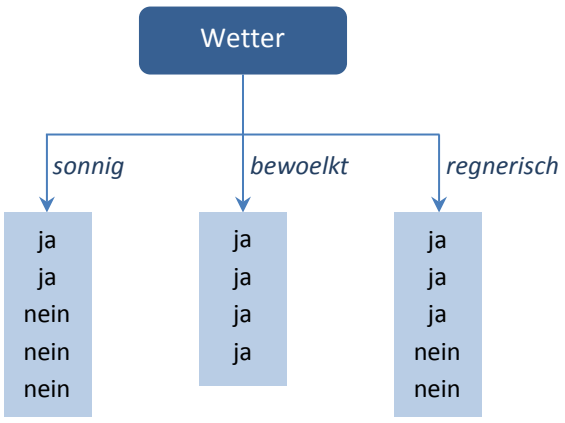


Abbildung 4-1: Das Attribut Wetter als Wurzelknoten

Es wird nun ein neues Attribut als *Knoten* gewählt (z.B. *Temperatur*), um mit dessen Attribut-Werten die Untereinheiten weiter aufzuteilen. Dieser Vorgang wird solange fortgesetzt, bis alle Attribute verbraucht wurden (jedes kommt nur einmal vor!), oder bis innerhalb einer Untereinheit nur noch eine einzige Klasse vorkommt. In diesem Fall wird die Untereinheit als *Blatt* bezeichnet und der *Ast* (= alles vom Wurzelknoten bis zum Blatt) des Entscheidungsbaumes terminiert.

Die 5 Instanzen, mit *Wetter = sonnig*, lassen sich also z.B. durch das Attribut *Temperatur* in drei Untereinheiten aufteilen. Zwei dieser Untereinheiten (*heiss* und *kalt*) sind Blätter. Die Untereinheit mit dem Attribut-Wert *Temperatur = mild*, müsste durch ein weiteres Attribut unterteilt werden. Der abgebildete Entscheidungsbaum würde also Instanzen mit den Attribut-Werten *Wetter = sonnig* und *Temperatur = heiss* in die Klasse *nein* einteilen, unabhängig davon, welche anderen Attribut-Werte sonst noch vorhanden sind.

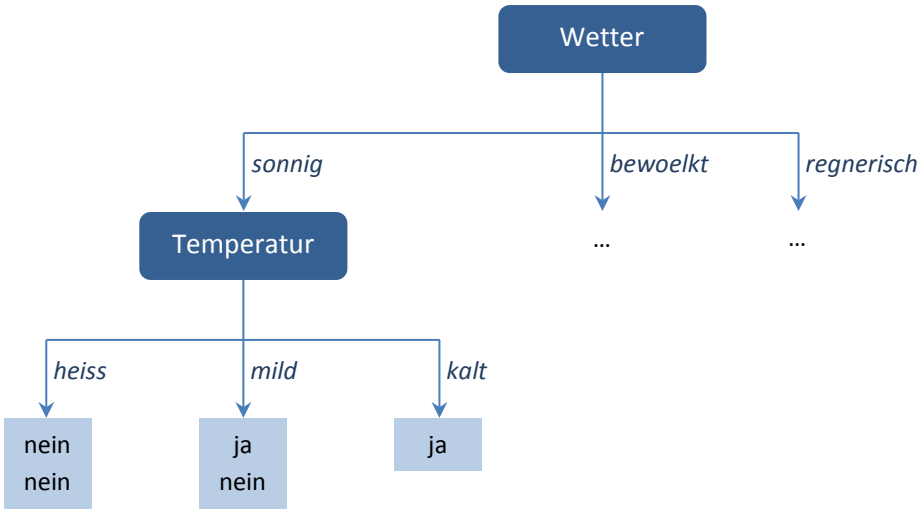


Abbildung 4-2: Entscheidungsbaum mit dem Wurzelknoten *Wetter* und dem Knoten *Temperatur*

Im obigen Beispiel wurde der Wurzelknoten *Wetter* und der Knoten *Temperatur* willkürlich ausgesucht, die Wahl hätte aber an beiden Stellen auf ein anderes Attribut fallen können, z.B. auf *Luftfeuchtigkeit*. Dies hätte zu einem kompakteren Entscheidungsbaum geführt, da die beiden entstehenden Untereinheiten je aus nur einer Klasse bestehen, damit Blätter sind und den Ast des Entscheidungsbaumes abschliessen würden.

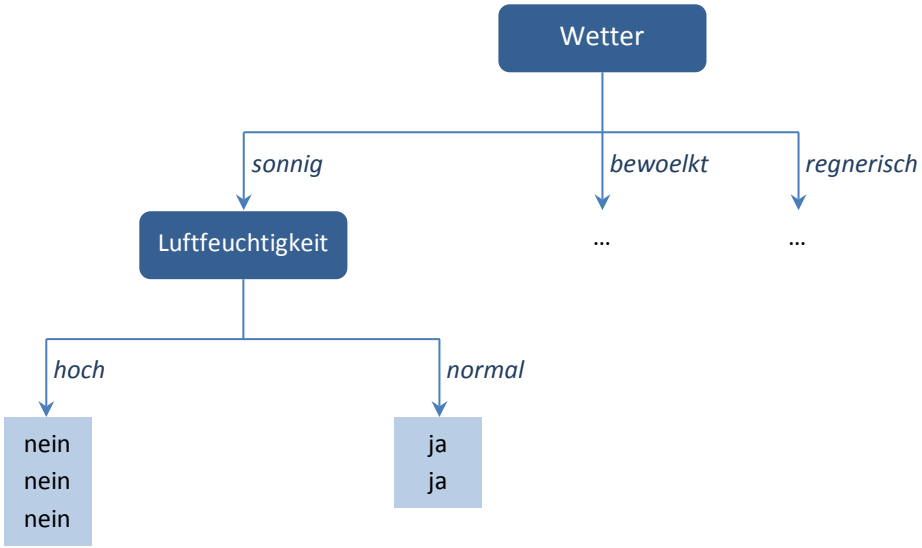


Abbildung 4-3: Der Entscheidungsbaum wird kompakter mit dem Knoten *Luftfeuchtigkeit*

Es stellt sich also die Frage, wie man die Attribute für die Knoten am geschicktesten wählt, so dass der Entscheidungsbaum möglichst kompakt wird. Das Ziel besteht also darin, möglichst schnell zu „klassenreinen“ Untereinheiten zu kommen. Unten sind alle vier möglichen Wurzelknoten für das «Wetter-Problem» abgebildet. Welcher Wurzelknoten führt wohl zum kompaktesten Entscheidungsbaum? Überlegen Sie!

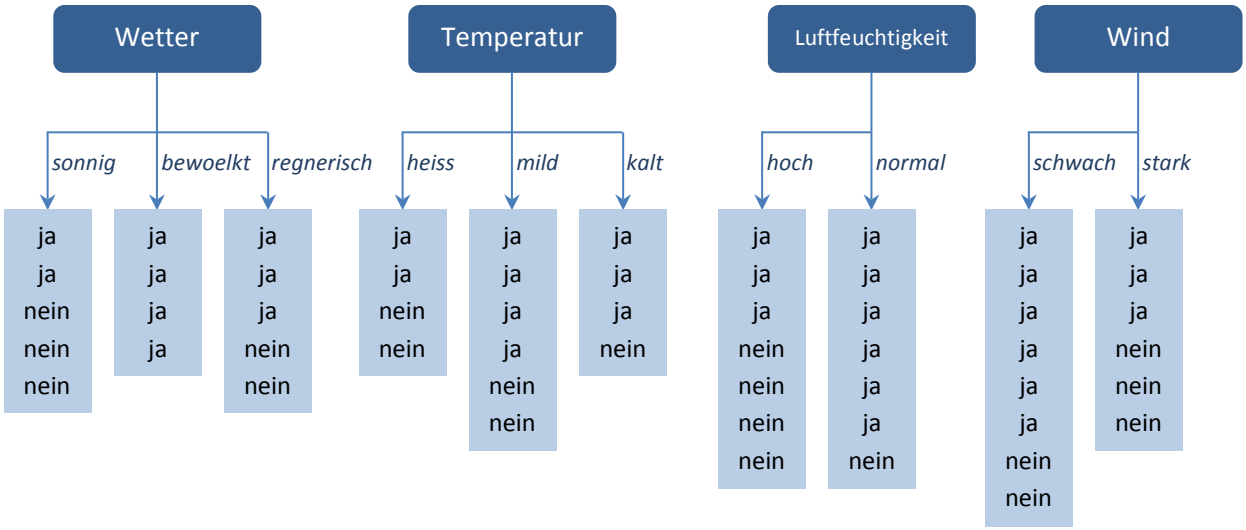


Abbildung 4-4: Alle möglichen Wurzelknoten für das Wetter-Problem

Für das «Wetter-Problem» müsste sich das Attribut *Wetter* wohl am ehesten für den Wurzelknoten eignen, da es als einziges Attribut bereits zu einer klassenreinen Untereinheit führt. Die Fähigkeit eines Attributes, zu möglichst klassenreinen Untereinheiten zu führen lässt sich mit Hilfe des *Informationsgewinns* (engl. *information gain*) messen. Was man darunter zu verstehen hat, erfahren Sie im nächsten Kapitel.

4.1 INFORMATIONSGEWINN (INFORMATION GAIN)

Im Folgenden wird ein Zufallsgenerator betrachtet, der die Symbole A, B, C, oder D mit jeweils gleicher Wahrscheinlichkeit (1/4) ausgibt. Dadurch wird eine zufällige Kette von Symbolen, z.B. BAACBADA… generiert. Diese Symbolkette soll nun binär übermittelt werden, d.h. sie muss in Bits umgewandelt werden. Da ein Bit jeweils nur zwei Werte aufweisen kann (0 oder 1), werden zur Codierung der vier Symbole insgesamt 2 Bit (2 x 2 = 4 Möglichkeiten) benötigt.

Symbol X	A	B	C	D
Wahrscheinlichkeit P(X)	1/4	1/4	1/4	1/4
Codierung	00	01	10	11

Tabelle 4-2: Zufällige Symbolkette aus A, B, C und D (gleiche Wahrscheinlichkeit)

Gemäss der Tabelle oben, würde also die Symbolfolge BAACBADA als 0100001001001100 übermittelt werden. Für den Fall, dass alle Symbole mit derselben Wahrscheinlichkeit auftreten, entspricht die Verwendung von 2 Bit pro Buchstabe der optimalen Codierung. Was aber, wenn die Wahrscheinlichkeiten für A, B, C und D unterschiedlich sind?

Symbol X	A	B	C	D
Wahrscheinlichkeit P(X)	1/2	1/4	1/8	1/8
Codierung	00	01	10	11

Tabelle 4-3: Zufällige Symbolkette aus A, B, C und D (unterschiedliche Wahrscheinlichkeit)

Auch in diesem Fall würde die durchschnittliche Anzahl Bit pro Symbol 2 betragen. Allerdings tritt A doppelt so häufig auf wie B und sogar viermal häufiger als C resp. D. Wenn man möglichst wenige Bits übertragen wollte, müsste man also bestrebt sein, die Datenmenge für A zu senken und im Gegenzug jene für C und D zu erhöhen. Da A viel häufiger auftreten wird als C und D, sollten unter dem Strich weniger Bits übertragen werden müssen. Folgende Verteilung wäre also eine Überlegung wert:

Symbol X	A	B	C	D
Wahrscheinlichkeit P(X)	1/2	1/4	1/8	1/8
Codierung	0	10	110	111

Tabelle 4-4: Asymmetrische Verteilung der Codierung

Natürlich muss darauf geachtet werden, dass die Symbole eindeutig codiert werden. Für B könnte man also z.B. nicht 11 verwenden, da die Folge 110 dann entweder BA oder C bedeuten könnte. Berechnet man die durchschnittliche Anzahl Bits pro übertragenem Symbol, erhält man 1.75 Bit.

$$1/2 \cdot 1 + 1/4 \cdot 2 + 1/8 \cdot 3 + 1/8 \cdot 3 = 1.75 \text{ Bit}$$

$$P(A) \cdot \text{Anzahl Bits (A)} + P(B) \cdot \text{Anzahl Bits (B)} + P(C) \cdot \text{Anzahl Bits (C)} + P(D) \cdot \text{Anzahl Bits (D)}$$

Sobald also nicht mehr alle Symbole mit derselben Wahrscheinlichkeit auftreten, scheint die Menge an Bits abzunehmen, welche übertragen werden müssen. Im Extremfall, dass die Wahrscheinlichkeit für A = 1 ist und für B = C = D = 0 ist, muss nichts (0 Bit) übertragen werden, da klar ist, dass immer A generiert wird.

Symbol X	A	B	C	D	Bit
Wahrscheinlichkeit P(X)	1	0	0	0	**0.00**
Wahrscheinlichkeit P(X)	1/2	1/4	1/8	1/8	**1.75**
Wahrscheinlichkeit P(X)	1/4	1/4	1/4	1/4	**2.00**

Tabelle 4-5: Anzahl Bit pro übertragenem Symbol in Abhängigkeit der Auftretenswahrscheinlichkeit der Symbole

Aufgabe 4-1: Finden Sie für folgendes Beispiel eine Codierung, die mit weniger als den vorgeschlagenen 2.00 Bit auskommt. Theoretisch liesse es sich mit 1.585 Bit realisieren, wir sind aber zufrieden, wenn Sie es mit 1.67 Bit schaffen.

Symbol X	A	B	C	Bit
Wahrscheinlichkeit P(X)	1/3	1/3	1/3	2.00
Codierung (nicht optimal)	00	01	10	

Symbol X	A	B	C	Bit
Wahrscheinlichkeit P(X)	1/3	1/3	1/3	1.67
Codierung (optimiert)				

4.1.1 DIE ENTROPIE H(X)

CLAUDE SHANNON, Begründer der Informationstheorie, hat sich in der ersten Hälfte des letzten Jahrhunderts unter anderem mit der Frage befasst, wie gross die durchschnittliche Anzahl Bit pro übertragenem Symbol mindestens sein muss. Für den allgemeinen Fall kam er zu folgendem Ergebnis:

Symbol X	X_1	X_2	X_3	...	X_n
Wahrscheinlichkeit P(X)	p_1	p_2	p_3	...	p_n

Tabelle 4-6: Asymmetrische Verteilung der Codierung

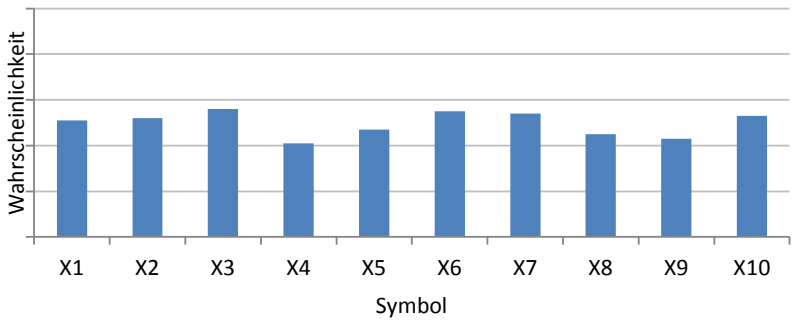
$$H(X) = - p_1 \cdot \log_2(p_1) - p_2 \cdot \log_2(p_2) - \quad ... \quad - p_n \cdot \log_2(p_n) = -\sum_{j=1}^{n} p_j \cdot \log_2\left(p_j\right)$$

H(X) wird als *Entropie* bezeichnet und ist ein Mass für den mittleren Informationsgehalt pro Symbol eines Symbolsatzes X.

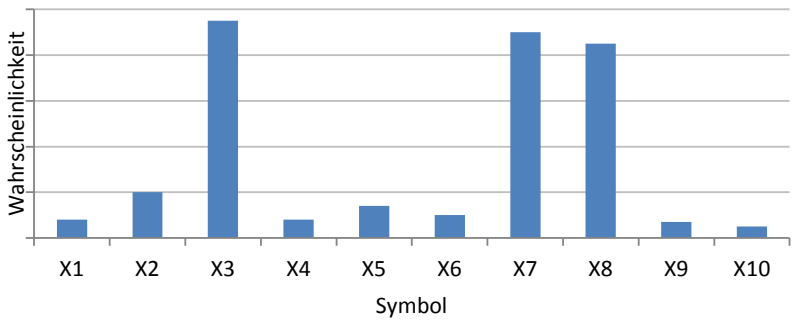
$$H(X) = -\sum_{j=1}^{n} p_j \cdot \log_2\left(p_j\right)$$

Für den Fall, dass eine Wahrscheinlichkeit $p_j = 0$ beträgt, müsste man den binären Logarithmus von 0 berechnen, welcher aber nicht definiert ist. Daher definiert man $0 \cdot \log_2 0 = 0$.

- **Ist der Wert für H(X) gross**, dann sind die Wahrscheinlichkeiten p_1 - p_n, mit welchen die verschiedenen Symbole X_1 - X_n vorkommen, ähnlich gross. Der Symbolsatz X weist also eine uniforme (gleichmässige) Verteilung der Symbole X_1 - X_n auf.



- **Ist der Wert für H(X) klein**, dann unterscheiden sich die Wahrscheinlichkeiten p_1 - p_n, mit welchen die verschiedenen Symbole X_1 - X_n vorkommen, stark. Der Symbolsatz X weist also eine sehr unterschiedliche Verteilung der Symbole X_1 - X_n auf.



4.1.2 DIE BEDINGTE ENTROPIE H(Y|X)

Die *bedingte Entropie* H(Y|X) ist ein Mass für den Informationsgehalt pro Symbol eines Symbolsatzes Y, wenn der Informationsgehalt des Symbolsatzes X bekannt ist.

$$H(Y|X) = -\sum_{j=1}^{n} p_j \cdot H(Y|X_j)$$

Alles klar? Wohl kaum und deshalb betrachten wir dazu ein hypothetisches Beispiel, in welchem untersucht wird, ob acht verschiedene Personen, in Abhängigkeit ihres Wohnkantons, gerne Ski fahren oder nicht.

Wohnkanton = X	ZH	SG	GR	ZH	ZH	GR	SG	ZH
Skifahren = Y	ja	nein	ja	nein	nein	ja	nein	ja

Tabelle 4-7: Bedingte Entropie H(Y|X)

Der Symbolsatz X enthält die drei Symbole *ZH*, *SG* und *GR* mit den Wahrscheinlichkeiten $p_{ZH} = 4/8 = 1/2$ und $p_{SG} = p_{GR} = 2/8 = 1/4$. Somit beträgt die Entropie für X:

H(X) = $- p_{ZH}*\log_2(p_{ZH}) - p_{SG}*\log_2(p_{SG}) - p_{GR}*\log_2(p_{GR}) = -\frac{1}{2}*\log_2(\frac{1}{2}) - \frac{1}{4}*\log_2(\frac{1}{4}) - \frac{1}{4}*\log_2(\frac{1}{4})$ = **1.50 Bit**

Der Symbolsatz Y enthält die beiden Symbole *ja* und *nein*. Da sowohl *ja* als auch *nein* gleich oft vorkommen (je 4 Mal), ist die Wahrscheinlichkeit $p_{ja} = p_{nein} = 4/8 = 1/2$ und somit beträgt die Entropie:

H(Y) = $- p_{ja}*\log_2(p_{ja}) - p_{nein}*\log_2(p_{nein}) = -\frac{1}{2}*\log_2(\frac{1}{2}) - \frac{1}{2}*\log_2(\frac{1}{2})$ = **1.00 Bit**

Man kann sich nun die Frage stellen, wie gross die Entropie von Y ist, wenn X = *ZH* gewählt wird. Dazu existieren nur noch 4 Einträge, wovon zwei *ja* und ebenfalls zwei *nein* entsprechen.

H(Y|X_{ZH}) = $- p_{ja}*\log_2(p_{ja}) - p_{nein}*\log_2(p_{nein}) = -\frac{1}{2}*\log_2(\frac{1}{2}) - \frac{1}{2}*\log_2(\frac{1}{2})$ = **1.00 Bit**

Für die anderen Fälle ergibt sich analog:

H(Y|X_{SG}) = $- p_{ja}*\log_2(p_{ja}) - p_{nein}*\log_2(p_{nein}) = - 0*\log_2(0) - 1*\log_2(1)$ = **0.00 Bit**

H(Y|X_{GR}) = $- p_{ja}*\log_2(p_{ja}) - p_{nein}*\log_2(p_{nein}) = - 1*\log_2(1) - 0*\log_2(0)$ = **0.00 Bit**

Bei den beiden letzten Werten ist die Entropie also minimal, d.h. wenn beide Seiten wissen welchen Wohnkanton die Person hat, muss keine zusätzliche Information mehr übermittelt werden, da klar ist, dass von den beiden St. Gallern sowieso keiner Ski fährt. Bei den Bündnern ist der Ausgang ebenfalls klar, da beide Skifahren. Nur bei den Zürchern muss noch zusätzliche Information übermittelt werden, da von diesen vier Personen nur die Hälfte gerne Ski fährt. Aus diesen Werten lässt sich nun die bedingte Entropie H(Y|X) berechnen:

$$\mathbf{H(Y|X)} = -\sum_{j=1}^{n} p_j \cdot H(Y|X_j) = p_{ZH} \cdot H(Y|X_{ZH}) + p_{SG} \cdot H(Y|X_{SG}) + p_{GR} \cdot H(Y|X_{GR}) = \frac{1}{2} \cdot 1.00 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 0 = \mathbf{0.50\ Bit}$$

Bezogen auf das Beispiel erhalten wir dadurch eine Aussage darüber, wie die Verteilung von *ja* und *nein* ist, wenn die Verteilung der Wohnkantone X bekannt ist. Für den Fall, dass auch alle Zürcher skifahren würden, hätten wir ein H(Y|X) = 0.00 Bit und damit keine Verteilung. Das würde heissen, dass sobald der Wohnkanton einer Person bekannt ist, auch automatisch klar ist, ob sie gerne Ski fährt oder nicht.

