



Steffen Herbold

Data Science Crashkurs

Eine interaktive
und praktische Einführung

Mit Jupyter
Notebooks

dpunkt.verlag

Inhalt

Cover

Über den Autor

Titel

Impressum

Vorwort

Inhaltsübersicht

Inhaltsverzeichnis

1 Big Data und Data Science

1.1 Einführung in Big Data

1.1.1 Volumen

1.1.2 Velocity/Geschwindigkeit

1.1.3 Variety/Vielfalt

1.1.4 Innovative Informationsverarbeitungsmethoden

1.1.5 Wissen generieren, Entscheidungen treffen, Prozesse automatisieren

1.1.6 Noch mehr Vs

1.2 Einführung in Data Science

1.2.1 Was gehört zu Data Science?

1.2.2 Beispielanwendungen

1.3 Fähigkeiten von Data Scientists

2 Der Prozess von Data-Science-Projekten

2.1 Der generische Data-Science-Prozess

2.1.1 Discovery

2.1.2 Datenvorbereitung

2.1.3 Modellplanung

- 2.1.4 Modellerstellung
- 2.1.5 Kommunikation der Ergebnisse
- 2.1.6 Operationalisierung

2.2 Rollen in Data-Science-Projekten

- 2.2.1 Anwenderin
- 2.2.2 Projektsponsorin
- 2.2.3 Projektmanagerin
- 2.2.4 Dateningenieurin
- 2.2.5 Datenbankadministratorin
- 2.2.6 Data Scientist

2.3 Deliverables

- 2.3.1 Sponsorenpräsentation
- 2.3.2 Analystenpräsentation
- 2.3.3 Quelltext
- 2.3.4 Technische Spezifikation
- 2.3.5 Daten

3 Allgemeines zur Datenanalyse

- 3.1 Das No-free-Lunch-Theorem
- 3.2 Definition von maschinellem Lernen
- 3.3 Merkmale
- 3.4 Trainings- und Testdaten
- 3.5 Kategorien von Algorithmen
- 3.6 Übung

4 Erkunden der Daten

- 4.1 Texteditoren und die Kommandozeile
- 4.2 Deskriptive Statistik
 - 4.2.1 Lagemaße
 - 4.2.2 Variabilität

4.2.3 Datenbereich

4.3 Visualisierung

4.3.1 Anscombes Quartett

4.3.2 Einzelne Merkmale

4.3.3 Beziehungen zwischen Merkmalen

4.3.4 Scatterplots für hochdimensionale Daten

4.3.5 Zeitliche Trends

4.4 Übung

5 Assoziationsregeln

5.1 Der Apriori-Algorithmus

5.1.1 Support und Frequent Itemsets

5.1.2 Ableiten von Regeln

5.1.3 Confidence, Lift und Leverage

5.1.4 Exponentielles Wachstum

5.1.5 Die Apriori-Eigenschaft

5.1.6 Einschränkungen für Regeln

5.2 Bewertung von Assoziationsregeln

5.3 Übung

6 Clusteranalyse

6.1 Ähnlichkeitsmaße

6.2 Städte und Häuser

6.3 k-Means-Algorithmus

6.3.1 Der Algorithmus

6.3.2 Bestimmen von k

6.3.3 Probleme des k-Means-Algorithmus

6.4 EM-Clustering

6.4.1 Der Algorithmus

6.4.2 Bestimmen von k

6.4.3 Probleme des EM-Clustering

6.5 DBSCAN

6.5.1 Der Algorithmus

6.5.2 Bestimmen von ϵ und minPts

6.5.3 Probleme bei DBSCAN

6.6 Single Linkage Clustering

6.6.1 Der SLINK-Algorithmus

6.6.2 Dendrogramme

6.6.3 Probleme bei SLINK

6.7 Vergleich der Algorithmen

6.7.1 Clusterformen

6.7.2 Anzahl der Cluster

6.7.3 Ausführungszeit

6.7.4 Interpretierbarkeit und Darstellung

6.7.5 Kategorische Merkmale

6.7.6 Fehlende Merkmale

6.7.7 Korrelierte Merkmale

6.7.8 Zusammenfassung des Vergleichs

6.8 Übung

7 Klassifikation

7.1 Binäre Klassifikation und Grenzwerte

7.2 Gütemaße

7.2.1 Die Confusion Matrix

7.2.2 Die binäre Confusion Matrix

7.2.3 Binäre Gütemaße

7.2.4 Die Receiver Operator Characteristic (ROC)

7.2.5 Area Under the Curve (AUC)

7.2.6 Micro und Macro Averages

7.2.7 Jenseits der Confusion Matrix

7.3 Decision Surfaces

7.4 k-Nearest Neighbor

7.5 Entscheidungsbäume

7.6 Random Forests

7.7 Logistische Regression

7.8 Naive Bayes

7.9 Support Vector Machines (SVMs)

7.10 Neuronale Netzwerke

7.10.1 Exkurs: CNNs zum Erkennen von Zahlen

7.11 Vergleich der Klassifikationsalgorithmen

7.11.1 Grundidee

7.11.2 Decision Surfaces

7.11.3 Ausführungszeit

7.11.4 Interpretierbarkeit und Darstellung

7.11.5 Scoring

7.11.6 Kategorische Merkmale

7.11.7 Fehlende Merkmale

7.11.8 Korrelierte Merkmale

7.11.9 Zusammenfassung des Vergleichs

7.12 Übung

8 Regression

8.1 Güte von Regressionen

8.1.1 Visuelle Bewertung der Güte

8.1.2 Gütemaße

8.2 Lineare Regression

8.2.1 Ordinary Least Squares (OLS)

8.2.2 Ridge

8.2.3 Lasso

8.2.4 Elastic Net

8.2.5 Auswirkung der Regularisierung

8.3 Jenseits von linearer Regression

8.4 Übung

9 Zeitreihenanalyse

9.1 Box-Jenkins-Verfahren

9.2 Trends und saisonale Effekte

9.2.1 Regression und das saisonale Mittel

9.2.2 Differencing

9.2.3 Vergleich der Ansätze

9.3 Autokorrelationen mit ARMA

9.3.1 Autokorrelation und partielle Autokorrelation

9.3.2 AR, MA und ARMA

9.3.3 Auswahl von p und q

9.3.4 ARIMA

9.4 Jenseits von Box-Jenkins

9.5 Übung

10 Text Mining

10.1 Preprocessing

10.1.1 Erstellung eines Korpus

10.1.2 Relevanter Inhalt

10.1.3 Zeichensetzung und Großschreibung

10.1.4 Stoppwörter

10.1.5 Stemming und Lemmatisierung

10.1.6 Visualisierung des Preprocessing

10.1.7 Bag-of-Words

10.1.8 Inverse Document Frequency

10.1.9 Jenseits des Bag-of-Words

10.2 Herausforderungen des Text Mining

10.2.1 Dimensionalität

10.2.2 Mehrdeutigkeiten

10.2.3 Weitere Probleme

10.3 Übung

11 Statistik

11.1 Hypothesentests

11.1.1 t-Test

11.1.2 Das Signifikanzniveau

11.1.3 Wichtige Hypothesentests

11.1.4 Anwendung der Tests

11.1.5 Übliche Fehler bei Hypothesentests

11.2 Effektstärke

11.3 Konfidenzintervalle

11.4 Gute Beschreibung von Ergebnissen

11.5 Übung

12 Big Data Processing

12.1 Parallelisierung

12.2 Verteiltes Rechnen zur Datenanalyse

12.3 Datenlokalität

12.4 MapReduce

12.4.1 map()

12.4.2 shuffle()

12.4.3 reduce()

12.4.4 Worthäufigkeiten mit MapReduce

12.4.5 Parallelisierung

12.5 Apache Hadoop

12.5.1 HDFS

12.5.2 YARN

12.5.3 MapReduce mit Hadoop

12.5.4 Streaming Mode

12.5.5 Weitere Komponenten von Hadoop

12.5.6 Grenzen von Hadoop

12.6 Apache Spark

12.6.1 Architektur

12.6.2 Datenstrukturen

12.6.3 Infrastruktur

12.6.4 Worthäufigkeiten mit Spark

12.7 Jenseits von Hadoop und Spark

13 Weiterführende Konzepte

Anhang

A Selbst ausführen

B Notationen

C Abkürzungen

D Literatur

Index

4 Erkunden der Daten

In diesem Kapitel befassen wir uns mit Methoden, die wir einsetzen können, um die Daten zu verstehen. Man spricht auch vom *Lernen der Daten*. Das heißt, dass wir uns Wissen über die Struktur, den Umfang (z.B. Größe und Anzahl von Datenpunkten) sowie den Wertebereich der Daten aneignen müssen. Eventuell wird auch ein Verständnis der Datenquellen benötigt, wenn die Richtigkeit und Zuverlässigkeit der Daten relevant sind oder wir die Gründe für unvollständige Daten näher beleuchten wollen.

Es gibt sehr viele Methoden, die man zur Erkundung von Daten einsetzen kann. Die Erkundung selbst ist ein durch Werkzeuge unterstützter interaktiver Prozess. Es ist üblich, verschiedene Werkzeuge zu benutzen, abhängig vom Aspekt, den man aktuell erkunden möchte. Im Folgenden stellen wir einige wichtige Werkzeuge vor, die uns dabei helfen, Daten zu verstehen.

Im Folgenden nutzen wir die Notation $x = (x_1, \dots, x_n)$ mit $x_1, \dots, x_n \in \mathbb{R}$. Der Wert x beschreibt also einen Vektor von reellen Zahlen. Wenn es sich bei x um eine Teilmenge der möglichen Daten handelt, nennt man x auch *Stichprobe* (engl. *sample*). Der Einfachheit halber können Sie sich x auch einfach als Menge von Zahlen vorstellen. Dies ist zwar für die Vorstellung oft einfacher, jedoch formal nicht korrekt, da Mengen nicht das gleiche Element mehrfach enthalten können, Datensätze jedoch schon.

4.1 Texteditoren und die Kommandozeile

Bereits sehr einfache Werkzeuge können uns helfen, die Daten zu verstehen. Texteditoren und einfache Kommandozeilenbefehle wie **head**, **more** und **less** ermöglichen es, sich den Inhalt der Textdateien direkt anzusehen. Hierdurch kann man beliebige Daten, die nicht im Binärformat vorliegen, inspizieren. Das ist durchaus häufig der Fall, zum Beispiel bei *Comma-Separated Values* (CSV), der *eXtensible Markup Language* (XML) oder der *JavaScript Object Notation* (JSON). Wir könnten uns zum Beispiel direkt den Quelltext dieses Kapitels

anschauen, um das Format der Daten, die in Jupyter Notebooks gespeichert werden, zu verstehen:

```
!head kapitel_04.ipynb
```

```
{  
  
  "cells": [  
  
    {  
  
      "cell_type": "markdown",  
  
      "id": "ceramic-tiger",  
  
      "metadata": {},  
  
      "source": [  
  
        "# Erkunden der Daten\n",  
  
        "\n",  
  
        "In diesem Kapitel befassen wir uns mit Methoden, die wir einsetzen können,  
  
        um die Daten zu verstehen. Man spricht auch vom *Lernen der Daten*. Das  
  
        heißt, dass wir uns Wissen über die Struktur, den Umfang (z.B. Größe und  
  
        Anzahl von Datenpunkten) sowie den Wertebereich der Daten aneignen
```

müssen. Eventuell wird auch ein Verständnis der Datenquellen benötigt,

wenn die Richtigkeit und Zuverlässigkeit der Daten relevant sind oder wir

die Gründe für unvollständige Daten näher beleuchten wollen. \n",

Durch diesen einfachen Befehl, der nur die ersten Zeilen der Datei anzeigt, können wir viel über die Daten lernen: Die Daten sind im JSON-Format als Liste von Zellen (**cells**) gespeichert. Für jede Zelle werden der Typ, ein Identifier, eventuelle Metadaten sowie der Quelltext der Zelle als Liste gespeichert, wobei jeder Listeneintrag eine Zeile des Quelltextes darstellt. All das erfahren wir nur mit einem einfachen Kommandozeilenbefehl.

Bemerkung:

Man kann durch ein Ausrufezeichen als Präfix beliebige Kommandozeilenbefehle (z.B. bash) direkt aus einem Jupyter Notebook aufrufen. Daher ist **!head** in einem Jupyter Notebook äquivalent zur Eingabe von **head** direkt in der Kommandozeile. Welche Befehle zur Verfügung stehen, hängt von der Ausführungsumgebung ab. Weil die Kommandozeile von Windows den Befehl **head** nicht kennt, würde dies zum Beispiel dort eventuell nicht funktionieren.

Dies ist natürlich nur eine Möglichkeit, die Art, wie Jupyter Notebooks Daten speichern, zu betrachten. Und wie bei den meisten sehr einfachen Lösungen gibt es natürlich auch Schwächen. Es ist zum Beispiel nicht klar, ob es noch weitere Informationen gibt, die für Zellen gespeichert werden könnten, da der betrachtete Auszug eventuell nicht alle Möglichkeiten des Datenformats ausschöpft. Daher ist es auch wichtig, dass die Metadaten betrachtet werden, in diesem Fall die Dokumentation des Datenformats¹. Dort findet man zum Beispiel heraus, dass es noch weitere Felder gibt, wie zum Beispiel das Feld **nbformat**, um die Version des Datenformats anzuzeigen. Eine gute Dokumentation der Metadaten beschreibt das Datenformat vollständig, das heißt, welche Felder es gibt, welche Informationen in einem Feld gespeichert werden und wie die Felder zusammenhängen.

Die Metadaten sind nicht notwendigerweise auf die Beschreibung der Daten selbst beschränkt. Es könnten zum Beispiel auch Links zu weiterführenden Informationen zur Verfügung gestellt werden. Außerdem könnten auch die Datenquellen beschrieben sein. Was man in den Metadaten üblicherweise nicht

vorfndet, ist eine Beschreibung der Werte, die die Daten in einem Datensatz haben. Hier findet man hchstens Beispiele, die veranschaulichen, welche Werte angenommen werden knnen. Dies ist ein Nachteil gegenber der direkten Betrachtung der Daten, wobei die Erkenntnisse ber die Werte der Daten, die man mithilfe eines Texteditors bekommen kann, ebenfalls stark limitiert sind, gerade bei groen Datenmengen. Hierfr sind die Statistiken und Visualisierungen, die wir im Folgenden betrachten, deutlich besser geeignet.

4.2 Deskriptive Statistik

Die *deskriptive* Statistik beschftigt sich mit der Beschreibung von Eigenschaften eines Datensatzes durch einzelne, hufig numerische Werte. Deskriptive Statistiken sollten nicht mit *induktiver* Statistik verwechselt werden, bei der es darum geht, Eigenschaften von Daten vorherzusagen. Entsprechend sollte man nicht annehmen, dass deskriptive Statistiken fr zuverlssige Vorhersagen von zuknftigen Werten geeignet sind.

