

# HANSER



## Leseprobe

zu

## Maschinelles Lernen mit R

von Uli Schell

Print-ISBN: 978-3-446-47165-8  
E-Book-ISBN: 978-3-446-47244-0  
E-Pub-ISBN: 978-3-446-47323-2

Weitere Informationen und Bestellungen unter  
<https://www.hanser-kundencenter.de/fachbuch/artikel/9783446471658>

sowie im Buchhandel

© Carl Hanser Verlag, München

# Inhalt

Vorwort .....	XIII
Zusatzmaterial online .....	XIV
<b>Teil I Einstieg</b> .....	<b>1</b>
<b>1 Einleitung</b> .....	<b>3</b>
1.1 Informatik und künstliche Intelligenz .....	3
1.2 Expertensysteme .....	4
1.3 Maschinelles Lernen .....	5
1.3.1 Überwachtes Lernen .....	7
1.3.2 Unüberwachtes Lernen .....	7
1.3.3 Verstärkendes Lernen .....	7
1.4 Methoden und Werkzeuge für das maschinelle Lernen .....	7
1.5 Zielsetzungen dieses Buchs und Vorgehensweise .....	9
<b>2 Einführung in R und RStudio</b> .....	<b>11</b>
2.1 Installation unter Windows .....	11
2.2 Installation unter Ubuntu Linux .....	12
2.3 Die RStudio-Oberfläche .....	15
2.3.1 Das Konsolenfenster .....	15
2.3.2 Das Source-Fenster .....	16
2.3.3 Das Files-/Packages-Fenster .....	18
2.3.3.1 Die Files-Registerkarte .....	18
2.3.3.2 Die Plot-Registerkarte .....	20
2.3.3.3 Die Packages-Registerkarte .....	20
2.3.3.4 Das Environment-Fenster .....	22
2.3.4 Daten einlesen und speichern .....	23

2.4	Zuweisungen, Variablen und elementare Datentypen .....	26
2.5	Zusammengesetzte Datentypen .....	27
2.5.1	Vektoren .....	27
2.5.1.1	Union, intersect, setdiff .....	29
2.5.1.2	Umwandeln des Datentyps, Faktorisierung .....	30
2.5.2	Matrizen .....	31
2.5.3	Frames .....	32
2.5.3.1	Hinzufügen von Spalten .....	32
2.5.3.2	Hinzufügen von Zeilen .....	33
2.5.3.3	Auswahl von Spalten .....	33
2.5.3.4	Auswahl von Zeilen und Spalten .....	33
2.5.3.5	Löschen von Spalten oder Zeilen .....	34
2.5.4	tibbles .....	34
2.5.5	Listen .....	35
2.5.6	Zeichenketten .....	36
2.5.7	Datum und Zeit .....	36
2.6	Programmablaufsteuerung .....	37
2.6.1	Blöcke .....	37
2.6.2	Bedingte Ausführung .....	37
2.6.3	Schleifen .....	38
2.6.4	Apply() .....	38
2.6.5	Gefahren und Benachrichtigungen .....	39
2.6.6	Funktionen .....	39
2.6.7	Der Pipe-Operator .....	41
2.7	Debugging .....	41
2.8	Programmierstil .....	43
2.8.1	Variablen- und Funktionsbenennung .....	43
2.8.2	Abstände .....	44
2.8.3	Blöcke .....	44
2.8.4	Abschnitte .....	45
2.9	Formatierhilfen .....	45
2.9.1	Styler .....	45
2.9.2	Lint .....	45
2.10	Zum Nachschlagen .....	46
2.11	Einstiegsaufgabe .....	46

<b>Teil II Datenvorbereitung</b> .....	<b>47</b>
<b>3 Daten visualisieren und vorbereiten</b> .....	<b>49</b>
3.1 Plots mit ggplot2 .....	50
3.1.1 Plot-Architekturen .....	50
3.1.2 Erster Plot mit ggplot2.....	50
3.1.3 Aufbau eines Plots.....	51
3.1.4 Ebenen.....	54
3.1.5 Geome .....	55
3.1.6 Stats .....	57
3.1.7 Themes .....	60
3.2 Datenaufbereitung .....	61
3.3 Strukturierung der Daten .....	62
3.3.1 Werte in Spaltenüberschriften .....	63
3.3.2 Mehrere Werte in einer Spalte .....	63
3.3.3 Imputation und Grafikdarstellung.....	65
3.3.4 Überwachung von Datentypen/Bereichsgrenzen .....	66
3.4 Datenaufbereitung an Zeitreihen – ein Beispiel .....	66
3.4.1 Erster Überblick .....	66
3.4.2 Messfehler.....	67
3.4.3 Überprüfen der zeitlichen Abfolge.....	68
3.4.4 Zeitserie generieren.....	69
3.4.4.1 Bestimmung der mittleren Tagestemperatur .....	69
3.4.4.2 Bestimmung der mittleren Jahrestemperatur .....	70
3.4.4.3 Imputation – Ersetzen von NA-Werten .....	70
3.5 Zwei kleine Checklisten .....	72
3.6 Aufgaben .....	73
<b>4 Datenplausibilität</b> .....	<b>75</b>
4.1 Hypothesen-Betrachtung.....	76
4.2 Allgemeine Kenngrößen als Hilfsmittel.....	77
4.2.1 Mittelwert und Median .....	77
4.2.2 Varianz.....	79
4.2.3 Momente höherer Ordnung .....	80
4.3 Grafische Hilfsmittel .....	81
4.3.1 Histogramme .....	81
4.3.2 Box-Plots .....	81
4.3.3 Summenhäufigkeiten und QQ-Plots .....	82
4.4 R-Packages zum Erkennen von Ausreißern .....	83

4.4.1	Das Package „Outliers“ .....	83
4.4.1.1	scores() .....	84
4.4.1.2	Dixon() .....	85
4.4.1.3	Cochran.test() .....	86
4.4.1.4	Grubbs-Test .....	87
4.4.1.5	Weitere Funktionen .....	88
4.5	Datenplausibilität bei mehreren Variablen und weitere Funktionen .....	88
4.6	Aufgaben .....	89
<b>Teil III Statistische Lernmodelle .....</b>		<b>91</b>
<b>5</b>	<b>Regression .....</b>	<b>93</b>
5.1	Regression mit einer unabhängigen Variablen .....	94
5.1.1	Methode der kleinsten Fehlerquadrate .....	94
5.1.2	Homoskedastizität .....	96
5.1.3	Modellvalidierung .....	98
5.1.3.1	Residuen .....	101
5.1.4	Vorbeugende Wartung .....	105
5.1.4.1	Das Modellbeispiel .....	105
5.1.4.2	Etwas mehr Residuenanalyse .....	106
5.1.4.3	Vorhersagen .....	110
5.1.5	Erweiterung der Regression auf nichtlineare Funktionen .....	113
5.1.6	Kreuzvalidierung .....	114
5.2	Regression mit mehreren unabhängigen Variablen .....	116
5.2.1	Der Boston-Datensatz .....	116
5.2.2	Durchführung der Regression .....	122
5.2.3	Modellvalidierung .....	124
5.2.4	Regressionen robuster machen .....	128
5.2.4.1	Regularisierung .....	128
5.2.4.2	M-Schätzer .....	130
5.2.4.3	Weitere Alternativen .....	131
5.3	Aufgaben .....	133
<b>6</b>	<b>Klassifikation .....</b>	<b>135</b>
6.1	Logistische Regression .....	136
6.2	Der Perceptron-Algorithmus .....	140
6.3	Support Vector Machines .....	143
6.3.1	Der Iris-Datensatz .....	145

6.4	Entscheidungsbaumverfahren.....	146
6.4.1	Entscheidungsbäume .....	146
6.4.2	Der Iris-Datensatz.....	148
6.4.3	Bagging .....	148
6.4.4	Random Forests .....	149
6.4.5	Boosted Regression Trees .....	149
6.5	Naive Bayes-Klassifikatoren .....	150
6.5.1	Multinomiale naive Bayes-Klassifikatoren .....	150
6.5.2	Das spam-Beispiel .....	152
6.5.3	Likelihood .....	155
6.5.4	Gauß'sche naive Bayes-Klassifizierer .....	156
6.6	KNN Nächste-Nachbarn-Klassifikation .....	158
6.7	Modellbewertung: Devianzen und universellere Kenngrößen .....	161
6.7.1	Receiver Operating Characteristic, ROC und AUC.....	161
6.7.2	Die $R^2$ -Metrik.....	163
6.7.3	Eine generalisierte Metrik.....	164
6.7.3.1	Pseudo- $R^2$ .....	164
6.7.3.2	Devianzen.....	167
6.8	Aufgaben .....	167
<b>7</b>	<b>Objekte clustern, Merkmale reduzieren und Zeitreihen zerlegen....</b>	<b>169</b>
7.1	K-means-Clustering .....	170
7.2	Korrelationen und Merkmalsreduktion durch Hauptkomponentenanalyse .....	173
7.2.1	Der beste Standpunkt .....	173
7.2.2	Kovarianzen.....	174
7.2.3	Kovarianzmatrizen.....	176
7.2.4	Anwendungen .....	180
7.2.4.1	Eine kleine Weinprobe .....	180
7.2.4.2	Der Boston-Datensatz.....	185
7.2.4.3	Erweiterungen der Hauptkomponentenanalyse .....	188
7.2.4.4	Ausreißerererkennung.....	189
7.3	Zeitreihen .....	189
7.3.1	Komponentenmodelle .....	189
7.3.2	Glättungsverfahren bei Zeitreihen.....	190
7.3.2.1	Gleitender Durchschnitt .....	191
7.3.2.2	Exponentielle Glättung.....	192
7.3.2.3	Holt-Winters-Glättung .....	194
7.3.2.4	Weitere Glättungsmethoden .....	195

7.3.3	AR-Modelle.....	196
7.3.3.1	Autokorrelationsfunktionen.....	197
7.3.3.2	Partielle Autokorrelationsfunktion .....	198
7.3.4	MA-Modelle.....	198
7.3.5	ARMA- und ARIMA-Modelle .....	200
7.4	Aufgaben.....	202
<b>Teil IV Lernen mit neuronalen Netzen.....</b>		<b>203</b>
<b>8</b>	<b>Neuronale Netze .....</b>	<b>205</b>
8.1	Das Perceptron.....	205
8.2	Layers.....	207
8.3	Aktivierungsfunktionen.....	208
8.4	Warum das Stapeln linearer Funktionen nicht sinnvoll ist .....	211
8.5	Der Regelkreis eines Lernvorgangs.....	213
8.5.1	Verlustfunktionen und Metriken.....	213
8.5.2	Epochen und Batches .....	214
8.5.3	Gradienten, lokale Gradienten und Autodiff .....	215
8.5.3.1	Manuelle Bestimmung des Gradienten eines Perceptrons ....	216
8.5.3.2	Berechnung des Gradienten mit CAS oder Differenzenquotienten.....	217
8.5.3.3	Berechnung des Gradienten mit Forward-Mode Autodiff .....	217
8.5.3.4	Berechnung des Gradienten mit Backpropagation Autodiff...	218
8.5.4	Der Optimizer .....	224
8.5.4.1	Stochastisches Gradientenverfahren (SGD).....	225
8.5.4.2	Momenten-Update .....	226
8.5.4.3	Nesterov-Moment .....	226
8.5.4.4	Adagrad .....	226
8.5.4.5	Rmsprop .....	227
8.5.4.6	Adadelta.....	227
8.5.4.7	Adam .....	228
8.5.4.8	ADAmx.....	228
8.5.4.9	Was tun, wenn ...? .....	229
8.6	Regularisierung und Dropouts.....	230
8.6.1	Regularisierung.....	230
8.6.2	Dropouts .....	230
8.7	Aufgaben.....	231