4.1.3 DER INFORMATIONSGEWINN IG(Y|X)

Aus der Differenz zwischen der Entropie H(Y) und der bedingte Entropie H(Y|X), lässt sich nun der Informationsgewinn IG(Y|X) berechnen.

$$IG(Y|X) = H(Y) - H(Y|X)$$

Der Informationsgewinn entspricht der Menge an Bits, die bei der Übermittlung von H(Y) eingespart werden können, wenn beide Kommunikationspartner X kennen. Wie es der Name «Informationsgewinn» suggeriert, gewinnt man also durch das Bekanntwerden von X zusätzliche Information. Je grösser IG, desto mehr Information wird durch X beigesteuert. Angewendet auf das Skifahren-Beispiel des vorigen Kapitels ergibt sich:

H(Y) $= - p_{ja}*\log_2(p_{ja}) - p_{nein}*\log_2(p_{nein}) = - ½*\log_2(½) - ½*\log_2(½) =$ **1.00 Bit**

H(Y|X) $= -\sum_{j=1}^{n} p_j \cdot H(Y|X_j) = p_{ZH} \cdot H(Y|X_{ZH}) + p_{SG} \cdot H(Y|X_{SG}) + p_{GR} \cdot H(Y|X_{GR}) = \frac{1}{2} \cdot 1.00 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 0 =$ **0.50 Bit**

IG(Y|X) $= H(Y) - H(Y|X) =$ **0.50 Bit**

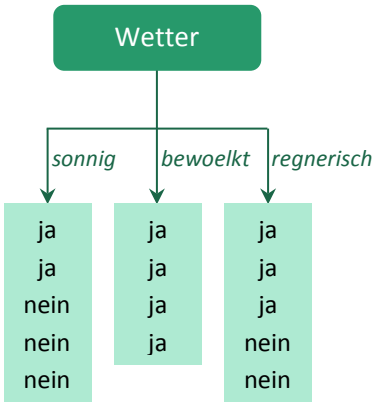
Ist also beiden Kommunikationspartnern der Wohnkanton X bekannt, können Sie sich die Hälfte der zu übertragenden Information sparen.

Auch im Data-Mining findet der Informationsgewinn Verwendung. So lässt sich der Einfluss eines Attributs auf die Klasse bestimmen. Möchten wir z.B. eine Voraussage über die *Lebenserwartung* (= Klasse) einer Person machen und haben wir dazu die *Attribute Haarfarbe*, *Raucher*, *Geschlecht* und *AHV-Nummer* zur Verfügung, lassen sich folgende IG's berechnen:

IG(*Lebenserwartung*|*Haarfarbe*) = 0.01 Bit
IG(*Lebenserwartung*|*Raucher*) = 0.20 Bit
IG(*Lebenserwartung*|*Geschlecht*) = 0.25 Bit
IG(*Lebenserwartung*|*AHV-Nummer*) = 0.00 Bit

Daraus lässt sich schliessen, dass das Attribut *Geschlecht*, am meisten Information dazu liefert, wie alt eine Person werden kann, dicht gefolgt von der Frage, ob jemand *Raucher* ist. Die *AHV-Nummer* scheint hingegen keine Information zu liefern, mit der sich etwas über die *Lebenserwartung* einer Person aussagen lässt.

Aufgabe 4-2: Berechnen Sie den Informationsgewinn IG(Anlass|Wetter) für den dargestellten Wurzelknoten des «Wetter-Problems».



Für die Erstellung von Entscheidungsbäumen, ist der Informationsgewinn IG eine entscheidende Grösse. Mit seiner Hilfe lässt sich die von jedem Attribut (Knoten) die Fähigkeit messen, zu möglichst klassenreinen Untereinheiten zu führen. Dabei werden vom Algorithmus im ersten Schritt die Informationsgewinne aller Attribute bestimmt, wobei jenes mit dem höchsten Informationsgewinn die Aufgabe des Wurzelknotens übernimmt. Für das «Wetter-Problem» ergibt sich folgendes Bild:

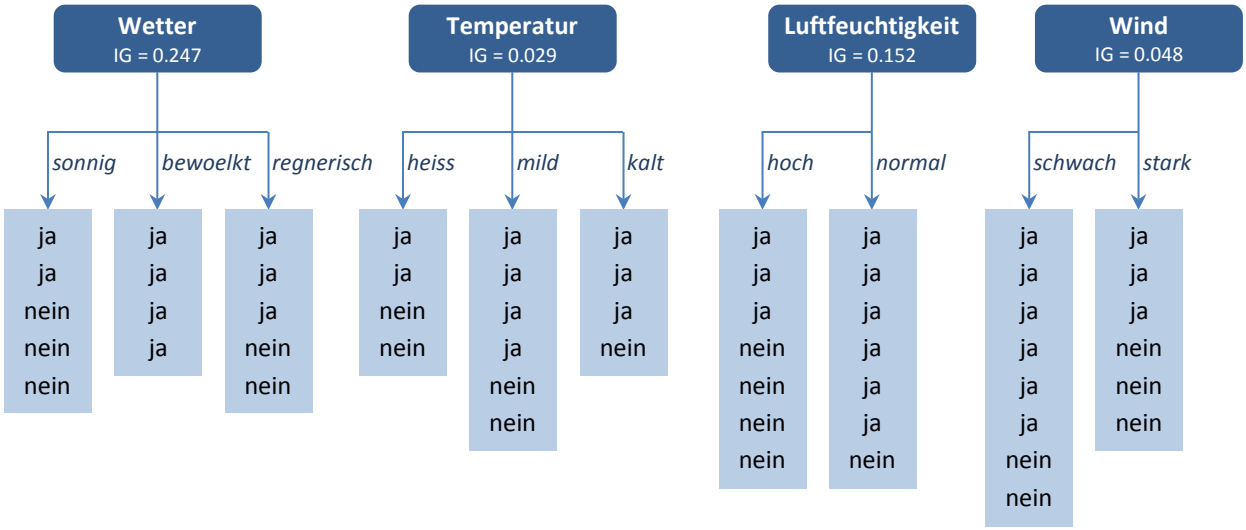


Abbildung 4-5: Informationsgewinn (IG) aller möglichen Wurzelknoten für das Wetter-Problem

Offenbar eignet sich also das Attribut *Wetter* mit einem Informationsgewinn von 0.247 Bit am besten für den Wurzelknoten. Dies liegt insbesondere daran, dass es als einziges Attribut bereits zu einer klassenreinen Untereinheit führt. Die Informationsgewinne lassen sich mit Hilfe von Weka einfach berechnen. Dazu wird unter **REPROCESS** der gewünschte Datensatz geladen und anschliessend zum Reiter **SELECTED ATTRIBUTES** gewechselt. Dort lässt sich über **CHOOSE** ❶ der Punkt *InfoGainAttributeEval* auswählen. Weka wählt dann für ❷ automatisch den *Ranker*. Nach dem Einstellen der gewünschten Klasse ❸, kann über **START** die Berechnung der Informationsgewinne sämtlicher Attribute eingeleitet werden. Die Ausgabe (IG und Rangfolge) erfolgt im Protokollfenster ❹.

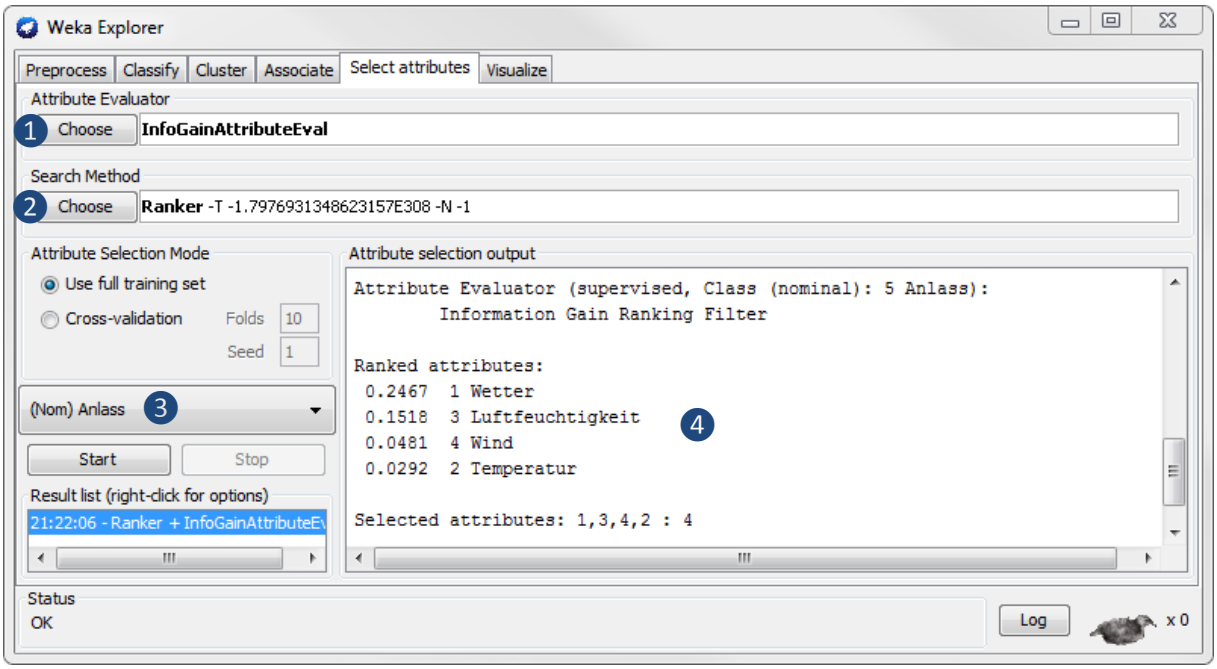


Abbildung 4-6: Informationsgewinn (IG) berechnen mit Weka

Wird für den Wurzelknoten das Attribut *Wetter* gewählt, so führt dies unter anderem zur Untereinheit *sonnig*, welche mehrklassig ist und folglich weiter unterteilt werden muss. Dazu stehen die restlichen drei Attribute zur Verfügung. Auch in diesem Fall entscheidet der Informationsgewinn darüber, welches dieser Attribute schlussendlich den Knoten bildet.

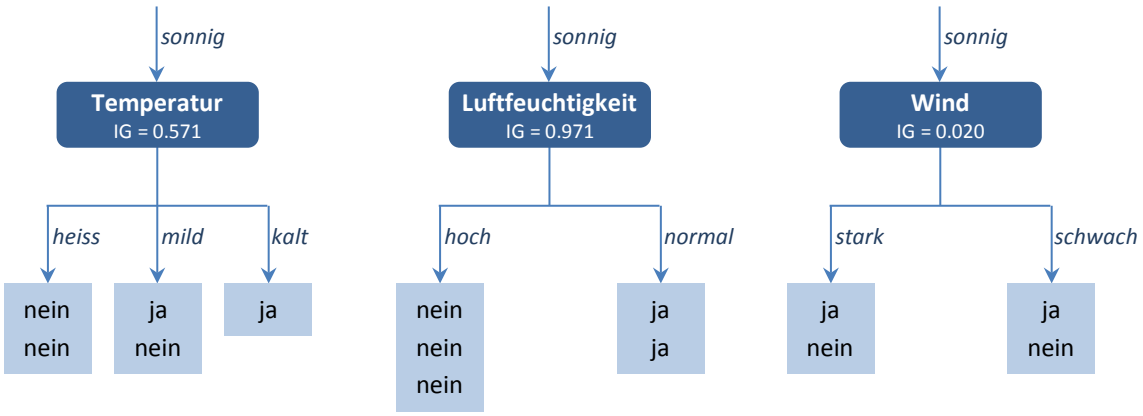


Abbildung 4-7: Informationsgewinn (IG) aller möglichen Knoten für die Untereinheit *sonnig*

Das Attribut *Luftfeuchtigkeit* wird in diesem Fall den Knoten bilden und schliesst den Ast gleichzeitig ab, da es zu zwei klassenreinen Untereinheiten führt.

Im nächsten und zugleich letzten Schritt wird nun noch der beste Knoten für die Untereinheit *regnerisch* gesucht. Von den zwei verbleibenden Attributen weist dabei *Wind* den höchsten Informationsgewinn auf, führt zu zwei klassenreinen Untereinheiten und schliesst damit den gesamten Entscheidungsbaum ab.

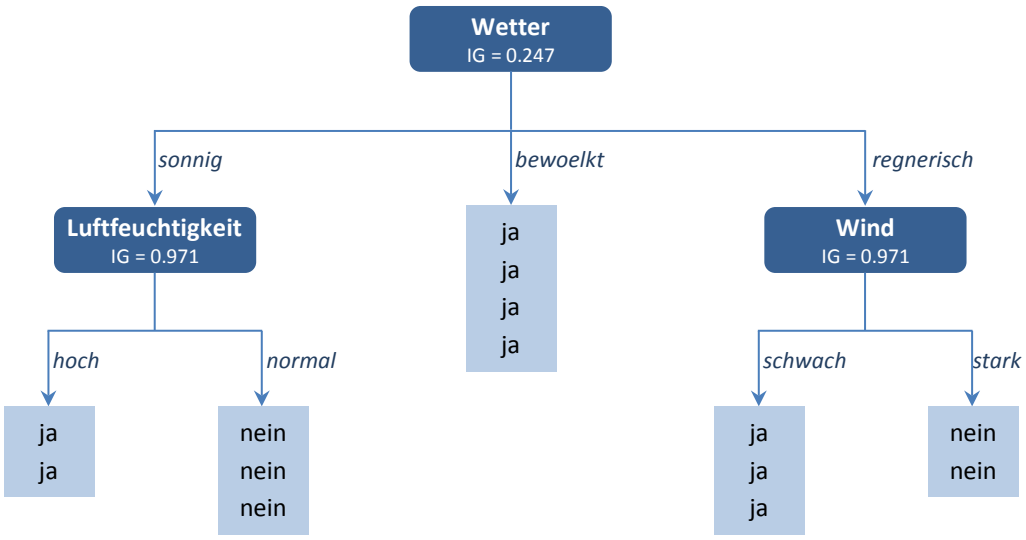


Abbildung 4-8: Fertiger Entscheidungsbaum für das Wetter-Problem

4.2 WEITERE ÜBUNGEN ZUM ENTSCHEIDUNGSBAUM

Aufgabe 4-3: *Laden Sie den Datensatz WetterProblem.csv und lassen Sie Weka den Entscheidungsbaum dazu lernen. Wählen Sie dazu unter dem Reiter **CLASSIFY** über **CHOOSE** den Classifier: trees » J48. Verwenden Sie als Test-Set das Trainings-Set*

*Der Entscheidungsbaum lässt sich darstellen, indem Sie in der **RESULT-LIST** (unten links) einen Rechtsklick auf den Eintrag machen und «Visualize tree» auswählen.*

Aufgabe 4-4: *Laden Sie mit Weka den Datensatz Iris.csv und klassifizieren Sie ihn mit Hilfe des Entscheidungsbaum-Algorithmus J48 (findet sich unter trees » J48).*

Stellen Sie den Entscheidungsbaum dar und vergleichen Sie die Resultate dieser Klassifikations-Methode mit OneR und NaiveBayes.

Aufgabe 4-5: *Laden Sie mit Weka den Datensatz auto-lkm.csv. Dabei handelt es sich um einen Datensatz, welcher 392 verschiedene Autos mit den folgenden Attributen beschreibt:*

- *Name des Autos*
- *Herkunft (Asien, Europa, Nordamerika)*
- *Baujahr (70-74, 75-78, 79-83)*
- *Zylinder (3,4,5,6,7,8)*
- *Hubraum (gering, mittel, hoch)*
- *PS (gering, mittel, hoch)*
- *Gewicht (gering, mittel, hoch)*
- *Beschleunigung (gering, mittel, hoch)*
- *Treibstoffverbrauch (mehr als 10 Liter, weniger als 10 Liter).*

Das Attribut «Treibstoffverbrauch» beschreibt, wie viele Liter Treibstoff ein Auto pro 100 km Fahrstrecke verbraucht und weist die beiden Werte «mehr als 10 Liter» resp. «weniger als 10 Liter» auf.

1) *Klassifizieren Sie die Daten nach dem «Treibstoffverbrauch» mit Hilfe des Entscheidungsbaum-Algorithmus J48. Wählen Sie dazu als Test-Option «Cross-Validation» aus. Linksklicken Sie die Classifier-Parameter «J48 -C 0.25 -M 2» und setzen Sie minNumObj = 4.*
   Lassen Sie sich von Weka den Entscheidungsbaum anzeigen und machen Sie eine Aussage über die Qualität des Classifiers.
2) *Klassifizieren Sie den Datensatz nun auch mit NaiveBayes und mit OneR und verwenden Sie bei beiden Methoden ebenfalls die Kreuzvalidierung als Test-Option. Vergleichen Sie die Qualität der Classifier mit jener von J48.*
3) *Experimentieren Sie mit verschiedenen Werten für minNumObj beim J48-Algorithmus. Was bewirkt dieser Parameter?*

5 CLUSTER-ANALYSE

Suchen wir mit einer Suchmaschine den Begriff „Decke“, so erhalten wir Treffer wie „Microfaser-Decke“, „unter der Decke“, aber auch „Stuck an der Decke“, „wie streiche ich eine Decke“, etc. Die erhaltene Menge der Textdokumente handelt es sich um zwei deutlich voneinander unterscheidbare Cluster. Google listet zum jetzigen Zeitpunkt die Suchresultate noch unstrukturiert. Besser wäre, wenn die Treffer von der Suchmaschine in Cluster unterteilt würden. In der Regel interessieren bei einer Recherche nur Treffer innerhalb eines Clusters.

Besonders beim Clustering ist, dass die Trainingsdaten nicht klassifiziert sind. Es hat somit keine Vorstrukturierung der Daten stattgefunden. Und um das Finden dieser Strukturen geht es genau beim Clustering. Im Raum der Trainingsdaten sollen Häufungen von Daten gefunden werden (Abb. 5-1).

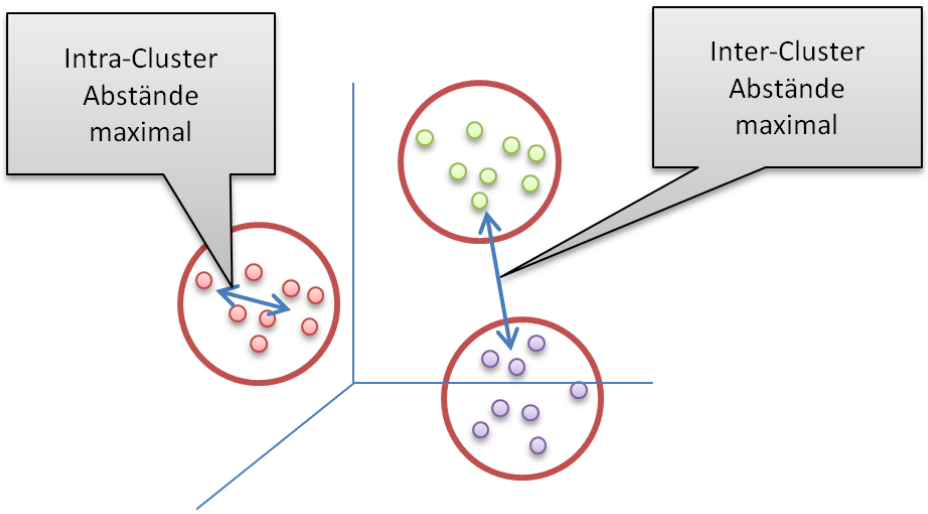


Abbildung 5-1: Cluster

Ziel der Cluster-Analyse ist es, die Datenobjekte eines Datensatzes in Gruppen (Cluster) einzuteilen, die sinnstiftend (die natürliche Struktur der Datenobjekte widerspiegelnd, um diese zu verstehen) oder nützlich (d.h. für andere Zwecke, z.B. einer Zusammenfassung von Datenobjekteigenschaften) sind.

BEISPIELE FÜR „SINNSTIFTENDES CLUSTERING“:

Biologie

- Erarbeitung einer Taxonomie über Species: *Art, Familie, Ordnung, Klasse, Stamm, Reich*
- Clustern von Genen mit jeweils ähnlichen Funktionen

Information Retrieval

- effizientes Suchen im WWW durch Clustern der Suchbegriffe, z.B. *Film* in die Cluster *Reviews*, *Trailer* und *Stars*

Klimaforschung

- durch Cluster-Analyse wurde der Einfluss der Luftdruckverhältnisse in den Polarregionen und über den Weltmeeren auf das Klima an Land aufgedeckt

Medizin und Psychologie

- pathologische Erscheinungen haben gewöhnlich mehrere Ausprägungen, solche Sub-Kategorien (z.B. Typen von Depressionen) sind durch Cluster- Analyse identifiziert worden

Wirtschaft

- Kunden können durch Cluster-Analyse partitioniert werden in Gruppen mit ähnlichen Interessen, um das Marketing kundenspezifischer (und damit effektiver) zu machen

BEISPIELE FÜR „NÜTZLICHES CLUSTERING“:

Zusammenfassungen (effiziente Datenverarbeitung)

Viele Analysetechniken haben eine Komplexität $O(n^2)$ oder schlimmer. Clustert man die Daten zuerst und wendet diese Techniken dann nur noch auf Cluster-Prototypen an, werden sie effizienter

Datenkompression

Ähnliches geschieht bei der Vektor-Quantisierung von Bild-, Ton- oder Videodaten. Für jedes Cluster wird nur ein Repräsentant nebst seinen Positionen im Datenstrom abgelegt

Effiziente Ermittlung nächstliegender Nachbarn

Der paarweise Vergleich von Abstandsmassen macht die Ermittlung nächster Nachbarn in großen Datenmengen sehr komplex. Wird dies nur mit Cluster-Prototypen, wird die Ermittlung effizienter.