In diesem Kapitel betrachten wir die folgenden deskriptiven Statistiken:

- Die *Lage* der Daten (engl. *central tendency*) durch das *arithmetische Mittel*, den *Median* und den *Modus*
- Die *Variabilität* der Daten durch die *Standardabweichung*, den *Interquartilsabstand (IQR)* und den *Median der absoluten Abweichung vom Median (MAD)*
- Der *Datenbereich* (engl. *range*) durch das *Minimum* und *Maximum*

Es gibt noch viele weitere deskriptive Statistiken fr die obigen Eigenschaften, die wir in diesem Kapitel nicht betrachten, zum Beispiel das harmonische Mittel fr die Lage. Auerdem gibt es noch weitere Eigenschaften, die sich durch deskriptive Statistiken erfassen lassen, zum Beispiel die *Form* (engl. *shape*) der Daten durch die *Kurtose* (Wlbung) und die *Schiefe* (engl. *skewness*). Diese gehen jedoch ber die Betrachtungen dieses Kapitels hinaus und werden hufig auch nicht bentigt, insbesondere wenn man deskriptive Statistiken zusammen mit Visualisierungen benutzt.

4.2.1 Lagemaße

Die Lage der Daten ist ein wichtiges statistisches Merkmal, das den *typischen Wert* im Zentrum der Daten beschreibt. Dies bedeutet nicht, dass viele Datenpunkte exakt diesen Wert haben oder dass man in der Zukunft erwarten

könnte, häufig diesen Wert zu beobachten. Stattdessen markiert die Lage die »Mitte« der Daten. Wenn Sie sich die Daten als Stadt vorstellen, markiert die Lage also nur den Mittelpunkt der Stadt. Das heißt jedoch nicht, dass dort besonders viele Häuser stehen.

Es gibt verschiedene Wege, die Lage von Daten zu definieren. Das *arithmetische Mittel* ist definiert als

$$\text{mean}(x) = \frac{1}{n} \sum_{i=1}^n x_i,$$

also die Summe der Datenpunkte geteilt durch ihre Anzahl. Häufig spricht man auch einfach vom *Mittelwert* der Daten (engl. *mean*). Das arithmetische Mittel ist eine gute Beschreibung der Lage, wenn die Daten normalverteilt oder gleichverteilt sind. Im Allgemeinen sollte man das arithmetische Mittel nur verwenden, wenn zwei Eigenschaften erfüllt sind:

Es gibt keine »Lücken« in den Daten, also keine größeren Bereiche, in denen keine Werte liegen. Dies bedeutet insbesondere auch, dass die Daten keine Ausreißer haben sollten.

Die Daten sind symmetrisch um das arithmetische Mittel verteilt, das heißt, dass die Verteilung links und rechts des arithmetischen Mittels in etwa gleich sein sollte.

Statistische Methoden, die derartige Annahmen an die Verteilung von Daten machen, nennt man auch *parametrische* Statistiken. Wenn diese Eigenschaften nicht erfüllt sind, sollte man das arithmetische Mittel nicht verwenden, da der Wert eventuell die Lage nicht gut beschreibt. Wir werden uns später noch anhand von *Anscombes Quartett* anschauen, wie die Werte von Statistiken irreführend sein können.

Eine Alternative zum arithmetischen Mittel ist der *Median*, der definiert ist als

$$\text{median}(x) = \begin{cases} \bar{x}_m & \text{wenn } n \text{ ungerade ist mit } m = \frac{n+1}{2} \\ \frac{1}{2} (\bar{x}_m + \bar{x}_{m+1}) & \text{wenn } n \text{ gerade ist mit } m = \frac{n}{2}, \end{cases}$$

wobei \bar{x} ein nach der Größe sortierter Vektor der Werte von x ist. Entsprechend ist der Median wortwörtlich in der Mitte der Daten: 50 % der Daten sind kleiner oder gleich dem Median und 50 % sind größer oder gleich dem Median. Im Gegensatz zum arithmetischen Mittel ist der Median eine *nicht parametrische* Statistik. Das bedeutet, dass der Median unabhängig von der Verteilung

eingesetzt werden kann, zum Beispiel wenn die Annahmen an das arithmetische Mittel nicht erfüllt sind. Insbesondere ist der Median robust gegen Ausreißer, kann also nicht durch wenige besonders hohe oder niedrige Werte beeinflusst werden.

Dies kann man sich auch an einem einfachen Beispiel verdeutlichen. Zuerst betrachten wir das arithmetische Mittel und den Median für Daten, bei denen die Annahmen des arithmetischen Mittels erfüllt sind:

```
import statistics # we use the statistics from the Python
standard library
```

```
data = [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84,
4.82, 5.68]
```

```
print('Daten ohne Ausreißer:', data)
```

```
print('mean: ', statistics.mean(data))
```

```
print('median:', statistics.median(data))
```

```
Daten ohne Ausreißer: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
7.24, 4.26, 10.84,
```

```
4.82, 5.68]
```

```
mean: 7.500909090909091
```

```
median: 7.58
```

In diesem Beispiel sind die Werte vom Median und vom arithmetischen Mittel ähnlich und beide sind gut geeignet, um die Lage zu beschreiben. Jetzt erweitern wir die Daten um einen einzelnen Ausreißer.

```
data.append(100)
```

```
print('Daten mit Ausreißer:', data)
```

```
print('mean: ', statistics.mean(data))
```

```
print('median:', statistics.median(data))
```

Daten mit Ausreißer: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
7.24, 4.26, 10.84,

4.82, 5.68, 100]

mean: 15.209166666666667

median: 7.81

Durch den Ausreißer wird das arithmetische Mittel stark beeinflusst, sodass es jetzt deutlich höher ist. Es ist sogar höher als alle Datenpunkte, die wir ursprünglich betrachtet haben. Das ist offensichtlich keine gute Beschreibung der Mitte der Daten. Im Vergleich dazu hat sich der Median kaum verändert und ist nur zum Mittelwert mit dem nächsthöheren Wert geworden, da wir jetzt eine gerade Anzahl von Datenpunkten haben.

Eine berechtigte Frage ist, warum wir das arithmetische Mittel überhaupt betrachten, wenn der Median scheinbar aufgrund seiner Robustheit überlegen ist. Die Gründe hierfür kommen aus der Stochastik. Das arithmetische Mittel ist eng verwandt mit dem Erwartungswert von Zufallsvariablen. Die Normalverteilung kann man vollständig durch den Mittelwert und die Standardabweichung beschreiben. Hieraus folgt, dass das arithmetische Mittel für die Normalverteilung die bestmögliche Schätzung für die Mitte der Daten ist. Trotzdem ist es ratsam, den Median zu verwenden, wenn wir die Verteilung der Daten nicht kennen oder wir befürchten, dass es Ausreißer gibt.

Das arithmetische Mittel und der Median sind nur für numerische Daten definiert. Es gibt jedoch auch nicht numerische Daten. Die Größe von Kleidungsstücken wird häufig in den Kategorien »Small«, »Medium« und »Large« angegeben. Man spricht hierbei von kategorischen Daten (siehe Kap. 3). Der *Modus* der Daten kann benutzt werden, um die Lage von kategorischen Daten zu bestimmen. Der Modus ist definiert als der Wert, den man am häufigsten in einer Stichprobe x beobachtet.

```
data = ['small', 'medium', 'small', 'large', 'large',  
        'medium', 'medium']
```

```
print('Daten:', data)
```

```
print('Modus:', statistics.mode(data))
```

```
Daten: ['small', 'medium', 'small', 'large', 'large',  
        'medium', 'medium']
```

```
Modus: medium
```

Bitte beachten Sie, dass der Modus zwar gut für kategorische Daten (und manchmal auch für diskret verteilte numerische Daten) geeignet ist, aber irreführend sein kann, wenn die Daten *bimodal* oder *multimodal* sind. Derartige Daten setzen sich aus zwei oder mehr Verteilungen zusammen und haben daher mehrere »Mittelpunkte«, was sich dadurch äußert, dass mehrere Werte besonders häufig auftreten. Im Extremfall sind die Kategorien gleichverteilt, sodass es gleich viele Beobachtungen jeder Kategorie gibt. Da dann jede Kategorie die gleiche Wahrscheinlichkeit hat, ist der Modus zufällig. Außerdem sollte man den Modus nicht für kontinuierliche numerische Daten benutzen, da es hier sehr unwahrscheinlich ist, mehrfach den exakt gleichen Wert zu beobachten.

4.2.2 Variabilität

Die Variabilität misst, wie stark die Daten verstreut sind. Eine kleine Variabilität heißt, dass sich viele Daten in der Nähe der Mitte befinden. Eine hohe Variabilität bedeutet dagegen, dass die Daten sich über einen großen Bereich verteilen.

Das am häufigsten verwendete Maß für die Variabilität ist die *Standardabweichung*, die als

$$sd(x) = \sqrt{\frac{\sum_{i=1}^n (x_i - \text{mean}(x))^2}{n-1}}$$

definiert ist. Die Standardabweichung ist die Quadratwurzel des arithmetischen Mittels von der Abweichung der einzelnen Datenpunkte vom arithmetischen

Mittel, nur dass bei der Standardabweichung durch $n - 1$ statt n geteilt wird. Der Grund hierfür sind die sogenannten *Freiheitsgrade*, auf die wir hier jedoch nicht näher eingehen. Auch wenn die Formulierung etwas kompliziert klingt, ist das Konzept einfach: Die Standardabweichung misst, wie stark die beobachteten Werte vom arithmetischen Mittel abweichen. Dadurch, dass die quadratischen Abweichungen betrachtet werden, beeinflussen große Abweichungen die Standardabweichung stärker als kleine Abweichungen. Da die Standardabweichung die Variabilität in Bezug auf das arithmetische Mittel misst, ist auch klar, dass die Standardabweichung nur dann benutzt werden sollte, wenn das arithmetische Mittel eine geeignete Repräsentation für die Lage der Daten ist. Hieraus folgt auch, dass die Standardabweichung ebenfalls ein parametrisches Maß für die Variabilität ist.

Ebenso wie es mit dem Median ein nicht parametrisches Gegenstück zum arithmetischen Mittel gibt, gibt es auch nicht parametrische Statistiken für die Variabilität. Der *Interquartilsabstand* (engl. *interquartile range*) ist definiert als

$$IQR(X) = Q_{upper} - Q_{lower},$$

wobei Q_{upper} und Q_{lower} das obere und untere *Quartil* der Daten sind. Die Quartile sind analog zum Median definiert, nur mit 75 % bzw. 25 % der Daten: 75 % der Daten sind kleiner oder gleich dem oberen Quartil und 25 % der Daten sind kleiner oder gleich dem unteren Quartil. Entsprechend sind mindestens 50 % der Daten größer oder gleich dem unteren Quartil und gleichzeitig kleiner oder gleich dem oberen Quartil. Der Interquartilsabstand misst also die Größe des Bereichs, in dem sich die mittleren 50 % der Daten befinden.

Eine gute Faustregel ist, dass Sie die Standardabweichung nutzen sollten, wenn Sie das arithmetische Mittel für die Lage verwenden, und den Interquartilsabstand, wenn Sie den Median betrachten. Das folgende Beispiel demonstriert, wie sich die Probleme mit Ausreißern auch in der Standardabweichung (*sd*) widerspiegeln.

```
from scipy import stats # we use scipy for the IQR
```

```
data = [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84,  
4.82, 5.68]
```

```
print('Daten ohne Ausreißer:', data)
```

```
print('sd: ', statistics.stdev(data))
```

```
print('IQR:', stats.iqr(data))

data.append(100)

print('Daten mit Ausreißer:', data)

print('sd: ', statistics.stdev(data))

print('IQR:', stats.iqr(data))
```

Daten ohne Ausreißer: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
7.24, 4.26, 10.84,

4.82, 5.68]

sd: 2.031568135925815

IQR: 2.2550000000000001

Daten mit Ausreißer: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
7.24, 4.26, 10.84,

4.82, 5.68, 100]

sd: 26.77235292350312

IQR: 2.465

Obwohl bis auf einen Datenpunkt alle Daten zwischen 4,26 und 10,84 liegen, ist die Standardabweichung ein Vielfaches von diesem Datenbereich durch den einen Ausreißer. Der Interquartilsabstand ist jedoch, ähnlich wie der Median, kaum beeinflusst.

Wenn die Annahmen an die Standardabweichung erfüllt sind, ist es jedoch ein sehr gutes Maß für die Variabilität der Daten. Ähnlich wie beim

arithmetischen Mittel, ist dies insbesondere bei normalverteilten Daten der Fall, die sich durch diese beiden Maße komplett beschreiben lassen. Außerdem kann man für die Normalverteilung beweisen, dass 68 % der Daten nicht mehr als eine Standardabweichung vom arithmetischen Mittel entfernt sind, 95 % der Daten nicht mehr als zwei Standardabweichungen und 99,7 % nicht mehr als drei Standardabweichungen. Dieser Umstand ist auch als *68-95-99,7-Regel* bekannt. Für andere Verteilungen kann man ähnliche, wenn auch schwächere Aussagen über die Verteilung mithilfe der Standardabweichung treffen. Die *Chebychev-Ungleichung* gilt für alle Verteilungen mit einem endlichen Erwartungswert (vereinfacht gesagt mit einem endlichen arithmetischen Mittel) und einer endlichen Varianz, die nicht null ist. Die Varianz ist nichts anderes als das Quadrat der Standardabweichung. Aus der Chebychev-Ungleichung folgt, dass 50 % der Daten nicht mehr als $\sqrt{2} \cdot sd$ vom arithmetischen Mittel entfernt sind. Hierdurch ergibt sich auch ein direkter Zusammenhang zum Interquartilsabstand, der ebenfalls eine Aussage über die mittleren 50 % der Daten macht.

Bitte beachten Sie, dass wir uns die Variabilität für kategorische Daten hier nicht im Detail anschauen [Kader & Perry 2017]. Die Grundidee ist, dass man betrachtet, wie häufig Daten in den einzelnen Kategorien vorkommen und wie stark diese Beobachtungen einer Gleichverteilung ähneln. Je gleichmäßiger sich die Daten auf die Kategorien verteilen, desto höher die Variabilität.

4.2.3 Datenbereich

Der Datenbereich gibt den Bereich an, in dem Werte beobachtet werden können. Für numerische Daten ist dieser durch das *Minimum* (den kleinsten möglichen Wert) und das *Maximum* (den größten möglichen Wert) definiert. Beide kann man anhand einer Stichprobe x schätzen, in der man den kleinsten und größten beobachteten Wert als *min* und *max* ermittelt. Die Differenz dieser Werte ist dann die Größe des Datenbereichs:

$$range = max - min.$$

Auch wenn diese statistischen Merkmale auf den ersten Blick trivial erscheinen, sind sie in der Praxis sehr hilfreich, insbesondere um Probleme mit den Daten zu erkennen. Wenn der Datenbereich nicht plausibel ist, stellt man zum Beispiel fest, dass es ungültige Werte gibt. Beobachtet man ein Minimum von -1 bei einem Merkmal, das das Alter in Jahren angibt, könnte dies auf fehlende Werte hindeuten. Bei einem zu hohen Maximum könnte es Fehler beim Übertragen der Daten gegeben haben. Da ungültige Werte nicht notwendigerweise Ausreißer

sind, bemerkt man diese unter Umständen nicht, wenn man nur die Lage und die Variabilität von Daten betrachtet.