<b>9</b>	<b>H2O</b> .....	<b>233</b>
9.1	Das Unternehmen H2O .....	234
9.2	Installation und erste Schritte .....	234
9.3	Univariate lineare Regression .....	236
9.3.1	Generalisierte lineare Modelle .....	237
9.3.1.1	Lambda-Suche .....	239
9.3.1.2	Grid-Suche .....	239
9.4	Entscheidungsbäume, Random Forests und Gradient Boosting .....	240
9.5	Neuronale Netze .....	245
9.6	Der Boston-Datensatz .....	249
9.6.1	AutoML .....	254
9.6.2	Explain .....	256
9.6.2.1	Residuenanalyse .....	257
9.6.2.2	Variablenwichtigkeit .....	257
9.6.2.3	Heatmap der Variablenwichtigkeit .....	258
9.6.2.4	Modell-Korrelation .....	258
9.6.2.5	Partielles Abhängigkeitsdiagramm .....	260
9.7	Iris .....	260
9.7.1	Stacked Ensemble .....	261
9.8	MNIST .....	266
9.8.1	Der Standarddatensatz .....	266
9.8.2	Bewertung .....	270
9.9	Aufgaben .....	271
<b>10</b>	<b>Keras/Tensorflow</b> .....	<b>273</b>
10.1	Einrichtung und Nutzung von Keras .....	274
10.1.1	Dimensionen und Tensoren .....	274
10.1.2	Normalisierung .....	276
10.2	Boston .....	276
10.2.1	Normalisierung .....	276
10.2.2	Modelldefinition .....	277
10.2.3	Compilierung .....	278
10.2.4	Fit .....	279
10.3	Diagnosemöglichkeiten und Optimierung .....	282
10.3.1	Eine kleine Regression .....	282
10.3.2	Speichern und Laden des Modells und der Gewichte .....	285
10.3.3	Auslesen des Modells .....	286
10.3.4	Callbacks .....	287



10.3.5	TensorBoard .....	287
10.4	Convolutional Networks .....	291
10.4.1	Faltung und Feature Learning .....	291
10.4.1.1	Diskrete 2D-Faltung .....	292
10.4.1.2	Aufbau von Mustern .....	294
10.4.1.3	Pooling Layers .....	296
10.4.2	Komposition der Layer .....	297
10.4.3	MNIST .....	298
10.4.4	ImageNet .....	300
10.5	Transferlernen .....	304
10.5.1	Modelldefinition .....	304
10.5.2	Datenbereitstellung .....	306
10.5.3	Augmentation .....	309
10.5.4	Lernvorgänge .....	311
10.6	Recurrent Networks .....	314
10.6.1	Simple Recurrent Networks .....	314
10.6.2	LSTM .....	316
10.6.3	Wettervorhersage .....	318
10.7	Aufgaben .....	326
<b>Teil V Anhang .....</b>		<b>329</b>
<b>A Basiswissen Statistik .....</b>		<b>331</b>
A.1	Beschreibende Statistik .....	331
A.1.1	Mittelwert und Median .....	331
A.1.2	Varianz .....	333
A.1.3	Momente höherer Ordnung .....	334
A.1.4	Histogramme, Summenhäufigkeiten und Quantile .....	334
A.1.5	Box-Plots .....	338
A.2	Schließende Statistik .....	338
A.2.1	Normalverteilung .....	339
A.2.2	t-(Student-)Verteilung .....	342
A.2.3	Chi-Quadrat-Verteilung .....	344
A.2.4	F-Verteilung .....	346
<b>Literatur .....</b>		<b>349</b>
<b>Stichwortverzeichnis .....</b>		<b>361</b>

# Vorwort

Künstliche Intelligenz ist einer der Haupt-Innovationstreiber der Gegenwart. Hätten Sie noch vor wenigen Jahren damit gerechnet, dass 2021 das erste Gesetz zum autonomen Fahren in Deutschland in Kraft treten würde? Ohne künstliche Intelligenz und maschinelles Lernen wäre das nicht möglich geworden.

Maschinelles Lernen – dieser Begriff weckt vielfältige Assoziationen. Zum einen erhofft man sich, dass damit der Alltag bequemer und sicherer gemacht wird. Zum anderen befürchtet man, dass immer noch intelligentere Geräte immer autonomer agieren. Vielleicht machen sie sich irgendwann sogar noch selbständig und stellen Unfug an? Um das besser beurteilen zu können, lohnt sich die Beschäftigung mit diesem Thema.

Und da liegt wahrscheinlich das eigentliche Problem: Es gibt noch wenig Einstiegsliteratur in das Thema „Maschinelles Lernen“, aber die technologischen Entwicklungen nehmen keine Rücksicht darauf. So kommt die Angst auf, dass man diesen rasanten Entwicklungen nicht mehr folgen kann, man fühlt sich „abgehängt“ und verschließt womöglich noch die Augen davor.

Ich denke, das muss nicht sein. Sie sehen das wahrscheinlich auch so, denn Sie haben sich dazu entschlossen, in dieses Thema einzusteigen, weil Sie gerade dieses Buch lesen.

Es war mir ein Ansporn, Ihnen diesen Weg mit vielen Beispielen zu ebnen und Ihnen möglichst viele Möglichkeiten einer Vertiefung zu bieten.

Allen Gesprächspartnern, die mich hierbei mit vielen guten Anregungen unterstützt haben, möchte ich an dieser Stelle sehr herzlich danken, insbesondere Frau Sylvia Hasselbach, Frau Irene Weilharth, Herrn Dr. Jochen Hirschle und Herrn Jürgen Dubau.

Trotz großer Sorgfalt lässt sich der eine oder andere Fehler nicht vermeiden. Wenn Sie also Kritik, Anmerkungen oder auch Wünsche haben, bitte ich Sie um eine Mail an [uli.schell@gmx.de](mailto:uli.schell@gmx.de), damit ich mich darum kümmern kann.

Ich wünsche Ihnen, dass Ihnen der Einstieg in das maschinelle Lernen gut gelingt, dass dieses Buch einen Beitrag dazu leisten konnte, und vor allem: dass Sie dabei Ihre Freude haben!

*Uli Schell, Dezember 2021*

# 6

## Klassifikation



### Fragen, die dieses Kapitel beantwortet:

- Wie kann man Objekte Gruppen zuordnen?
- Wie kann man Spam-Nachrichten anhand des Texts erkennen?
- Was kann man mit einem Bayes-Klassifikator noch machen?
- Was sind Support Vector Machines?
- Was versteht man unter Likelihood?
- Wie kann man eine allgemeinere Metrik für  $R^2$  formulieren?
- Was versteht man unter einer Residuendevianz?

Bei der Klassifikation werden Objekte aufgrund ihrer Merkmale verschiedenen existierenden Klassen zugeordnet. Nummeriert man die Klassen durch, ist die Lernvariable bei der Klassifikation ein diskreter Wert (im Gegensatz zum kontinuierlichen Wert bei der Regression).

Eine Klassifikation kann durch die logistische Regression als eine Generalisierung des linearen Modells durchgeführt werden. Dies wird in [Abschnitt 6.1](#) behandelt. Auch mit dem Perceptron-Algorithmus ([Abschnitt 6.2](#)) und mit Support Vector Machines ([Abschnitt 6.3](#)) kann diese Klassifikationsaufgabe erfüllt werden. Eine weitere Alternative für Klassifikationen sind Entscheidungsbaumverfahren, die in [Abschnitt 6.4](#) behandelt werden.

Die Grundlagen zur Klassifizierung nach Bayes wurden bereits 1763 gelegt [[Wik21g](#)]. Die Erkennung von Junk-E-Mails mithilfe der Wahrscheinlichkeiten nach Bayes stieß auf große Beachtung und wird in [Abschnitt 6.5](#) gezeigt.

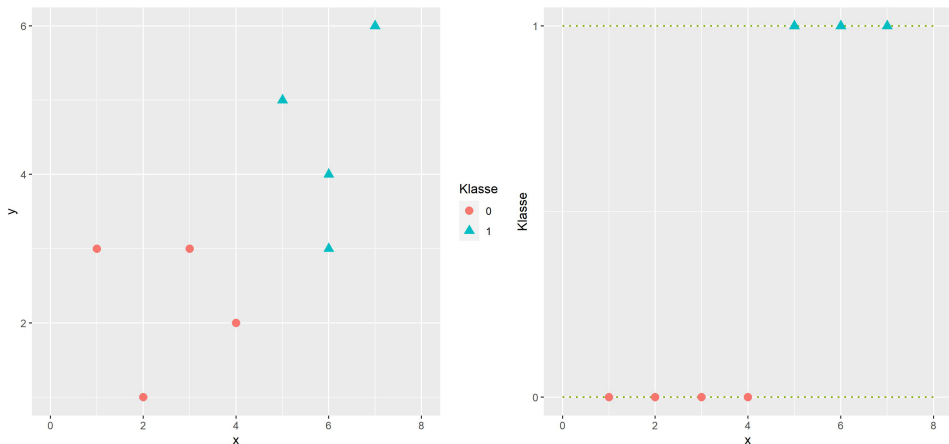
Das kNN-Verfahren, welches die Abstände von Datenpunkten zur Klassifikation nutzt, wird in [Abschnitt 6.6](#) gezeigt.

Das Kapitel schließt ab mit der Diskussion einer generalisierten Metrik, welche auf dem Prinzip der Log-Likelihood basiert.

## 6.1 Logistische Regression

Bei den Regressionen (Kapitel 5) haben wir das Verhalten einer kontinuierlichen abhängigen Variablen von einer oder mehreren unabhängigen Variablen beschrieben; wir hatten so etwas wie verbindende Eigenschaften gesucht. Im Gegensatz dazu wollen wir in diesem Kapitel eher trennende Eigenschaften suchen, also Trennlinien ziehen.

Betrachten wir Punkte auf einer Ebene (Bild 6.1 links). Im linken unteren Bereich befinden sich Punkte, die zu einer Klasse 0 gehören, die Dreiecke im rechten oberen Bereich gehören zu einer Klasse 1. Wir lassen jetzt die Information über die  $y$ -Werte weg und verwenden die  $y$ -Achse dazu, die Klassenzugehörigkeit darstellen zu können (Bild 6.1 rechts).



**Bild 6.1** Klassifikation

Zur Bestimmung einer Trennlinie könnten wir einem ersten Ansatz einfach versuchen, über eine lineare Regression diese Grenze zu bestimmen und einzuzichnen; die Grenze könnten wir bei 0.5 festlegen. Eine solche Gerade ist auf der linken Seite in Bild 6.2 gezeigt. Diese Methode hat aber den Nachteil, dass sehr kleine oder sehr große  $x$ -Werte zu Vorhersagewerten führen, die sehr weit unterhalb von 0 oder oberhalb von 1 liegen können. Weiterhin reduzieren sie aufgrund der Hebelwirkung die Steigung der Geraden, wodurch die Trennschärfe reduziert wird.

Deshalb suchen wir eine Abbildungsvorschrift, die für  $x \rightarrow -\infty$  gegen 0 geht, bei  $x = 0$  den Wert 0.5 hat und für  $x \rightarrow \infty$  gegen 1 geht. Eine Funktion, die das leistet, ist die logistische Funktion aus der Gruppe der Sigmoid-Funktionen (auch Schwanenhalsfunktionen, Fermi-funktionen oder S-Funktionen genannt).