5.1 UEBERBLICK

Wie bereits erwähnt, ist das Ziel beim Clustering, das Finden von Clustern, welche

- sich signifikant voneinander unterscheiden, d.h. welche einen maximalen Abstand voneinander haben, und
- innerhalb derer sich die DO nicht signifikant unterscheiden („dichtbeieinander“ sind), d.h. einen minimalen Abstand voneinander haben.

Die folgenden Clustering-Aufgaben zeigt keine eindeutige Lösung:

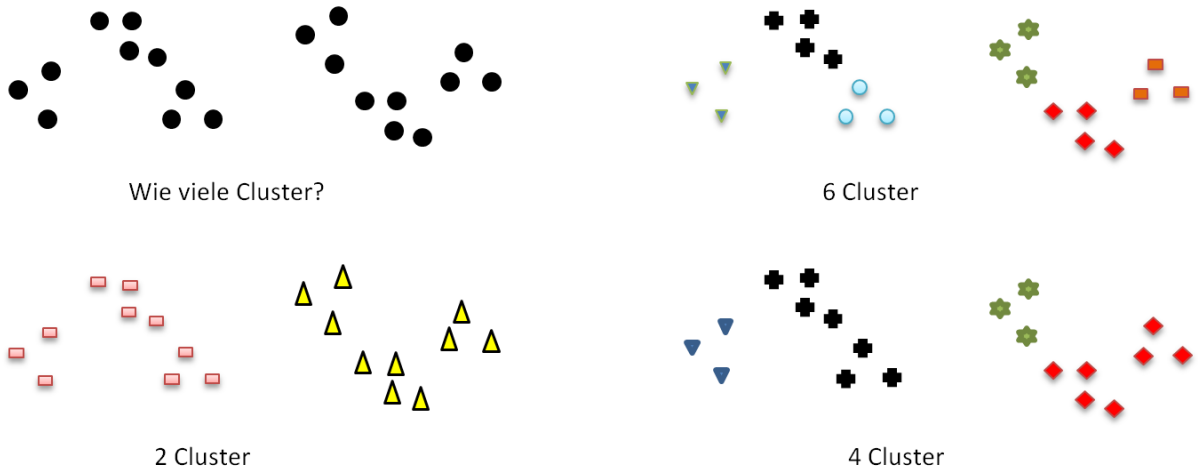


Abbildung 5-2: Clusteraufgabe

Im folgenden Abschnitt wenden wir uns verschiedenen Typen von Clustering zu.

5.1.1 TYPEN VON CLUSTERINGS

HIERARCHISCH VERSUS PARTITIONIERUNG

Partitionierung

Die Datenobjekte werden in disjunkte Teilmengen (Cluster) eingeteilt. Dabei wird jedes Datenelement genau einem Cluster zugeordnet (vgl. Abbildung 5-2).

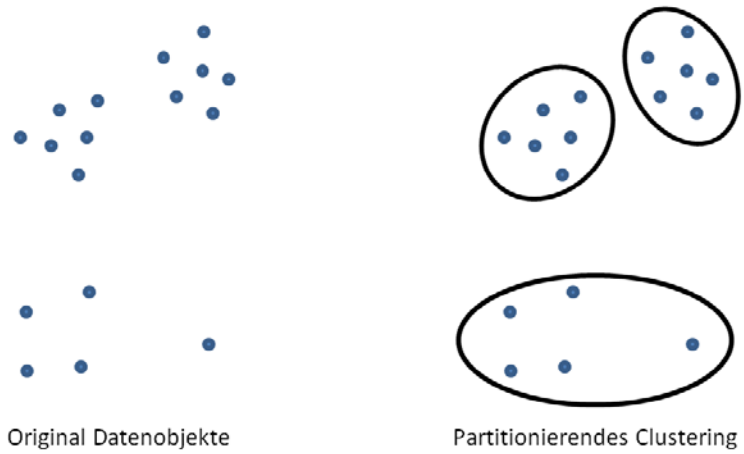


Abbildung 5-3: Partitionierendes Clustering

Hierarchisches Clustern

Eine baumförmige Cluster-Hierarchie wird ermittelt. Dabei ist jede Astgabel eines Baumes eine (weitere) Partitionierung. Jeder Knoten (Cluster) im Baum (mit Ausnahme der Blätter) enthält die Datenobjekte, welche sich aus der Vereinigung der Unterbäume ergibt. Die Cluster 2, 4 und 6 der Abbildung 5-2 könnten auch als baumförmige Cluster hierarchisch organisiert werden. Beim traditionellen Clustering (constrained Clustering) erhalten wir einen binären Baum. Jede Astgabel des Baumes identifiziert ein (fertiges) Cluster (Blattknoten) in einem Unterbaum und die restlichen im anderen Unterbaum.

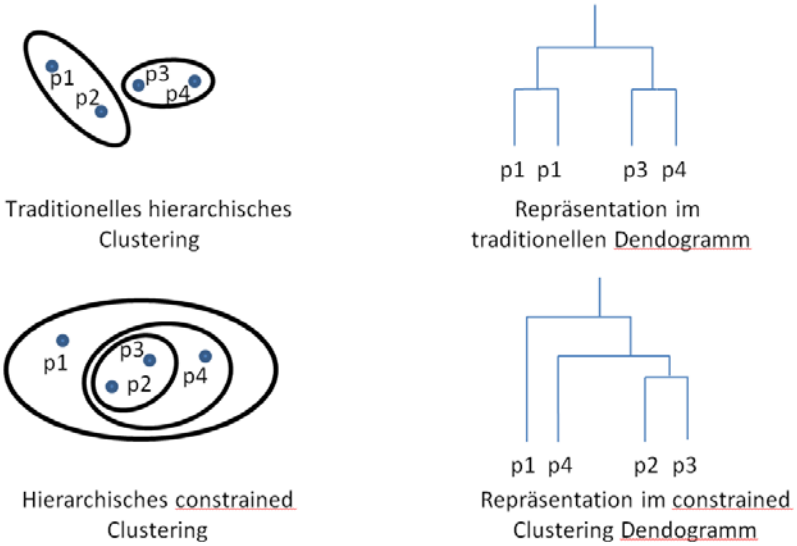


Abbildung 5-4: Hierarchisches Clustering

EXKLUSIV VERSUS ÜBERLAPPEND

Ein exklusives Clustering liegt vor, wenn jedes Datenobjekt genau einem Cluster angehört. Wenn ein Datenobjekt mehreren Clustern zugehört, sprechen wir von überlappendem Clustering.

FUZZY VERSUS SCHARF

Von „fuzzy“ spricht man, wenn jedes Datenobjekt zu jedem Datenobjekt mit einem Gewicht g, von $0 \leq g \leq 1$ gehört und die Summe aller Gewichte für ein Datenobjekt gleich 1 ist. Von „scharf“ sprechen wir, wenn eine scharfe Zugehörigkeit gegeben ist (0 oder 1).

PARTIELL VERSUS TOTAL

Beim partiellen Clustering können einige Datenobjekte in keinem Cluster untergebracht werden. Hingegen kann beim totalen Clustering jedes Datenobjekt mindestens einem Cluster zugeordnet werden.

HETEROGEN VERSUS HOMOGEN

Wenn die Cluster verschieden Grössen, Formen und Dichten aufweisen, sprechen wir von einem heterogenen Clustering. Dagegen sind beim homogenen Clustering die Cluster in Grösse, Form und Dichte identisch.

5.1.2 TYPEN VON CLUSTERN

WOHL – SEPARIERTE CLUSTER

In diesem Fall ist jedes Datenobjekt näher bzw. ähnlicher zu jedem anderen Datenobjekt innerhalb des Clusters als zu jedem Datenobjekt ausserhalb des Clusters. Anders formuliert: Alle Punkte ausserhalb des Clusters sind entfernter, d.h. unähnlicher, als der entfernteste Punkt innerhalb des Clusters.

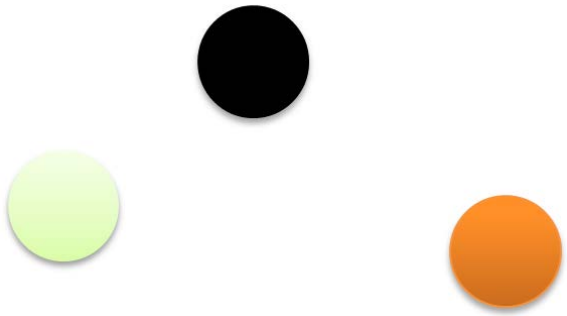


Abbildung 5-5: 3 wohl separiert Cluster

PROTOTYP – BASIERTE CLUSTER (CENTER-BASED CLUSTER)

Jedes Datenobjekt eines Clusters ist näher bzw. ähnlicher dem Zentrum (zum Prototyp – Datenobjekt) seines Clusters als zum Zentrum jedes anderen Clusters. Diese Clusterform hat den Hang zu einer Kugelform. Das Zentrum eines solchen Clusters wird oft als Centroid bezeichnet, wenn es für jedes Attribut den Durchschnitt der Werte aller Datenobjekte des Clusters hat. Wenn es das „repräsentativste“ Datenobjekt des Clusters bildet, wird es Medoid genannt.

Ein Centroid muss kein reales Datenobjekt sein. Dagegen muss ein Medoid eines der Datenobjekte sein.

Abbildung 5-6: 4 Prototyp basierte Cluster

ZUSAMMENHÄNGENDE CLUSTER (CONTIGUITY-BASED CLUSTER)

Jedes Datenobjekt eines Clusters ist zumindest einem Datenobjekt seines Clusters näher bzw. ähnlicher, als zu allen Datenobjekten ausserhalb des Clusters.

Abbildung 5-7: 8 zusammenhängende Cluster

DICHTE – BASIERTE CLUSTER (DENSITY-BASED CLUSTER)

Ein dichtebasiertes Cluster ist eine Region, die sich von angrenzenden Regionen des Raumes durch eine hohe Dichte von Datenobjekten auszeichnet. Die Cluster werden typischerweise durch Gebiete geringer Dichte getrennt. Diese Cluster werden meistens verwendet, wenn Cluster irreguläre Formen haben, Cluster ineinander verflochten sind und verrauschte Daten oder Aussenseiter im Datensatz existieren.

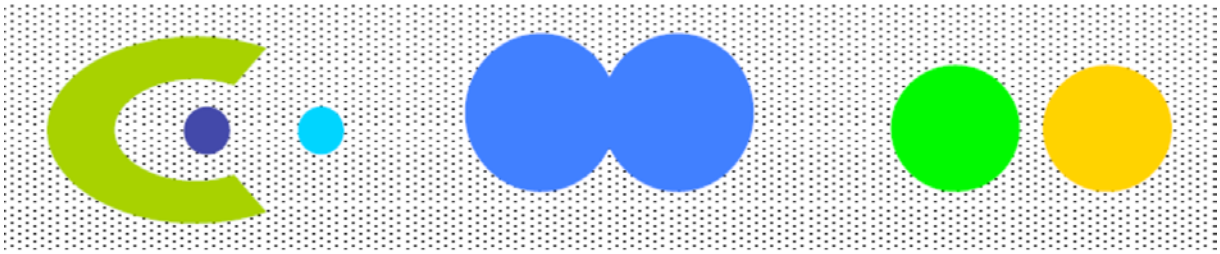


Abbildung 5-8: 6 dichte basierte Cluster

KONZEPTUELLE CLUSTER (CONCEPTUAL CLUSTER, SHARED-PROPERTY CLUSTER)

Ein konzeptuelles Cluster beinhaltet alle oben erwähnten Typen von Clustern. Die Cluster werden als Datenobjekte mit einer gemeinsamen Eigenschaft, welche nicht als Attribut definiert ist, definiert und nicht durch die Verteilung der Datenobjekte im Attributraum. Solche Cluster könne nichtleere Schnittmengen enthalten.

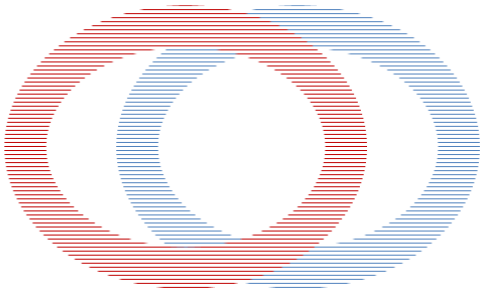


Abbildung 5-9: 2 Konzeptuelle Cluster

Im weiteren Verlauf werden die Clustering-Verfahren K-means und hierarchisches Clustering vorgestellt. Auf das dichtebasierte Clustering mit DBSCAN, dessen Vorteile die geringe Anfälligkeit gegenüber verrauschten Daten und das Erkennen von verschiedenen Clusterformen und –grössen sind, wird in dieser Arbeit nicht eingegangen.

5.2 K – MEANS (K – MEDOID)

Dieses Verfahren ist prototyp-basierend, partitionierend und nicht hierarchisierend.

5.2.1 GRUNDLEGENDER K – MEANS (K – MEDOID) ALGORITHMUS

Dieser funktioniert wie folgt:

1. Zuerst geht es darum (mehr oder wenig willkürlich) *K* Centroiden (Medoiden) festzulegen (*K:* gewünschte Anzahl von Clustern)
2. Als nächstes wird zyklisch
   I. jedes Datenobjekt wird dem nächstgelegenen Centroiden (Medoiden) zugeordnet
      - Nächstgelegen: Euklidischer Abstand, Kosinus-Koeffizient, Korrelation, …
   II. und der Centroid (Medoid) jedes Clusters wird danach neu berechnet
3. … solange, bis sich kein Centroid (Medoid) durch eine Neuberechnung mehr ändert. Und somit kein Datenobjekt seine Clusterzugehörigkeit mehr ändert.

Dieses Verfahren konvergiert typischerweise schnell, d.h. nach wenigen Zyklen. Wenn dies nicht so ist, muss das Stopp-Kriterium aufgeweicht werden. Im Sinne von: „ … solange, bis sich nur noch wenige Centroiden (Medoiden) ändern“.

- Komplexität ***Є o (n * K * I * d)***
  n: Anzahl der Datenobjekte***,*** ***K:*** Anzahl der Centroiden (Medoien), ***I:*** Anzahl der Iterationen, ***d:*** Anzahl der Attribute

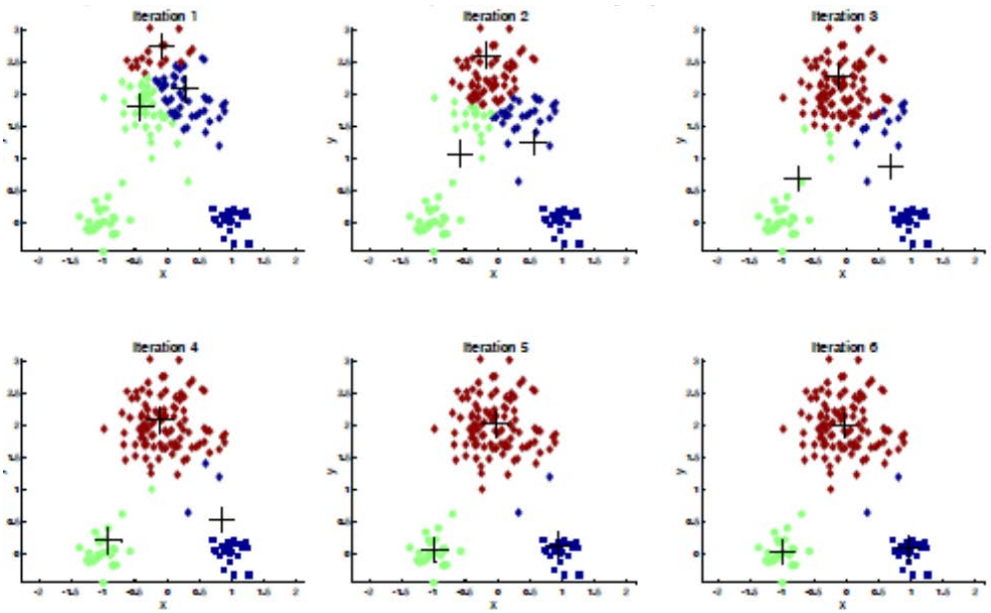


Abbildung 5-10: 2 Attribute, Euklidisches Abstandsmass, 3 Centroiden (+)

Bei der Zuordnung zum nächstgelegenen Centroiden am häufigsten verwendete Aehnlichkeitsmasse sind:

- Euklidischer (L_2) oder Manhatten (L_1) Abstand im Euklidischen Raum
- Kosinus-Koeffizient oder Jaccard Koeffizient für Dokumente

Die Auswahl von Initial-Centroiden ist von grosser Bedeutung. Von dieser Auswahl hängt ab, ob das Verfahren in das globale oder in ein lokales Minimum des SSE (sum of the squared error) konvergiert.

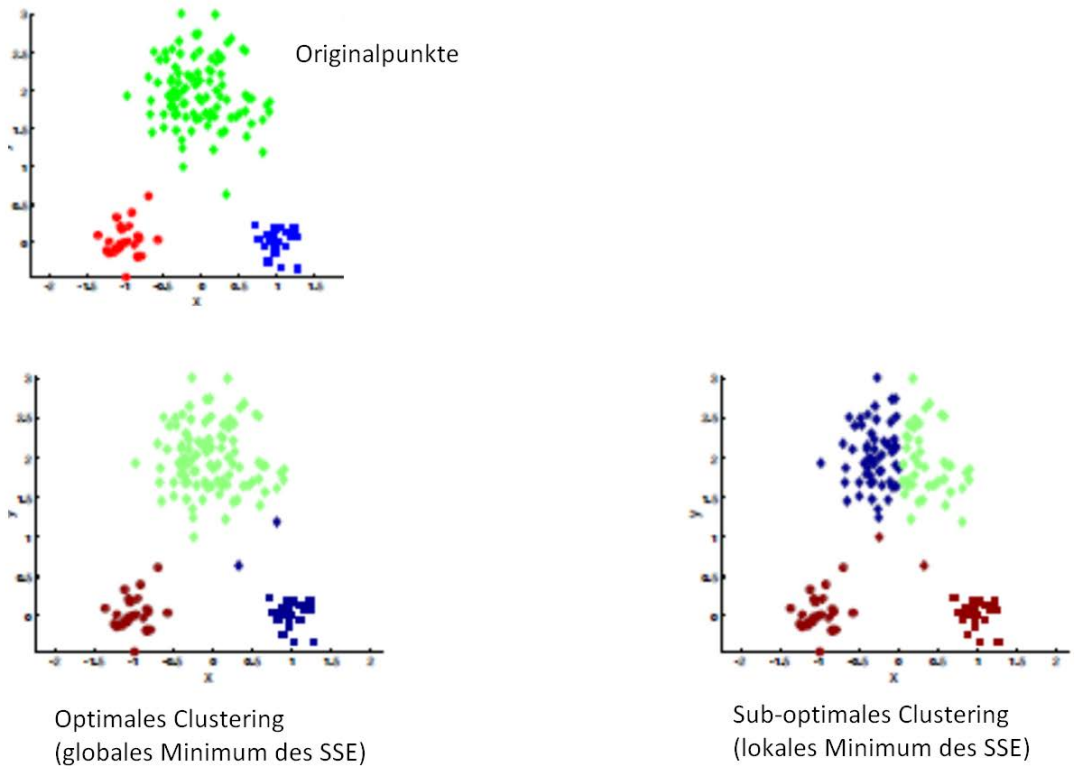


Abbildung 5-11: Auswahl von Centrodien

Ansatz zur Vermeidung lokaler SSE Minima

- wähle den ersten Centroid zufällig oder verwende den Centroiden aller Datenobjekte
- für alle weiteren Centroiden wähle jeweils einen Punkt, der am weitesten entfernt von allen bisher gewählten Centroiden ist

Vorteil: Initial-Centroiden sind wohl separiert
Nachteil: ggf. werden „Ausreißer“ zu Clustern erhoben (statt Regionen mit hoher Dichte)

Ein anderer Ansatz

- führe zunächst ein hierarchisches Clustering durch
- verwende die Centroiden der dabei entstehenden K Cluster and Initial-Centroide des partitionierenden Clusterings

Komplexität

$space \in O(((m + K) \bullet n) \quad time \in O(I \bullet K \bullet m \bullet n)$

mit **m** Anzahl der Datenobjekte
n Anzahl der Attribute
K Anzahl der gewünschten Cluster
I Anzahl der Iterationen (gewöhnlich gering, konvergiert recht schnell)

5.2.2 PROBLEME MIT K – MEANS BEI VERSCHIEDENEN CLUSTERTYPEN

5.2.2.1 CLUSTER VERSCHIEDENER GRÖSSE

Bei grösseren natürlichen Clustern, besteht die Tendenz, dass sie durch K – means aufgeteilt werden.