4.3 Visualisierung

Visualisierungen sind ein sehr mächtiges Werkzeug, um etwas über Daten zu lernen und diese zu verstehen. Hier sehen Sie ein Beispiel von einem Tortendiagramm (die Prozentzahlen sind ausgedacht und haben keine wissenschaftliche Bedeutung).

```
# we use matplotlib for the creation of visualizations
```

```
import matplotlib.pyplot as plt
```

```
# this is only required for Jupyter notebooks
```

```
%matplotlib inline
```

```
# data to plot
```

```
labels = ['Schnelleres Verständnis', 'Bessere  
Langzeitmemorierung',
```

```
         'Kann Muster zeigen', 'Kann komplexe Informationen  
vereinfachen',
```

```
         'Kann lustige Bilder beinhalten']
```

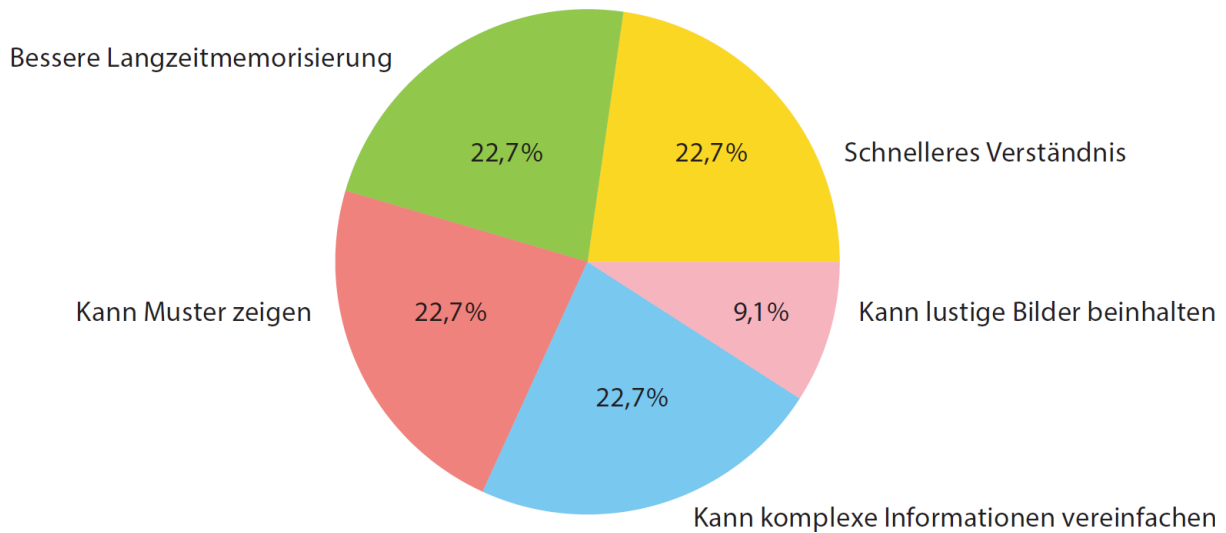
```
sizes = [25, 25, 25, 25, 10]
```

```
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue',  
'pink']
```

```
# plot
```

```
plt.pie(sizes, labels=labels, colors=colors,  
autopct='%1.1f%%')
```

```
plt.show()
```



Es gibt viele Vorteile von Visualisierungen gegenüber anderen Arten, Daten darzustellen, zum Beispiel Tabellen oder deskriptive Statistik. Im Allgemeinen verarbeitet man visuelle Informationen schneller. Hierdurch kann man in der Regel schneller etwas über die Daten lernen. Hinzu kommt, dass viele Menschen sich visualisierte Informationen besser merken können.

Die Vorteile von Visualisierungen gehen jedoch über die Verarbeitungsgeschwindigkeit und die Memorisierung hinaus. Visualisierungen können auch ein Verständnis der Daten ermöglichen, das man andernfalls nicht erlangen würde, zum Beispiel über Muster in Daten und komplexe Zusammenhänge, die man sonst nicht sehen könnte. Hierfür betrachten wir das folgende Beispiel. Alice kennt Bob und Dan, Dan kennt Bob, Bob kennt Carol und Alice, und Carol kennt Alice, Bob und Dan. Diese textuelle Beschreibung ist sehr komplex und schwer zu verstehen. Außerdem bekommt man kein intuitives Verständnis der Beziehung zwischen Alice, Bob, Carol und Dan. Jetzt betrachten wir das Gleiche als gerichteten Graphen:

```
import networkx as nx # networkx is a powerful library for  
working with graphs
```

```
# Create the graph by adding edges
```

```
# The vertices are implicitly defined through the endpoints of  
the edges
```

```
graph = nx.DiGraph()
```

```
graph.add_edge('Alice', 'Bob')
```

```
graph.add_edge('Alice', 'Dan')
```

```
graph.add_edge('Dan', 'Bob')
```

```
graph.add_edge('Bob', 'Carol')
```

```
graph.add_edge('Bob', 'Alice')
```

```
graph.add_edge('Carol', 'Alice')
```

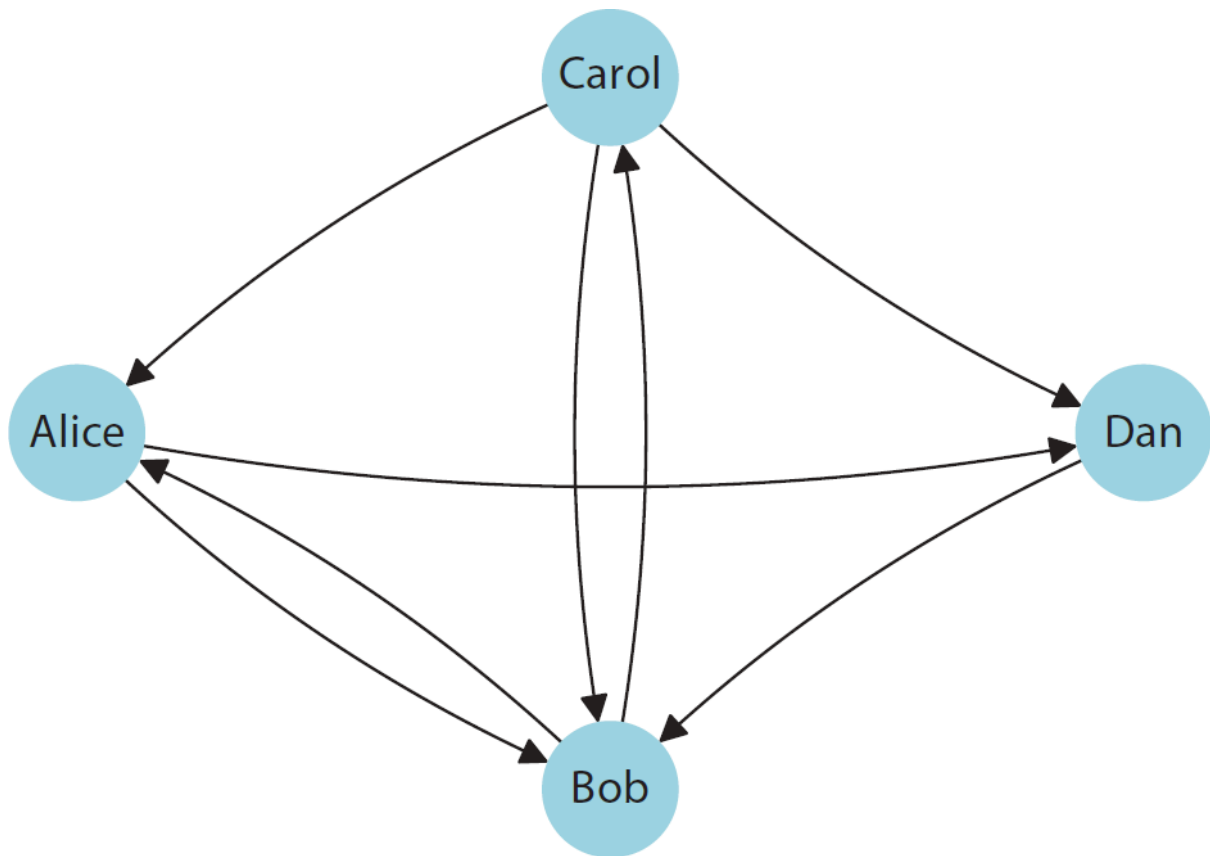
```
graph.add_edge('Carol', 'Bob')
```

```
graph.add_edge('Carol', 'Dan')
```

```
# Plot the graph with a shell layout
```

```
nx.draw_shell(graph, with_labels=True, node_size=2000,  
node_color='lightblue',
```

```
arrowsize=20, connectionstyle='arc3, rad = 0.1')
```



Dieser Graph ist einfach zu lesen und gibt uns ein intuitives Verständnis der Beziehungen.

Bitte beachten Sie, dass wir bei allen Grafiken in diesem Kapitel Legenden und andere Details bewusst weglassen. Stattdessen generieren wir die Grafiken mit möglichst wenig Quelltext, sodass sie trotzdem die gewünschten Erkenntnisse liefern. Der Grund hierfür ist, dass wir hier Visualisierungen als Werkzeug zur Erkundung von Daten betrachten. Wenn man ansonsten Visualisierungen für Texte (Bücher und sonstige Publikationen) oder Präsentationen erstellt, sind andere Aspekte ebenfalls relevant, zum Beispiel die konsistente Beschriftung, Farbwahl, Legenden und Titel.

4.3.1 Anscombes Quartett

Anscombes Quartett ist ein berühmtes Beispiel für Daten, in dem deskriptive Statistiken irreführend sind. Das Beispiel basiert auf vier Datensätzen $(x_1, y_1), \dots, (x_4, y_4)$ mit je elf Paaren aus zwei Variablen x_i und y_i , $i = 1, \dots, 4$. Tabelle 4-1 zeigt die Werte für jedes der Paare.

$x_{1,2,3}$	y_1	y_2	y_3	x_4	y_4
10	8,04	9,14	7,46	8	6,58
8	6,95	8,14	6,77	8	5,76
13	7,58	8,74	12,74	8	7,71
9	8,81	8,77	7,11	8	8,84
11	8,33	9,26	7,81	8	8,47
14	9,96	8,10	8,84	8	7,04
6	7,24	6,13	6,08	8	5,25
4	4,26	3,10	5,39	19	12,50
12	10,84	9,13	8,15	8	5,56
7	4,82	7,26	6,42	8	7,91
5	5,68	4,74	5,73	8	6,89

Tab. 4-1 Anscombes Quartett

Wenn wir das arithmetische Mittel und die Standardabweichung betrachten, sehen wir, dass die Werte für alle x_i und für alle y_i gleich sind.

```
import numpy as np # we now also need numpy, the commonly used
numerics
```

library for Python

```
x = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5]) # same
for x1, x2, and x3
```

```
y1 = np.array([8.04, 6.95, 7.58, 8.81, 8.33, 9.96,
               7.24, 4.26, 10.84, 4.82, 5.68])
```

```
y2 = np.array([9.14, 8.14, 8.74, 8.77, 9.26,
               8.10, 6.13, 3.10, 9.13, 7.26, 4.74])
```

```
y3 = np.array([7.46, 6.77, 12.74, 7.11, 7.81,
```

```

        8.84, 6.08, 5.39, 8.15, 6.42, 5.73])

x4 = np.array([8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8])

y4 = np.array([6.58, 5.76, 7.71, 8.84, 8.47, 7.04,
               5.25, 12.50, 5.56, 7.91, 6.89])

print('Arithmetisches Mittel')

print('x1, x2, x3 = ', statistics.mean(x))

print('x4          = ', statistics.mean(x4))

print('y1          = ', statistics.mean(y1))

print('y1          = ', statistics.mean(y2))

print('y1          = ', statistics.mean(y3))

print('y1          = ', statistics.mean(y4))

print('Standardabweichung')

print('x1, x2, x3 = ', statistics.stdev(x))

print('x4          = ', statistics.stdev(x4))

print('y1          = ', statistics.stdev(y1))

print('y1          = ', statistics.stdev(y2))

print('y1          = ', statistics.stdev(y3))

```

```
print('y1 = ', statistics.stdev(y4))
```

Arithmetisches Mittel

```
x1, x2, x3 = 9
```

```
x4 = 9
```

```
y1 = 7.500909090909091
```

```
y1 = 7.500909090909091
```

```
y1 = 7.5
```

```
y1 = 7.500909090909091
```

Standardabweichung

```
x1, x2, x3 = 3.3166247903554
```

```
x4 = 3.3166247903554
```

```
y1 = 2.031568135925815
```

```
y1 = 2.0316567355016177
```

```
y1 = 2.030423601123667
```

```
y1 = 2.0305785113876023
```

Es gibt sogar noch mehr Gemeinsamkeiten. Wenn wir eine lineare Regression von y_i durch x_i bestimmen würden (siehe Kap. 8), würden wir jedes Mal die gleiche Regressionsgerade $y = 3 + 0,5 \cdot x$ finden. Statistisch sind sich diese vier

Datensätze also sehr ähnlich. Wenn wir uns die Daten mit einem einfachen *Scatterplot* visualisieren, sehen wir, dass die Datensätze eigentlich sehr unterschiedlich sind.

```
xfit = np.array([4, 20])
```

```
yfit = 3+0.5*xfit
```

```
f, axes = plt.subplots(2, 2, sharey=True, sharex=True)
```

```
axes[0, 0].plot(x, y1, color='darkorange', marker='.',  
linestyle='none')
```

```
axes[0, 0].plot(xfit, yfit, color='navy', lw=1)
```

```
axes[0, 0].set_title('$x_1, y_1$')
```

```
axes[0, 1].plot(x, y2, color='darkorange', marker='.',  
linestyle='none')
```

```
axes[0, 1].plot(xfit, yfit, color='navy', lw=1)
```

```
axes[0, 1].set_title('$x_2, y_2$')
```

```
axes[1, 0].plot(x, y3, color='darkorange', marker='.',  
linestyle='none')
```

```
axes[1, 0].plot(xfit, yfit, color='navy', lw=1)
```