Die logistische Funktion wird beschrieben durch eine Gleichung

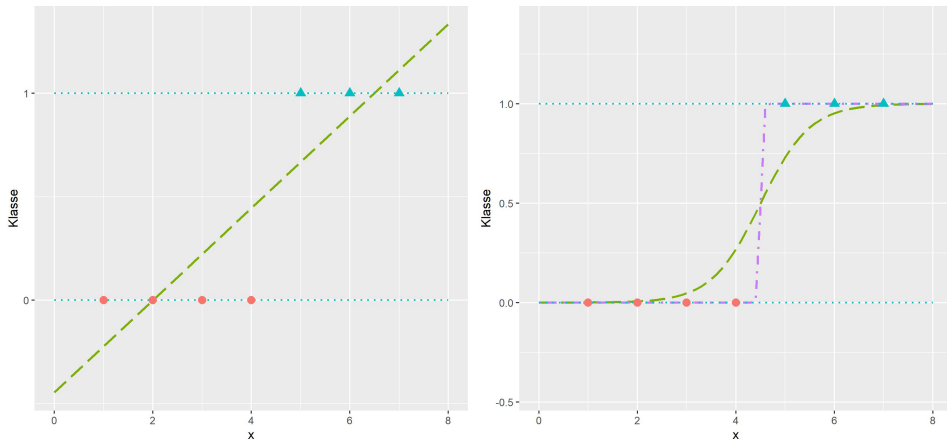
$$y = f(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

Die Umkehrfunktion der logistischen Funktion nennt man logit-Funktion:

$$x = \text{logit}(y) = f^{-1}(y) = \ln\left(\frac{y}{1-y}\right) = \ln(\text{odds})$$

Die Verwendung der Variablen *odds* ermöglicht eine kürzere Schreibweise.

Ein beispielhafter Verlauf der logistischen Funktion ist als gestrichelte Linie in [Bild 6.2](#) rechts gezeigt.



**Bild 6.2** Lineare versus logistische Funktion

Bei einer linearen Regression (s. [Kapitel 5](#)) nimmt man eine Normalverteilung für die Störgröße an und kann damit Aussagen über die Streuung machen. Ändert man nun die Abbildungsfunktion beispielsweise von einer linearen Funktion auf eine logistische Funktion ab, ist diese Annahme natürlich nicht mehr gültig. Dadurch sind manche Werte (wie z. B. der Vertrauensbereich) nicht mehr in der gewohnten Weise bestimmbar.

Die Einschränkung der Theorie der Regressionen auf normalverteilte Werte wurde durch die Arbeit von [\[NW72\]](#) aufgehoben und auf die gesamte Klasse der Exponentialfamilie erweitert. Neben der Gauß'schen Normalverteilung sind darin auch die Binomial-, Poisson-, Gamma- und inverse Gauß-Verteilung einbezogen. Diese Modelle nennt man generalisierte lineare Modelle (GLM).

In R können Regressionen dieser Klasse über die Funktion `glm()` aufgerufen werden:

```
> logistische_regression <- glm(label ~ x,
                               family = binomial("logit"))

Warnmeldung:
glm.fit: Angepasste Wahrscheinlichkeiten mit numerischem Wert 0 oder 1
aufgetreten
> summary(logistische_regression)
Call:
glm(formula = label ~ x, family = binomial("logit"))
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.364e-05 -2.110e-08  0.000e+00  2.110e-08  1.378e-05
:
:
:
:
:
```

Wir erhalten eine Warnmeldung, da die logistische Funktion erst für  $z \rightarrow \pm\infty$  die Werte 0 oder 1 annimmt und man solche Werte in der Praxis kaum vorfindet. Die Vorhersagewerte der Regression kann man erhalten über:

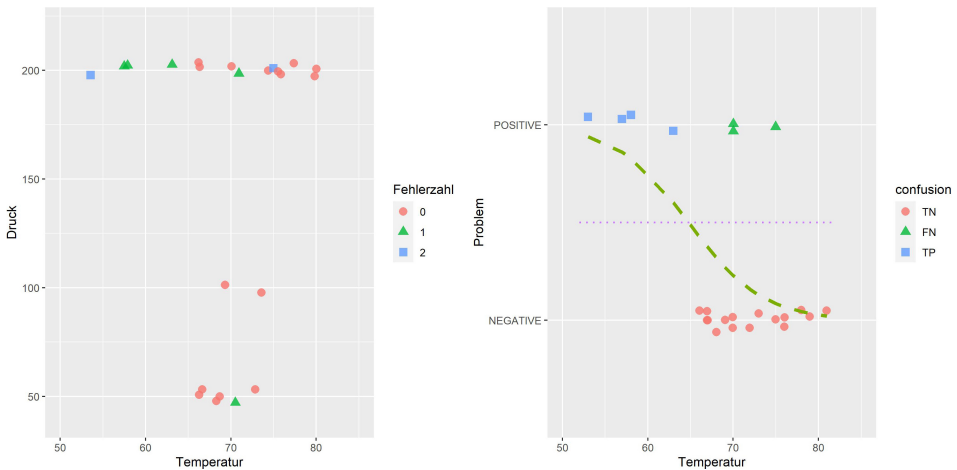
```
> predict_y <- predict(logistische_regression,x,type="response")
```

Der Verlauf der Trennlinie ist als strichpunktierte Linie in [Bild 6.2](#) rechts gezeigt.

Man erhält damit Werte zwischen 0 und 1; liegt ein Wert oberhalb von 0.5, zählt er zur Klasse 1, im anderen Fall zur Klasse 0.

Eine Anwendung der logistischen Regression ist das Problem mit den Dichtungen der Raumfähre „Challenger“ der NASA, welche im Jahr 1986 bei ihrem Start zerbrach. Im Datensatz `shuttle` des Package `SMPRACTICALS` sind die Fehlerzahlen bei bestimmten Temperaturen und Drücken festgehalten ([Bild 6.3](#) links).

Wir bilden aus den Fehlerzahlen die kategoriale Variable „Problem“. Sie ist TRUE, wenn mehr als ein Fehler auftritt. Wir tragen „Problem“ über der Temperatur auf und führen eine logistische Regression durch. Das Ergebnis ist in [Bild 6.3](#) rechts gezeigt.



**Bild 6.3** Logistische Regression der Dichtungen

Mit abnehmender Temperatur steigt die Wahrscheinlichkeit für ein Problem.

Sachverständige haben festgestellt, dass die mangelhafte Temperaturbeständigkeit der Dichtungsringe für diese Katastrophe mit verantwortlich war ([Wik21h](#)).

Aus den vorhergesagten und aktuellen Werten können wir eine Tabelle ([Bild 6.4](#)) aufbauen und daran einige Bewertungsmaße betrachten:

```
> ringe$problem_pred
[1] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> ringe$problem_actual
[1] 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0
```

Diese Tabelle vergleicht die Zahl der wirklichen mit den vorhergesagten Werten. In den Spalten trägt man die Häufigkeit der vorhergesagten Werte ein, verglichen mit den wirklichen Häufigkeiten (die aktuellen Klassenzugehörigkeiten) in den Zeilen. Diese Tabellen nennt man Konfusionsstabellen (oder auch Konfusionsmatrix, Confusion Matrix oder allgemein Kontingenztabelle):

		Predict	
		Positiv (PP)	Negativ (PN)
Aktuell	Positiv (P)	TP 4	FN 3
	Negativ (N)	FP 0	TN 16

Bild 6.4 Konfusionstabelle

„True positiv“ (TP, auch „hit“ genannt) bedeutet: Es wurde eine positive Vorhersage (predicted positive, PP) getroffen, wobei dies dem aktuellen Wert (positive, P) entspricht. Bei unserem Datensatz trat dies viermal auf.

Entsprechend bedeutet „True negative“ (TN, auch „korrekte Rückweisung“ oder „correct rejection“ genannt) die zutreffende Vorhersage eines negativen Ergebnisses.

„False positive“ (FP, falsch positiv) – hier wurde positiv vorhergesagt, der aktuelle Wert ist negativ. Zusammen mit „False negative“ erhält man so die Summe der Fehlklassifikationen.



Zeilen und Spalten von Konfusionstabellen sind nicht einheitlich geregelt. Sie sind häufig vertauscht.

Anhand dieser Konfusionsmatrix kann man eine Bewertung vornehmen, indem man die Zahl der Korrekt-Klassifikationen zur Gesamtzahl der Klassifikationen ins Verhältnis setzt. Diesen Wert nennt man Korrekt-Klassifikationsrate (auch Treffergenauigkeit oder Accuracy genannt).

Entsprechend ist die Fehlklassifikationsrate das Verhältnis der Fehlklassifikationen zur Gesamtzahl der Klassifikationen.

Unter der Sensitivität (auch Abdeckung, recall, hit rate oder true positive rate (TPR) genannt) versteht man das Verhältnis von true positive zu positive:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = \frac{4}{7}$$

Dieser Wert ist beeinflussbar durch das Verhältnis von actual positive (P) zu allen Fällen. Dieses Verhältnis nennt man die Prävalenz (PRV):

$$PRV = \frac{P}{P + N} = \frac{7}{23}$$

Ist die Prävalenz sehr groß, hätten wir schon dadurch gute Vorhersagewerte, indem wir immer „positiv“ tippen (dies macht man sich bei Glücksspielen zu eigen).

Die Falsch-Positiv-Rate (false positive rate, fall-out, FPR) ist das Verhältnis der Falsch-positiven Werte zu den negativen:

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = \frac{0}{16}$$

Ein anderes Maß ist die Spezifität (true negative rate, TNR), das Verhältnis von korrekten Rückweisungen zur Zahl aktueller negativer Werte (N):

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = \frac{16}{16 + 0}$$

Bei sehr kleinen Prävalenzwerten haben wir schon dadurch gute Vorhersagewerte, indem wir immer N vorhersagen. In diesem Fall ist TPR nur bedingt tauglich für eine Bewertung von Klassifikationen.

Ein anderes Bewertungsmaß ist die Präzision (auch positiver Vorhersagewert, positive predictive value, PPV):

$$PPV = \frac{TP}{TP + FP} = \frac{4}{4} = 1$$

Nach diesem Maßstab wäre unser Klassifikator in unserem Beispiel „präzise“, was der allgemeinen Vorstellung über Präzision sicherlich nicht entspricht. Dieses Phänomen wird als Genauigkeitsparadoxon bezeichnet.

Zur Vermeidung dieses Nachteils kann man das F1-Maß als harmonisches Mittel zwischen Recall und Precision verwenden:

$$F1 = 2 \cdot \frac{1}{\frac{1}{TPR} + \frac{1}{PPV}} = 2 \cdot \frac{TPR \cdot PPV}{TPR + PPV} = \frac{\frac{4}{7} \cdot 1}{\frac{4}{7} + 1} = \frac{4}{11}$$

Der Nachteil dieses Maßes ist die fehlende Berücksichtigung der TN-Fälle.

In [Abschnitt 6.7](#) werden wir uns mit der Frage der Metriken noch weiter auseinandersetzen.

Generalisierte Modelle können – wie im Fall der linearen Regression – auch für die Regression mit mehreren Variablen eingesetzt werden:

```
> multiple_logistische_regression <- lm(label ~ x1 + x2)
```

Auch wenn dieses Verfahren „logistische Regression“ heißt, zählt es zu den Klassifikationsverfahren.

Wir werden die logistische Regression in [Abschnitt 9.3](#) wieder verwenden.

## ■ 6.2 Der Perceptron-Algorithmus

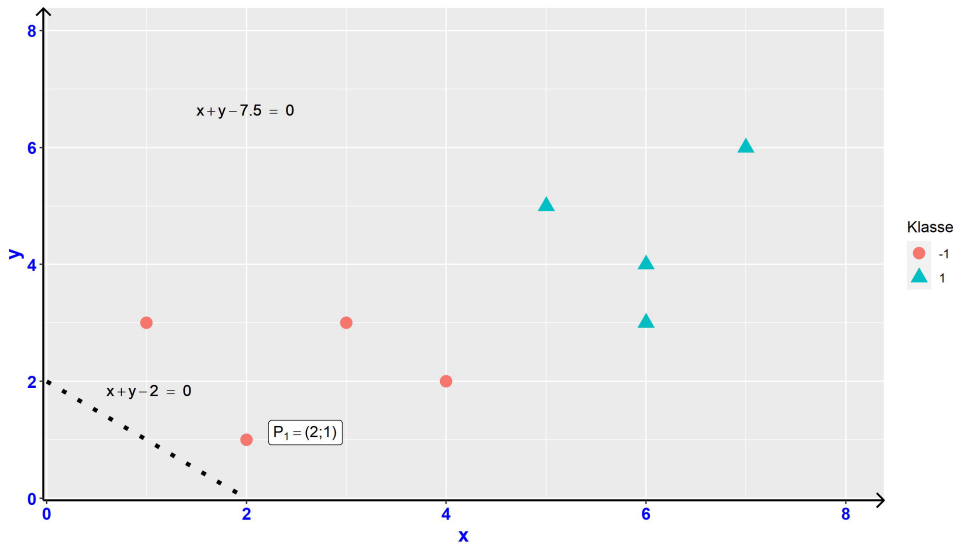
Eine andere Möglichkeit zur Durchführung von Klassifikationen ist der Perceptron-Algorithmus. Dieser Algorithmus ist ein historischer Ansatz von [\[ROS58\]](#), welcher die Grundlage neuronaler Netze gebildet hat.