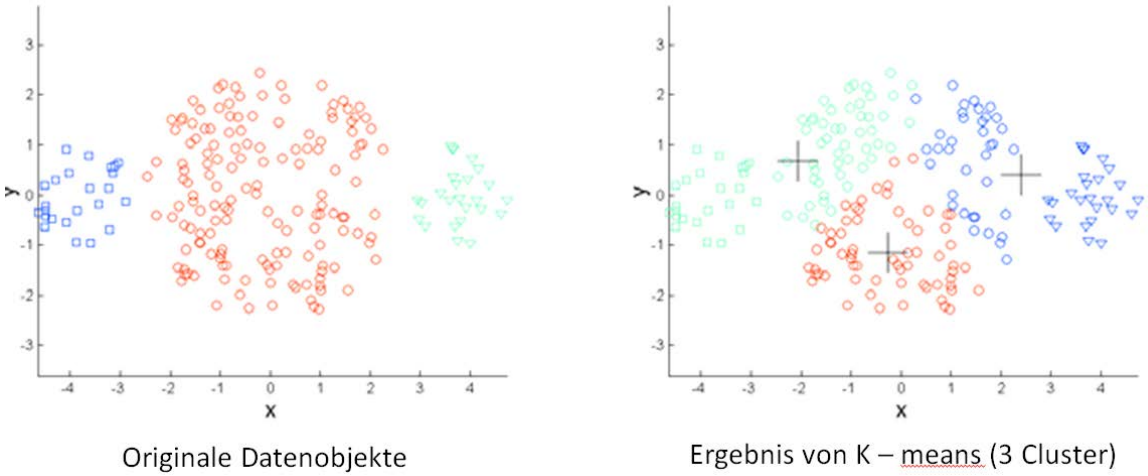


Abbildung 5-12: Aufteilung von natürlichen Clustern

5.2.2.2 CLUSTER VERSCHIEDENER DICHTE

Natürliche Cluster werden aufgrund unterschiedlicher Dichte nicht gefunden.

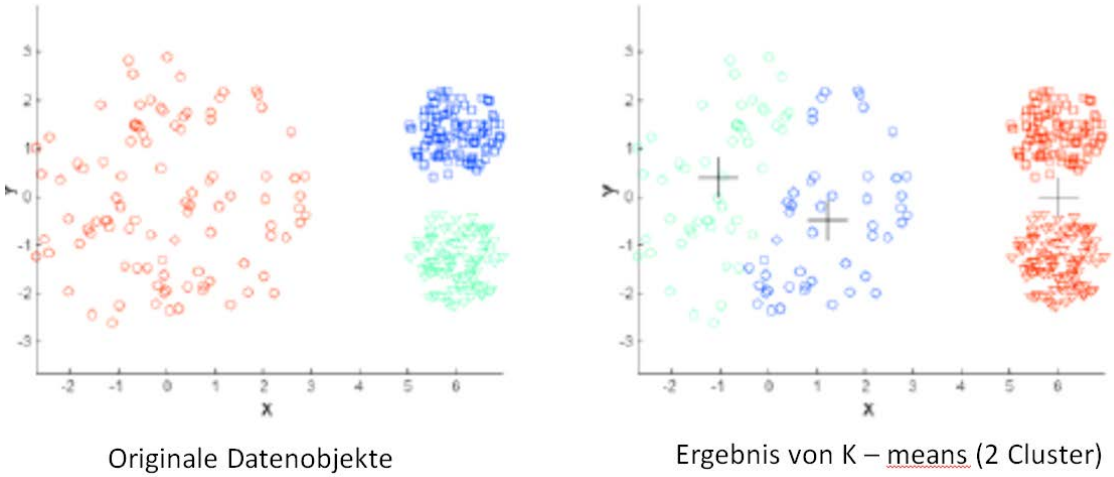


Abbildung 5-13: Probleme mit Clustern unterschiedlicher Dichte

5.2.3 NICHT GLOBULARE CLUSTER

Cluster, die ineinander verwunden und nicht-konvex sind, werden nicht gefunden. Das Zuordnungskriterium „Abstand zum Centroiden“ bewirkt, dass nicht globulare natürliche Cluster nicht erkannt werden.

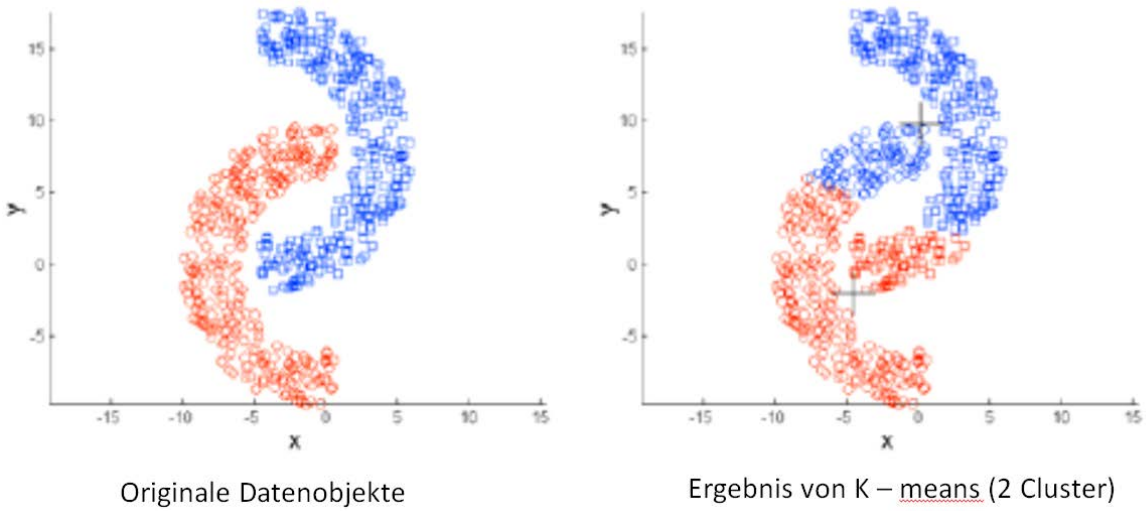


Abbildung 5-14: Verwundene und nicht konvexe Cluster

5.3 HIERARCHISCHES CLUSTERING

Zu den Stärken des hierarchischen Clusterings gehört, dass keine spezielle Anzahl an Clustern vorausgesetzt werden muss. Zudem können die Cluster sinnvollen Taxonometrien entsprechen. Als Beispiele aus der Biowissenschaft können hier die Tierwelt und die stammesgeschichtliche Entwicklung der Lebewesen oder auch Verwandtschaftsgruppen erwähnt werden. Das hierarchische Clustering erzeugt eine Menge baumartig geschachtelter Cluster. Es wird entweder mittels bottom-up oder top-down Verfahren durchgeführt.

Am meisten wird das agglomerative Clustering (bottom up) verwendet. Es beginnt mit einzelnen Datenobjekten als Cluster. Bei jedem Schritt werden diejenigen Datenobjekte als Cluster verschmolzen, die am nächsten beieinander liegen.

Ein selten verwendetes Verfahren ist das divisive Clustering (top down) verwendet. Es startet mit einem Cluster, das alle Datenobjekte enthält. Bei jedem Schritt wird ein Cluster in zwei neue Cluster aufgespalten. Dies geschieht solange, bis die gewünschte Anzahl von Clustern gebildet worden sind. Für die Auswahl der aufzuspaltenden Cluster wird jeweils zuerst das grösste Cluster gewählt und in der dann das Cluster mit dem grössten SSE (sum of he squared error).

VERANSCHAULICHUNG DER CLUSTER-HIERARCHIE

Das hierarchische Clustering erzeugt einen Satz von verschachtelten Clustern. Diese sind baumförmig hierarchisch organisiert. Sie können als Dendrogramm dargestellt werden. Die Höhe der Gabelungen im Dendogramm entspricht dem Abstand der verschmolzenen Cluster.

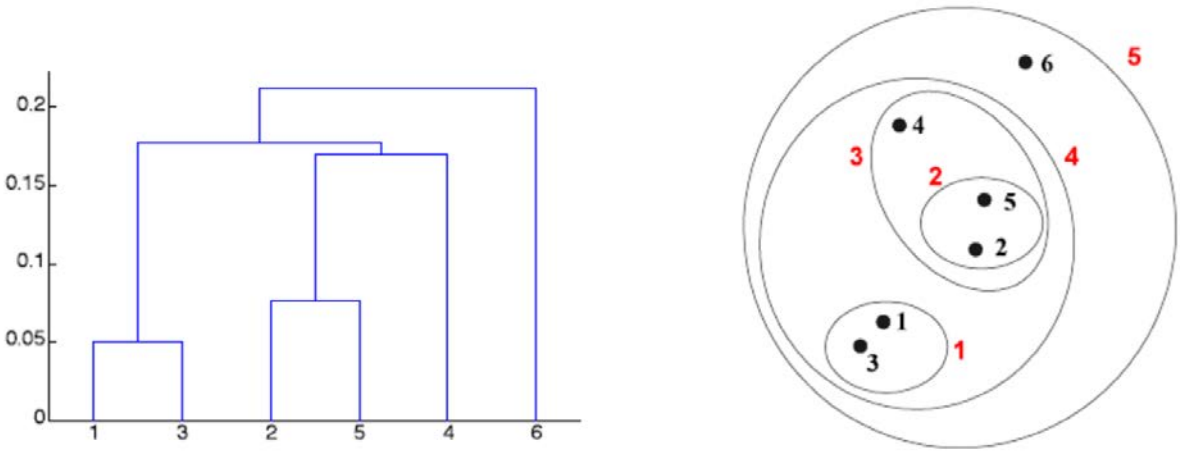


Abbildung 5-15: Dendogramm und Cluster Diagramm

Durch das Schneiden des Dendrogramms auf der entsprechenden Höhe, kann jede gewünschte Anzahl von Clustern erstellt werden.

5.3.1 HIERARCHISCHER CLUSTERING GRUND-ALGORITHMUS

Wie bereits erwähnt, wird jedes Datenobjekte zu Beginn als ein Cluster betrachtet und die beiden am nächsten zusammenliegenden Cluster verschmolzen. Diese wird solange wiederholt, bis nur noch ein Cluster übrig bleibt.

1. errechne eine Distanz-Matrix zwischen den Clustern
2. repeat
    - verschmelze die beiden Cluster, die am nächsten beieinander liegen
    - aktualisiere die Distanz-Matrix entsprechend (in der Zeile und/oder in der Spalte des neuen Clusters
3. until nur noch ein Cluster verfügbar ist

Die Schlüsselfunktion dieses Verfahrens, ist die Berechnung der Distanz zweier Cluster. Dafür gibt es verschiedene Ansätze bzw. Algorithmen.

Zu Beginn haben wir die Datenobjekte als individuelle Cluster mit einer Distanz-Matrix.

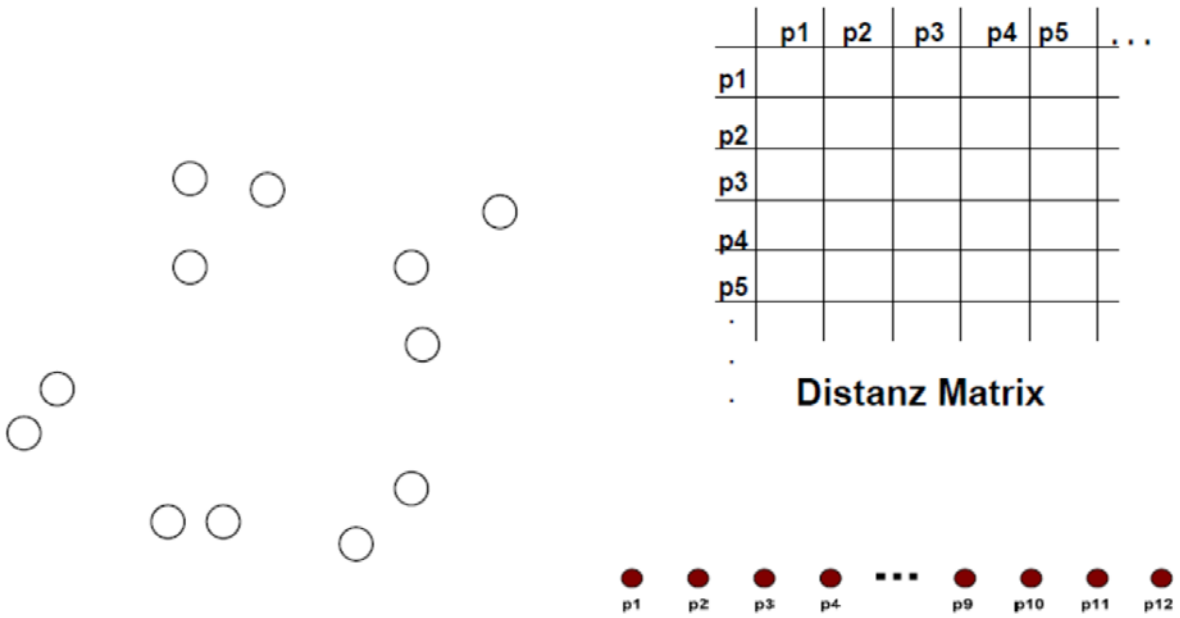


Abbildung 5-16: Individuelle Cluster mit Distanz Matrix

Nach ein paar Iterationen (Verschmelzungsschritten), werden einige Cluster sichtbar.

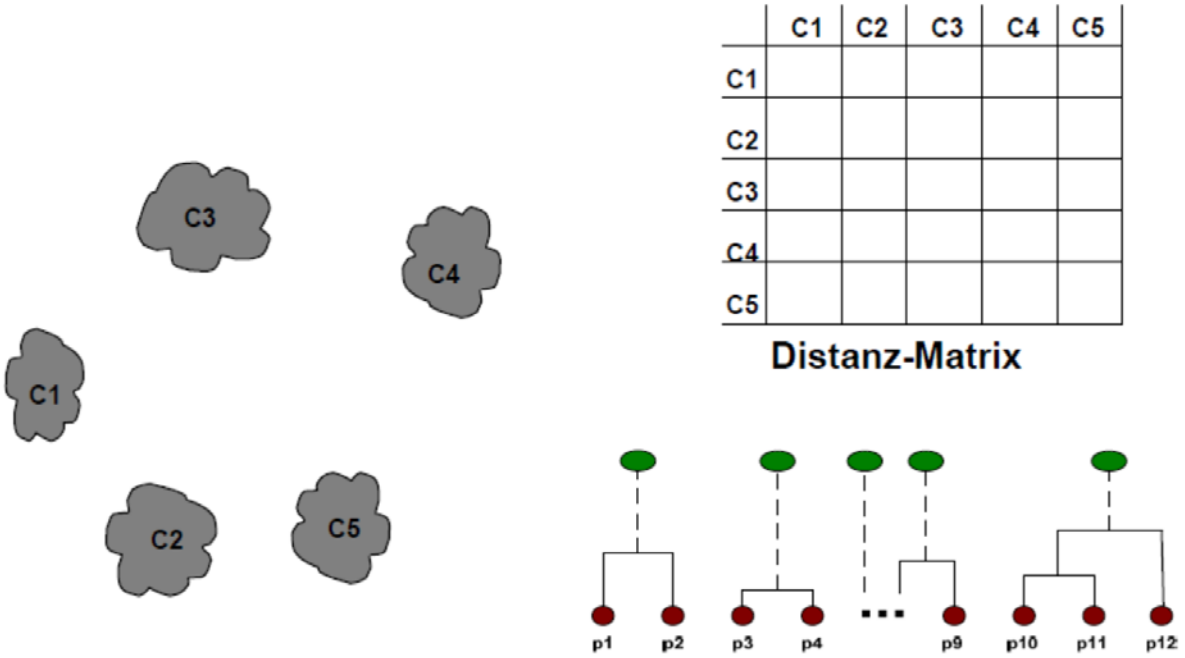


Abbildung 5-17: Situation nach einer Anzahl von Durchläufen

Nach weiteren Iterationen werden 2 nächstgelegene Cluster verschmolzen und die Distanz-Matrix wird aktualiesiert.

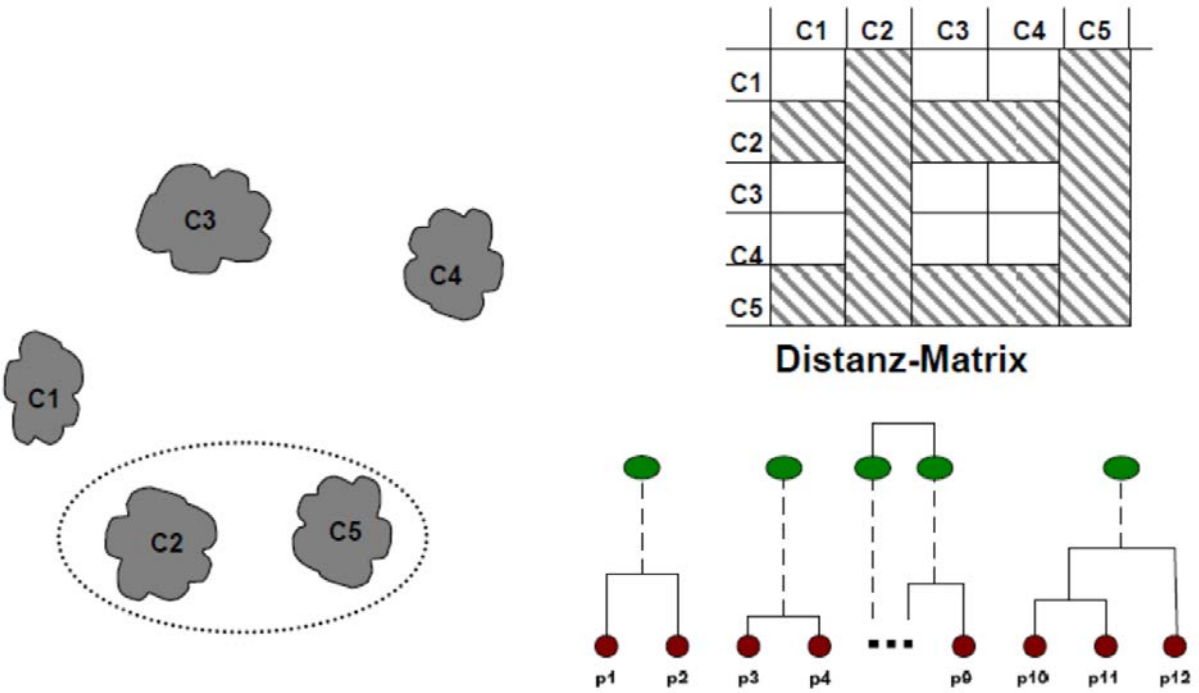


Abbildung 5-18: C2 und C5 verschmelzen

Nach der Verschmelzung von C2 und C5, stellt folgende Frage: Wie soll jetzt die Distanzmatrix aktualisiert werden? Oder anders gefragt: Wie finden wir Aehnlichkeiten zwischen den Clustern?