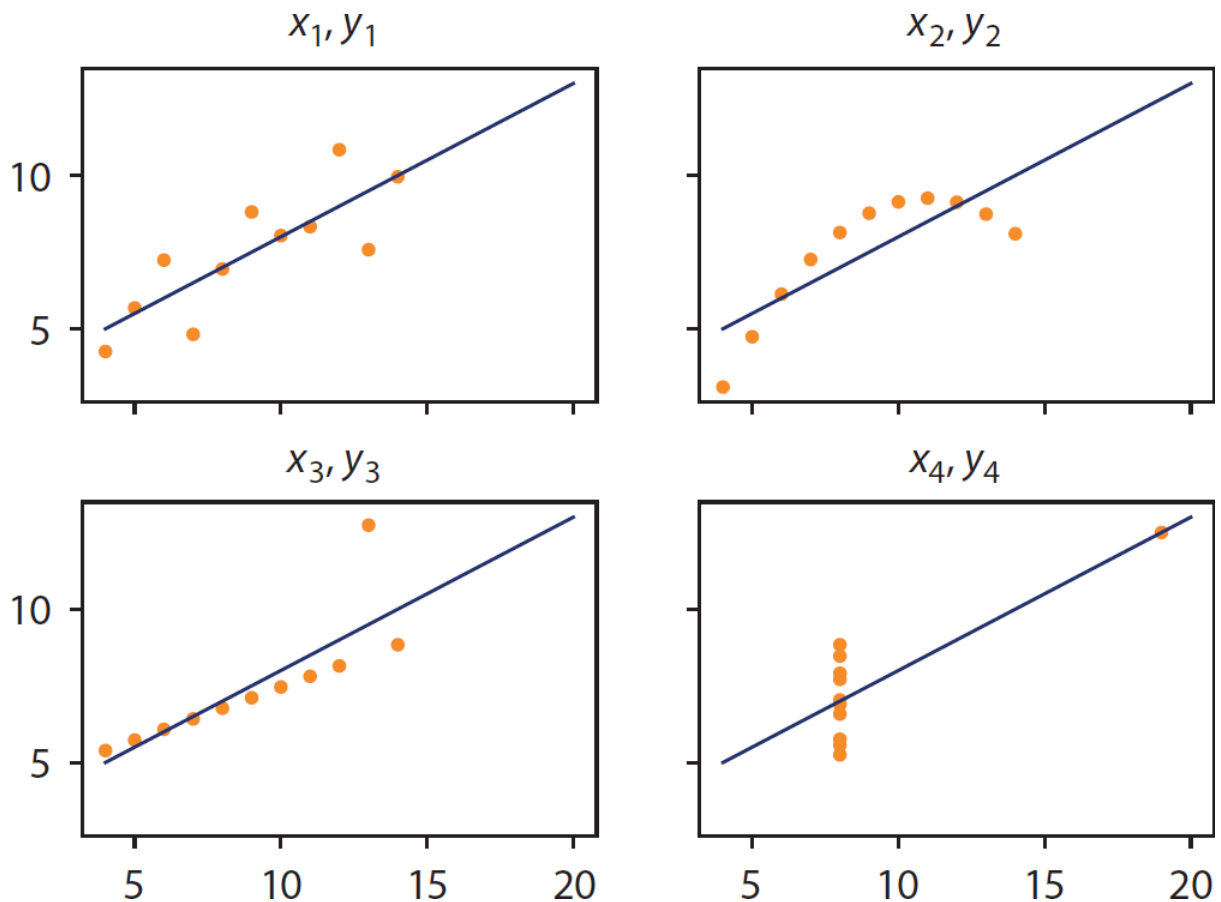
```
axes[1, 0].set_title('$x_3, y_3$')
```

```
axes[1, 1].plot(x4, y4, color='darkorange', marker='.',  
linestyle='none')
```

```
axes[1, 1].plot(xfit, yfit, color='navy', lw=1)
```

```
axes[1, 1].set_title('$x_4, y_4$')
```

```
plt.show()
```



Die orangen Punkte visualisieren die Daten selbst, die blauen Linien zeigen die Regressionsgerade für $y = 3 + 0,5 \cdot x$, die optimal ist für die Daten. Nur auf Basis der statistischen Informationen über die Daten würde man erwarten, dass die Daten ungefähr so wie beim Paar x_1, y_1 aussehen: Die Daten haben in etwa eine lineare Beziehung mit einer leichten Streuung ohne ein klar erkennbares Muster um die Regressionsgerade. Eine lineare Beziehung zwischen x und y bedeutet, dass wenn x sich verändert, sich y um ein konstantes Vielfaches von x verändert. Das bedeutet umgekehrt auch, dass man die Beziehung von x und y sich etwa als Gerade vorstellen kann. Beim Paar x_2, y_2 ist dies nicht der Fall, stattdessen sehen die Daten eher aus wie eine auf den Kopf gestellte Parabel. Das Paar x_3, y_3 ist eigentlich perfekt linear, bis auf einen einzelnen Datenpunkt, der nach oben ausreißt und dafür sorgt, dass die blaue Regressionsgerade nicht zur eigentlichen Geraden, auf der die Daten liegen, passt. Das Paar x_4, y_4 passt nicht

zu dem, was die Statistiken aussagen, die auch wieder von einem Ausreißer stark beeinflusst werden.

Die Aussage von Anscombes Quartett ist somit eindeutig: Auch wenn Statistiken gut geeignet sein können, Daten zusammenzufassen, ist es ebenso möglich, dass die Statistiken irreführend sind. Die kritische Leserin oder der aufmerksame Leser wird vielleicht bemerkt haben, dass die durch Ausreißer verursachten Probleme beim arithmetischen Mittel und der Standardabweichung zu erwarten sind. Die Probleme von Statistiken sind jedoch grundlegender und es gibt weitere Beispiele, die auch mehr statistische Marker berücksichtigen [Matejka & Fitzmaurice 2017].

4.3.2 Einzelne Merkmale

Eine grundlegende Betrachtung der Daten besteht in der Visualisierung einzelner Merkmale der Daten. Hierdurch kann man die Verteilung dieser Merkmale verstehen, ähnlich zur Beschreibung durch Statistiken. Hierzu schauen wir uns Histogramme, Densityplots, Rugs und Boxplots an.

Wir betrachten dieses Feature anhand von Daten über Hauspreise aus Boston, die 1978 veröffentlicht wurden, die wir im Folgenden einfach als Bostondaten bezeichnen werden.

```
# sklearn is a large machine learning library that we use
```

```
from sklearn import datasets
```

```
from textwrap import TextWrapper
```

```
# we use this wrapper to avoid printing lines that are too long because this
```

```
# should be readable as a book
```

```
# usually you would still just use print
```

```
wrapper = TextWrapper(width=65, replace_whitespace=False,  
break_long_words=False)
```

```
def wrap_print(string):  
    for line in string.split('\n'):  
        print('\n'.join(wrapper.wrap(line)))  
  
boston = datasets.fetch_openml(data_id=531)  
  
wrap_print(boston.DESCR)
```

****Author**:**

****Source**:** Unknown - Date unknown

****Please cite**:**

The Boston house-price data of Harrison, D. and Rubinfeld,
D.L.

'Hedonic

prices and the demand for clean air', J. Environ. Economics &

Management,

vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch,
'Regression

diagnostics

...', Wiley, 1980. N.B. Various transformations are used in
the

table on

pages 244-261 of the latter.

Variables in order:

CRIM per capita crime rate by town

ZN proportion of residential land zoned for lots over
25,000 sq.ft.

INDUS proportion of non-retail business acres per town

CHAS Charles River dummy variable (= 1 if tract bounds
river;

0 otherwise)

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to
1940

DIS weighted distances to five Boston employment centres

RAD index of accessibility to radial highways

TAX full-value property-tax rate per \$10,000

PTRATIO pupil-teacher ratio by town

B $1000(B_k - 0.63)^2$ where B_k is the proportion of
blacks

by town

LSTAT % lower status of the population

MEDV Median value of owner-occupied homes in \$1000's

Information about the dataset

CLASSTYPE: numeric

CLASSINDEX: last

Downloaded from openml.org.

Die obige Beschreibung ist ein Beispiel für Metadaten, in diesem Fall die Dokumentation der Daten.

Bemerkung:

Die Bostondaten werden immer häufiger kritisiert und wurden zum Beispiel auch als Beispieldatensatz aus scikit-learn entfernt. Der Grund hierfür ist, dass die Daten aus ethischer Sicht hochproblematisch sind: Die Daten sind teilweise rassistisch, insbesondere das Merkmal **B**. In diesem Buch verwenden wir diese Daten dennoch weiter und nutzen den Datensatz, um nicht nur zu zeigen, wie man Daten analysieren kann, sondern auch als Beispiel, dass man ethische Aspekte von Daten und Modellen niemals vergessen sollte.

Wir werfen jetzt einen genaueren Blick auf das Merkmal **MEDV**, den Median des Werts der Eigenheime in 1000 Dollar. *Histogramme* sind eine einfache und oft effektive Art, etwas über die Verteilung von Daten zu lernen.

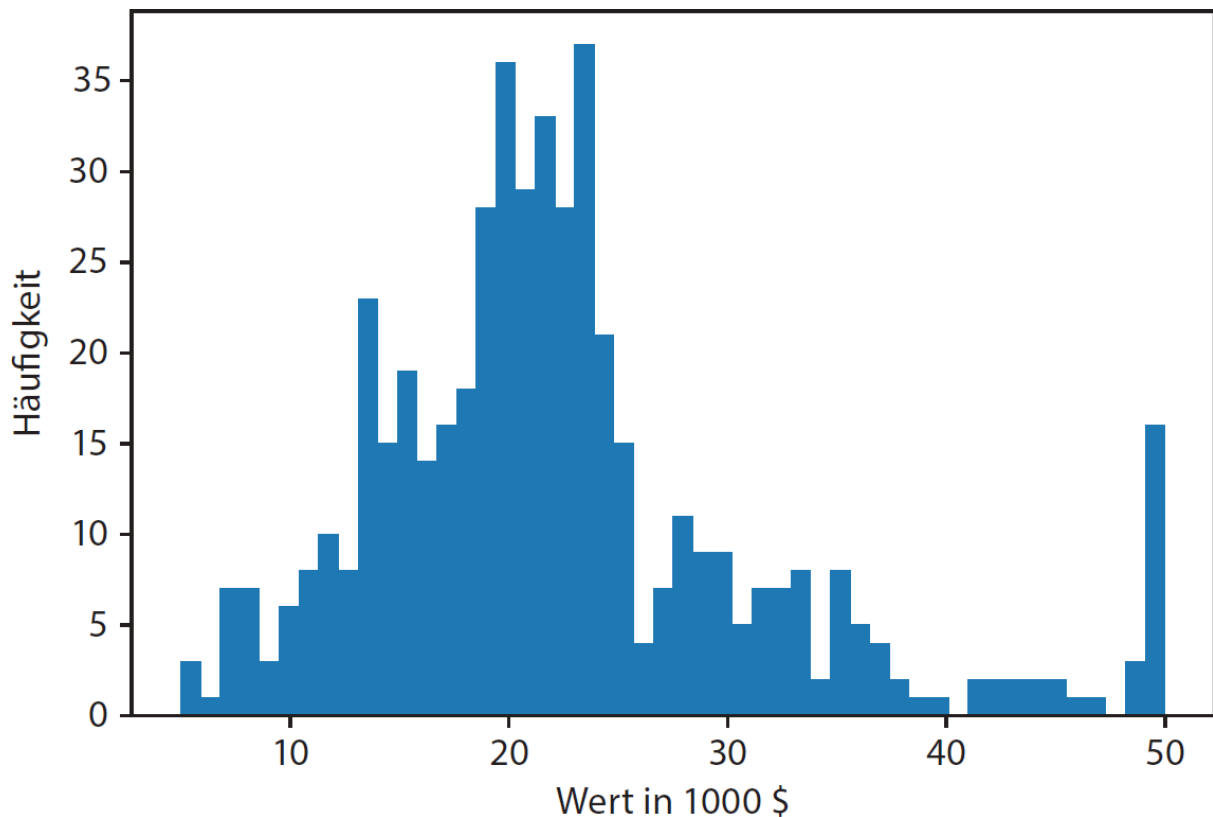
```
fig, ax = plt.subplots()
```

```
ax.hist(boston.target, bins=50)
```

```
ax.set_xlabel('Wert in 1000 $')
```

```
ax.set_ylabel('Häufigkeit')
```

```
plt.show()
```



Das Histogramm zeigt, wie häufig Werte vorkommen. Hierfür wird der Wertebereich in sogenannte *Bins* unterteilt. Der Begriff *Bin* ist vom englischen Wort für Gruppieren abgeleitet. Das Histogramm gibt an, wie viele Datenpunkte in jedem Bin liegen. Im obigen Beispiel haben wir 50 Bins. Das Histogramm verrät uns viel über die Daten.

- Die Daten zwischen 0 und 30 scheinen normalverteilt zu sein. Dies erkennt man daran, dass die Daten ungefähr eine *Glockenform* (engl. *bell-shape*) bilden, was für die Normalverteilung typisch ist. Der Mittelwert dieser Normalverteilung liegt ca. bei 22. Dies erkennt man am *Peak* in diesem Wertebereich. Die Standardabweichung kann man nicht genau ablesen, aber mithilfe der oben diskutierten 68-95-99-Regel grob abschätzen. Der Wert sollte sich zwischen 7 und 11 befinden.
- Neben dieser Normalverteilung gibt es noch viele Werte am rechten Rand der Grafik, also bei genau 50. Dies deutet darauf hin, dass es in Wirklichkeit vermutlich auch noch Werte oberhalb von 50 gibt, diese aber zusammengefasst wurden. Die eigentliche Bedeutung des Werts 50 sind somit nicht Häuser mit einem Wert von 50.000 Dollar, sondern Häuser, die mindestens 50.000 Dollar wert sind.

- Es gibt einen *Tail* auf der rechten Seite des Grafik, also Häuser, die teuer sind. Diese Häuser kommen zwar vor, lassen sich aber nicht mehr durch die Normalverteilung erklären. Solche Daten nennt man auch *Rechtsschief* (engl. *right skew*).