Wir betrachten wieder Punkte auf einer Ebene ([Bild 6.5](#)). Im linken unteren Bereich befinden sich Punkte, die zu einer Klasse  $-1$  gehören, die Dreiecke im rechten oberen Bereich gehören zu einer Klasse  $+1$ . Gesucht ist eine Gerade als Trennlinie.



Wir starten mit einer willkürlich festgelegten Geraden durch die Punkte (2;0) und (0;2) und verschieben und rotieren sie solange, bis sie beide Klassen möglichst gut trennt.

Bei einer Beschreibung der Geraden durch eine Gleichung der Form  $y = m \cdot x + b = -x + 2$  könnte eine Rotation durch Ändern von  $m$  und eine Verschiebung (Translation) durch Ändern von  $b$  durchgeführt werden (so haben wir es im Fall der Bestimmung einer Regressionsgeraden in [Abschnitt 5.1](#) praktiziert).



**Bild 6.5** Der Perceptron-Algorithmus

Diesmal wollen wir jedoch neben der Translation anstelle eines Dreh- und Angelpunkts zwei Dreh- und Angelpunkte (Pivot-Punkte) verwenden. Dies gelingt durch die Darstellung der Geraden in der Form

$$a \cdot x + b \cdot y + c = 0, \quad \text{also z. B.} \quad x + y - 2 = 0$$

Dann haben wir zwei Dreh- und Angelpunkte (Pivotpunkte), nämlich die Schnittpunkte mit der  $x$ - bzw.  $y$ -Achse, die wir verschieben:

- Erhöht man den Wert von  $a$ , verbleibt der Schnittpunkt mit der  $y$ -Achse unverändert, während sich der Schnittpunkt mit der  $x$ -Achse nach links verschiebt (und umkehrt).
- Erhöht man den Wert von  $b$ , verbleibt der Schnittpunkt mit der  $x$ -Achse unverändert, während sich der Schnittpunkt mit der  $y$ -Achse nach unten verschiebt (und umkehrt).
- Erhöht man den Wert von  $c$ , verschiebt man die Gerade (ohne Drehung) nach links unten.

Die gewählte Geradendarstellung kann man auch dazu verwenden, die Lage eines Punktes bezüglich der Geraden festzustellen. Dazu setzt man einfach seine Koordinatenwerte, beispielsweise den Punkt (2, 1), in die Geradengleichung ein:

$$\text{Lage} = a \cdot x + b \cdot y + c, \quad \text{also z. B.} \quad \text{Lage} = 1 + 2 - 2 = 1 > 0$$

*Lage* hat einen positiven Wert. Das bedeutet, dass sich der Punkt rechts von der Geraden befindet.

Nun müssen wir noch herausfinden, ob der Punkt auf der richtigen Seite der Geraden liegt. Hierzu verwenden wir seine Klassenzuordnung. Der Punkt  $(2, 1)$  gehört zur Klasse  $-1$ . Das bedeutet, er sollte links der Geraden liegen.

Das heißt, alle Punkte, deren Produkt von *Klasse*  $\cdot$  *Lage* größer als  $0$  ist, liegen auf der richtigen Seite. Diese Punkte werden dann nicht weiter berücksichtigt.

Der Punkt  $P_1$  im Ausschnitt von [Bild 6.6](#) liegt auf der falschen Seite. Wir könnten jetzt die Gerade auf einmal um den ganzen Abstand zum Punkt verschieben, damit der Punkt auf der richtigen Seite liegt. Hierzu müssten wir  $a$  um die  $x$ -Koordinate des Punkts verkleinern und  $b$  um die  $y$ -Koordinate. Aber wir bewegen die Gerade nur kleines Stück in Richtung Ziel, indem wir die Änderungen von  $a$  und  $b$  mit einem Bruchteil, der Lernrate  $\lambda$ , multiplizieren.  $c$  wird ebenso um diese Lernrate verkleinert. Dieses Prinzip wird als Hebb'sche Lernregel bezeichnet ([Heb50](#)).

Die Lernrate wird in der Regel auf einen Wert zwischen  $0.001$  und  $0.1$  festgelegt. Hat man eine zu kleine Lernrate gewählt, kann es lange dauern, bis eine Lösung gefunden wurde. Ist die Lernrate zu groß, „schießt man über das Ziel hinaus“. Wir wählen in unserem Beispiel für eine bessere Darstellung eine Lernrate von  $\lambda = 0.2$  (was einem sehr, sehr großen Wert entspricht).

Mit dieser Lernrate und der Lage des Punktes  $P_1$  erhalten wir dann die neuen Werte

$$a_{neu} = a - \lambda \cdot P_{x1} = 1 - 0.2 \cdot 2 = 0.6$$

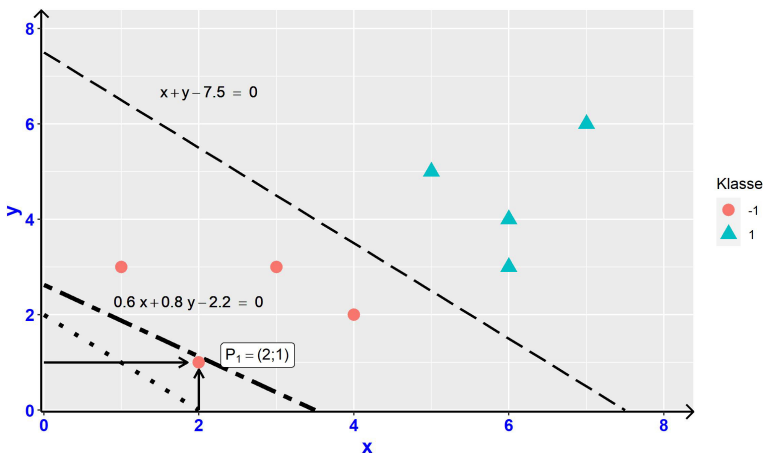
$$b_{neu} = b - \lambda \cdot P_{y1} = 1 - 0.2 \cdot 1 = 0.8$$

$$c_{neu} = c - \lambda = -2 - 0.2 = -2.2$$

Die neue Geradengleichung lautet also:  $0.6 \cdot x + 0.8 \cdot y - 2.2 = 0$ .

Sie liegt nun rechts von  $P_1 = (2; 1)$ , denn  $0.6 \cdot 2 + 0.8 \cdot 1 - 2.2 = 1.2 + 0.8 - 2.2 = -0.2 < 0$ .

Der Punkt  $P_1$  wird bei der Optimierung der Lage nicht weiter berücksichtigt, solange er auf der richtigen Seite liegt.

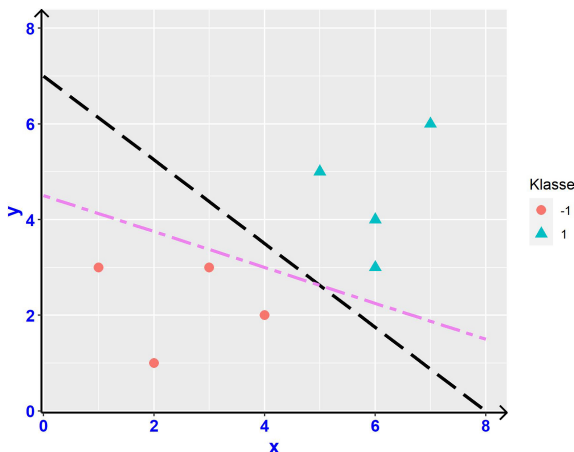


**Bild 6.6** Der Perceptron-Algorithmus

In weiteren Iterationsschritten werden zufällig andere Punkte ausgewählt. Liegen sie auf der falschen Seite, wird die Gerade auf die gleiche Weise weiter in Richtung einer Lösung manövriert, bis wir die beiden Bereiche getrennt haben (gestrichelte Linie in Bild 6.6).

## 6.3 Support Vector Machines

Mithilfe des Perceptron-Algorithmus konnten wir durch ein iteratives Verfahren zwei Bereiche voneinander trennen. Sobald aber beim Perceptron-Algorithmus eine Trennung erfolgreich war, endet der Algorithmus. Bild 6.7 zeigt schematisch die Ergebnisse zweier Durchläufe, welche bei verschiedenen Ausgangswerten gestartet wurden. Offenbar wird durch die gestrichelte Linie eine bessere Trennung als durch die strichpunktierte Linie erzielt. Aber der Perceptron-Algorithmus war schon zufrieden, wenn kein Punkt mehr auf der falschen Seite erkannt wird.



**Bild 6.7** Zwei verschiedene Trennlinien mit dem Perceptron-Algorithmus

Support Vector Machines ermöglichen es, bessere Trennlinien zu erzeugen [HCL03].

Man geht aus von der Gleichung einer Startgeraden:

$$a \cdot x + b \cdot y + c = 0, \quad \text{z. B.} \quad x + y - c = 0$$

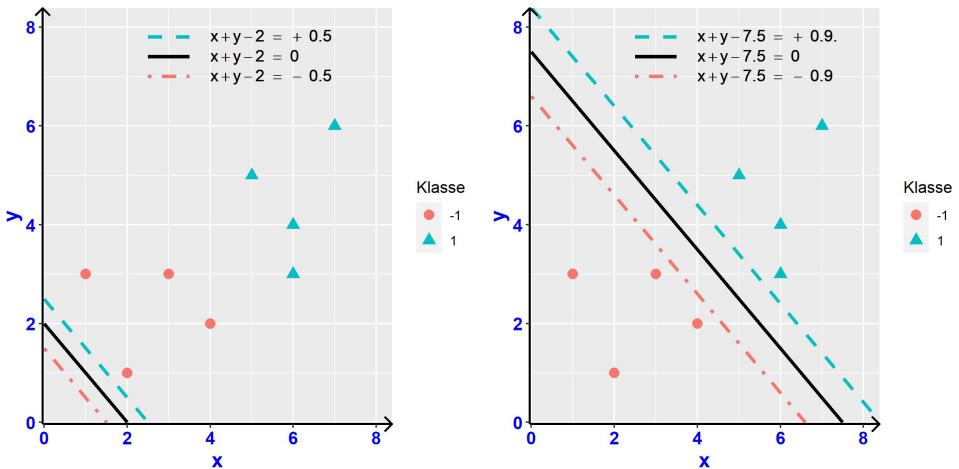
Von dieser Gleichung erzeugt man zwei Parallelen, indem man einen kleineren Wert  $e$  auf der rechten Seite einmal addiert und einmal subtrahiert.

$$a \cdot x + b \cdot y + c = -e, \quad \text{z. B.} \quad x + y - c = -0.5$$

$$a \cdot x + b \cdot y + c = e, \quad \text{z. B.} \quad x + y - c = 0.5$$

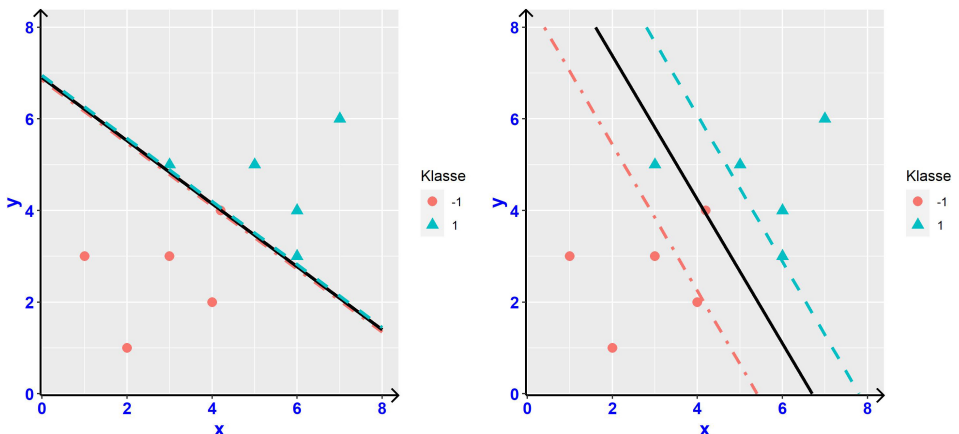
Nun wird der Perceptron-Algorithmus angewandt, sodass zum einen alle runden Punkte links von der strichpunktierten Linie und die Dreiecke alle rechts von der gestrichelten Linie zu liegen kommen. Gleichzeitig wird der Abstand der beiden Parallelen bei jeder Iteration um einen kleinen Betrag vergrößert, indem man die linke Seite der Gleichungen

mit einem Faktor um einen Wert nahe und kleiner als 1 (die Ausdehnungsrate) multipliziert. Zum Schluss hat man eine Gerade, die die Punktepaare durch Parallelen mit größtem möglichem Abstand zueinander trennt (Bild 6.8).