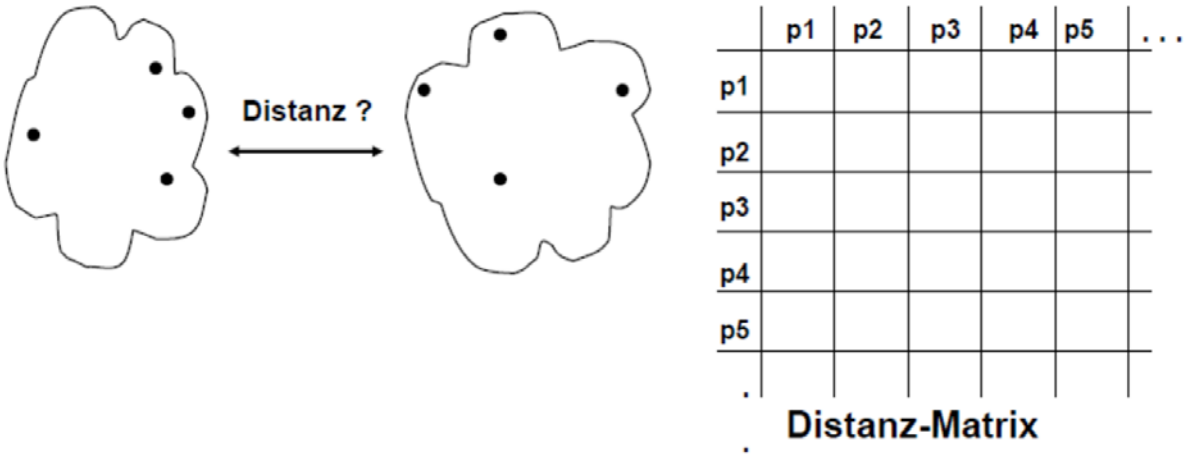


Abbildung 5-19: Distanzermittlung

Ansätze zur Distanzermittlung zwischen 2 Clustern aus dem Distanzmass zweier Datenobjekte:

1. **MIN:** Abstand zwischen den einander nächstgelegenen Datenobjekten der beiden Cluster
2. **MAX:** Abstand zwischen den einander entferntesten Datenobjekten der beiden Cluster
3. **AVERAGE:** Durchschnitt der Abstände jedes Paars von Datenobjekten aus je einem der beiden Cluster
4. **ABSTAND DER CENTROIDEN**
5. **WARD'S METHODE:** Centroid-basiert, Abstand ist definiert durch die Verschlechterung des SSE (sum of squared error) – Verbesserung beim Verschmelzen des Cluster-Paars

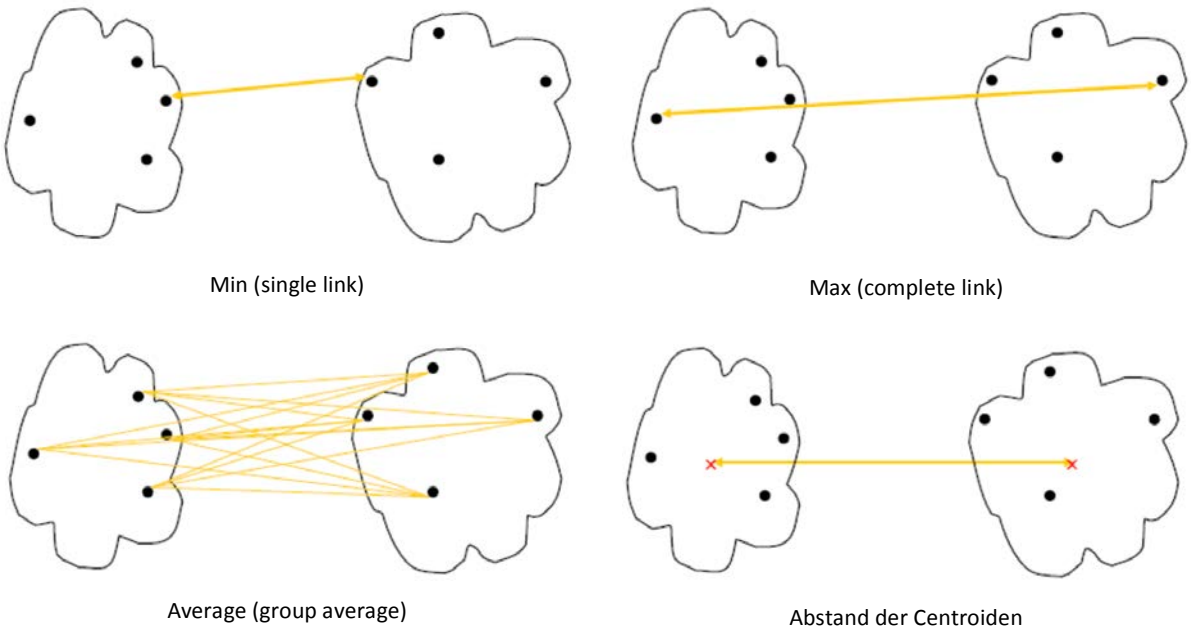
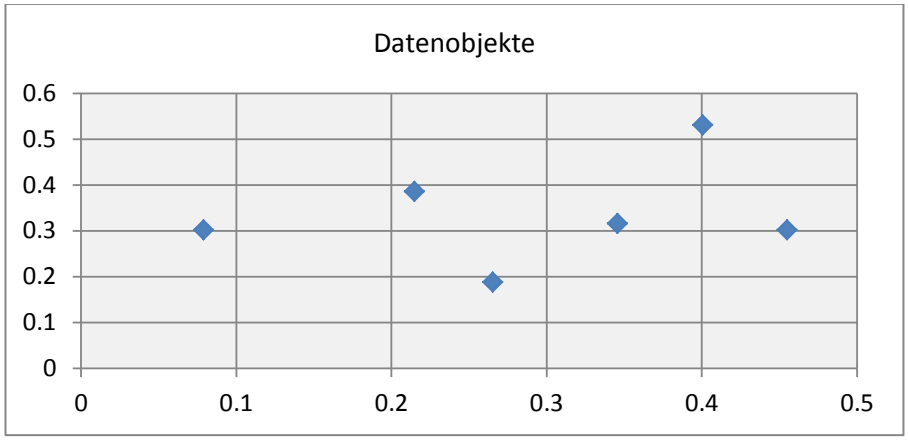


Abbildung 5-20: Verschiedene Ansätze zur Distanzermittlung

Beispiel einer Distanzermittlung mittels euklidischem Abstand:

Datensatz

DO	x	y
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.3022
p6	0.4548	0.3022



Distanz-Matrix mit euklidischem Abstand

	p1	p2	p3	p4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2843	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

Abbildung 5-21: Euklidisicher Abstand

CLUSTERBILDUNG NACH MIN –ANSATZ (SINGLE LINK APPROACH)

Wir ermitteln den geringsten Abstand zwischen p3 und p6:

D(p3, p6) = 0.1100 → erste Dendrogramm-Gabelung bei y = 0.1100

Ein weiteres Beispiel:

Dist({p3,p6}, {p2,p5} = min(d(p3,p2), d(p6,p2), d(p6,p5) = min(0.1483, 0.2843, 0.2540, 0.3921) = 0.1483. Die Dendrogramm-Gabelung liegt somit bei y = 0.1483.

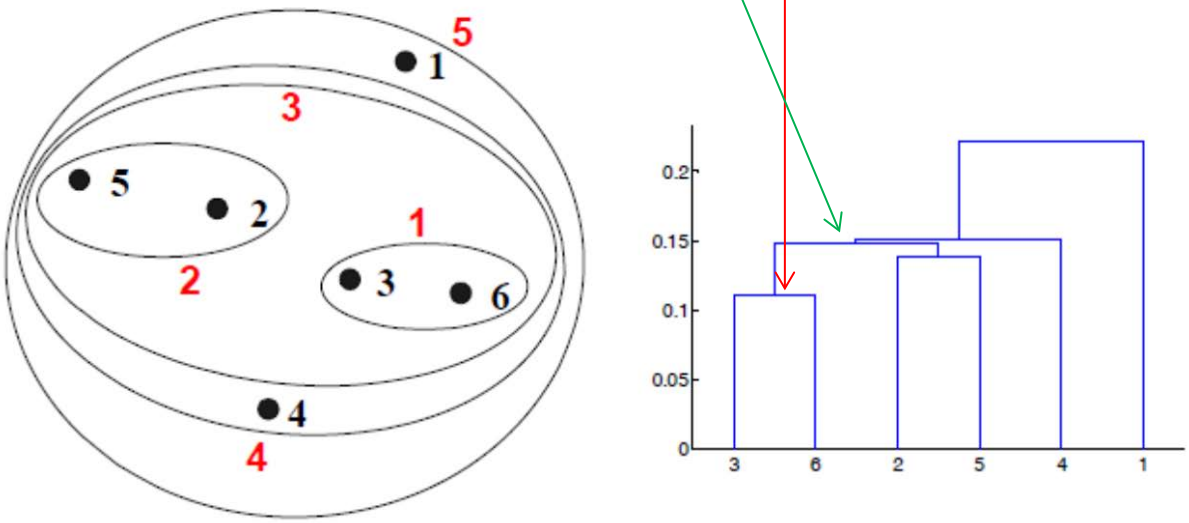


Abbildung 5-22: MIN-Ansatz

CLUSTERBILDUNG NACH MAX-ANSATZ (COMPLETE LINK APPROACH)

Wir ermitteln den geringsten Abstand zwischen p3 und p6:

D(p3, p6) = 0.1100 → die erste Dendrogramm-Gabelung liegt bei y = 0.1100

Ein weiteres Beispiel:

Dist({p2,p5}, {p1} = max(d(p2,p1), d(p5,p1) = max(0.2357, 0.3421) = 0.3421. Die Dendrogramm-Gabelung liegt somit bei y = 0.3421.

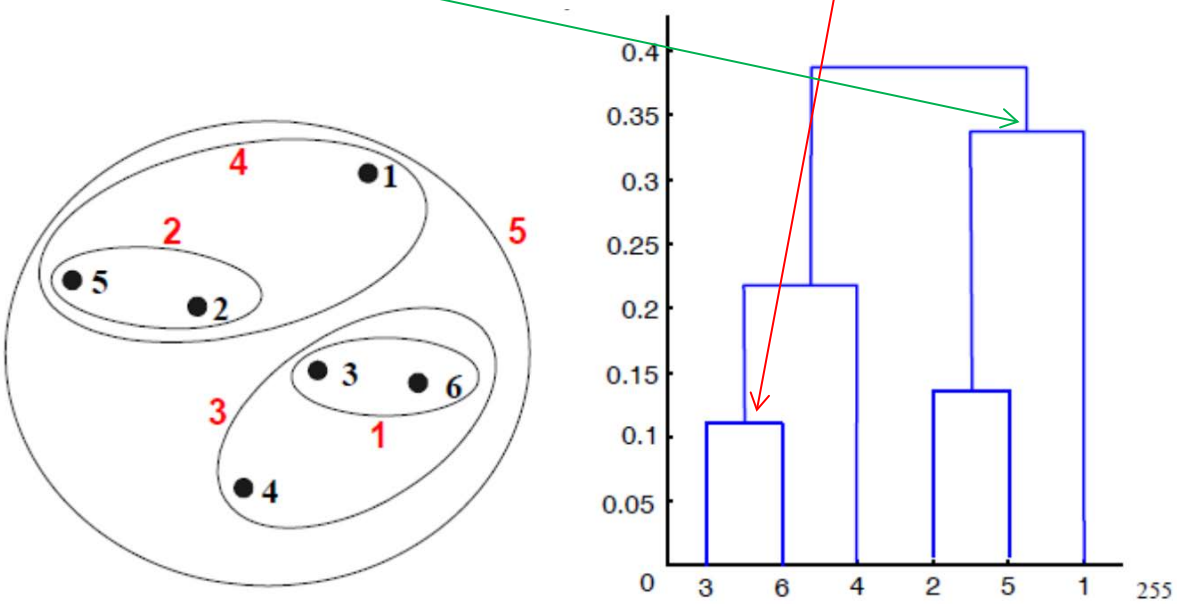


Abbildung 5-23: MAX-Ansatz

CLUSTERBILDUNG AVERAGE-ANSATZ (GROUP AVERAGE APPROACH = ABSTAND DER CENTROIDEN)

Wir ermitteln den geringsten Abstand zwischen p3 und p6:

D(p3, p6) = 0.1100 → erste Dendrogramm-Gabelung bei y = 0.1100

Ein weiteres Beispiel:

Dist({p3,p6,p4}, {p2,p5}
= d(p3,p2) + d(p3,p5) + d(p6,p2) + d(p6,p5) + d(p4,p2) + d(p4,p5) / 6
= 0.1483+0.2843+0.2540+0.3921+0.2042+0.2932 / 6 = 0,2627.

→ die erste Dendrogramm-Gabelung liegt bei y = 0.2627

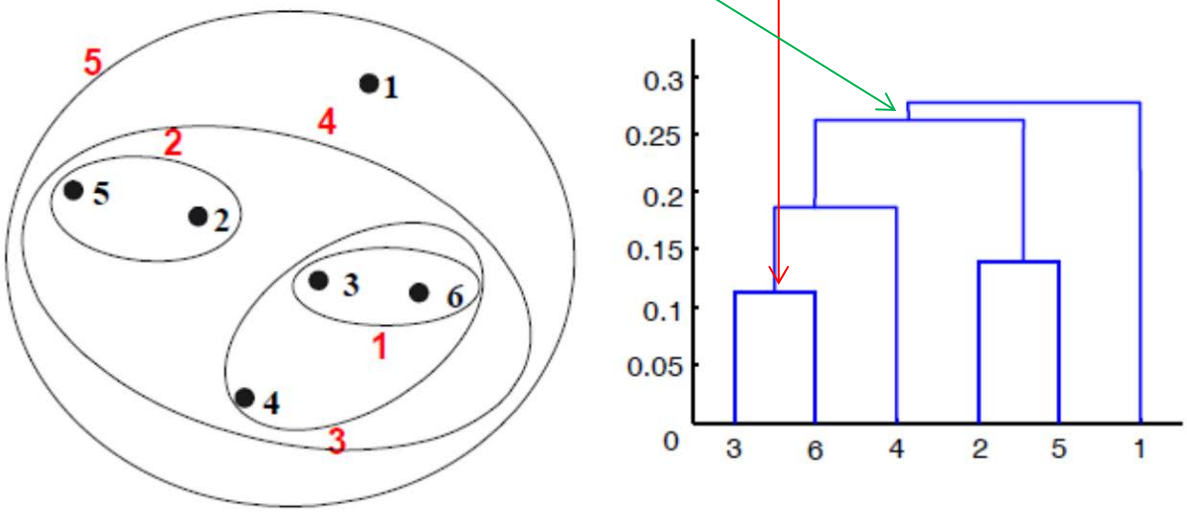


Abbildung 5-24: AVERAGE-Ansatz

CLUSTER NACH WARD'S METHODE

Nach WARD wird diejenige Verschmelzung jeweils durchgeführt, welche die kleinste Verschlechterung des SSE (sum of squared error) zur Folge hat.

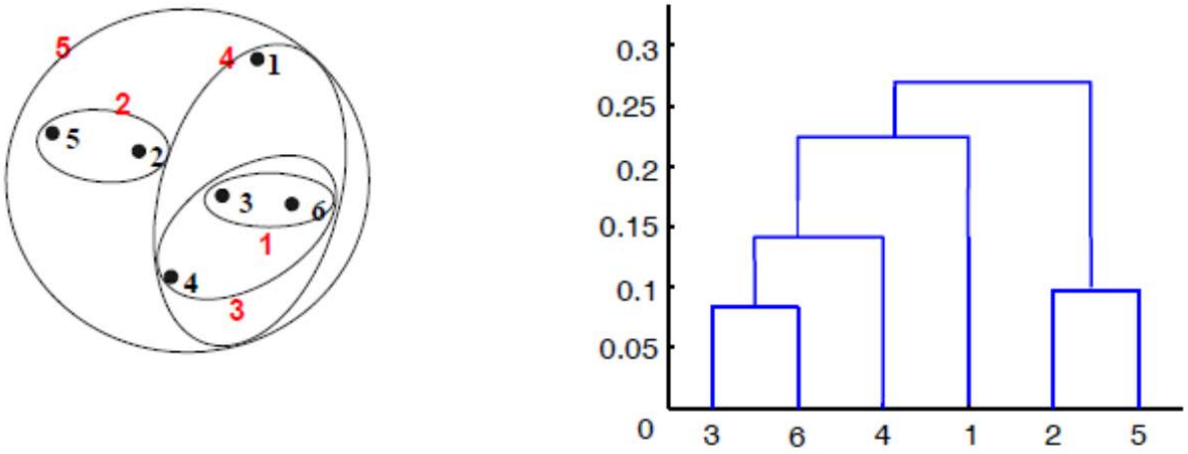


Abbildung 5-25: WARD's-Ansatz

UEBERBLICK ZU DEN GEZEIGTEN METHODEN

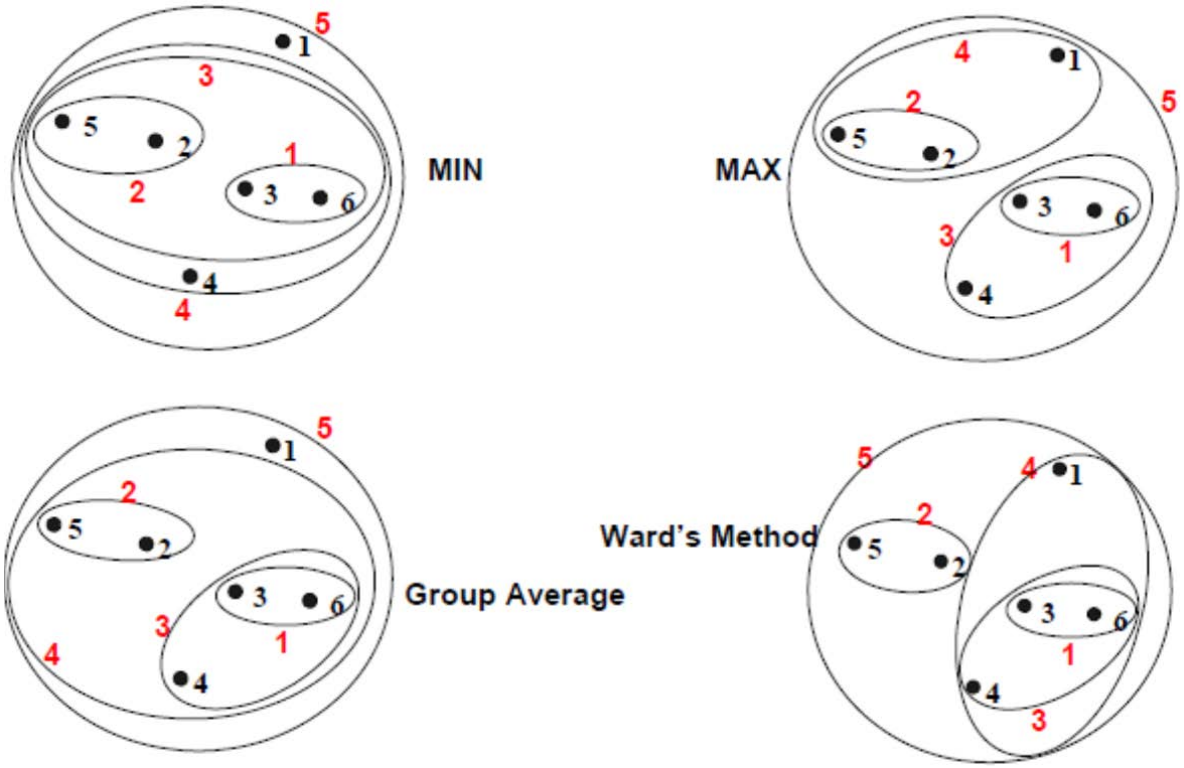


Abbildung 5-26: Vergleich der Methoden

6 ASSOZIATIONS-ANALYSE

Ihr Ziel ist, Assoziationen innerhalb einer Menge von Objekten zu finden. Bei einer gegebenen Transaktion (= Menge von Objekten bzw. einem Itemset) wird mittels Assoziations-Regeln nach Objekten gesucht, die typischerweise im Itemset enthalten sind. Ein Itemset ist eine Sammlung von einem oder mehreren Items (Artikel, Gegenstände, Posten). Beispiel: {Milch, Brot, Windeln}. Ein k-Itemset enthält k Items.

Beispiel: Einkaufskorb

Transaktion ID	*Itemset*
1	{Brot, Milch}
2	{Brot, Windeln, Bier, Eier}
3	{Milch, Windeln, Bier, Cola}
4	{Brot, Milch, Windeln, Bier}
5	{Brot, Milch, Windeln, Cola}

Assoziations-Regeln
{Windeln} → {Bier}
{Milch, Brot} → {Eier, Cola}
{Brot, Brot} → {Milch}
…

Mit der Implikation (Pfeil) innerhalb der Assoziationsregeln soll ein gemeinsames Auftreten verdeutlicht werden und keine Kausalität!

Typische Anwendungsbereiche der Assoziations-Analyse sind, die Warenkorbanalyse, die Klassifikation, das Clustering, das Cross Marketing, die Bioinformatik, medizinische Diagnose, Web Mining (Suchmaschinen) und die Analyse wissenschaftlicher Daten wie z.B. in der Klimaforschung.

In unserem Beispiel interessieren wir uns für die Assoziations-Regel {Milch, Windeln} → {Bier}. Diese besteht aus einem impliziten Ausdruck der Form X → Y. X und Y sind Itemsets.

Zur Regelauwertung verwenden wir die Messwerte Support (s) und Confidence (c), die in den folgenden Formeln umschrieben sind (|T| entspricht der Menge aller Transaktionen):

$$s = \frac{\delta\{\text{Milk, Windeln, Bier}\}}{|T|} = \frac{2}{5} = 0.4$$

Der Support (s) stellt den Bruchteil der Transaktionen dar, welche X und Y beinhalten. Der Support count (δ) bezeichnet die Häufigkeit des Auftretens eines Itemsets. Beispiel: δ({Brot, Milch, Windeln}) = 2. Ein Itemset dessen Support grösser oder gleich einem minsup Schwellenwert ist, bezeichnet man als häufiges Itemset.

$$c = \frac{\delta\{\text{Milch, Windeln, Bier}\}}{\delta\{\text{Milch, Windeln}\}} = \frac{2}{3} = 0.67$$

Mit Confidence (c) messen wir, wie häufig Items aus Y in Transaktionen erscheinen, welche X beinhalten.

Wir kommen nun zur eigentlichen Aufgabe des Assoziations-Mining:

Bei einer gegebenen Anzahl an Transaktionen T, gilt es alle Regeln zu finden, die die folgenden Bedingungen erfüllen:

- ✓ support ≥ minsup Schwellenwert
- ✓ confidence ≥ minconf Schwellenwert

Oder man wählt den Brute-Force-Ansatz:

1. alle möglichen Assoziations-Regeln werden aufgelistet
2. für jede Regel wird der Support und die Confidence berechnet
3. Regeln streichen, welche den minsup- und minconf-Schwellenwert nicht erreichen

Achtung: Der Brute-Force-Ansatz ist zu vermeiden, da er sehr rechenintensiv ist!

6.1 SUCHE NACH ASSOZIATIONS-REGELN

Beispiel-Regeln:

{Milch,Windeln} →{Bier} (s=0.4, c=0.67)
{ Milch, Bier } → { Windeln } (s=0.4, c=1.0)
{ Windeln, Bier } → { Milch } (s=0.4, c=0.67)
{ Bier } → { Milch, Windeln } (s=0.4, c=0.67)
{ Windeln } → { Milch, Bier } (s=0.4, c=0.5)
{ Milch } → { Windeln, Bier } (s=0.4, c=0.5)

Was können wir beobachten?

Alle Beispiel-Regeln sind binäre Aufteilungen von demselben Itemset. Regeln, welche von demselben Itemset abstammen, weisen identische Support-, aber unterschiedliche Confidence-Werte auf. So können die Support- und Confidence-Anforderungen entkoppelt werden.

Der 2-Schrittt-Approach

1. Ermittlung häufiger Item-Sets. Das heisst, es werden diejenigen gesucht, die einen Mindest - Support *minsup* aufweisen
2. Generierung starker Regeln aus den häufigen Item-Sets. Das heisst mit einer Verteilung der Items auf Prämisse und Konklusion, für die ein grosser Confidence-Wert entsteht.