Wir können uns **MEDV** auch mithilfe von einem Densityplot anschauen.

```
import seaborn as sns # seaborn is a visualization library  
build on top of
```

```
matplotlib
```

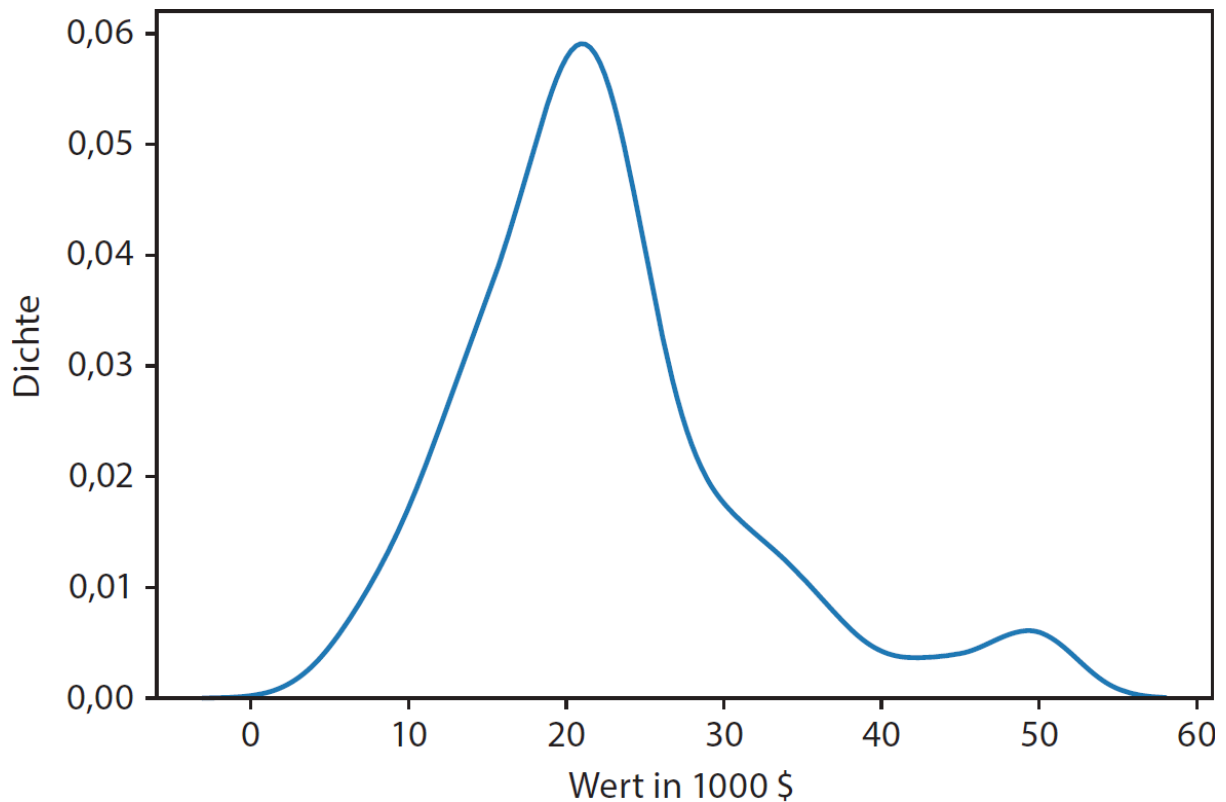
```
fig, ax = plt.subplots()
```

```
sns.kdeplot(boston.target, ax=ax)
```

```
ax.set_xlabel('Wert in 1000 $')
```

```
ax.set_ylabel('Dichte')
```

```
plt.show()
```



Vereinfacht gesagt, zeigen Densityplots eine Schätzung der Dichtefunktion der Wahrscheinlichkeitsverteilung der Daten. Wir können uns das als eine Art kontinuierliches Histogramm vorstellen. Der Vorteil von Densityplots gegenüber Histogrammen ist, dass es oft einfacher ist, die Verteilung der Daten zu erkennen. Im obigen Beispiel treten die Glockenform und die Position des Peaks deutlicher hervor und man kann dadurch die Normalverteilung in der linken Hälfte des Plots besser erkennen. Densityplots haben jedoch auch einige Nachteile im Vergleich zu Histogrammen. Ein kleiner Nachteil ist, dass es schwierig ist, die Werte der y-Achse zu interpretieren. Während ein Histogramm eine klare Aussage über die Anzahl der Datenpunkte in einem Bin erlaubt, sieht man im Densityplot lediglich die *Dichte* als Wert zwischen 0,0 und 1,0. Die Dichte ist mehr oder weniger der Anteil der Daten, der an einem bestimmten Punkt auf der x-Achse zu erwarten ist.

Der zweite Nachteil der Densityplots ist gravierender. Das Histogramm zeigt uns klar die vielen Datenpunkte mit dem Wert 50 und auch, dass es keine Datenpunkte mit einem höheren Wert gibt. Im Densityplot sieht man bei der 50 nur einen kleinen Peak, den man auch als weitere Normalverteilung mit dem Mittelwert 50 am rechten Rand der Daten interpretieren könnte. Das ist irreführend und versteckt die wahre Verteilung der Daten. Dieses Risiko lässt sich bei Densityplots auch nicht vermeiden, da es mit der Technik, wie die Dichte geschätzt wird, zusammenhängt. Solche Fehlinterpretationen der Daten sind

dann wahrscheinlich, wenn die Daten nicht sehr dicht verteilt sind (großer Datenbereich mit verhältnismäßig wenig Datenpunkten), sowie an den Grenzen der beobachteten Daten. Hier geht die geschätzte Dichtefunktion automatisch über den beobachteten Datenbereich hinaus, was eventuell aber der Interpretation eines Merkmals widerspricht. Man erkennt im obigen Plot zum Beispiel auch, dass die Werte kleiner 0 nicht eine Dichte von 0 haben, was bedeuten würde, dass einige Eigenheime einen negativen Wert hätten.

Bemerkung:

Densityplots werden mithilfe von *Kernel Density Functions* (KDE) erstellt, in der Regel mittels der Dichtefunktion der Normalverteilung. Diese Dichtefunktion wird dann an jedem Datenpunkt mit *Skalierungs-* und *Bandbreitenparametern* geschätzt. Anschließend werden alle diese geschätzten Dichtefunktionen aufaddiert, um den Densityplot zu erstellen. Hierdurch kann man auch das Verhalten an den Grenzen erklären: Die Dichtefunktion, die an den äußersten Punkten der Daten geschätzt wird, geht aufgrund der Symmetrie der Normalverteilung automatisch über den Datenbereich hinaus. Die Skalierung der Schätzung führt dazu, dass ein seltsames Verhalten, wie viele gleiche Werte an einer Grenze in der aufsummierten Dichtefunktion, nicht stark auffällt.

Ein einfaches Mittel, dieses Problem zu vermeiden, besteht darin, einen *Rug* (dt. Teppich) anzuzeigen. Der *Rug* hat seinem Namen daher, weil er wie ein Teppich unter den Plot gelegt wird.

```
fig, ax = plt.subplots()

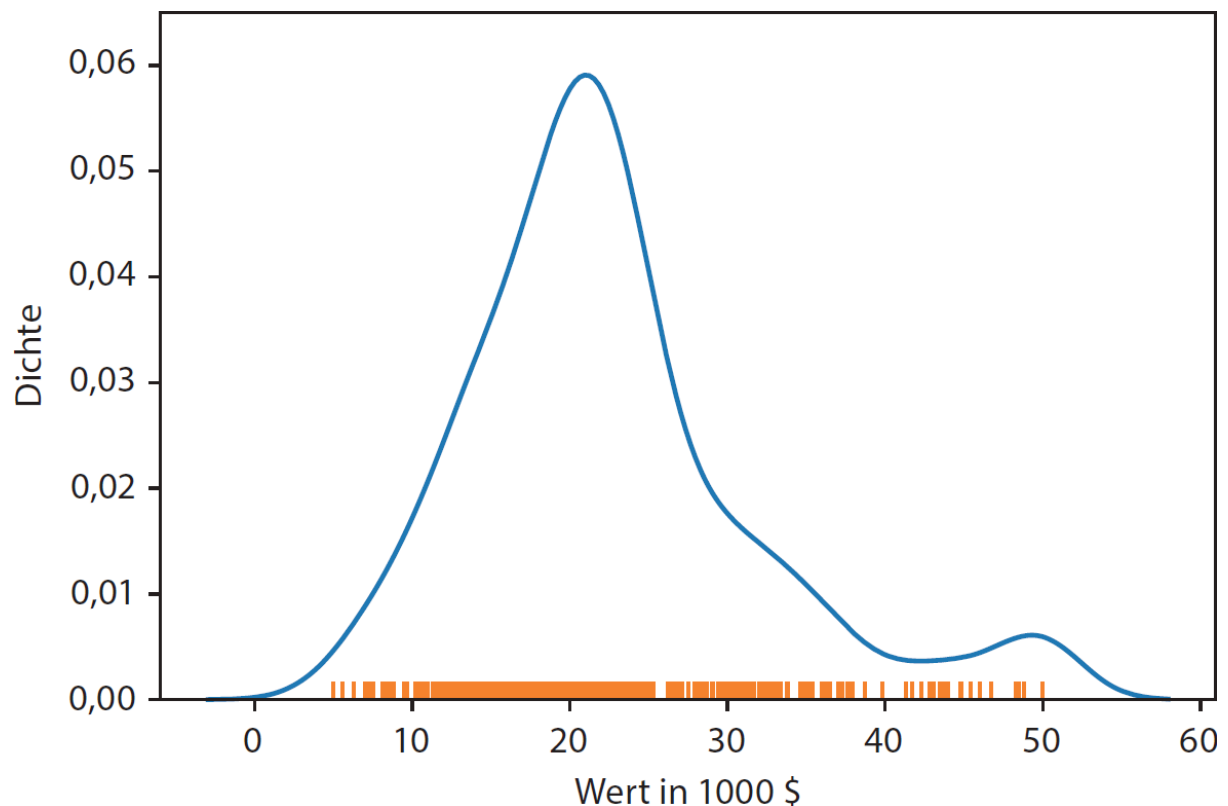
sns.kdeplot(boston.target, ax=ax)

sns.rugplot(boston.target, ax=ax)

ax.set_xlabel('Wert in 1000 $')

ax.set_ylabel('Dichte')

plt.show()
```



Der Rug zeigt, wo sich wirklich Datenpunkte befinden. In Kombination mit dem Densityplot können wir also sehen, dass es keine Datenpunkte kleiner als 5 oder größer 50 gibt. Wir erkennen auch, dass die Region zwischen 38 und 50 nur relativ dünn besiedelt ist, wobei die Punkte in diesem Bereich etwa gleichverteilt zu sein scheinen. Hierzu passt, dass der Densityplot in diesem Bereich fast parallel zur x-Achse verläuft. Dies zeigt, dass der Rug hilfreich ist, um weitere Erkenntnisse über die Daten zu gewinnen und Fehlinterpretationen zu vermeiden.

Man kann auch einfach alle oben betrachteten Ansätze kombinieren und Histogramm, Densityplot und Rug zusammen visualisieren.

```
fig, ax = plt.subplots()
```

```
# stat='density' scales the plot to match the y-axis of the density plot
```

```
# alpha=0.2 makes the bars transparent for better readability
```

```
sns.histplot(boston.target, stat='density', alpha=0.2,  
bins=50, ax=ax)
```

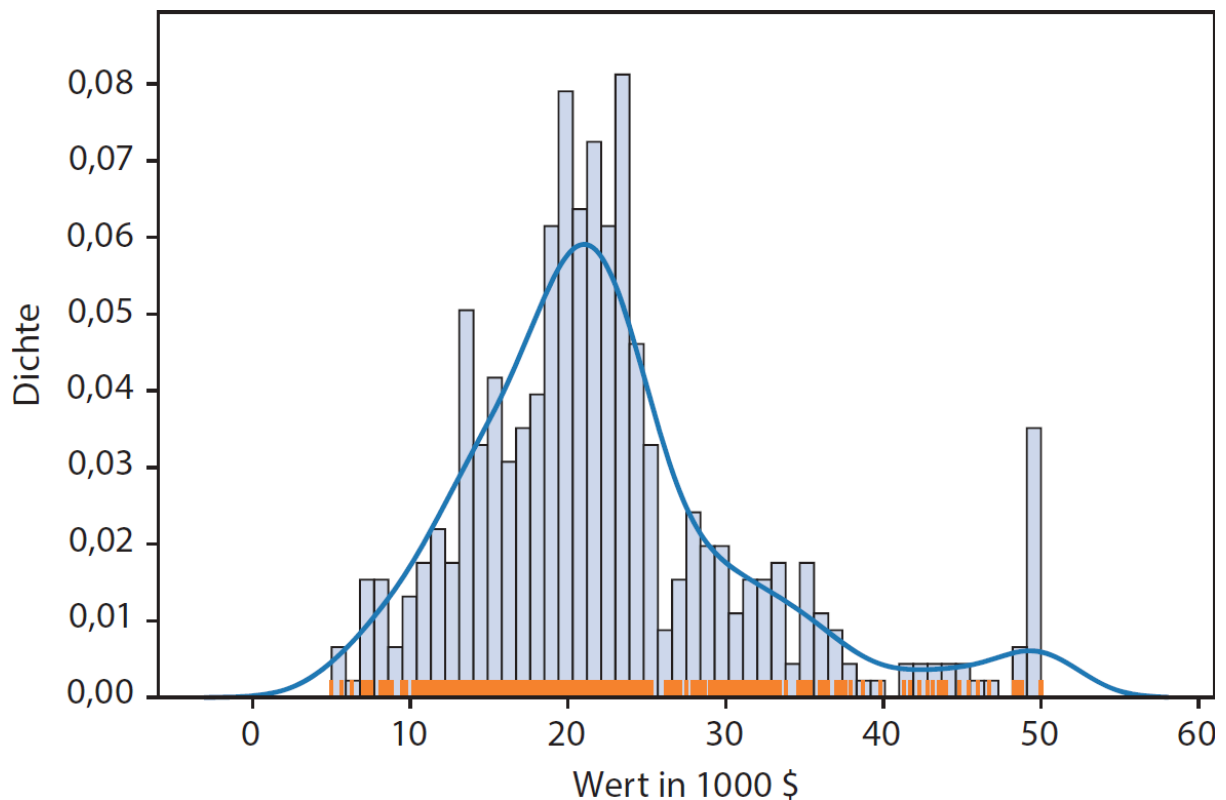
```
sns.kdeplot(boston.target, ax=ax)
```

```
sns.rugplot(boston.target, ax=ax)
```

```
ax.set_xlabel('Wert in 1000 $')
```

```
ax.set_ylabel('Dichte')
```

```
plt.show()
```



In diesem Plot können wir sehen, wo es Datenpunkte gibt (Rug), wie viele Daten in jedem Bereich liegen (Bins), und bekommen außerdem noch eine Schätzung der Dichtefunktion (Densityplot). Wir kombinieren also alle Vorteile von Histogrammen und Densityplots. Lediglich das Problem der Interpretation der y-Achse der Densityplots bleibt, wir können also nicht ablesen, wie viele Datenpunkte genau in einem Bin liegen. Falls dies relevant ist, muss man auf ein reines Histogramm zurückgreifen.