**Bild 6.8** Trennung durch den Support-Vector-Algorithmus

Je nach Datenlage ist die korrekte Trennung der Punkte nicht das Optimum. Betrachten wir den Fall, dass der Bereich zwischen den Punkten sehr gering geworden ist (Bild 6.9). Ist nun das linke oder das rechte Ergebnis besser? Wir können dies bei den vorhandenen Informationen nicht entscheiden. Man kann jedoch einen Wert für das Verhältnis zwischen dem mittleren Abstand der falsch klassifizierten Punkte und dem Abstand zwischen den beiden Parallelen als Ziel vorgeben.



**Bild 6.9** Welche Trennlinien sind besser?

### 6.3.1 Der Iris-Datensatz

Wir wollen mithilfe der Support-Vektor-Maschine versuchen, Trennebenen zwischen die verschiedenen Typen der Schwertlilien-Arten des Iris-Datensatzes zu ziehen. Dass dies nicht einfach werden dürfte, zeigt beispielhaft der Plot, bei dem für die Gruppen Ellipsen zur Abgrenzung eingezeichnet wurden.



**Bild 6.10** Der Iris-Datensatz mit dem ellipse-Geom

Diese Ellipsen werden über das Geom `stat_ellipse` erstellt. Die Bestimmung der Richtungen haben wir bei der Hauptkomponentenanalyse gesehen, die Länge der Halbachsen wird über die 5 %- und 95 %-Quantile für die `Petal.width` und `Petal.length` bestimmt:

```
> ggplot(iris, aes(x = Petal.Width,
+                 y = Petal.Length,
+                 color = Species, shape = Species)) +
+   geom_point(size = 2) +
+   geom_jitter() +
+   theme_gray() +
+   stat_ellipse(geom = "polygon", aes(fill = Species),
+               type = "norm",
+               alpha = 0.2,
+               show.legend = FALSE,
+               level = 0.90)
```

Das Package `e1071` enthält verschiedene Funktionen, darunter auch die Unterstützung von Support Vector Machines. Diese wollen wir für diese Klassifikation einsetzen.

Während die bisherigen Verfahren von einem statistischen Modell (z. B. einer Normalverteilung) ausgingen, setzen Support Vector Machines keines dieser Modelle voraus. Deswegen kann man die Güte auch nicht über die sonst üblichen Kennzahlen abschätzen. Stattdessen werden die Daten in Trainings- und Testdaten unterteilt. Wir verwenden dafür pseudo-zufällige Zahlen (mittels `set.seed()`) in einem Verhältnis von 7/3. Hierbei achten wir darauf, dass die Anteile der verschiedenen Iris-Typen ausgewogen sind:

```
> set.seed(9)
> train <- sample(nrow(iris), 0.7 * nrow(iris))
```

```

> iris_train <- iris[train, ]
> iris_validate <- iris[-train, ]
> table(iris_train$Species)
setosa versicolor virginica
   37      36      32
> table(iris_validate$Species)
setosa versicolor virginica
   13      14      18

```

Über `svm()` rufen wir die Funktion auf und lassen uns mittels `predict()` die Werte für die Testdaten anzeigen:

```

> fit_svm <- svm(Species ~ ., data = iris_train)
> fit_svm
Call:
svm(formula = Species ~ ., data = iris_train)
Parameters:
SVM-Type: C-classification
SVM-Kernel: radial
cost: 1
Number of Support Vectors: 45

> svm_pred <- predict(fit_svm, na.omit(iris_validate))
> svm_perf <- table(na.omit(iris_validate)$Species,
+                  svm_pred, dnn = c("Actual", "Predicted"))
> svm_perf

```

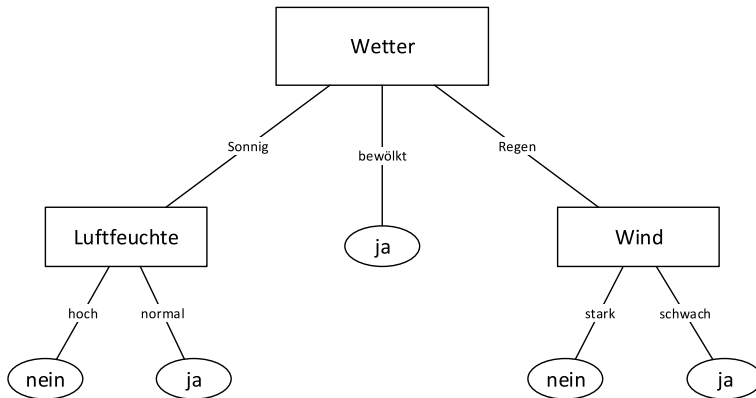
Actual	Predicted		
	setosa	versicolor	virginica
setosa	13	0	0
versicolor	0	13	1
virginica	0	1	17

Bei der Vorhersage für die Testdaten sind lediglich zwei Fehlklassifikationen zu verzeichnen, was einer Fehlklassifikationsrate von 4.4% entspricht.

## ■ 6.4 Entscheidungsbaumverfahren

### 6.4.1 Entscheidungsbäume

Entscheidungsbäume (Decision Trees) sind geordnete und gerichtete Bäume. Darin werden Entscheidungsregeln abgelegt. Sie bestehen aus einem Wurzelknoten (oben) und beliebig vielen inneren Knoten (Rechtecke in [Bild 6.11](#)). An jedem dieser Knoten kommt man durch eine Entscheidung über eine Kante zu einem Folgeknoten. Ist an einem Knoten keine Entscheidung mehr zu treffen, befindet man sich an einem Blatt (Oval in [Bild 6.11](#)), welches das Ergebnis beinhaltet.



**Bild 6.11** Entscheidungsbaum

Will man für einen Datensatz einen Entscheidungsbaum aufbauen, werden für die einzelnen Prädiktorvariablen jeweils getrennte Partitionen erstellt. Der Einfachheit halber teilen wir hier jeweils in zwei Bereiche auf (binärer Split). Diese Stelle kann man beispielsweise finden, indem man vom Mittelwert der Ergebnisvariablen ausgeht und alle Prädiktorvariablen auswählt, deren Ergebnisse in eine erste Hälfte fallen. Derjenige Prädiktor wird für eine Aufteilung herangezogen, bei dem der Fehler nach einer erfolgten Aufteilung minimal wird. So wird rekursiv der Baum aufgebaut. Da dieses Aufspalten wegen des exponentiellen Wachstums sehr schnell zu sehr großen Bäumen führt, schneidet man jedoch die Stellen zurück (Cost complexity pruning).

Entscheidungsbäume können auch für numerische Prädiktorvariablen erzeugt werden.

Die Vorteile von Entscheidungsbäumen sind [JWHT17]:

- Bäume können sehr einfach erklärt werden (einfacher als die lineare Regression).
- Oft hat man den Eindruck, dass Entscheidungsbäume die menschliche Entscheidungsfindung besser widerspiegeln als andere Regressions- und Klassifikationsansätze.
- Bäume können grafisch dargestellt werden und sind auch von einem Laien leicht zu interpretieren, insbesondere wenn sie klein sind.
- Bäume können problemlos qualitative Prädiktoren verarbeiten, ohne dass man Dummy-Variablen erstellen muss.

Diesen Vorteilen stehen die Nachteile gegenüber:

- Bäume haben im Allgemeinen nicht die gleiche Vorhersagegenauigkeit wie andere Regressions- und Klassifikationsansätze.
- Bei großen Strukturen besteht die Gefahr des „Overfitting“.
- Bäume sind oft nicht robust, es entstehen zu viele fragile Regeln: Eine kleine Änderung der Daten kann eine große Änderung des endgültigen geschätzten Baums bewirken.

Die Vorhersagegenauigkeit kann durch Methoden wie Random Forests oder Boosting ([Unterabschnitt 6.4.4](#)) verbessert werden.

## 6.4.2 Der Iris-Datensatz

Im Package `party` können Entscheidungsbäume zur Klassifikation aufgebaut und ausgewertet werden. Wir sehen uns dies wieder am Iris-Datensatz an (mit den gleichen Trainings- und Testdaten wie bei den SVM):

```
> iris_ctree = ctree(Species ~ ., data = iris_train)
> iris_ctree

Model formula:
Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width

Fitted party:
[1] root
| [2] Petal.Length <= 1.9: setosa (n = 37, err = 0.0%)
| [3] Petal.Length > 1.9
| | [4] Petal.Width <= 1.6
| | | [5] Petal.Length <= 4.8: versicolor (n = 32, err = 0.0%)
| | | [6] Petal.Length > 4.8: virginica (n = 7, err = 42.9%)
| | [7] Petal.Width > 1.6: virginica (n = 29, err = 3.4%)

Number of inner nodes: 3
Number of terminal nodes: 4
```

Dieses Verfahren weist nur eine Fehlklassifikation auf:

```
> ctree_pred <- predict(iris_ctree, iris_validate, type = "response")
> ctree_perf <- table(iris_validate$Species, ctree_pred,
+                    dnn = c("Actual", "Predicted"))
> ctree_perf
```

Actual	Predicted		
	setosa	versicolor	virginica
setosa	13	0	0
versicolor	0	13	1
virginica	0	0	18

Mithilfe von `autoplot` erhält man eine Übersicht über den erzeugten Baum ([Bild 6.12](#)).

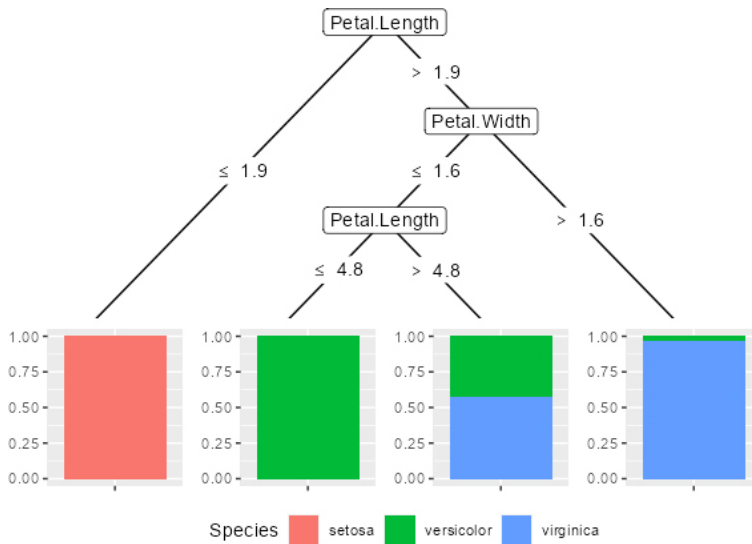
```
> autoplot(iris_ctree, main = "Entscheidungsbaum", type = "simple")
```

## 6.4.3 Bagging

Ein Hauptnachteil von Entscheidungsbäumen ist die große Varianz. Man kann die Varianz von unabhängigen Beobachtungen jedoch dadurch reduzieren, dass man mehrere Beobachtungen macht, denn die Varianz des Mittelwerts von  $n$  Beobachtungen mit einer Varianz  $\sigma^2$  beträgt  $\sigma^2/n$ . Man kann also die Varianz dadurch verkleinern, dass man Mittelwerte von Teilmengen einer Beobachtung bildet.