Achtung: Auch dieser Approach ist rechnerisch sehr aufwändig und daher teuer!

Methode der Wahl: Reduzierung der Komplexität bei der Erzeugung häufiger Itemsets

1. Effiziente Generierung von Kandidat–Itemsets
   - Anwendung des *„a priori" Prinzips*
2. Effiziente Repräsentation der Kandidat–Itemsets
   - kompaktere Repräsentation der Kandidaten-Sets als Hash-Baum
3. Effizientes Vergleichen der Kandidaten mit den Transaktionen
   - effiziente Traversierung des Hash-Baumes

6.1.1 EFFEKTIVE ERZEUGUNG VON ITEMSETS, A PRIORI-PRINZIP

A Priori-Prinzip

Wenn ein Itemset **häufig** (**selten**) ist, dann sind auch alle seine **Teilmengen** (**Super-Mengen**) **häufig** (**selten**).

Dies erscheint logisch zu sein, da sich die Teilmengen-Relation und Support-Relation sich anti-monoton verhalten. Das heisst:

- $\forall X \forall Y : (X \subseteq Y) \rightarrow s(X) \geq s(Y)$, *anders formuliert*
- Eine Verkleinerung eines Itemsets kann nur zu einer Erhöhung seines Support-Wertes führen.
- Eine Vergrösserung eines Itemsets kann nur zu einer Verringerung seines Support-Wertes führen.

Nach dem A Priori-Prinzip kann somit gesagt werden, dass falls ein Itemset minsup unterschreitet, keine Untersuchung aller seiner Super-Sets nötig ist.

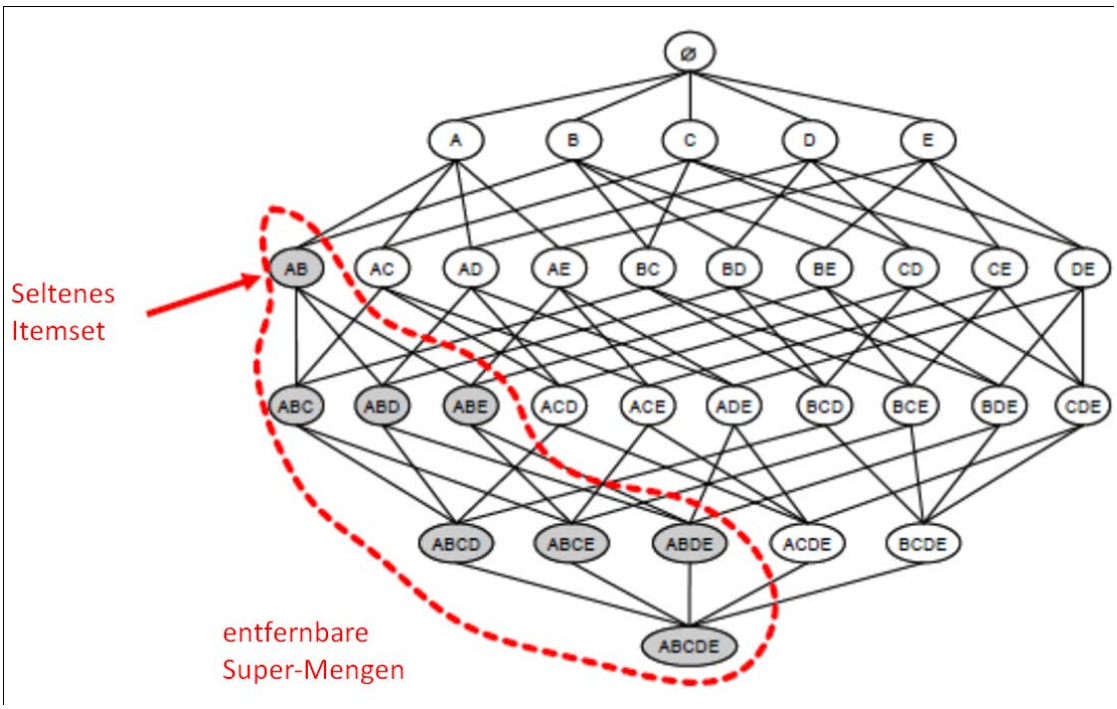


Abbildung 6-1: A Priori-Prinzip

A Priori-Algorithmus in Pseudo-Code

```
k := 1
F_k := {i: i ∈ I, σ(i)/N ≥ minsup} ; alle 1-elementigen Sets mit support ≥ minsup
repeat
    k:= k+1
    Ck:=apriori_gen(F_{k-1}) ; bilde alle k-elementigen Kandidaten aus den (k-1) elementigen
    for each transaction t ∈ T do
        C_t:=subset(C_k,t) ; filtere Kandidaten, die ⊆ t sind (in Transaktionen vorkommen)
        for each c ∈ C_t do   ; ermittle den Support count jedes Kandidaten
            σ(c):= σ(c) + 1 for each time c ⊆ t was found
        end (for)
    end (for)
    F_k:={c: c ∈ C_k, σ(c)/N ≥ minsup} ; filtere Kandidaten mit support ≥ minsup
until F_k = ∅
    result := ∪ F_k
```

n' Beispiel: *minsup = 0.6, d.h.* $\sigma_{min} = 3$

	Transaktion ID
1	{Brot, Milch}
2	{Brot, Windeln, Bier, Eier}
3	{Milch, Windeln, Bier, Cola}
4	{Brot, Milch, Windeln, Bier}

Die grün markierten Itemsets entfallen wegen nicht ausreichendem Support.

Itemset F1	***σ***
{Bier}	3
{Brot}	4
{Cola}	2
{Eier}	1
{Milch}	4
{Windeln}	4

2-el. Itemset F2	***σ***
{Bier, Brot}	2
{Bier, Milch}	2
{Bier, Windeln}	3
{Brot, Milch}	3
{Brot, Windeln}	3
{Milch, Windeln}	3

3-el. Itemset F3	***σ***
{Brot, Milch, Windeln}	2

Ergebnis

- ✓ 6+6+1 = 13 Kandidaten wurden gebildet
- ✓ Ohne Pruning der support-armen Itemsets wären es 6+15+20 = 41 (für bis zu 3-elementige Itemsets) gewesen.

7 BIG PRIVACY: DATENSCHUTZ IN BIG DATA

Gewaltige Datenmengen über Individuen – demographische Informationen, Internet Aktivitäten, Energiegebrauch, Verhaltensmuster der Kommunikation und soziale Interaktionen, um einige zu nennen – werden von statistischen Ämtern, Umfrageunternehmungen, medizinische Zentren, Web- und Social-Networking Organisationen gesammelt. Eine breite Verbreitung von solchen Mikrodaten (Daten mit der Granularität von Individuen) erleichtern Erkenntnisse in Wissenschaft und Politik. Sie helfen Bürgern über ihre Gesellschaften zu lernen und ermöglichen Schülern Skills zur Datenanalyse zu entwickeln! Aber oft können Sammelnde solcher Mikrodaten, diese nicht veröffentlichen. Wenn sie dies tun würden, könnten sensible Daten enthüllt werden. Wie wir wissen, kann die Vernachlässigung des Datenschutzes die Privatsphäre von Individuen verletzen. Das kann insbesondere im Fall von Regierungen und Forschungseinrichtungen nicht nur illegal sein, sondern auch beträchtliche Konsequenzen mit sich bringen. Zum Beispiel ist es in den USA so, dass falls die Privatsphäre eines Individuums, im Sinne des U.S. Confidential Information Protection and Statistical Efficiency Act, verletzt wird, mit einer Busse von bis zu $250‘000 und einer fünfjährigen Haftstrafe gerechnet werden muss.

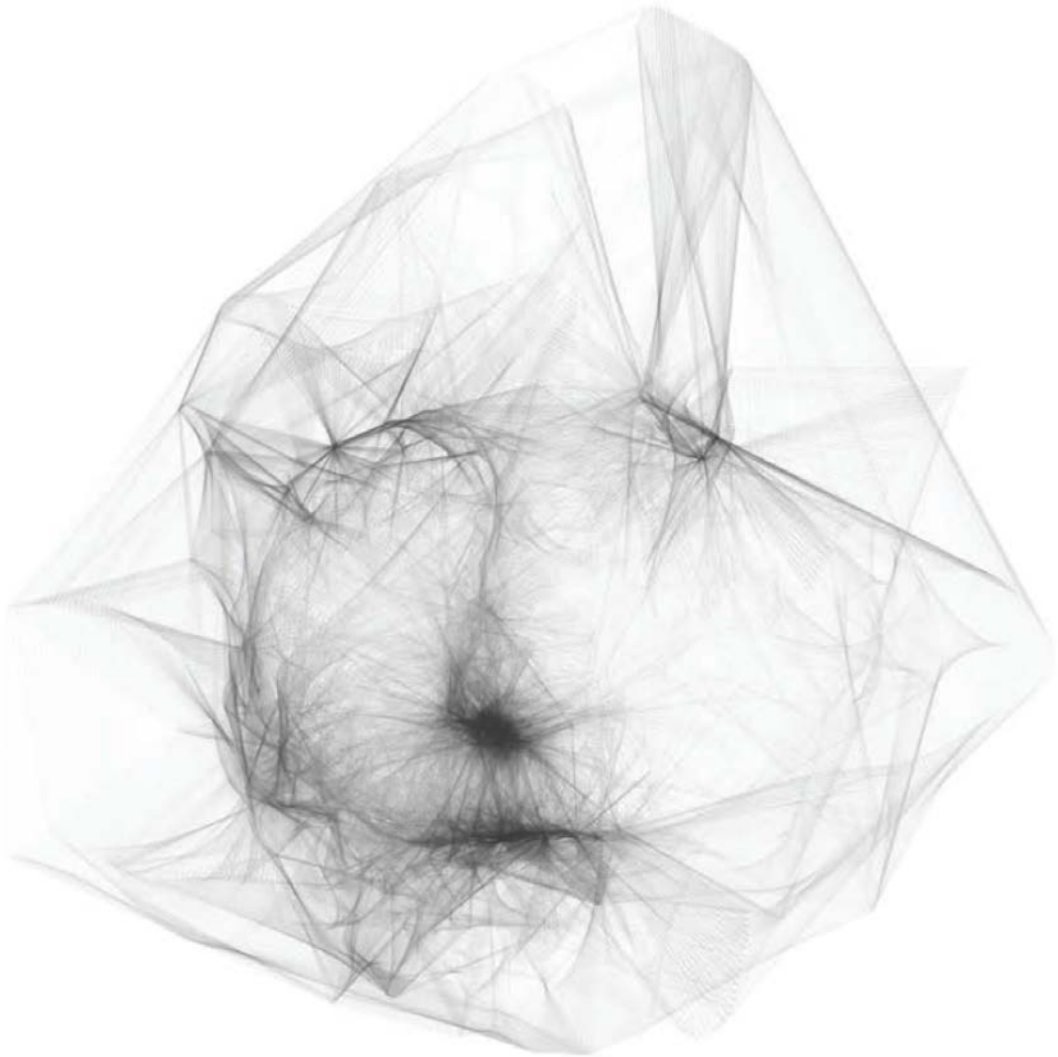


Abbildung 7-1: Digitalisiertes Gesicht[8]

[8] Crossroads (The ACM Magazine for Students), Vol.19 – No.1, Herbst 2012

Beim ersten Anschein, scheint das zur Verfügung stellen von Daten einfach zu sein. Man entfernt eindeutige Kennzeichen wie Namen, Adressen und AHV-Nummern aus den Datensätzen, bevor diese weitergereicht werden. Dies alleine ist aber häufig nicht ausreichend, da andere verfügbare Werte, wie gesammelte geografische oder demografische Daten auf dem File verbleiben. Diese Quasi-Marker können benutzt werden, um veröffentlichte Daten mit anderen Datenbanken in Übereinstimmung zu bringen. Zum Beispiel konnte von einer Informatikerin gezeigt werden, dass für 97% der öffentlich zugänglichen Daten der registrierten Wähler aus Cambridge in Massachusetts, eine eindeutige Identifikation der Individuen mittels Verwendung von Geburtsdatum und Postleitzahl möglich war. Mittels Abgleichung der Informationen in den Listen, identifizierte sie den Gouverneur William Weld in einer anonymisierten medizinischen Datenbank!

Die wohl grösste und viel Aufsehen erregende Verletzung der Privatsphäre geschah 2006. AOL stellte 20 Millionen Suchanfragen, welche über einen Zeitraum von 3 Monaten getätigt wurden, für Forschungszwecke zur Verfügung. Dabei wussten sie, dass die Information zu den Webrecherchen sensible Informationen, wie AHV-Nummern und Kreditkartennummern enthielten. Deshalb versuchten sie die Daten zu anonymisieren, indem Benutzererkennungen mit zufälligen Nummern ersetzt wurden. Innert ein paar Stunden nach Veröffentlichung der anonymisierten Daten, gelang zwei Reportern der New York Times, die Identifikation des Nutzers mit Nr. 4417749. Dies anhand der einfachen Suchanfrage: „Landscapers in Lilburn, GA“, Menschen mit dem Nachnamen Arnold und „numb fingers.“ Dies hatte zur Folge, dass einige der Topkader von AOL entlassen wurden. Zudem streben sich nun Suchunternehmungen dagegen, ähnliche Daten zu Forschungszwecken freizugeben. Sogar Forscher sind vorsichtig, beim Verwenden der nun öffentlich verfügbaren AOL-Daten. Ähnliche Wieder-Identifizierungen sind möglich anhand von Daten aus sozialen Netzwerken, der Geolokalisierung und des Stromverbrauchs. Obiges Beispiel zeigt, dass ein neugieriger Nachbar grosse Datenbestände auf Individuen anwenden kann und diese damit auch schädigen könnte. Kreditoren oder Marketingfachleute könnten grosse Datenbestände zur Identifikation guter, schlechter oder möglicher Kunden durchwühlen. Bösartige Hacker könnten sogar versuchen, ganze Unternehmungen zu diskreditieren, indem sie durch die Verwendung öffentlicher Daten, Individuen identifizieren.

Diese Bedrohungen haben faszinierende Forschungsfelder für ambitionierte Informatiker, Mathematiker, Statistiker und Sozialwissenschaftler kreiert. Es ist ein Gebiet, wo die Herausforderungen für die Forschenden gross und interdisziplinär sind. Um der Interdisziplinarität gerecht zu werden, werden in der Folge Themen aus der Informatik und Statistik. Es gibt viele andere Themen zu „Big Privacy“, die an dieser Stelle aber nicht ausgeführt werden. Dies sind, z.B. Systeme für das private Sammeln von Daten, die Webzugangskontrolle, Anwendungen sozialer Netzwerke, die Datensicherheit, die Kryptographie und Protokolle für sichere Berechnungen. Das sind ebenso reiche und komplementäre Forschungsgebiete, die für den sicheren und vertraulichen Umgang mit Daten wichtig sind.

7.1 VERTRAULICHKEITSRISIKEN: DEFINITION UND MASSNAHMEN

Beide, die Informatik und die Statistik haben eine Vielfalt von Kriterien und Methoden entwickelt, um diese Risiken zu quantifizieren. Davon werden einige wichtige in der Folge dargestellt.

Die in der Statistik üblicherweise verwendeten Massnahmen tendieren dazu, informeller und heuristischer Natur zu sein. Die allgemeinste und mathematisch formale Methode zur Offenlegung von Risikobewertungen, basiert auf der Wahrscheinlichkeit der Wiedererkennung nach Bayes, womit versucht wird, die Wahrscheinlichkeit abzuschätzen, mit welcher Eindringlinge zu Informationen über Individuen gelangen könnten. Dies unter Verwendung der veröffentlichten Daten und einer Auswahl von Annahmen über das Verhalten und Wissen von potentiellen Eindringlingen. Diejenigen, die Daten verbreiten wollen, können die Messwerte, unter Verwendung diverser Wissensszenarien der Eindringlinge, berechnen. So können kritische Attribute identifiziert und Entscheidungen, unter Berücksichtigung statistischer Unsicherheiten, zur Veröffentlichung der Daten getroffen werden. Diese Wahrscheinlichkeiten zu berechnen, ist computertechnisch aufwendig und verlangt nach innovativen Technologien insbesondere für grosse Datenmengen.

In der Informatik zielten frühe Anstrengungen darauf ab, die Vertraulichkeitsrisiken zu quantifizieren und sicherzustellen, dass keine singulären Werte eines Individuums in einem Datensatz auftreten, um Wiedererkennungsattacken zu vereiteln. Daraus entwickelte sich die gängige Verwendung der „K-Anonymität“. Sie verlangt, dass kein Individualwert sich von mindestens K-1 anderen Individualwerten unterscheiden darf. Dadurch ist das Problem aber noch nicht gelöst. Noch können Eindringlinge auf sensible Eigenschaften Rückschlüsse ziehen. Nehmen wir an, dass ein Spital K-anonyme Mikrodaten über Patienten veröffentlicht. Sie wissen, dass Ihr Nachbar Bob im Datensatz befindet. Wenn die Individuen des Datensatzes alle entweder Grippe oder Krebs aufweisen, und Sie wissen, dass Bob diese Grippe nicht hatte, können Sie daraus folgern, dass Bob Krebs hat. Die K-Anonymität wurde daher erweitert, um dieses Manko zu beheben. Aktuell werden Datensätze, bevor sie offengelegt werden, mittels einer Metrik der differentiellen Privatsphäre geprüft. Differentielle Privatsphäre erfüllt das wichtige Kriterium der Zusammensetzbarkeit. Wenn M1 und M2 zwei Mechanismen sind, welche die Kriterien der differentiellen Privatsphäre mit den Parametern $\epsilon 1$ und $\epsilon 2$ erfüllen, ergeben sich aus den Outputs von M1 und M2, die zusammen der differentiellen Privatsphäre mit den Parametern $\epsilon 1 + \epsilon 2$ genügen. Andere bekannte Messwerte der Privatsphäre, wie z.B. die K-Anonymität und L-Diversität) genügen nicht dem Kriterium der Zusammensetzbarkeit.

7.2 METHODEN ZUM SCHUTZ OEFFENTLICHER DATEN

Zur Risikominderung entwickelten Informatiker wie auch Statistiker Methoden, um Datensätze vor der Veröffentlichung zu verändern. Beide Fachgebiete entwickelten unabhängig voneinander ähnliche Methoden. Wenn es um das Design von Methoden zum Schutze der Privatsphäre geht, müssen vor allem zwei wichtige Aspekte berücksichtigt werden. Einerseits ist es wichtig, dass die Methode Resultate liefert, welche nützliche Informationen über den Input enthalten. Durch den Schutz der Privatsphäre geht immer ein gewisser Informationsgehaltsverlust des zur Verfügung gestellten Datensets einher. Wer Daten klug veröffentlichen will, muss immer einen guten Kompromiss zwischen Risiko und Qualität finden. Andererseits, sollte eine Methode, die zum Schutz der Privatsphäre angewendet wurde, nicht erkennbar sein: Dabei muss von einem Angreifer angenommen werden, dass ihm die Methode zum Schutz der Privatsphäre bekannt ist. In der Folge wird auf einige wichtige Methoden zum Schutz der Privatsphäre eingegangen.

7.2.1 AGGREGATION

Die Aggregation verringert das Enthüllungsrisiko indem atypische Werte, welche ein grosses Entdeckungsrisiko in sich tragen, in typische Werte verändert werden. Zum Beispiel kann es sein, dass eine einzige Person ein typisches demographisches Merkmal in einer Stadt hat, aber viele Personen mit diesem Merkmal innerhalb eines Kantons. Für diese Person birgt die Veröffentlichung geografischer Daten auf Städteebene ein grosses Risiko, identifiziert zu werden, wobei die Veröffentlichung auf kantonaler Ebene risikoärmer wäre. Unglücklicherweise, machen Aggregationen das Analysieren auf tieferen Stufen schwierig und oft unmöglich. Und es entsteht das Problem ökologischer Interferenzen, d.h. Zusammenhänge welche auf aggregiertem Level sichtbar sind, sind auf nicht aggregierte Level nicht anwendbar.

7.2.2 SUPPRESSION

Agenturen können sensible Werte von veröffentlichen Daten löschen. Ganze Variablen oder nur gefährdete Datenwerte würden ausgeblendet. Die Unterdrückung von einzelnen Werten, generiert fehlende Datenwerte, was saubere Datenanalysen wiederum erschwert. Zum Beispiel, wenn Einkommenswerte, weil sie gross sind, herausgelöscht wurden, werden Berechnungen zu Einkommensverteilungen basierend auf den veröffentlichten Daten nahezu unmöglich.

7.2.3 DATA SWAPPING

Andererseits können Agenturen Werte aus ausgewählten Datensätzen vertauschen, um Nutzer vom Abgleich der Daten abzuhalten, da die Abgleiche auf falschen Datenwerten gründen. Das Austauschen, wird vor allem von Regierungsagenturen vorgenommen. Es wird angenommen, dass Swapping auf eine geringe Anzahl von Datensätzen angewendet wird, da eine hohe Rate an ausgetauschten Werten, Beziehungen zwischen den Datensätzen unbrauchbar werden lässt.

7.2.4 ZUFÜGEN VON ZUFÄLLIGEM RAUSCHEN

Zu den gesammelten Datenwerten, werden zufällig gewählte zusätzliche Datensätze hinzugefügt. Dadurch können die Möglichkeiten zur genauen Übereinstimmung von Datensätzen innerhalb von „getrübten“ Datensätzen eingeschränkt und sensible Daten können verzehrt werden.