Das Merkmal **MEDV** verhält sich aus statistischer Sicht relativ gut: Es ist größtenteils normalverteilt, wenn auch etwas rechtsschief. Außerdem gibt es noch eine Gruppierung aller Werte, die größer sind als 50. Dies kann unter Umständen später zu Problemen führen und sollte berücksichtigt werden. Jetzt schauen wir uns ein zweites Merkmal an, bei dem es nicht ganz so einfach ist. **CRIM** ist in den Bostondaten definiert als die Kriminalitätsrate in einer Gegend.

```
fig, ax = plt.subplots()
```

```
sns.histplot(boston.data['CRIM'], stat='density', alpha=0.2,  
bins=50, ax=ax)
```

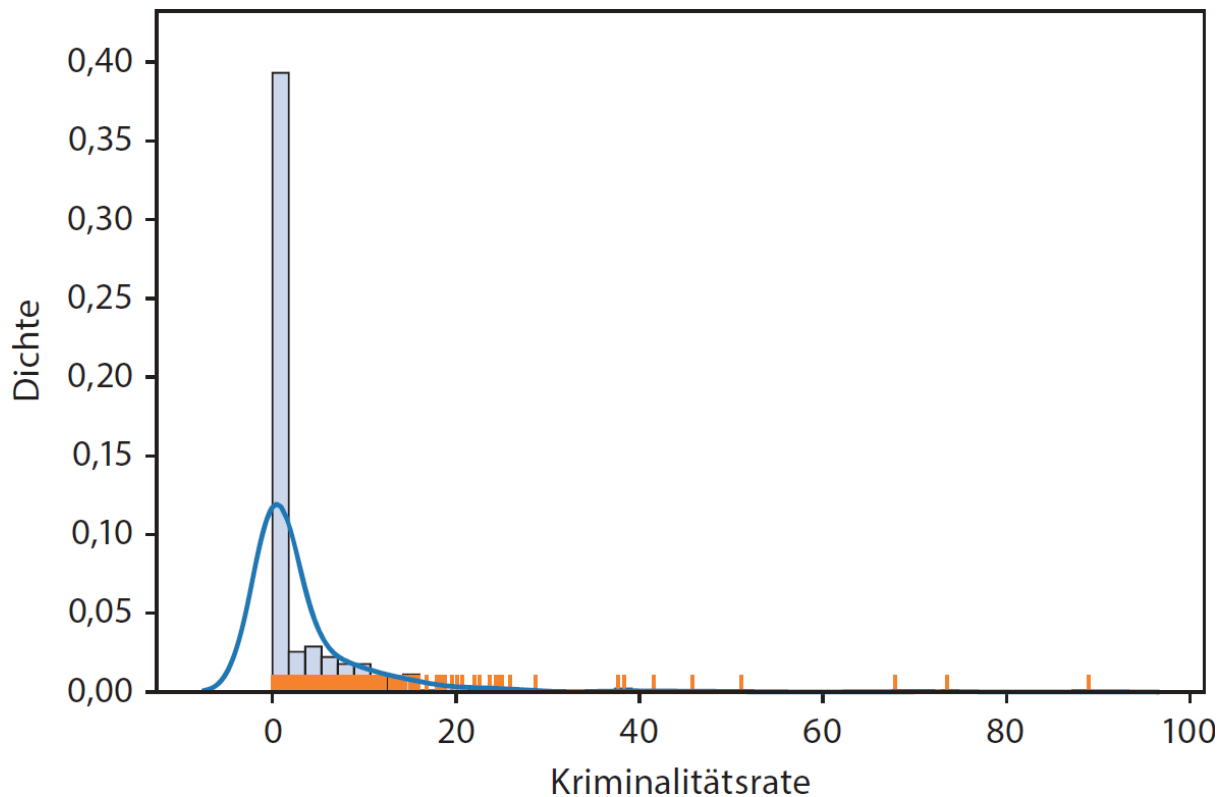
```
sns.kdeplot(boston.data['CRIM'], ax=ax)
```

```
sns.rugplot(boston.data['CRIM'], ax=ax)
```

```
ax.set_xlabel('Kriminalitätsrate')
```

```
ax.set_ylabel('Dichte')
```

```
plt.show()
```



Der Plot zeigt, dass die meisten Daten sehr nah bei null liegen, der Datenbereich bis ca. 15 jedoch insgesamt sehr dicht besiedelt ist. Bei höheren Werten scheint es sich eher um Ausreißer zu handeln. Da die Daten nahe null so dicht sind, ist es sehr schwer, mehr zu erkennen. Bei solchen Daten könnte es sich um exponentiell verteilte Daten handeln. Daher sollte man einen einfachen Trick ausprobieren und den Logarithmus der Daten betrachten. Wir betrachten also nicht die Daten x direkt, sondern stattdessen $\log(x + 1)$. Die Addition von eins ist ein Glättungsparameter, der auch als *Laplace Glättung* bekannt ist. Durch diesen Glättungsfaktor vermeiden wir das Problem, dass der Logarithmus von 0 undefiniert ist. Da $\log(0 + 1) = 0$, was bedeutet, dass sich der Nullpunkt der Daten nicht verschiebt.

Bemerkung:

Laplace Glättung ist sogar etwas generischer als die einfache Addition von eins. Im Allgemeinen kann jede beliebige Konstante addiert werden. Die obige Variante wird daher auch oft als *Add-One Glättung* bezeichnet. Diese Art der Glättung des Logarithmus ist so verbreitet, dass die Bibliothek `numpy` sogar extra die Funktion `log1p` sowie auch die Umkehrfunktion `exp1m` zur Verfügung stellt.

Betrachten wir jetzt also den Logarithmus der Kriminalitätsrate.

```
fig, ax = plt.subplots()
```

```
sns.histplot(np.log1p(boston.data['CRIM']), stat='density',  
alpha=0.2, bins=50,
```

```
ax=ax)
```

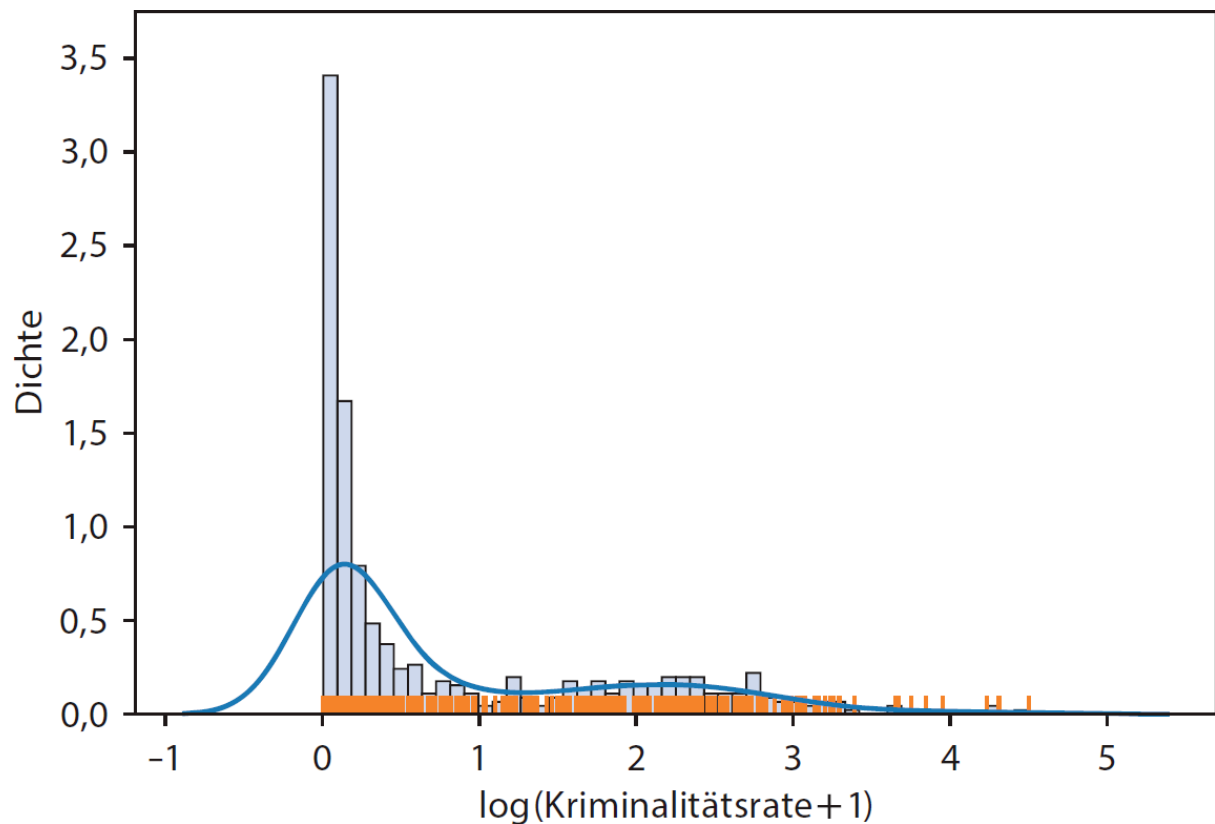
```
sns.kdeplot(np.log1p(boston.data['CRIM']), ax=ax)
```

```
sns.rugplot(np.log1p(boston.data['CRIM']), ax=ax)
```

```
ax.set_xlabel('log(Kriminalitätsrate+1)')
```

```
ax.set_ylabel('Dichte')
```

```
plt.show()
```



In diesem Plot erkennt man schon mehr Informationen über die Kriminalitätsrate:

- Etwa 80% der Gegenden sind nahezu ohne Kriminalität. Dass viele Daten nah bei null liegen, konnte man zwar bereits ohne den Logarithmus erkennen, aber den Anteil der Daten kann man jetzt klarer sehen. Man erkennt auch, dass die Kriminalitätsrate zwischen null und eins nahezu monoton abfällt, so dass dieser Teil der Daten in etwa wie die rechte Hälfte der Glockenform einer Normalverteilung aussieht.
- Die restlichen 20% der Daten ab ca. dem Wert eins scheinen einer *Lognormal*-Verteilung zu folgen. Man nennt eine Verteilung lognormal, wenn die Daten normalverteilt sind, nachdem man sie logarithmiert hat. Dies scheint hier der Fall zu sein, wie man an der sehr flachen Glockenform in der Mitte des Plots erkennt. Der Mittelwert scheint bei ca. 2,2 zu liegen und die Standardabweichung scheint relativ groß zu sein, weshalb die Daten weit streuen. Bitte beachten Sie, dass der Mittelwert von 2,2 für die logarithmierten Kriminalitätsraten gilt, das heißt, wir müssen erst die Umkehrung des Logarithmus berechnen, um den eigentlichen Wert zu ermitteln. Dadurch finden wir heraus, dass der lognormale Abschnitt der Daten bei einer Kriminalitätsrate von etwa $e^1 - 1 \approx 1,7$ beginnt, mit einem Mittel bei ca. $e^{2,2} - 1 \approx 8,0$.

Um unsere Einschätzung, dass der hintere Teil der Daten lognormal ist, zu überprüfen, können wir uns diese Daten auch noch einmal genauer anschauen, indem wir den Anzeigebereich einschränken und hierdurch den hinteren Bereich in der Anzeige vergrößern.

```
fig, ax = plt.subplots()

sns.histplot(np.log1p(boston.data['CRIM']), stat='density',
             alpha=0.2, bins=50,

             ax=ax)

sns.kdeplot(np.log1p(boston.data['CRIM']), ax=ax)

sns.rugplot(np.log1p(boston.data['CRIM']), ax=ax)

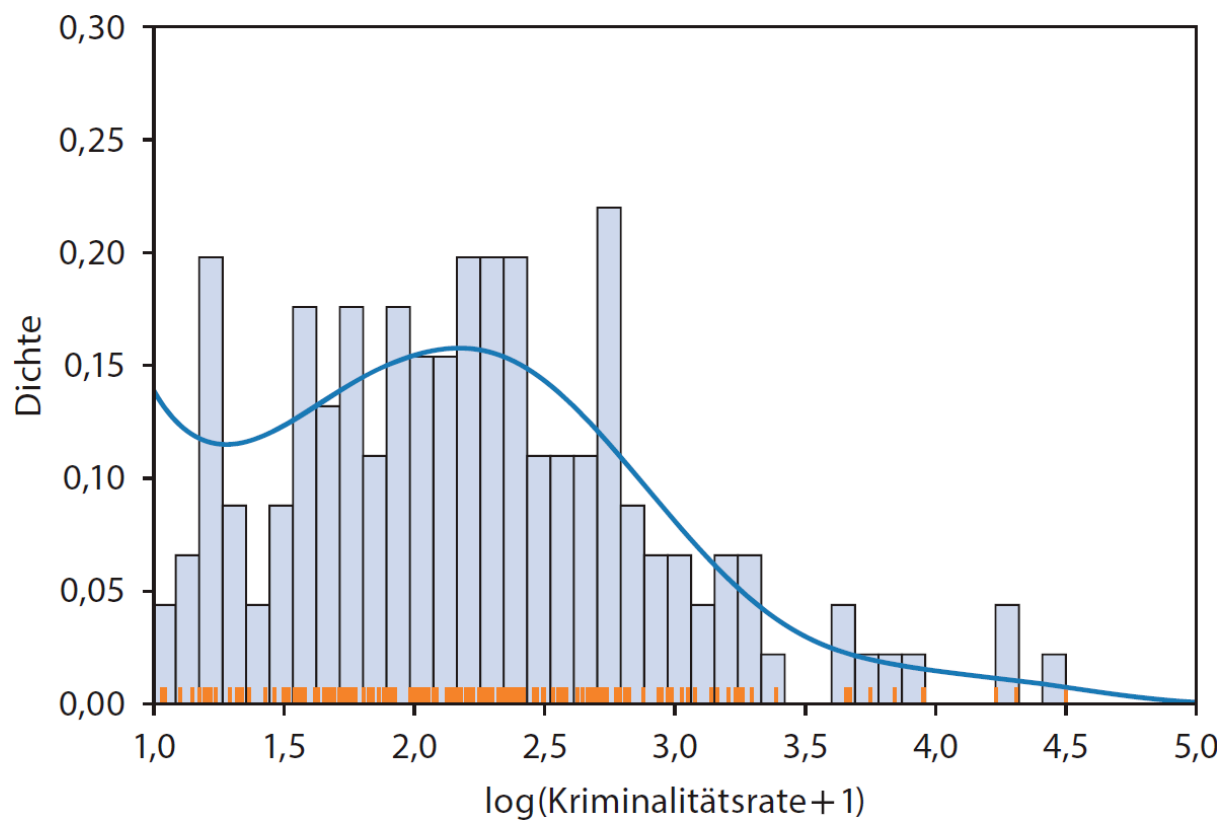
ax.set_xlabel('log(Kriminalitätsrate+1)')
```

```
ax.set_ylabel('Dichte')
```

```
ax.set_xlim(1,5)
```

```
ax.set_ylim(0,0.3)
```

```
plt.show()
```



Wie man sieht, wird die Einschätzung hierdurch bestätigt.

Für die nächsten Beispiele nutzen wir einen anderen Datensatz, nämlich die *Irisdaten*.

```
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
wrap_print(iris.DESCR)
```

```
.. _iris_dataset:
```

Iris plants dataset

****Data Set Characteristics:****

:Number of Instances: 150 (50 in each of three classes)

:Number of Attributes: 4 numeric, predictive attributes
and

the class

:Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica

:Summary Statistics:

```
=====
=====
                Min  Max   Mean   SD   Class Correlation
=====
=====

sepal length:   4.3  7.9   5.84   0.83   0.7826

sepal width:    2.0  4.4   3.05   0.43  -0.4194

petal length:   1.0  6.9   3.76   1.76   0.9490 (high!)

petal width:    0.1  2.5   1.20   0.76   0.9565 (high!)

=====
=====
```

:Missing Attribute Values: None

:Class Distribution: 33.3% for each of 3 classes.

:Creator: R.A. Fisher

:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken

from Fisher's paper. Note that it's the same as in R, but not as

in the UCI

Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and

is referenced frequently to this day. (See Duda & Hart, for example.) The

data set contains 3 classes of 50 instances each, where each class refers to a

type of iris plant. One class is linearly separable from the other 2; the

latter are NOT linearly separable from each other.

.. topic:: References

- Fisher, R.A. "The use of multiple measurements in taxonomic

problems"

Annual Eugenics, 7, Part II, 179-188 (1936); also in

"Contributions to

Mathematical Statistics" (John Wiley, NY, 1950).