Man nimmt dazu zufällig ausgewählte Untermengen der Trainingsdaten und baut Entscheidungsbäume auf, die unabhängig voneinander sind. Bei der Vorhersage wird dann der Mittelwert der einzelnen Ergebnisse gebildet. Dies nennt man Bagging.





**Bild 6.12** Entscheidungsbaum

## 6.4.4 Random Forests

Entsprechend eines Vorschlags von Breiman [Bre01] werden bei Random Forests – ähnlich wie beim Bagging – Teilmengen zum Aufbau von Bäumen verwendet [LW01].

Während bei Standardentscheidungsbäumen an jedem Knoten die beste Aufteilung unter allen Variablen vorgenommen wird, wird bei Random Forests die beste Aufteilung einer zufällig ausgewählten Teilmenge aller Prädiktorvariablen vorgenommen. Sind jetzt für eine Eingabe alle Bäume ausgewertet, wird das Ergebnis verwendet, welches bei den meisten Bäumen ermittelt wurde.

In <https://www.r-bloggers.com/2017/01/random-forest-classification-of-mushrooms/> wurde diese Methode zur Erkennung essbarer/giftiger Pilze angewandt. Die Korrekt-Klassifikationsrate war erstaunlich gut. Der Autor rät jedoch von der Anwendung dieses Modells ab, weil er keine Aussage über die Zuverlässigkeit der Daten machen konnte.

Wir werden diese Methode an einem anderen Beispiel in [Abschnitt 9.4](#) in Aktion sehen.

## 6.4.5 Boosted Regression Trees

Boosted Regression Trees kombinieren Bagging und Boosting-Methoden [JWHT17]: Wie bei Bagging werden mehrere Entscheidungsbäume aufgebaut als Wald (forest). Dies erfolgt jedoch im Gegensatz zu Random Forest sequentiell. Hierbei werden die Daten und die Ergebnisse vorhergehender Bäume modifiziert: Man gewichtet die Daten, welche große Fehler verursachen, und verwendet die Gewichte, um bei der Bildung eines nachfolgenden Datensatzes diese Daten mit größerer Wahrscheinlichkeit auszuwählen. Dadurch wird versucht, die Genauigkeit der Bäume sequenziell zu vergrößern.

## ■ 6.5 Naive Bayes-Klassifikatoren

Bayes-Klassifikatoren bestimmen aufgrund bedingter Wahrscheinlichkeiten ein Ergebnis. Man kann sie unterteilen in multinomiale und Gauß'sche Klassifikatoren.

### 6.5.1 Multinomiale naive Bayes-Klassifikatoren

Beim Bayes-Klassifikator nutzt man die Eigenschaften bedingter Wahrscheinlichkeiten aus. Die Eigenschaften bedingter Wahrscheinlichkeiten kann man sich sowohl bei kategorialen als auch bei kontinuierlichen Variablen zunutze machen. Bei kategorialen Variablen bedient man sich des Urnenmodells, woraus sich die Binomialverteilung (bzw. bei mehreren Ergebnissen eine multinomiale Verteilung) ergibt. Daher hat dieser Klassifikator seinen Namen.

Zur Behandlung der bedingten Wahrscheinlichkeiten betrachten wir das Vorkommen bestimmter Worte in einer Nachricht. Anhand der Auftrittswahrscheinlichkeiten der Worte soll abgeschätzt werden, ob es sich bei einer Nachricht um eine Spam-Nachricht handelt.

Angenommen, wir haben im Lauf der Zeit 50 spam-Nachrichten ( $S$ ) und 150 Nicht-spam („ham“)-Nachrichten ( $H$ ) erhalten. In den Nachrichten wurde das Auftreten der beiden Worte „gewonnen“ ( $g$ ) und „Konto“ ( $k$ ) (jeweils in spam und in ham) gezählt (Tabelle 6.1). Dann können wir daraus eine a-priori-Wahrscheinlichkeit von  $p(S) = \frac{50}{50+150} = 0,25$  für spam und  $p(H) = 1 - 0,25 = 0,75$  für ham abschätzen.

Wir wollen das Auftreten von „gewonnen“ ( $g$ ) und „Konto“ ( $k$ ) heranziehen, um zu entscheiden, ob es sich um eine ham- oder spam-Nachricht handelt.

**Tabelle 6.1** Auftreten von Worten in spam und ham

	spam		ham	
Auftreten	n	$p( S)$	n	$p( H)$
insgesamt	50		150	
$g$	40		30	
$k$	25		15	

Die Wahrscheinlichkeit für das Auftreten von  $g$  in einer Nachricht beträgt

$$p(g) = \frac{40 + 30}{50 + 150} = \frac{7}{20}.$$

Jetzt betrachten wir die Nachrichten unter der Bedingung, dass die Nachricht  $g$  zutrifft: Wie groß ist dann die Wahrscheinlichkeit, dass es sich um spam handelt? Dies wird ausgedrückt als  $p(S|g)$  (die bedingte Wahrscheinlichkeit für  $S$  unter der Bedingung  $g$ ). Wir können dies einfach aus den Zahlen in der Zeile  $g$  ausrechnen:

$$p(S|g) = \frac{\text{Zahl der günstigen Fälle}}{\text{Zahl der möglichen Fälle}} = \frac{40}{40 + 30} = \frac{4}{7}$$

Entsprechend ist

$$P(H|g) = \frac{30}{40 + 30} = \frac{3}{7}$$

Vertauschen wir jetzt Bedingung ( $g$ ) und Variable ( $S$ ), dann ist die Wahrscheinlichkeit für  $g$  unter der Bedingung  $S$

$$p(g|S) = \frac{40}{50} = 0,8$$

Bei ham beträgt sie

$$p(g|H) = \frac{30}{150} = 0,2$$

Bei diesem Zahlenbeispiel haben wir bei der Bestimmung der Wahrscheinlichkeiten die Bedingungen ausgetauscht. Dies ist die Aussage des Satzes von Bayes:

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}.$$

Damit können wir analog  $p(g|S)$  auch bestimmen über

$$p(g|S) = \frac{p(S|g) \cdot p(g)}{p(S)} = \frac{\frac{4}{7} \cdot \frac{7}{20}}{0,25} = 0,8.$$

Entsprechend kann man die Wahrscheinlichkeiten für  $k$  abschätzen:

$$p(k|S) = \frac{25}{50} = 0,5 \quad \text{und} \quad p(k|H) = \frac{15}{150} = 0,1$$

Nun nehmen wir an, dass die Texte über keinerlei Grammatik (keine Reihenfolge etc.) verfügen und die Worte statistisch unabhängig voneinander auftreten. Wir sprechen dann von einem naiven Klassifikator.

Wie wahrscheinlich ist es, dass  $g$  und  $h$  in den Nachrichten zusammen vorkommen? Hier gilt

$$p(g \cap k|S) = p(g|S) \cdot p(k|S) \quad \text{und} \quad p(g \cap k|H) = p(g|H) \cdot p(k|H)$$

für den Fall unabhängiger Ereignisse (naives Modell).

Die erwartete Anzahl der spam-Nachrichten, die sowohl  $g$  als auch  $k$  aufweisen, können wir damit einfach über Multiplikation von  $p(g \cap k|S)$  mit der Zahl  $n$  der spam-Nachrichten erhalten (Tabelle 6.2).

**Tabelle 6.2** Wahrscheinlichkeiten von Worten in spam und ham

	spam		ham	
Auftreten	n	p(  S)	n	p(  H)
insgesamt	50		150	
$g$	40	0,8	30	0,2
$k$	25	0,5	15	0,1
$g \cap k$	20	0,4	1	0,02

Wenn wir jetzt entscheiden wollen, ob eine Nachricht  $H$  oder  $S$  ist, müssen wir  $p(S|g \cap k) = \frac{20}{20+1} \approx 0,95$  in das Verhältnis setzen zu  $p(H|g \cap k) = \frac{1}{20+1} \approx 0,05$ .

Aufgrund dieses Verhältnisses können wir von einer spam-Nachricht ausgehen. In Formeln und unter Verwendung des Satzes von Bayes können wir angeben:

$$p(S|g \cap k) = \frac{p(g \cap k|S) \cdot p(S)}{p(g \cap k|S) \cdot p(S) + p(g \cap k|H) \cdot p(H)},$$

ausmultipliziert:

$$p(S|g \cap k) = \frac{p(g|S) \cdot p(k|S) \cdot p(S)}{p(g|S) \cdot p(l|S) \cdot p(S) + p(g|H) \cdot p(k|H) \cdot p(H)}.$$

Diese Formel kann man auf weitere Variablen erweitern. Kommen also bestimmte Worte in einer Nachricht vor (bzw. kommen nicht vor), kann man über den Satz von Bayes die bedingten Wahrscheinlichkeiten für Ham oder Spam ausrechnen und anhand des Verhältnisses über diese Frage entscheiden.

Es kann nun sein, dass die bedingte Wahrscheinlichkeit für eine der Variablen Null ergeben hat. Dann würden wir in beiden Fällen das Produkt 0 erhalten. Um dem vorzubeugen, wird bei allen Variablen ein sehr kleiner Wert hinzuaddiert. Dies verursacht zwar Fehler in der Berechnung, ermöglicht aber verwertbare Ergebnisse.

## 6.5.2 Das spam-Beispiel

Eine Studie hat bei der Erkennung von spam bei SMS-Nachrichten mithilfe des Bayes-Klassifikators viel Beachtung gefunden [HHSD98]. Hier wurden 5574 SMS-Nachrichten klassifiziert als ham bzw. als spam. Der Datensatz kann von <https://www.kaggle.com/hdza1991/sms-spam> abgerufen werden.

```
> sms_raw <- read.csv("sms_spam.csv",
+                     stringsAsFactors = FALSE)
> sms_raw$type <- factor(sms_raw$type)
> table(sms_raw$type)
  ham spam
4827  747
```

Der erste Schritt besteht darin, die Texte in eine Form zu bringen, dass sie für den Klassifikator verwendbar sind. Hierzu laden wir den gesamten Text zur besseren Verwaltung in den Hauptspeicher:

```
> sms_corpus <- VCorpus(VectorSource(sms_raw$text))
```

An diesem Datensatz werden nun Bereinigungen vorgenommen wie

- Leerzeichen entfernen,
- Zahlen entfernen,
- Stoppwörter (z. B. wie, auch, und ...) entfernen,
- alle Satzzeichen entfernen,
- Bildung von Wortstämmen mit dem Stemmer-Algorithmus [CKLN10].

Danach wird eine Matrix aufgebaut, in der die Dokumentennummern als Zeilen und die einzelnen (bereinigten) Wörter als Spalten abgelegt werden. Die Zahlen in der Matrix geben an, wie oft der Begriff in jedem Dokument vorkommt. All diese Schritte werden von der Funktion `DocumentTermMatrix` vorgenommen:

```
> sms_dtm <- DocumentTermMatrix(sms_corpus, control = list(
+                               tolower = TRUE,
+                               removeNumber = TRUE,
+                               stopwords = TRUE,
+                               removePunctuation = TRUE,
+                               stemming = TRUE,
+                               stripWhitespace=TRUE))
```

Wenn Sie mehr Details über die Textvorbereitung erfahren möchten, bietet Ihnen neben der package-Dokumentation zu `tm` auch <https://confessionsofadatascientist.com/text-analysis-with-r-repro-report.html> sowie [https://github.com/solariz/german\\_stopwords](https://github.com/solariz/german_stopwords) weitergehende Informationen.