7.2.5 HINZUFÜGEN KÜNSTLICHER DATEN

Die Grundidee synthetischer Daten ist es, Originaldatenwerte zu ersetzen, welche ein hohes Risiko in sich tragen, durch die Anwendung von Wahrscheinlichkeitsverteilungen entdeckt zu werden. Es gibt dabei die Möglichkeit partiell oder komplett synthetische Daten einzusetzen. Partiell synthetische Daten umfassen die ursprünglich erhobenen Einheiten mit einigen zusätzlichen Datensätzen simulierter Natur. Synthetische Daten enthalten komplett simulierte (synthetische) Datensätze; original erfasste Einheiten sind nicht im File enthalten. Synthetische Datensätze können eine sichere Privatsphäre mit hoher Qualität garantieren.

7.2.6 HERAUSFORDERUNGEN FÜR DIE FORSCHUNG

Bis anhin wurde viel in die Erforschung im Bereich der Privatsphäre investiert, wobei die einzelnen Datensätze typischerweise als voneinander unabhängig betrachtet wurden. Ein riesiges Problem ist die Gewährleistung der Privatsphäre verlinkter relationaler Daten. Zum Beispiel in sozialen Netzwerkstrukturen; in denen Personen unter einander verlinkt sind. Das Räsonieren in diesem Gebiet wird dadurch erschwert, dass Informationen über Individuen über Links zu anderen Individuen gelangen können. Ein anderes interessantes Problem ist das regelmässige Freigeben von Daten im Verlaufe der Zeit. Angreifer könnten per Analyse sequentieller Daten Informationen über Individuen gewinnen, welche sie über eine einzelne Sequenz unmöglich finden könnten. Zum Schluss: So wie Datenbestände extrem mehrdimensional sind, so benötigen wir Techniken, welche die Privatsphäre wie auch die Nützlichkeit der Daten garantieren können!

8 QUELLEN

ZUR ERSTELLUNG DIESER ARBEIT, WURDEN FOLGENDE QUELLEN GENUTZT:

- Witten I.H., Eibe F., Hall M.A. 2011. *Data Mining – practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

- Ertel W., 2009. *Grundkurs Künstliche Intelligenz.* Vieweg + Teubner.

- Machanavajjhala A. und Reiter J.P., 2012. *Big Privacy: Protecting Confidentiality in Big Data.* Crossroads-The ACM Magazine for Students, 20-23.

- Moore A., 2006. *Statistical Data Mining Tutorials.* *http://www.autonlab.org/tutorials/* (12.1.2013)

SÄMTLICHE VERWENDETEN DATENSÄTZE STAMMEN VON:

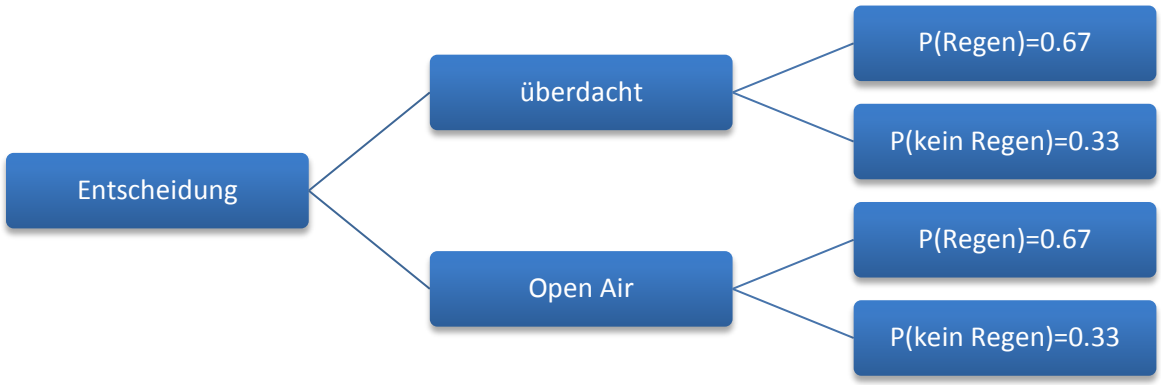
http://archive.ics.uci.edu/ml/datasets.html (12.01.2013)

9 LÖSUNGEN ZU DEN AUFGABEN

9.1 KAPITEL 1: EINFÜHRUNG

Aufgabe 1-1: Berechnen Sie den Erwartungswert (EW) mit perfekter Information (PI) über den Regen. Wie viel sollten Sie bereit sein, für diese Information (PI) zu bezahlen?

Um den Erwartungswert zu bestimmen, wird am besten ein Entscheidungsbaum verwendet:



EW überdacht = 130'000, EW Open Air = 185'000
Wie viel sollten Sie bereit sein, für diese Information (PI) zu bezahlen?
EWPI = EW mit PI – EW ohne PI = 18'150.-
EW mit PI = 0.67 x 130'000.- + 0.33 x 185'000.- = 148.150.-
EW ohne PI = 130'000 (mit dem bedeckten Stadion werden auf jeden Fall 130'000 eingenommen)

Aufgabe 1-2: Betrachten Sie die Tabelle 1-5 bzw. Abbildung 1-7 genau. Welche Schlüsse können für die verschiedenen Altersbereiche gezogen werden? Zur Erinnerung: Die Datensätze stammen aus den USA.

Bis zum 20-ten Lebensjahr wird sichtbar, dass viele sich in der Ausbildung befinden. Nachher, d.h. bis zum 40-ten Lebensjahr ist die Erwerbsphase mit steigenden Einkünften erkennbar. Zwischen dem 40-ten und 50-ten Lebensjahr wird der „Babyboom“ sichtbar. Über die 60-ten und 70-ten Lebensjahre ist die Problematik der US-Rentenpolitik sichtbar. Zum neunten Lebensjahrzehnt passt die Aussage: „Wer reich ist, lebt länger.“

Aufgabe 1-3: Wir haben 16 Attribute, a) wie viele 1-dimensionale Kohärenztabellen erhalten wir damit? b) Wie viele 2-dimensionale Kohärenztabellen? c) Wie viele 3-dimensionale Kohärenztabellen? d) Falls wir 100 Attribute hätten, wie viele 3-dimensionale Kohärenztabellen hätten wir? e) Welchen Schluss ziehen Sie in für grössere Kontingenztabellen?

a) 16
b) 16 x 15 / 12 = 120
c) 16 x 15 x 14 / 1 x 2 x 3 = 560
d) 100 x 99 x 98 / 3 x 2 x 1 = 161‘700 (das sind relativ viele Kohärenztabellen! :-(
e) Im Falle grösserer Kontingenztabellen wird der Erkenntnisgewinn zusehends beschwerlicher. Es könnte in diesem Fall interessant sein, andere, z.B. automatisierte Lernmechanismen einzusetzen.

9.2 KAPITEL 2: KLASSIFIZIERUNG – 1R

Aufgabe 2-1: *Klassifizieren Sie mit Hilfe der Regel-Sets zu den Attributen «Wetter» und «Luftfeuchtigkeit» den Datensatz aus der Tabelle 2-4. Vergleichen Sie anschliessend die Resultate miteinander. Was fällt dabei auf?*

Wetter:
sonnig → nein
bewoelkt → ja
regnerisch → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	nein
2	regnerisch	heiss	hoch	schwach	ja

Luftfeuchtigkeit:
hoch → nein
normal → ja

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
1	sonnig	heiss	normal	stark	ja
2	regnerisch	heiss	hoch	schwach	nein

Aufgabe 2-2: *Kommentieren Sie folgende Aussage:*
„Offensichtlich ist es so, dass ein Algorithmus besser lernen kann, je grösser das Trainingsset ist. Damit wäre es doch für das «Wetter-Problem» eine gute Lösung, wenn man sämtliche 20 Einträge zum Trainieren verwenden würde. Um das Regelset zu testen könnten von denselben 20 Einträgen doch einfach 6 zufällig ausgewählt werden“.

Dies ist eine sehr schlechte Idee, weil der Algorithmus bereits auf diesen Daten trainiert hat. Dadurch wird er gezwungenermassen gut abschneiden und dem naiven Anwender damit vorgaukeln, besser zu sein, als er in Tat und Wahrheit ist. Trainings- und Testdaten müssen aus diesem Grund immer strikt voneinander getrennt werden.

Aufgabe 2-3: Wenden Sie das Regel-Set zum Attribut Luftfeuchtigkeit auf die Testdaten an und vervollständigen Sie untenstehende Tabelle. Bestimmen Sie anschliessend die Fehlerquote resp. Genauigkeit und vergleichen Sie Ihre Resultate mit der Tabelle 2-7.

Luftfeuchtigkeit: *hoch → nein*
normal → ja

(Totale Fehlerquote = 1/6 | Genauigkeit = 5/6)

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass	Luftf. Regel-Set	
15	sonnig	mild	hoch	stark	nein	nein	✓
16	bewoelkt	mild	normal	schwach	nein	ja	×
17	bewoelkt	kalt	hoch	stark	nein	nein	✓
18	bewoelkt	heiss	normal	stark	ja	ja	✓
19	regnerisch	heiss	normal	stark	ja	ja	✓
20	regnerisch	heiss	hoch	schwach	nein	nein	✓

Aufgabe 2-4: Tragen Sie unten schrittweise die Intervallgrenzen ein, welche 1R für das Attribut Luftfeuchtigkeit erstellen würde, wenn minBucketSize = 3 ist. Geben Sie anschliessend das Regel-Set und dessen Fehlerquote an.

Schritt 1:

Luftfeuchtigkeit:	65	70	70	70	75	80	80	85	86	90	90	91	95	96
	ja	nein	ja	ja	ja	ja	ja	nein	ja	ja	nein	nein	nein	ja

Schritt 2:

Luftfeuchtigkeit:	65	70	70	70	75	80	80	85	86	90	90	91	95	96
	ja	nein	ja	ja	ja	ja	ja	nein	ja	ja	nein	nein	nein	ja

Regelset:

Luftffeuchtigkeit: < 82.5 → ja
< 95.5 → nein
≥ 95.5 → ja

Fehlerquote:

Totale Fehlerquote = 3/14
Genauigkeit = 11/14

Aufgabe 2-5: *Laden Sie mit Weka den Datensatz zum «Wetter-Problem» und wählen Sie als Classifier **OneR** mit minBucketSize = 6 aus. Probieren Sie nun alle vier Testverfahren durch, welche unter **TEST OPTIONS** ❷ angeboten werden. Als Testset steht Ihnen die Datei WetterProblem-TestSet.csv zur Verfügung.*

Vergleichen und interpretieren Sie die Resultate, insbesondere die Prozentzahlen unter Summary » Correctly Classified Instances resp. Incorrectly Classified Instances und die Wahrheitsmatrizen.

Notieren Sie für alle vier Testverfahren den Wert bei Summary » Kappa statistics. Sie werden diese Zahlen in der nächsten Übung benötigen.

In allen vier Fällen wird für den Classfier folgendes Regel-Set gewählt:

```
=== Classifier model (full training set) ===

Wetter:
   sonnig   -> nein
   bewoelkt -> ja
   regnerisch  -> ja
(10/14 instances correct)
```

Allerdings unterscheiden sich die Teststatistiken der einzelnen Methoden stark. Es wird nur allzu deutlich klar, dass es gefährlich ist, die Leistungsfähigkeit eines Classifiers voraussagen zu wollen wenn man nur die Fehlerrate bei den Trainingsdaten betrachtet.

USE TRAINING SET

```
Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %
Kappa statistic                          0.3778
=== Confusion Matrix ===
 a b   <-- classified as
 3 2 | a = nein
 2 7 | b = ja
```

SUPPLIED TEST SET (WETTERPROBLEM-TESTSET.CSV)

```
Correctly Classified Instances           3               50      %
Incorrectly Classified Instances         3               50      %
Kappa statistic                          0.1818
=== Confusion Matrix ===
 a b   <-- classified as
 1 3 | a = nein
 0 2 | b = ja
```

CROSS-VALIDATION (10 FOLDS)

```
Correctly Classified Instances           5               35.7143 %
Incorrectly Classified Instances         9               64.2857 %
Kappa statistic                         -0.1455
=== Confusion Matrix ===
 a b   <-- classified as
 3 2 | a = nein
 7 2 | b = ja
```

PERCENTAGE SPLIT (66 %)

```
Correctly Classified Instances           2               40      %
Incorrectly Classified Instances         3               60      %
Kappa statistic                         -0.3636
=== Confusion Matrix ===
 a b   <-- classified as
 0 2 | a = nein
 1 2 | b = ja
```

Aufgabe 2-6: Vergleichen Sie die Werte für κ, welche Sie bei der letzten Aufgabe notiert haben und machen Sie eine Aussage über die Brauchbarkeit der verschiedenen Classifier.

Sämtliche Werte für κ liegen unter 0.40 womit die erhaltenen Classifier ziemlich unbrauchbar sind. Dies ist aber nicht weiter verwunderlich, da das «Wetter-Problem» einerseits nur über sehr wenige Instanzen verfügt, andererseits kein realer Datensatz ist, sondern nur zur Veranschaulichung verschiedener Data-Mining-Techniken dient.

Interessant ist aber der Umstand, dass κ auch negative Werte annehmen kann. Das bedeutet dann, dass ein Zufalls-Classifier in diesem Fall die Daten besser klassifizieren würde als der trainierte Classifier – auf realen Datensätzen ein deutliches Alarmzeichen!

Aufgabe 2-7: Berechnen Sie κ für das «Wetter-Problem» von Hand und interpretieren Sie den Wert. Verwenden Sie für Ihre Überlegungen die gegebene Wahrheitsmatrix.

	Classifier teilt in Klasse *nein* ein	**Classifier teilt in Klasse *ja* ein.**
Gehört in die Klasse *nein*	3	2
Gehört in die Klasse *ja*	2	7

Um κ für das «Wetter-Problem» zu berechnen, benötigen wir die Wahrheitsmatrix aus Tabelle 2-11.

BERECHNUNG VON PC

Trefferquote des trainierten Classifiers: (3+7)/14 = 0.7143

BERECHNUNG VON PZ

Gehört in Klasse die *nein*: (3+2)/14 = 0.3571
Vom Classifier eingeteilt in die Klasse *nein*: (3+2)/14 = 0.3571
W'keit dass der Classifier zufällig in Klasse *nein* einteilt: 0.3571*0.3571 = 0.1276

Gehört in Klasse die *ja*: (2+7)/14 = 0.6429
Vom Classifier eingeteilt in die Klasse *ja*: (2+7)/14 = 0.6429
W'keit dass der Classifier zufällig in Klasse *ja* einteilt: 0.6429*0.6429 = 0.4133

p_Z = W'keit dass der Zufalls-Classifier korrekt in Klassen *ja* und *nein* einteilt = 0.1276+0.4133 = 0.5408

BERECHNUNG VON KAPPA

$$\kappa = \frac{p_C - p_Z}{1 - p_Z} = \frac{0.7143 - 0.5408}{1 - 0.5408} = 0.3778$$

Dieser Wert entspricht auch dem Kappa aus der Übung 3-5 (Testmethode: Use training set).

Aufgabe 2-8: Berechnen Sie κ für ein Classifier, welcher folgende Wahrheitsmatrix liefert.

	Classifier teilt in Klasse *a* ein	**Classifier teilt in Klasse *b* ein.**	**Classifier teilt in Klasse *c* ein.**
Gehört in die Klasse *a*	88	10	2
Gehört in die Klasse *b*	14	40	6
Gehört in die Klasse *c*	18	10	12

BERECHNUNG DER ANZAHL INSTANZEN

Anzahl Instanzen: 88+10+2+14+40+6+18+10+12 = 200

BERECHNUNG VON PC

Trefferquote des trainierten Classifiers: (88+40+12)/200 = 0.70

BERECHNUNG VON PZ

Gehört in Klasse die *a*: (88+10+2)/200 = 0.50
Vom Classifier eingeteilt in die Klasse *a*: (88+14+18)/200 = 0.60
W'keit dass der Classifier zufällig in Klasse *a* einteilt: 0.50*0.60 = 0.30

Gehört in Klasse die *b*: (14+40+6)/200 = 0.30
Vom Classifier eingeteilt in die Klasse *b*: (10+40+10)/200 = 0.30
W'keit dass der Classifier zufällig in Klasse *b* einteilt: 0.30*0.30 = 0.09

Gehört in Klasse die *c*: (18+10+12)/200 = 0.20
Vom Classifier eingeteilt in die Klasse *c* (2+6+12)/200 = 0.10
W'keit dass der Classifier zufällig in Klasse *c* einteilt: 0.20*0.10 = 0.02

p_Z = W'keit dass der Zufalls-Classifier korrekt in Klassen *a, b* und *c* einteilt: 0.30+0.09+0.02 = 0.41

BERECHNUNG VON KAPPA

$$\kappa = \frac{p_C - p_Z}{1 - p_Z} = \frac{0.70 - 0.41}{1 - 0.41} = 0.49$$

Der trainierte Classifier teilt 49 % der, von einem Zufalls-Classifier falsch klassifizierten Instanzen, zusätzlich korrekt ein. Das Regel-Set ist damit nicht überragend, aber dennoch ausreichend gut.

Aufgabe 2-9: *Der «Iris-Datensatz» umfasst 150 Instanzen mit den Daten von drei verschiedenen Schwertlilien-Arten (Iris Setosa, Iris Virginica und Iris Versicolor), wobei je 50 Instanzen auf eine der drei Arten entfallen. Der Datensatz enthält vier Attribute: Länge und Breite des Sepalum (Kelchblatt) und Länge und Breite des Petalum (Kronblatt).*

Verwenden Sie den 1R-Algorithmus um ein Regel-Set zu finden, welches mit Hilfe einer der vier Attribute, die Instanzen korrekt klassifiziert. Die Schwertlilien-Art entspricht in diesem Beispiel der Klasse. Verwenden Sie den Datensatz iris.csv für Ihre Arbeit.

Notieren Sie das Regel-Set und machen Sie eine Aussage über dessen Qualität wenn Sie als Testmethode die Kreuzvalidierung verwenden.

```
=== Run information ===
Scheme:       weka.classifiers.rules.OneR -B 6
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===
petalwidth:
   < 0.8 -> Iris-setosa
   < 1.65   -> Iris-versicolor
   >= 1.65  -> Iris-virginica
(144/150 instances correct)
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances         140               93.3333 %
Incorrectly Classified Instances        10                6.6667 %
Kappa statistic                          0.9
Mean absolute error                      0.0444
Root mean squared error                  0.2108
Relative absolute error                 10      %
Root relative squared error             44.7214 %
Coverage of cases (0.95 level)          93.3333 %
Mean rel. region size (0.95 level)      33.3333 %
Total Number of Instances              150

=== Confusion Matrix ===
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 43  7 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

Mit einem Kappa-Wert von 0.90 ist der trainierte Classifier in der Lage, 90 % der von einem Zufalls-Classifier falsch klassifizierten Instanzen korrekt zuzuordnen. Das Regel-Set eignet sich daher hervorragend, um den Iris-Datensatz zu klassifizieren. Sehr gut ist dies auch an der Wahrheitsmatrix zu erkennen, welche zeigt, dass es nur zwischen den Klassen *b* und *c* zu Fehlzuteilungen kommt, diese aber ebenfalls sehr gering ausfallen.

9.3 KAPITEL 3: KLASSIFIZIERUNG – NAÏVE BAYES

Aufgabe 3-1: Ermitteln Sie die Klasse, welcher der zweite Eintrag der Tabelle 3-3 zugeordnet werden müsste.

	Wetter	Temperatur	Luftfeuchtigkeit	Wind	Anlass
2	regnerisch	heiss	hoch	schwach	?

Wetter			Temperatur			Luftfeuchtigkeit			Wind			Anlass	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
sonnig	3	2	heiss	2	2	hoch	4	3	schwach	2	6	5	9
bewoelkt	0	4	mild	2	4	normal	1	6	stark	3	3		
regnerisch	2	3	kalt	1	3								
sonnig	3/5	2/9	heiss	2/5	2/9	hoch	4/5	3/9	schwach	2/5	6/9	5/14	9/14
bewoelkt	0/5	4/9	mild	2/5	4/9	normal	1/5	6/9	stark	3/5	3/9		
regnerisch	2/5	3/9	kalt	1/5	3/9								

Chance für *ja* = 3/9 (regnerisch ja) x 2/9 (heiss ja) x 3/9 (hoch ja) x 6/9 (schwach ja) x 9/14 (Klasse ja) = 0.01058

Chance für *nein* = 2/5 (regnerisch nein) x 2/5 (heiss nein) x 4/5 (hoch nein) x 2/5 (schwach nein) x 5/14 (Klasse nein) = 0.01829

$$\text{Wahrscheinlichkeit für ja} = P(\text{ja}) = \frac{0.01058}{0.01058+0.01829} = 0.367 = 36.7\ \%$$

$$\text{Wahrscheinlichkeit für nein} = P(\text{nein}) = \frac{0.01829}{0.01058+0.01829} = 0.633 = 63.3\ \%$$

Aufgabe 3-2: Die Wahrscheinlichkeit, dass eine Person eine bestimmte Krankheit in sich trägt ist P(K) = 0.0002. Mit Hilfe eines neuartigen Tests soll ermittelt werden, ob eine Person von dieser Krankheit betroffen ist. Der Test T erkennt die Krankheit mit 99 prozentiger Sicherheit. Von 10'000 untersuchten Personen, wurden 102 positiv getestet. Wie viele von Ihnen sind tatsächlich krank?

$$P(K \mid T = \text{positiv}) = \frac{P(T = \text{positiv} \mid K) \cdot P(K)}{P(T = \text{positiv})} = \frac{0.99 \cdot 0.0002}{102/10'000} = 0.019 = 1.9\ \% = 2\ \text{Personen}$$

Eine Person, die vom Test positiv getestet wird, hat eine 98 %ige Chance gesund zu sein! Dieses erstaunliche Resultat liegt darin begründet, dass die Wahrscheinlichkeit erkrankt zu sein mit 0.02 % um das fünfzigfache geringer ist, als die Wahrscheinlichkeit eines falschen Testergebnisses (102/10'000 = 1.0 %).

Aufgabe 3-3: Berechnen Sie P(nein | bewoelkt, heiss, hoch, schwach) und erläutern Sie, was von diesem Resultat zu halten ist.

Wetter			Temperatur			Luftfeuchtigkeit			Wind			Anlass	
	nein	ja		nein	ja		nein	ja		nein	ja	nein	ja
sonnig	3	2	heiss	2	2	hoch	4	3	schwach	2	6	5	9
bewoelkt	0	4	mild	2	4	normal	1	6	stark	3	3		
regnerisch	2	3	kalt	1	3								
sonnig	3/5	2/9	heiss	2/5	2/9	hoch	4/5	3/9	schwach	2/5	6/9	5/14	9/14
bewoelkt	0/5	4/9	mild	2/5	4/9	normal	1/5	6/9	stark	3/5	3/9		
regnerisch	2/5	3/9	kalt	1/5	3/9								

Chance für *ja* = 4/9 (bewoelkt ja) x 2/9 (heiss ja) x 3/9 (hoch ja) x 6/9 (schwach ja) x 9/14 (Klasse ja) = 0.01411

Chance für *nein* = **0/5** (bewoelkt nein) x 2/5 (heiss nein) x 4/5 (hoch nein) x 2/5 (schwach nein) x 5/14 (Klasse nein) = **0**

Die Wahrscheinlichkeit, den Attribut-Wert Wetter = bewoelkt in der Klasse nein zu finden beträgt 0. Somit würde auch die Wahrscheinlichkeit, dass die Instanz in die Klasse nein gehört, auf 0 sinken, was aber sicher nicht der Fall sein kann.

Aufgabe 3-4: Laden Sie mit Weka den Datensatz Spam-Basis.arff, welcher 4'600 E-Mails enthält, die aufgrund von 55 verschiedenen Attributen (Worte und Zeichen) als Spam resp. nicht Spam klassifiziert wurden.

1) Trainieren Sie aufgrund dieses Datensatzes einen NaiveBayes-Classifier. Wählen Sie als Testmethode «Cross-Validation». Wie gut schneidet der Classifier ab?