- Duda, R.O., & Hart, P.E. (1973) Pattern Classification
and

Scene Analysis.

(Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See
page

218.

- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A

New System

Structure and Classification Rule for Recognition in

Partially Exposed

Environments". IEEE Transactions on Pattern Analysis and

Machine

Intelligence, Vol. PAMI-2, No. 1, 67-71.

- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".

IEEE Transactions

on Information Theory, May 1972, 431-433.

- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's

AUTOCLASS II

conceptual clustering system finds 3 classes in the data.

- Many, many more ...

Oben sehen wir die Beschreibung der Irisdaten. Es handelt sich um einen Datensatz über drei verschiedene Arten der Blume Iris (Setosa, Versicolour, Virginica) mit jeweils vier Merkmalen. Diese Merkmale sind die Länge und Breite der Blütenblätter (engl. *sepal*) und der Kelchblätter (engl. *petal*).

Boxplots sind eine weitere Möglichkeit, wie wir einzelne Merkmale, aber auch mehrere Merkmale gleichzeitig visualisieren können.

```
import pandas as pd # pandas is a powerful library for
working with data
```

```
# we need that data as dataframe for the pairplot feature of
seaborn
```

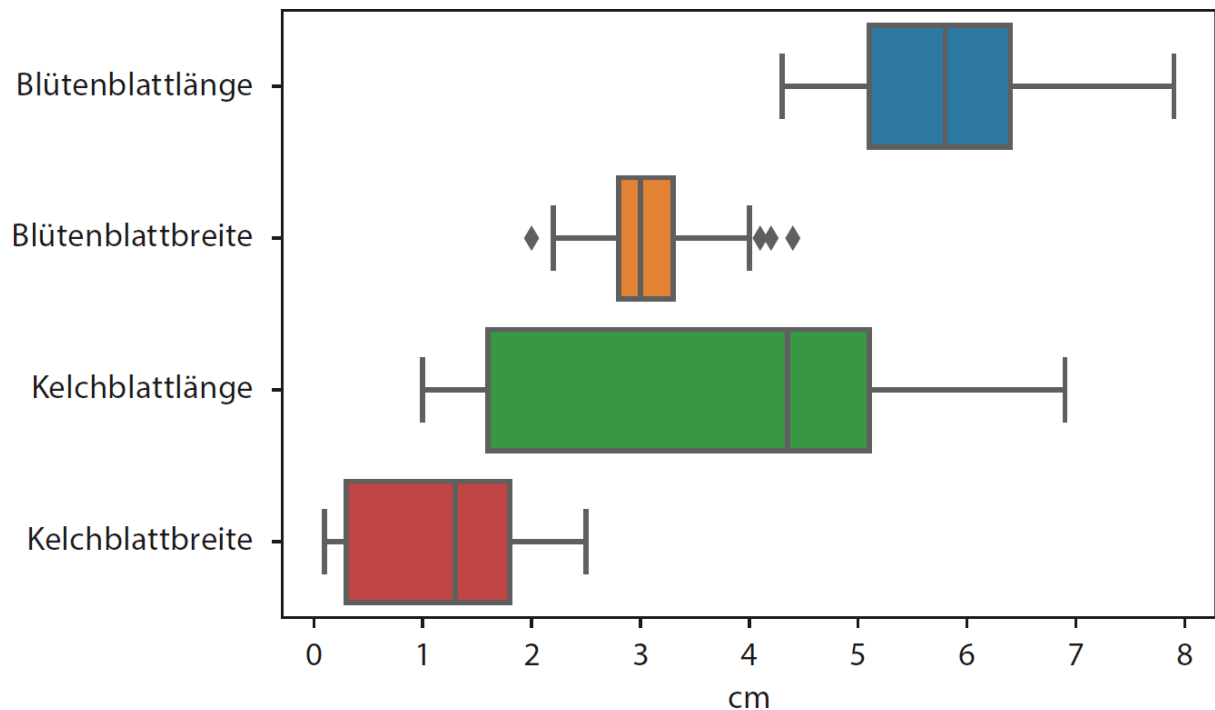
```
iris_df = pd.DataFrame(iris.data, columns=[
    'Blütenblattlänge', 'Blütenblattbreite',
    'Kelchblattlänge',
    'Kelchblattbreite'])
```

```
fig, ax = plt.subplots()
```

```
sns.boxplot(data=iris_df, orient='h')
```

```
ax.set_xlabel('cm')
```

```
plt.show()
```



Ein Boxplot besteht aus einer Box, die an jeder Seite noch eine Linie hat, die sogenannten *Whisker*. Daher sind diese Plots auch als *Box-Whisker-Plots* bekannt. Die Grenzen der Box sind durch das untere Quartil Q_{lower} und das obere Quartil Q_{upper} bestimmt. Daraus folgt, dass die Länge der Box der Interquartilsabstand ist. In der Box sieht man noch eine Linie. Diese Linie markiert den Median der Daten. Die Definition der Whisker ist etwas komplizierter. Das Ende der linken Linie ist der kleinste beobachtete Datenpunkt, der maximal das 1,5-Fache des Interquartilsabstands vom unteren Quartil entfernt ist, also

$$\min\{x_j: x_j > Q_{lower} - 1,5 \cdot IQR, i = 1, \dots, n\}.$$

Analog ist das Ende der rechten Linie der größte beobachtete Datenpunkt, der maximal das 1,5-Fache des Interquartilsabstands vom oberen Quartil entfernt ist, also

$$\max\{x_j: x_j > Q_{upper} + 1,5 \cdot IQR, i = 1, \dots, n\}.$$

Die Idee der Whisker ist es, den Datenbereich ohne Ausreißer darzustellen. Entsprechend ist jeder Wert, der mehr als das 1,5-Fache des Interquartilsabstands von den Quartilen entfernt ist, ein Ausreißer. Sollte es Ausreißer geben, werden diese als Punkte dargestellt, wie man sie zum Beispiel bei der **Blütenblattbreite** sieht.

Im Endeffekt ist ein Boxplot also nur eine Visualisierung von nicht parametrischen deskriptiven Statistiken und von Ausreißern. Dennoch kann

man mit Boxplots auf einen Blick viel über die Daten lernen:

- Die Lage der Boxen zeigt den Bereich an, in dem sich der Großteil der Daten befindet.
- Die Größe der Boxen zeigt, wie stark der mittlere Bereich gestreut ist.
- Die Länge der Whisker zeigt den Datenbereich (ohne Ausreißer).
- Es werden die Existenz, die Anzahl und die Lage von Ausreißern dargestellt. Hierdurch erkennt man insbesondere auch, ob Ausreißer sich in der Nähe der Whisker befinden oder weit von den Daten entfernt liegen.
- Die Lage des Medians sowie die Länge der Whisker geben darüber Auskunft, ob die Daten möglicherweise symmetrisch sein könnten. Dies erkennt man, indem man überprüft, ob der Median etwa in der Mitte der Box liegt und die Whisker gleich lang sind.

Über die Irisdaten können wir zum Beispiel folgendes lernen:

- Die Merkmale haben nicht den gleichen Datenbereich. Der Datenbereich der Kelchblattlänge ist mehr als doppelt so groß wie der Bereich der Kelch- und Blütenblattbreite und ebenfalls deutlich größer als der Datenbereich der Blütenblattlänge.
- Es gibt nur wenige Ausreißer in den Daten. Nur bei der Blütenblattbreite gibt es einige Ausreißer. Diese liegen aber sehr nah an den Whiskern. Außerdem ist der Interquartilsabstand sehr klein, wodurch natürlich auch das 1,5-Fache des Interquartilsabstands sehr klein ist. Es handelt sich bei diesen Ausreißern daher nicht um nicht plausibel große Datenpunkte, sondern vielmehr um ein Artefakt infolge der Strategie, wie die Ausreißer bestimmt werden.

Wie man sieht, ist die Art der Informationen anders als bei Histogrammen und Densityplots. Histogramme und Densityplots sind gut geeignet, um die Verteilung von Daten zu verstehen und ein sehr detailliertes Bild der Daten zu bekommen. Bei Boxplots liegt der Fokus auf allgemeinen statistischen Eigenschaften der Daten. Mithilfe eines Boxplots kann man schnell statistische Merkmale der Verteilungen von mehreren Merkmalen erfassen und miteinander vergleichen.

4.3.3 Beziehungen zwischen Merkmalen

Bisher haben wir Merkmale nur unabhängig voneinander visualisiert. Bei den Boxplots haben wir zwar bereits Boxen für mehrere Merkmale in einer Visualisierung angezeigt, die Boxen haben sich jedoch nur auf ein einzelnes Merkmal bezogen. Man kann zu einem gewissen Grad die Beziehung zwischen einem kategorischen Merkmal und numerischen Merkmalen mithilfe von Boxplots darstellen. Wir können zum Beispiel die Kelchblattlänge für die verschiedenen Arten der Iris darstellen.

```
fig, ax = plt.subplots()

sns.boxplot(x=iris_df['Kelchblattlänge'], y=iris.target,
orient='h', ax=ax)

ax.set_title('Boxplots der Kelchblattlänge für jede Art')

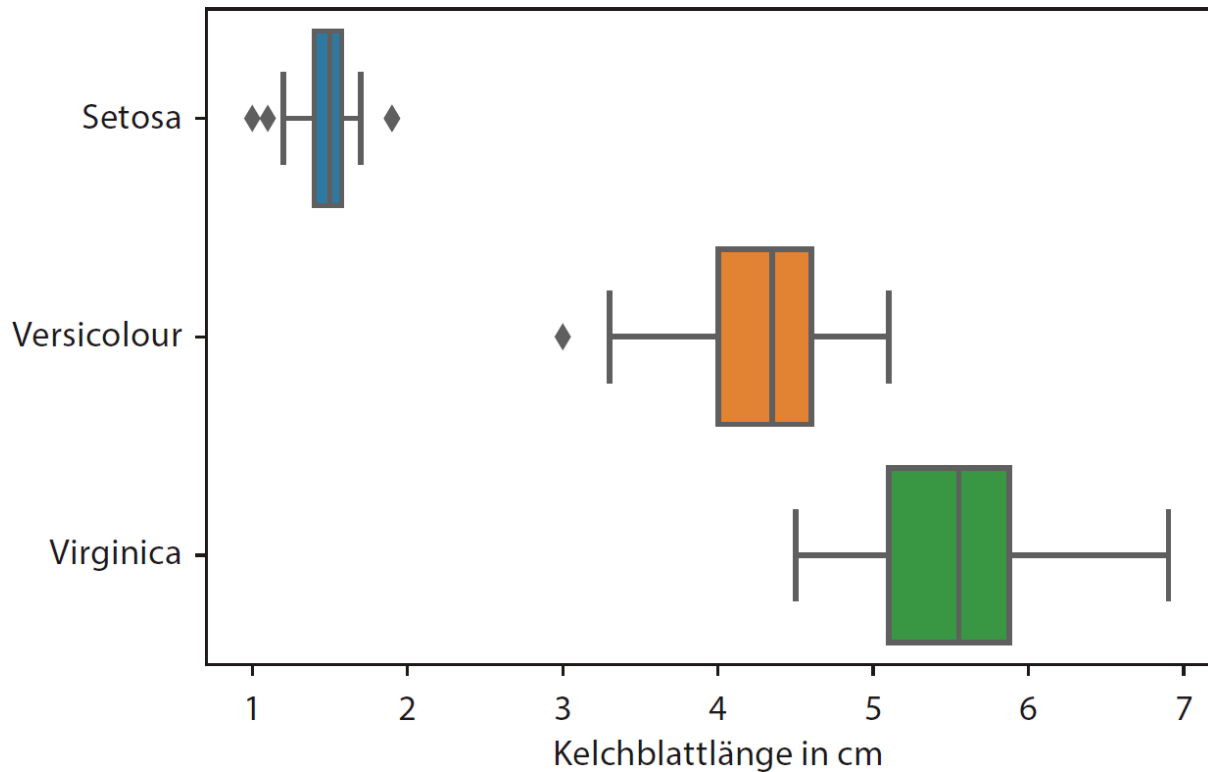
ax.set_xlabel('Kelchblattlänge in cm')

ax.set_yticks((0,1,2))

ax.set_yticklabels(('Setosa', 'Versicolour', 'Virginica'))

plt.show()
```

Boxplots der Kelchblattlänge für jede Art



Aus diesem Boxplot können wir viel über die Beziehung der Kelchblattlänge zu den Arten der Iris lernen.

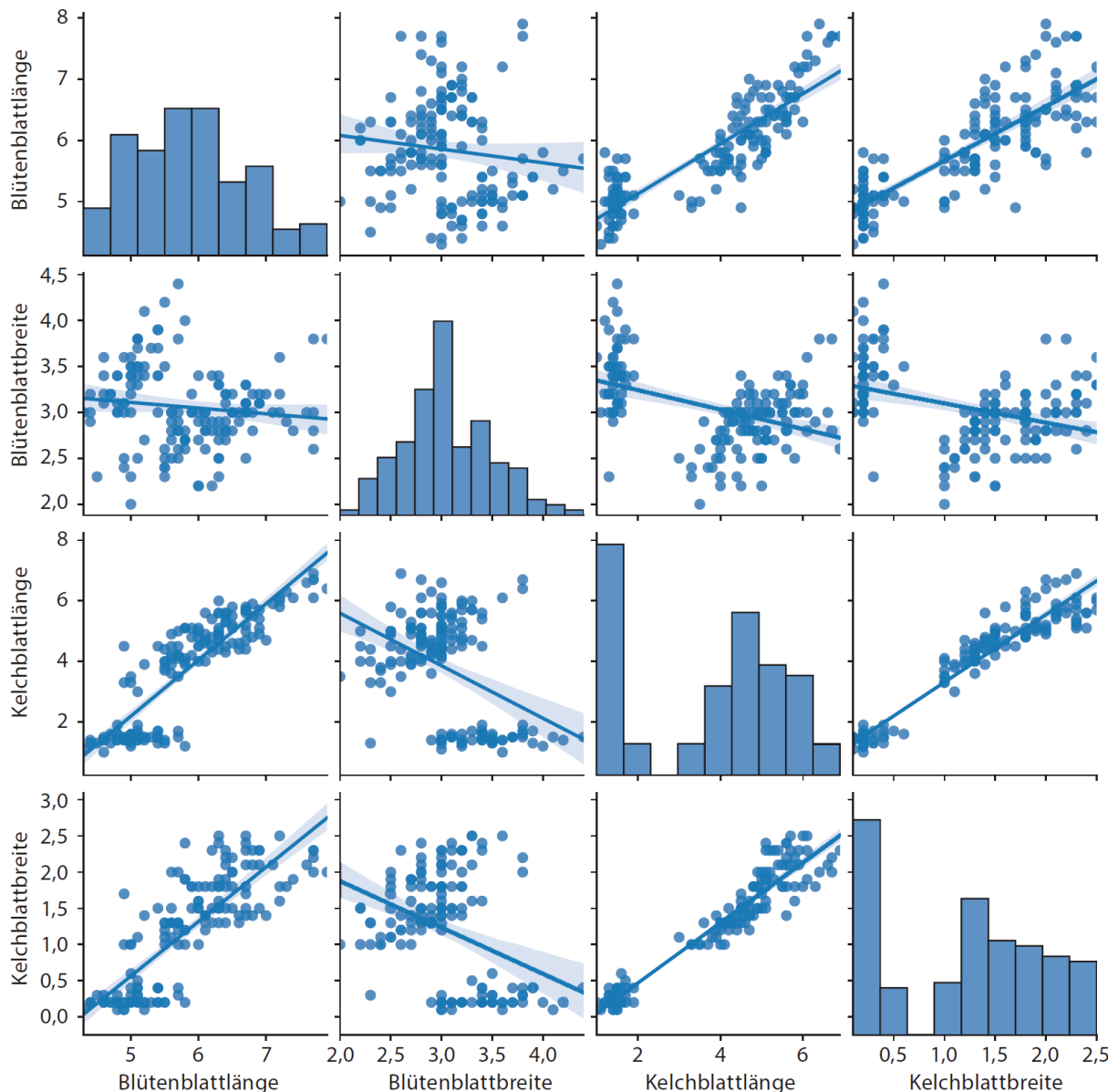
- Die Kelchblattlänge von Setosa ist immer kleiner als 2,5cm. Die anderen Arten haben immer eine Kelchblatlänge, die größer als 2,5cm ist.
- Die Virginica hat immer eine Kelchblattlänge größer als 4,3cm. Aus diesen beiden Beobachtungen folgt, dass alle Iris mit einer Kelchblattlänge zwischen 2,5cm und 4,3cm Versicolours sein müssen.
- Die Kelchblattlänge der Versicolour ist immer kleiner als 5,1cm. Daraus folgt also, dass alle Iris mit einer Kelchblattlänge größer als 5,1cm Virginica sein müssen.
- Iris mit einer Kelchblattlänge zwischen 4,3cm und 5,1cm könnten sowohl Versicolour als auch Virginica sein.