Nach der Verarbeitung der Texte wird der Datensatz unterteilt in Trainings- und Testdaten sowie die zugehörigen Labels. Über `prop.table()` prüfen wir nach, ob das Verhältnis von `ham` und `spam` zwischen den Trainings- und Testdaten ausgewogen ist:

```
> sms_dtm_train <- sms_dtm[1:4169, ] # Training dataset
> sms_dtm_test <- sms_dtm[4170:5559, ] #Test dataset
> sms_train_labels <- sms_raw[1:4169, ]$type #Training Label
> sms_test_labels <- sms_raw[4170:5559, ]$type # Test Label
> prop.table(table(sms_train_labels))
sms_train_labels
ham      spam
0.8647158 0.1352842
> prop.table(table(sms_test_labels))
sms_test_labels
ham      spam
0.8697842 0.1302158
```

Unsere Matrix hat 7935 Spalten, von denen etliche nur wenige oder gar keine Einträge aufweisen. Um den Rechenaufwand zu verkleinern, entfernen wir mittels `findFreqTerms()` alle Spalten, in denen ein Wort nicht mindestens fünfmal vorkommt, und reduzieren so die Matrix auf 1234 Spalten:

```
> sms_freq_words <- findFreqTerms(sms_dtm_train, 5)
> sms_dtm_freq_train <- sms_dtm_train[, sms_freq_words]
> sms_dtm_freq_test <- sms_dtm_test[, sms_freq_words]

> inspect(sms_dtm_freq_train)
<<DocumentTermMatrix (documents: 4169, terms: 1234)>>
Non-/sparse entries: 25986/5118560
Sparsity             : 99%
Maximal term length: 15
Weighting            : term frequency (tf)
Sample              :
```

```

Terms
Docs  call can come free get just know ltgt now will
1086  0  0  1  0  1  0  0  0  0  9
1580  0  0  0  0  0  0  0  18  0  0
:      :      :      :      :      :      :

```

Bei der Betrachtung des Bayes-Theorems hatten wir für eine Variable nur eine Ja/Nein-Entscheidung für das Auftreten in einer Nachricht betrachtet, nicht aber die Häufigkeit des Auftretens. Deswegen wird jetzt mithilfe einer selbst definierten Funktion `convert_counts()` Felder mit Zahlen größer als Null auf „Yes“ gesetzt, der Rest auf „No“:

```

> convert_counts <- function(x) {
+   x <- ifelse(x > 0, "Yes", "No")
+ }
> sms_train <- apply(sms_dtm_freq_train, MARGIN = 2, convert_counts)
> sms_test <- apply(sms_dtm_freq_test, MARGIN = 2, convert_counts)

```

Damit sind wir vorbereitet für den Klassifikator `naiveBayes` und können die Ergebnisse über `CrossTable` betrachten:

```

> sms_classifier <- naiveBayes(sms_train, sms_train_labels)
> sms_test_pred <- predict(sms_classifier, sms_test)
> head(sms_test_pred)
[1] spam ham ham ham ham
Levels: ham spam
> CrossTable(sms_test_pred, sms_test_labels,
+            prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
+            dnn = c('predicted', 'actual'))
Cell Contents
|-----|
|                N |
|                N / Col Total |
|-----|
Total Observations in Table: 1390

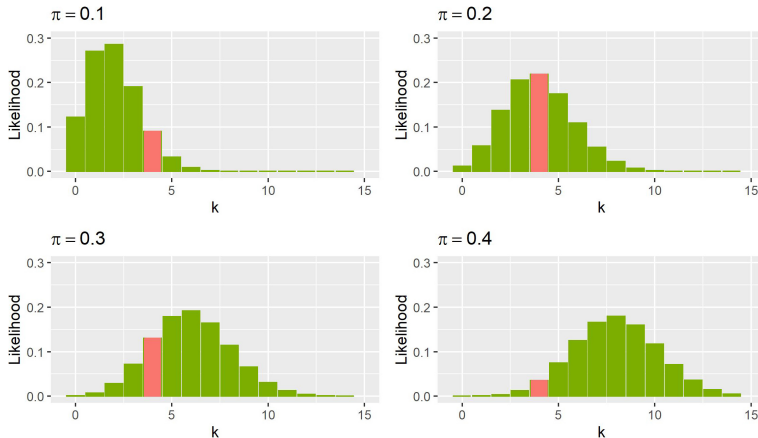
```

predicted	actual		Row Total
	ham	spam	
ham	1204 0.996	28 0.155	1232
spam	5 0.004	153 0.845	158
Column Total	1209 0.870	181 0.130	1390

Wir haben mit dieser Methode eine respektable Korrekt-Klassifikationsrate von  $\frac{25+5}{1390} \approx 98\%$  erreicht.

Im Zusammenhang mit der Verarbeitung von Texten ist auch das Package `wordcloud` geschrieben worden, mit dem Worte nach Häufigkeit in einer Wolke angeordnet werden ([Bild 6.13](#)).





**Bild 6.14** Wahrscheinlichkeiten/Likelihoods für verschiedene  $\pi$

oder allgemeiner

$$L_k(\pi) = \binom{n}{k} \cdot \pi^k \cdot (1 - \pi)^{n-k}.$$

Dies ist nichts anderes als die Binomialverteilung. „Normalerweise“ hält man  $\pi$  und  $n$  fest und variiert  $k$ , woraus sich die Wahrscheinlichkeit ergibt, ein bestimmtes  $k$  zu erhalten.

Man kann aber auch  $k$  festhalten und  $\pi$  variieren. In diesem Fall wird die Wahrscheinlichkeit Likelihood genannt. Der Name ergibt sich also aus der Frage, welches die abhängige Variable ist.

Im Fall kontinuierlicher Variablen sieht es anders aus. Betrachten wir noch einmal das Beispiel der Normalverteilung ([Unterabschnitt A.2.1](#)): Dort ist die Wahrscheinlichkeit, dass sich eine normalverteilte Variable innerhalb des Bereichs  $0 \leq z \leq 0.674$  befindet, die Fläche unter der Häufigkeitsdichte ([Bild 6.15](#) links). Die Likelihood für  $z = 0.674$  ist bei kontinuierlichen Verteilungen die Wahrscheinlichkeitsdichte (der Y-Wert des Punkts bei  $z = 0.674$  in [Bild 6.15](#) rechts).

## 6.5.4 Gauß'sche naive Bayes-Klassifizierer

Der Gauß'sche naive Bayes-Klassifizierer wendet das Bayes-Theorem auf Variablen an, von denen angenommen wird, dass sie einer Gauß-Verteilung folgen.

Hierzu ein einfaches Beispiel:

Wir betrachten zwei Klassen von Fahrzeugen, Coupés und SUV, von denen wir den Mittelwert und die Standardabweichung der Massen  $m$  und Höchstgeschwindigkeiten  $v$  kennen ([Tabelle 6.3](#)). Es gibt dreimal so viele Coupés wie SUV. Nun soll für ein unbekanntes Fahrzeug der Masse 1.2 to und der Höchstgeschwindigkeit 210 km/h der Typ bestimmt werden. Hierzu schätzen wir aus den Anteilen der Typen die a-priori-Wahrscheinlichkeiten  $p = \frac{3}{4}$  bzw.  $p = \frac{1}{4}$ . Aus der Wahrscheinlichkeitsdichtefunktion der Gauß-Verteilung bestimmen



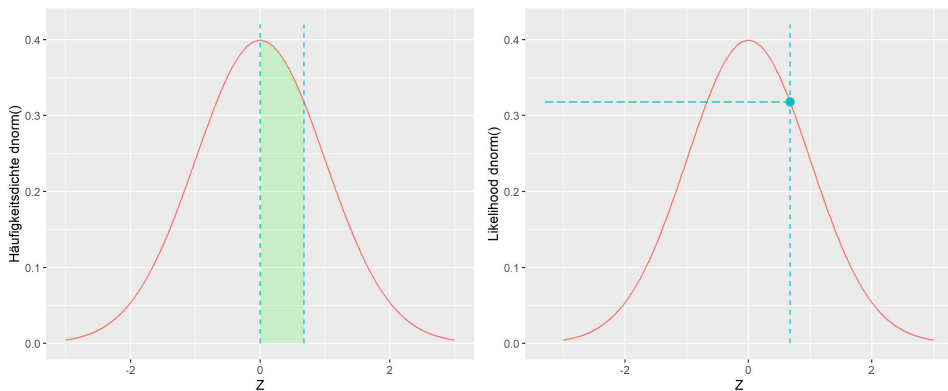


Bild 6.15 Wahrscheinlichkeit und Likelihood

Tabelle 6.3 Beschreibende Größen von Coupés und SUV

Typ	Masse $m$		Höchstgeschwindigkeit $v$	
	$\mu$ [to]	$\sigma$ [to]	$\mu$ [km/h]	$\sigma$ [km/h]
Coupé	1	0.2	220	10
SUV	2	0.4	180	20

wir zunächst mit `dnorm()` die Likelihoods für die Massen und Höchstgeschwindigkeiten für beide Fahrzeugtypen (Bild 6.16).

```
> l_mass_coupe <- dnorm(1.2, mean = 1, sd = 0.2)
> l_mass_coupe
[1] 1.209854
:
:
> l_speed_suv <- dnorm(210, mean = 180, sd = 20)
> l_speed_suv
[1] 0.00647588
```

Wir nehmen nun an, dass die Likelihoods unabhängig voneinander sind. Dann könnten wir einfach das Produkt der Likelihoods für beide Fahrzeugtypen berechnen. Da aber oft sehr kleine Werte vorkommen und Computer sehr kleine Zahlen als Nullen behandeln, kann dies zu Fehlern führen. Um dieses Problem zu umgehen, bildet man den Logarithmus der Likelihoods (Log-Likelihoods). Damit kann man anstelle der Multiplikation durch Nutzung der Logarithmenregeln einfach die Log-Likelihoods addieren. Da die Likelihoods Werte zwischen 0 und 1 annehmen, liegen Log-Likelihoods zwischen  $-\infty$  und 0:

```
> log(3 / 4) + log(l_mass_coupe) + log(l_speed_coupe)
[1] -3.818706
> log(1 / 4) + log(l_mass_suv) + log(l_speed_suv)
[1] -8.428613
```

Man stellt fest, dass die Log-Likelihood in diesem Falle für das Coupé größer ist als die des SUVs. Wir schließen deshalb auf ein Coupé.