```
Correctly Classified Instances        4074               88.546  %
Incorrectly Classified Instances       527               11.454  %
Kappa statistic                          0.7568
=== Confusion Matrix ===
    a    b   <-- classified as
 2596  192 |    a = 0
  335 1478 |    b = 1
```

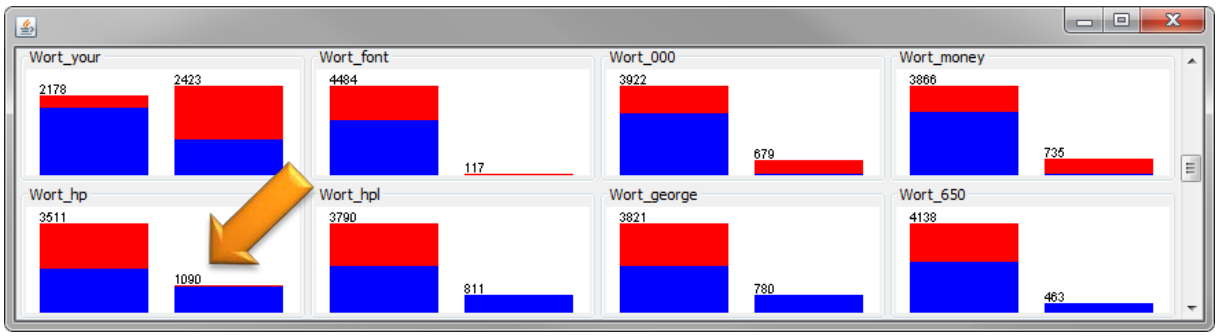
2) Berechnen Sie die bedingte Wahrscheinlichkeit dafür dass ein E-Mail, welches das Wort «3d» enthält Spam ist P(3d|spam). Die Zahlen, welche Sie dafür benötigen, können Sie dem Report zum Classifier entnehmen.

```
Instances:    4601
...
                     Class (Spam)
Attribute                0      1
                     (0.61) (0.39)
================================
...
Wort_3d
  0                  2781.0 1775.0
  1                     9.0   40.0
  [total]            2790.0 1815.0
```

$$P(3d|spam) = \frac{P(spam|3d) \cdot P(3d)}{P(spam)} = \frac{40}{40+9} \cdot \frac{9+40}{2781+1775} = 2.5\,\%$$

3) Wählen Sie nun als Testmethode «Supplied test set» aus und laden Sie als Testset die Datei Spam-Test.arff. Das Test-Set enthält zwei Instanzen (E-Mails), wobei eine als Spam, die andere als nicht Spam klassifiziert wird. Erstaunlich ist, dass die beiden E-Mails sich nur in einem Wort unterscheiden. Finden Sie heraus, welches Wort dies ist und erklären Sie, weshalb es für die Klassifizierung einen solch grossen Unterschied macht.

Das erste E-Mail wird in die Klasse Spam = 1 (ist Spam), das zweite E-Mail in die Klasse Spam = 0 (kein Spam) eingeteilt. Der einzige Unterschied zwischen den beiden E-Mails besteht darin, dass das Wort «hp» im zweiten E-Mail vorkommt, im ersten aber nicht. Betrachtet man die Balkendiagramme aller 55 Attribute von Spam-Basis.arff, stellt man fest, dass das Wort «hp» praktisch nicht in Spam-Mails vorkommt (rot, rechts), in E-Mails die kein Spam sind aber trotzdem häufig zu finden ist (links, rot). Ein E-Mail, das also «hp» enthält, verliert an Wahrscheinlichkeit, ein Spam-Mail zu sein.



Aufgabe 3-5: Laden Sie mit Weka den Datensatz WetterProblem.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes (findet sich unter bayes » NaiveBayes). Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert (More options … » Output predictions » Choose » PlainText).

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

OneR

```
=== Predictions on training set ===
 inst#     actual  predicted error prediction
     1     1:nein     1:nein       1
     2     1:nein     1:nein       1
     3       2:ja       2:ja       1
     4       2:ja       2:ja       1
     5       2:ja       2:ja       1
     6     1:nein       2:ja   +   1
     7       2:ja       2:ja       1
     8     1:nein     1:nein       1
     9       2:ja     1:nein   +   1
    10       2:ja       2:ja       1
    11       2:ja     1:nein   +   1
    12       2:ja       2:ja       1
    13       2:ja       2:ja       1
    14     1:nein       2:ja   +   1
=== Summary ===
Correctly Classified Instances          10
71.4286 %
Incorrectly Classified Instances         4
28.5714 %
Kappa statistic
0.3778
```

NaiveBayes

```
=== Predictions on training set ===
 inst#     actual  predicted error prediction
     1     1:nein     1:nein       0.704
     2     1:nein     1:nein       0.847
     3       2:ja       2:ja       0.737
     4       2:ja       2:ja       0.554
     5       2:ja       2:ja       0.867
     6     1:nein       2:ja   +   0.737
     7       2:ja       2:ja       0.913
     8     1:nein     1:nein       0.588
     9       2:ja       2:ja       0.786
    10       2:ja       2:ja       0.845
    11       2:ja       2:ja       0.568
    12       2:ja       2:ja       0.667
    13       2:ja       2:ja       0.925
    14     1:nein     1:nein       0.652
=== Summary ===
Correctly Classified Instances          13
92.8571 %
Incorrectly Classified Instances         1
7.1429 %
Kappa statistic
0.8372
```

Aufgabe 3-6: *Laden Sie mit Weka den Datensatz WetterProblem-Numerisch.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes (findet sich unter bayes » NaiveBayes). Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert (More options … » Output predictions » Choose » PlainText).*

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

OneR

```
=== Predictions on training set ===
 inst#     actual  predicted error prediction
     1     1:nein     1:nein       1
     2     1:nein     1:nein       1
     3       2:ja       2:ja       1
     4       2:ja       2:ja       1
     5       2:ja       2:ja       1
     6     1:nein       2:ja   +   1
     7       2:ja       2:ja       1
     8     1:nein     1:nein       1
     9       2:ja     1:nein   +   1
    10       2:ja       2:ja       1
    11       2:ja     1:nein   +   1
    12       2:ja       2:ja       1
    13       2:ja       2:ja       1
    14     1:nein       2:ja   +   1

=== Summary ===
Correctly Classified Instances          10
71.4286 %
Incorrectly Classified Instances         4
28.5714 %
Kappa statistic
0.3778
```

NaiveBayes

```
=== Predictions on training set ===
inst#     actual  predicted error prediction
     1     1:nein     1:nein       0.723
     2     1:nein     1:nein       0.831
     3       2:ja       2:ja       0.765
     4       2:ja       2:ja       0.512
     5       2:ja       2:ja       0.787
     6     1:nein       2:ja   +   0.798
     7       2:ja       2:ja       0.952
     8     1:nein     1:nein       0.638
     9       2:ja       2:ja       0.84
    10       2:ja       2:ja       0.75
    11       2:ja       2:ja       0.647
    12       2:ja       2:ja       0.719
    13       2:ja       2:ja       0.904
    14     1:nein     1:nein       0.594

=== Summary ===
Correctly Classified Instances          13
92.8571 %
Incorrectly Classified Instances         1
7.1429 %
Kappa statistic
0.8372
```

Aufgabe 3-7: *Laden Sie mit Weka den Datensatz Iris.csv und klassifizieren Sie ihn mit Hilfe von OneR und NaiveBayes (findet sich unter bayes » NaiveBayes). Sorgen Sie dafür, dass Ihnen Weka die Klassifizierung der einzelnen Instanzen im Report mitliefert (More options … » Output predictions » Choose » PlainText).*

Vergleichen Sie die Resultate der beiden Klassifikations-Methoden.

OneR

```
=== Summary ===
Correctly Classified Instances         144
96      %
Incorrectly Classified Instances         6
4       %
Kappa statistic
0.94
```

NaiveBayes

```
=== Summary ===
Correctly Classified Instances         144
96      %
Incorrectly Classified Instances         6
4       %
Kappa statistic
0.94
```

9.4 KAPITEL 4: KLASSIFIZIERUNG – ENTSCHEIDUNGSBAUM

Aufgabe 4-1: Finden Sie für folgendes Beispiel eine Codierung, die mit weniger als den vorgeschlagenen 2.00 Bit auskommt. Theoretisch liesse es sich mit 1.585 Bit realisieren, wir sind aber zufrieden, wenn Sie es mit 1.67 Bit schaffen.

Symbol X	A	B	C	Bit
Wahrscheinlichkeit P(X)	1/3	1/3	1/3	2.00
Codierung (nicht optimal)	00	01	10	

1/3 · 2 + 1/3 · 2 + 1/3 · 2 = 2.00 Bit

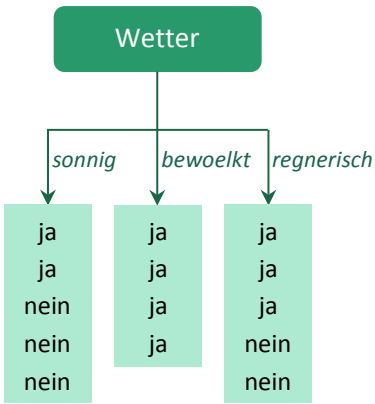
P(A) · Anzahl Bits (A) + P(B) · Anzahl Bits (B) + P(C) · Anzahl Bits (C)

Symbol X	A	B	C	Bit
Wahrscheinlichkeit P(X)	1/3	1/3	1/3	1.67
Codierung (optimiert)	0	10	11	

1/3 · 1 + 1/3 · 2 + 1/3 · 2 = 1.67 Bit

P(A) · Anzahl Bits (A) + P(B) · Anzahl Bits (B) + P(C) · Anzahl Bits (C)

Aufgabe 4-2: Berechnen Sie den Informationsgewinn IG(Anlass|Wetter) für den dargestellten Wurzelknoten des «Wetter-Problems».



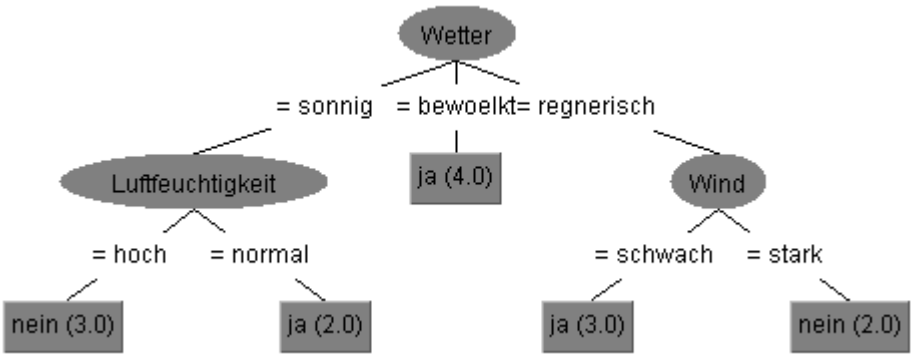
H(Y)	Anzahl ja	Anzahl nein	ja + nein			**0.9403**
	9	5	14			

H(Y|X)	Anzahl ja	Anzahl nein	ja + nein	H(Y|Xi)	H(Y|Xi)*pi	**0.6935**
sonnig	2	3	5	0.9710	0.3468	
bewölkt	4	0	4	0.0000	0.0000	
regnerisch	3	2	5	0.9710	0.3468	

IG(Y|X)	**0.2467**

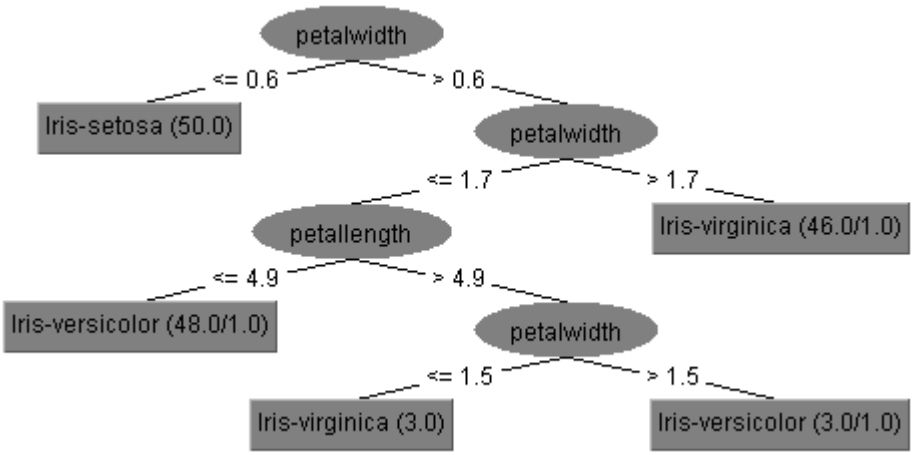
Aufgabe 4-3: *Laden Sie den Datensatz WetterProblem.csv und lassen Sie Weka den Entscheidungsbaum dazu lernen. Wählen Sie dazu unter dem Reiter **CLASSIFY** über **CHOOSE** den Classifier: trees » J48. Verwenden Sie als Test-Set das Trainings-Set*

*Der Entscheidungsbaum lässt sich darstellen, indem Sie in der **RESULT-LIST** (unten links) einen Rechtsklick auf den Eintrag machen und «Visualize tree» auswählen.*



Aufgabe 4-4: *Laden Sie mit Weka den Datensatz Iris.csv und klassifizieren Sie ihn mit Hilfe des Entscheidungsbaum-Algorithmus J48 (findet sich unter trees » J48).*

Stellen Sie den Entscheidungsbaum dar und vergleichen Sie die Resultate dieser Klassifikations-Methode mit OneR und NaiveBayes.



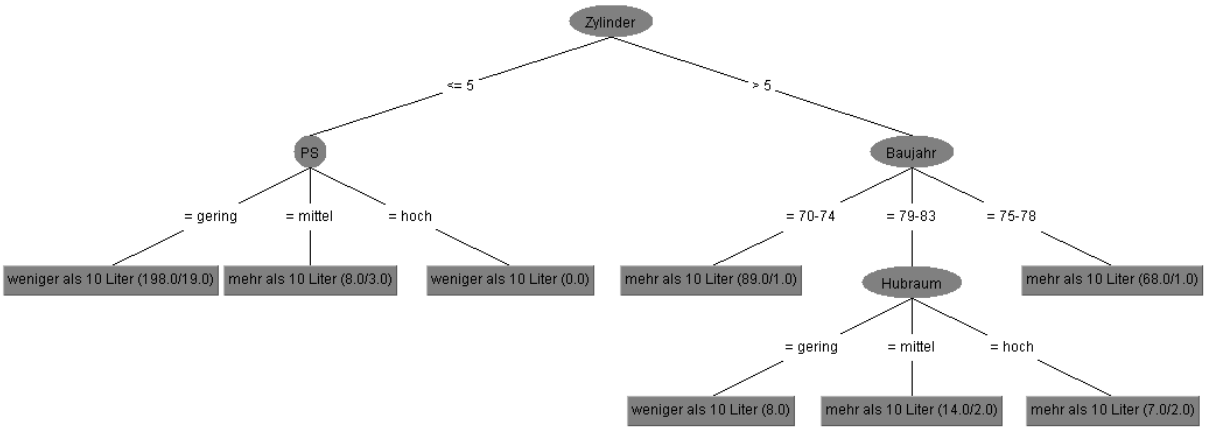

```
=== Summary ===
Correctly Classified Instances         147               98      %
Incorrectly Classified Instances         3                2      %
Kappa statistic                          0.97

=== Confusion Matrix ===
  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 49  1 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Aufgabe 4-5: *Laden Sie mit Weka den Datensatz auto-lkm.csv. Dabei handelt es sich um einen Datensatz, welcher 392 verschiedene Autos mit den folgenden Attributen beschreibt:*

- Name des Autos
- Herkunft (Asien, Europa, Nordamerika)
- Baujahr (70-74, 75-78, 79-83)
- Zylinder (3,4,5,6,7,8)
- Hubraum (gering, mittel, hoch)
- PS (gering, mittel, hoch)
- Gewicht (gering, mittel, hoch)
- Beschleunigung (gering, mittel, hoch)
- Treibstoffverbrauch (mehr als 10 Liter, weniger als 10 Liter).

Das Attribut «Treibstoffverbrauch» beschreibt, wie viele Liter Treibstoff ein Auto pro 100 km Fahrstrecke verbraucht und weist die beiden Werte «mehr als 10 Liter» resp. «weniger als 10 Liter» auf.

1) Klassifizieren Sie die Daten nach dem «Treibstoffverbrauch» mit Hilfe des Entscheidungsbaum-Algorithmus J48. Wählen Sie dazu als Test-Option «Cross-Validation» aus. Linksklicken Sie die Classifier-Parameter «J48 -C 0.25 -M 2» und setzen Sie minNumObj = 4.
Lassen Sie sich von Weka den Entscheidungsbaum anzeigen und machen Sie eine Aussage über die Qualität des Classifiers.




```
=== Summary ===
Correctly Classified Instances         356               90.8163 %
Incorrectly Classified Instances        36                9.1837 %
Kappa statistic                          0.8163

=== Confusion Matrix ===
   a   b   <-- classified as
 185  11 |   a = weniger als 10 Liter
  25 171 |   b = mehr als 10 Liter
```

2) Klassifizieren Sie den Datensatz nun auch mit NaiveBayes und mit OneR und verwenden Sie bei beiden Methoden ebenfalls die Kreuzvalidierung als Test-Option. Vergleichen Sie die Qualität der Classifier mit jener von J48.

OneR (Cross Validation):

```
=== Summary ===
Correctly Classified Instances         252               64.2857 %
Incorrectly Classified Instances       140               35.7143 %
Kappa statistic                          0.2857

=== Confusion Matrix ===
   a   b   <-- classified as
 189   7 |   a = weniger als 10 Liter
 133  63 |   b = mehr als 10 Liter
```

NaiveBayes (Cross Validation):

```
=== Summary ===
Correctly Classified Instances         355               90.5612 %
Incorrectly Classified Instances        37                9.4388 %
Kappa statistic                          0.8112

=== Confusion Matrix ===
   a   b   <-- classified as
 183  13 |   a = weniger als 10 Liter
  24 172 |   b = mehr als 10 Liter
```

3) Experimentieren Sie mit verschiedenen Werten für minNumObj beim J48-Algorithmus. Was bewirkt dieser Parameter?

Dieser Parameter setzt die Menge an Instanzen fest, über welche ein Knoten mindestens verfügen muss. Je grösser dieser Wert, desto weniger Knoten (= einfacher lesbar) weist der Entscheidungsbaum auf. Wird dieser Wert zu gross gewählt, dann hat der Entscheidungsbaum zu wenige Knoten um die Daten genügend gut klassifizieren zu können. Ist der Wert zu klein, ergeben sich viele Knoten, die nur eine einzige Instanz enthalten. Dies führt zu grossen, unübersichtlichen Entscheidungsbäumen, die auf realen Daten meist nicht besser funktionieren, als einfachere.