# Stichwortverzeichnis

## A

a-priori-Wahrscheinlichkeit 150  
Abdeckung 139  
Abhängige Variable 96  
Abhängigkeitsdiagramm, partielles 260  
Abstraktions-Level 297  
Accuracy 139  
ACF-Werte 197  
Achsen 274  
Achsenbeschriftungen 52  
Adadelta 227  
Adagrad 227  
Adam 228  
ADAMax 228  
ADF-Test 190  
aes() 51  
aggregate() 69  
aktuelles Modell 6  
 $\alpha$ -Fehler 76  
ANOVA 347  
apply() 38  
AR(p)-Prozess 196  
AR-Modell 196  
Arbeitsverzeichnis festlegen 18  
Area Under Curve 162  
ARMA-Modell 200  
Assert 66  
ästhetisches Mapping 51  
AUC 162  
Auftauern 305  
Augmentation 310  
Ausreißer 75  
Auswahl von Zeilen und Spalten 33  
Autodiff 218  
Autokorrelationsfunktion 197  
– partielle 198  
AutoML 254

autoplot 69  
Average Pooling 296

## B

Backpropagation 218  
Bagging 148  
Bartlett-Test 179  
Batch 214  
Batch-Optimierung 214  
Bayes, Satz von 151  
Bayes-Klassifikator 150  
Bayes-Klassifikator, multinomialer naiver 150  
Bayes-Klassifizierer, Gauß'scher naiver 156  
Bedingte Ausführung 37  
Bedingte Wahrscheinlichkeit 150  
Bereichsgrenzenüberwachung 66  
Bestimmtheitsmaß 106  
 $\beta$ -Fehler 76  
Bias 207  
Bilder kopieren 307  
Bildklassifikation 305  
Bildtransformationen 309  
Binärer Split 147  
Biplot 185  
Blöcke 37, 301  
Boosted Regression Trees 149  
Boston-Datensatz 116  
Box-Plots 81, 338  
Breakpoints 42

## C

Callbacks 286  
Chi-Quadrat-Anpassungstest 344  
Chi-Quadrat-Verteilung 344  
clipnorm 225  
clipvalue 225  
Clustering 5  
Clusterverfahren 169

Cochran-Statistik 86  
Code-Analyse 45  
Coercing 30  
Compilierung 278  
Confusion Matrix 138  
Conv\_2D-Layer 294  
Convolution 291  
Convolutional Networks 291  
Cooks-Distanz 104  
Cosine Proximity 213  
CUDA-Treiber 11

## D

Datei lesen 23  
Datenaufbereitung 49  
Datentypen 27  
Datum 36  
Debugging 41  
Decision Trees 146  
Deep Learning 207  
Devianz 167  
dfbeta() 99  
DFFITS-Werte 103  
Distributions 289  
Distributionsdiagramm 291  
Dixon 85  
Durchschnitt, gewichteter 193  
Durchschnitt, gleitender 191

## E

Ebene 53  
Eigenvektoren 178  
Eigenwert 178  
Einfrieren 305  
eingeschränktes Modell 6  
Elastic Nets 130  
Ellipsen 57, 145  
ELU 210  
Encoding 16  
Entscheidungsbäume 146  
Epoche 279  
Euklidische Distanz 158  
Excess 80, 334  
Expertensystem 5  
Explodierender Gradient 225  
Exponentielle Lineare Einheit 210  
Extrapolationen 6

## F

F-Verteilung 346  
F1-Maß 140  
Facette 54

Faktoranalyse 5  
Faktorisierung 31  
Faktorladung 179, 183  
Falsch-Positiv-Rate 140  
Faltung 291  
Faltungsnetze 291  
Farb-Codes 53  
Feature 95  
Feature Learning 298  
Feature Map 295  
Fehler erster Art 76  
Fehler zweiter Art 76  
Fehlerhafte Bilder 307  
Fehlklassifikation 139  
Fehlklassifikationsrate 139  
Fermifunktion 136  
Fit-Modell 166  
Formatierung 45  
Forward-Mode Autodiff 217  
Frames 32  
Freiheitsgrad 343  
Funktional 309  
Funktionen 39  
– logistische 136  
Funktionsgraph 216

## G

Gauß-Verteilung 339  
Genauigkeitsparadoxon 140  
generalisierte lineare Modelle 137  
geom\_jitter() 56  
geom\_point() 55  
geom\_text() 55  
geom\_tile() 55  
Geom 55  
gesättigtes Modell 166  
getAnywhere 41  
Gewichte auslesen 286  
Gewichte setzen 286  
ggplot2 50  
ggsave 56  
Glättung, exponentielle 192  
GLM 137  
GPU 274  
Gradient 215  
Graphs 289  
grid search 239  
group 56  
Grubbs-Test 87

## H

H2O 234

h2o-Objekt 235  
Häufigkeitspolygone 335  
Harmonisches Mittel 140  
Hauptkomponentenanalyse 173  
HDFS-Speicherort 236  
Heatmap der Variablenwichtigkeit 258  
Hebb'sche Lernregel 142, 225  
Hebelwirkung 101  
Helligkeitsumkehr 293  
Heteroskedastizität 96  
Hidden Layer 207  
Histogramme 81, 334  
Histogrammansicht 291  
hit rate 139  
Holt-Winters-Glättung 194  
Homoskedastizität 96

**I**

ILSVRC 300  
ImageNet 300  
Imputation 65  
Input Layer 207  
Interpolationspolynom, Newton'sches 114  
intersect() 29  
Intervalle 34  
Iris-Datensatz 145

**K**

k-means 169  
k next neighbours 158  
Kaiser-Meyer-Olkin-Kriterium 180  
Kalenderdaten 64  
Keras 274  
Kern 292  
Klassenzugehörigkeit 307  
Klassifikationen 5  
Klassifikator, naiver 151  
Klassifizierung 301  
KMO-Kriterium 180  
knn-Klassifikator 158  
Konfidenzschätzung 343  
Konfusionsmatrix 138  
Konfusionstabelle 138  
Kontingenztabellen 138  
Koordinatensystem 54  
Korrekt-Klassifikation 139  
Korrekt-Klassifikationsrate 139  
Korrelogramm 197  
Kovarianz 174  
Kovarianzmatrix 176  
Kreisbögen 57  
Kreuzvalidierung 115

Kurtosis 80, 334

**L**

$L_1$ -Norm 130  
 $L_2$ -Norm 130  
Löschen einer Zeile oder Spalte 34  
Label 96, 208  
Labels 57  
Ladungsdiagramm 183  
lag 197  
Landkarte 58  
LASSO-Regularisierung 130  
latex2exp 57  
Layer 207  
layer\_flatten() 298  
Lernrate 142, 225  
Levene-Test 97  
Leverage 101  
Likelihood 156  
linkssteil 80, 334  
lintr 45  
loess 196  
Log-Likelihoods 157  
Logistische Funktion 136  
logit-Funktion 136  
lowess 196  
LQS-Regression 131  
LTS-Regression 132  
Lupen 57

**M**

M-Schätzer 131  
MA-Modell 198  
Mapping, ästhetisches 51  
 $MA(q)$ -Prozess 199  
Maschinelles Lernen 5  
Matrix 31  
Max-Pooling 296  
Maxout-Funktion 210  
McFadden  $R^2$ -Bestimmtheitsmaß 163  
Measure of Sampling Adequacy 180  
Median 78, 332  
Messfehler 68  
Metrik 278  
Mini-Batch-Gradientenabstiegsverfahren 214  
Mini-Batch-Gradientenverfahren 225  
Mittelwert 77, 331  
mittlerer absoluter Fehler 95  
mittlerer quadratischer Fehler 95  
MNIST-Datensatz 266  
Modalwert 79, 333  
Modell, gesättigtes 166

Modell laden 285  
 Modell speichern 285  
 Modell-Korrelations-Diagramm 258  
 Modelle, generalisierte lineare 137, 237  
 Modus 79, 333  
 Momente höherer Ordnung 80, 334  
 Momenten-Update 226  
 Mycin 4

**N**

NA 116  
 NA-Eintrag 29  
 NA-Werte 68  
 naan 287  
 Nesterov accelerated Gradient 226  
 Nichtlinearitäten 252  
 Normalverteilung 340  
 Nulldevianz 167  
 Nullmodell 6, 164

**O**

odds 136  
 One-Hot-Codierung 299  
 ONNX® 8  
 Operationscharakteristik 161  
 Optimierer 214, 278  
 Oszillationen 291  
 out-of-sample error 115  
 Outliers 83, 84  
 Output Layer 207  
 Overfitting 115

**P**

PACF-Funktion 198  
 Packages-Registerkarte 20  
 pad() 69  
 Padding 292  
 Partitionen 147  
 Perceptron 205  
 Perceptron-Algorithmus 140  
 Perzentile 337  
 Pipe-Operator 41  
 Pivot-Punkte 141  
 plotnine 50  
 Pooling 296  
 Pooling Layers 296  
 positive predictive value 140  
 Positiver Vorhersagewert 140  
 POSIXct 37  
 Prädiktorvariable 95  
 Prävalenz 139  
 Präzision 140

predict() 111  
 PRV 139  
 Pseudo- $R^2$  166  
 Pseudo-Bestimmtheitsmaße 163

**Q**

QQ-Plot 83, 341  
 Quantile 337  
 Quartil 336

**R**

$R^2$ -Bestimmtheitsmaß 163  
 $R^2$ -Metrik 163  
 Random Forests 149  
 Range 79, 333  
 recall 139  
 Receiver Operating Characteristic 161  
 Rechtsgipfelig 80, 334  
 Rechtssteil 80, 334  
 Rectifier-Funktion 208  
 Regression 5  
 – logistische 137  
 – nichtlineare 113  
 Reguläre Ausdrücke 289, 308  
 Reguläres Modell 6  
 Regularisierung 129  
 ReLU 208  
 Representational State Transfer API 235  
 Residuenanalyse 98, 257  
 Residuendevianz 167  
 Residuenquadrate, geordnete 131  
 Residuum 94  
 REST-API 236  
 RIDGE-Regularisierung 129  
 RMSprop 227  
 ROC 161  
 RStudio 15  
 RTools 12

**S**

S-Funktion 136  
 Saison 189  
 Sample 205  
 Saturiertes Modell 6  
 Schicht 207  
 Schiefe 80, 334  
 Schleife 38  
 Schwanenhalsfunktion 136  
 Scoreplot 185  
 Scree-Plot 182  
 Selektionskriterien 34  
 Sensitivität 139

setdiff() 29  
Sigmoid-Funktion 136, 208  
Skalarprodukt 206  
Skalierung 54, 277, 340  
skewness 80, 334  
Sobel-Filter 293  
Spalierbildung 54  
Spannweite 79, 333  
Speichern 24  
Spezifizität 140  
Splines 195  
Stacked Ensemble 261  
Standardpaletten 53  
Stereogramme 57  
Stochastisches Gradientenabstiegsverfahren 214  
Stochastisches Gradientenverfahren 225  
Strafterm 129  
Streudiagramm 50, 55  
Streuung 79, 333  
Striding 293  
Struktur-entdeckende Verfahren 92  
Struktur-prüfende Verfahren 92  
Student-Verteilung 342  
Studentisierte Residuen 101  
Style Guide 45  
styler 45  
Summenhäufigkeit 335  
Super-Learning 261  
Super-Zuweisungsoperator 40  
Support Vector Machine 143

**T**

t-Residuum, gelöschttes 103  
t-Residuum, internes 101  
t-Verteilung 342  
Tangens Hyperbolicus 208  
Tensor 274  
TensorBoard 287  
Textvorbereitung 153  
Thema 54  
Themes 60  
tibble 35  
tidy data 62  
tidyverse 51  
TNR 140  
TPR 139  
Transferlernen 304

Treffergenauigkeit 139  
Trend 189  
true negative rate 140  
true positive rate 139  
type\_convert() 64

## U

Überanpassung 115, 129  
Überlauf 287  
Ubuntu Linux 12  
Uhrzeit 36  
Unabhängige Variable 95  
Uneingeschränktes Modell 6  
union() 29  
Unit 207

## V

Validierung 98  
Variablennamen 26, 43  
Variablenwichtigkeit 257  
Varianz 79, 333  
Varianzanalyse 347  
Vektoren 28  
Verdeckte Schicht 207  
Verkettung 216  
Verlustfunktion 213, 278  
Versteckte Variablen 40  
Verteilungsdichte 340  
Verteilungsfunktion 340  
Vertrauensintervall 343  
Vertrauensniveau 111  
Violindiagramme 335  
Voll verbundenes neuronales Netz 207  
Vollmodell 6, 166  
Voronoi-Diagramm 158

## W

Wölbung 80, 334  
Werte in Spaltenüberschriften 63  
Whisker 81, 338  
wordcloud 154  
WordNet® 300

## Z

Z-Transformation 340  
Zeichenketten 36  
Zeitreihen 189  
Zeitreihe 69