Der Nachteil von Boxplots ist, dass man Merkmale nur anhand von wenigen statistischen Eigenschaften vergleichen kann. Außerdem sind die Vergleiche beschränkt, da man nur ein kategorisches mit einem numerischen Merkmal vergleichen kann. Außerdem sollte das kategorische Merkmal nicht zu viele Kategorien haben, da es sonst sehr viele Boxen geben würde.

Eine weitere Möglichkeit, sich Beziehungen zwischen Merkmalen anzuschauen, sind *paarweise Scatterplots*.

```
sns.pairplot(iris_df, kind='reg')
```

```
plt.show()
```



In einem paarweisen Scatterplot hat man nicht nur einen einzigen Plot, sondern eine Sammlung von Plots, die als Matrix organisiert sind. Auf der Diagonale findet man die Histogramme der Merkmale. Die anderen Plots zeigen die paarweisen Beziehungen zwischen je zwei Merkmalen. Die Geraden geben den linearen Zusammenhang zwischen den Daten wieder, wie wir es schon von

Anscombes Quartett kennen. Die Schattierung um die Geraden zeigt zusätzlich noch die Unsicherheit dieses Zusammenhangs an. Je näher die Datenpunkte an den Geraden liegen, desto kleiner ist die Unsicherheit und desto größer die Wahrscheinlichkeit, dass es tatsächlich einen linearen Zusammenhang gibt. Wenn die Steigung der Geraden negativ ist (abwärts), dann ist die Korrelation zwischen den Merkmalen negativ. Wenn die Steigung positiv ist, dann ist Korrelation zwischen den Merkmalen positiv. Die Werte welcher Merkmale auf der x-Achse bzw. y-Achse aufgetragen sind, kann man am unteren bzw. linken Rand des Plots erkennen. Der zweite Plot in der ersten Zeile zeigt zum Beispiel die Beziehung zwischen der Blütenblattbreite auf der x-Achse und der Blütenblattlänge auf der y-Achse.

Ein paarweiser Scatterplot ermöglicht es uns, viel über die Beziehungen zwischen Merkmalen zu lernen, aber auch über einzelne Merkmale über die Histogramme auf der Diagonale.

- Die Blütenblattbreite scheint normalverteilt zu sein, die Kelchblattbreite sieht eher wie ein Mix von zwei Normalverteilungen aus.
- Die Beziehung zwischen der Blütenblattbreite und der Blütenblattlänge sieht zufällig aus. Dies erkennt man daran, dass sich die Punkte über die komplette Fläche des Plots verteilen.
- Die Kelchblattbreite hat eine positive Korrelation mit der Kelchblattlänge. Dies bedeutet, dass wenn sich die Länge erhöht, sich die Breite ebenfalls erhöht und umgekehrt. Diese Korrelation scheint linear zu sein, was man daran erkennt, dass alle Datenpunkte sehr nah an der Regressionslinie liegen. Es gibt also ein festes Verhältnis zwischen der Kelchblattbreite und Kelchblattlänge.
- Man kann in den Plots der Kelchblätter zwei Gruppen in den Daten erkennen: eine kleinere Gruppe mit niedrigen Werten und eine größere Gruppe mit höheren Werten. Zwischen den Gruppen ist eine größere Lücke. Ähnliche Gruppen findet man in nahezu allen Plots, wenn auch nicht so eindeutig.

Bemerkung:

Wir betrachten das Konzept der Korrelation hier nicht im Detail. Die Idee ist aber relativ einfach: Man nennt zwei Merkmale X und Y *positiv korreliert*, wenn bei steigendem Wert von X in der Regel auch der Wert von Y steigt. Je häufiger dies der Fall ist, desto stärker ist die Korrelation. Umgekehrt nennt man eine Korrelation *negativ*, wenn bei fallendem Wert von X in der Regel auch der Wert von Y fällt. Eine *lineare Korrelation* bedeutet zusätzlich noch, dass es ein Muster gibt, sodass $X \approx a + b \cdot Y$ für zwei Konstanten $a, b \in \mathbb{R}$. Die Korrelation wird

häufig mithilfe von *Korrelationskoeffizienten* gemessen, deren Werte zwischen [-1,1] liegen. Ein Korrelationskoeffizient von 0 bedeutet, dass zwei Merkmale nicht korreliert sind. Je näher die Werte an -1/1 sind, desto stärker ist die negative/positive Korrelation.

Für kategorische Variablen wie die Art der Iris kann man noch weitere Informationen aus einem paarweisen Scatterplot holen, indem man die Kategorien als Farben nutzt. Zusätzlich kann man dann auch noch Densityplots pro Kategorie auf der Diagonale anzeigen.

```
iris_df['Art'] = iris.target
```

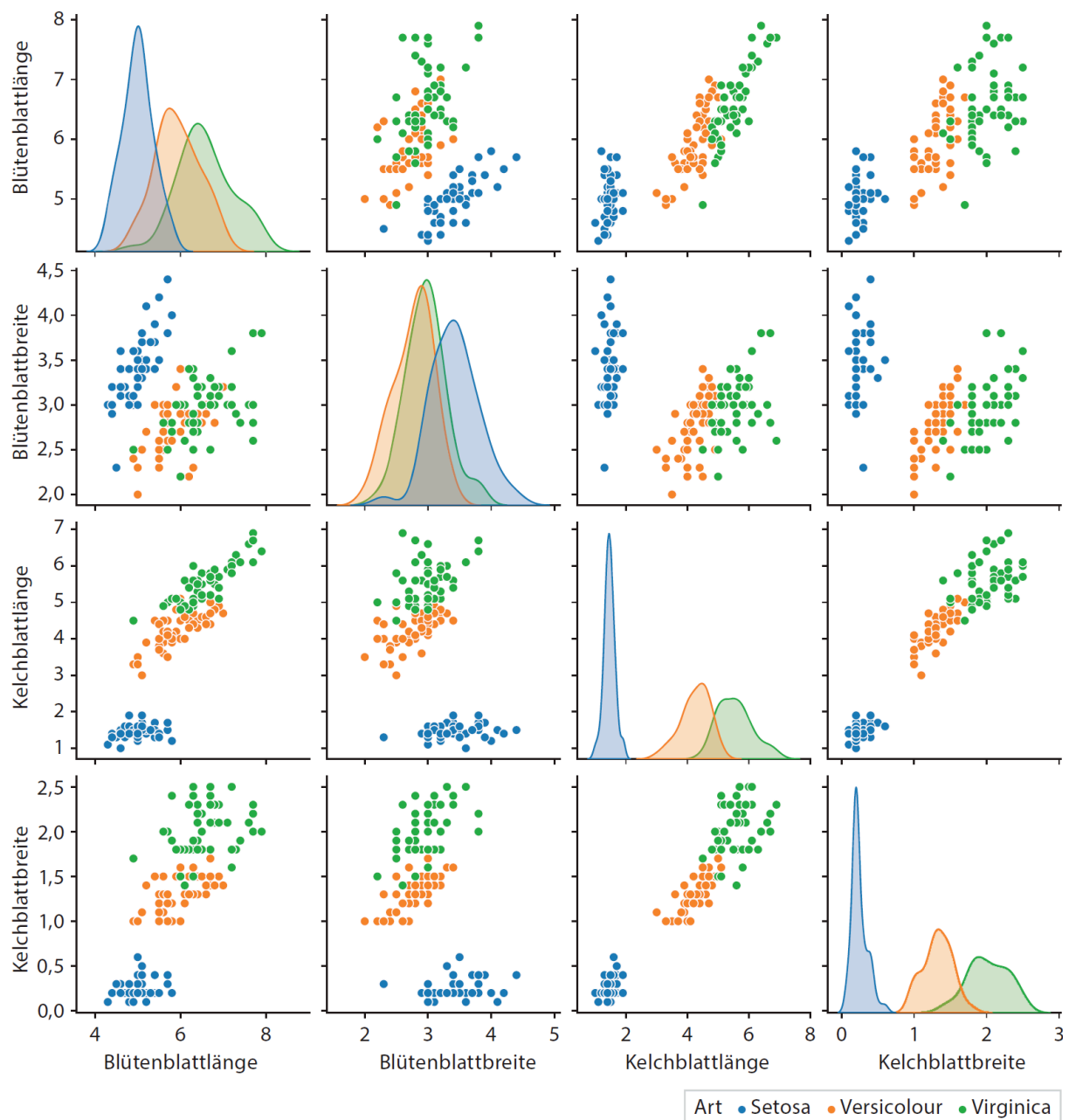
```
iris_df.loc[iris_df['Art']==0, 'Art'] = 'Setosa'
```

```
iris_df.loc[iris_df['Art']==1, 'Art'] = 'Versicolour'
```

```
iris_df.loc[iris_df['Art']==2, 'Art'] = 'Virginica'
```

```
sns.pairplot(iris_df, hue='Art')
```

```
plt.show()
```



Durch die Farben können wir die Bedeutung der Gruppen erkennen, die wir oben schon gesehen haben: Die kleinere Gruppe sind die Setosas, während die größere Gruppe aus den Versicolours und Virginicas besteht. Diese Trennung ist ähnlich zu dem, was wir bereits im Boxplot gesehen haben. Wir erkennen auch, dass man die Arten nur schwer mithilfe der Blütenblätter unterscheiden kann, insbesondere nicht zwischen Versicolours und Virginicas. Man sieht aber auch, dass man die drei Arten fast perfekt anhand der Kelchblätter unterscheiden kann.

Ein mögliches Problem von Scatterplots ist, dass man häufig sehr viele Datenpunkte in einem kleinen Bereich hat, sodass sich die Daten überlappen.

Man spricht dann auch von einem *dichten* Bereich. Wenn man solche Daten als Scatterplot darstellt, sieht man häufig nur einen großen Fleck.

```
np.random.seed(0)

n = 10000

x = np.random.standard_normal(n)

y = 2.0 + 3.0 * x + 4.0 * np.random.standard_normal(n)

xmin = x.min()

xmax = x.max()

ymin = y.min()

ymax = y.max()

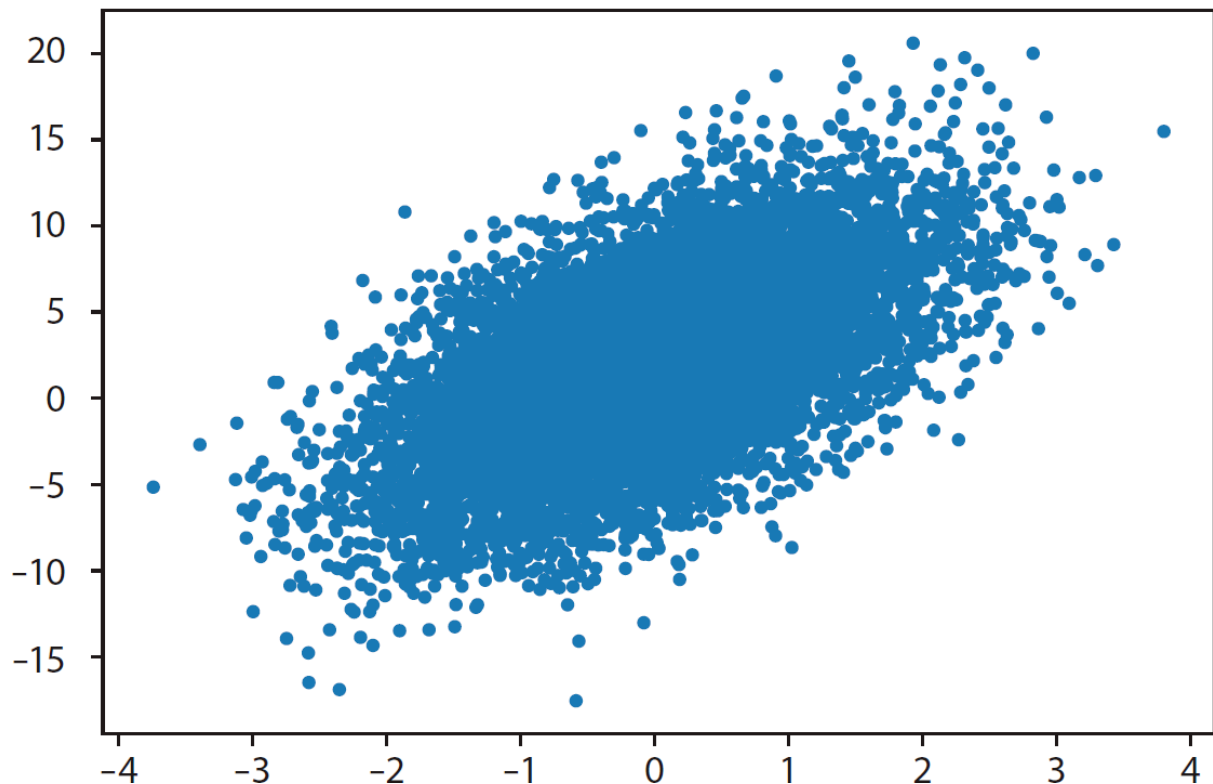
plt.figure()

plt.title('Scatterplot von 10000 Datenpunkten')

plt.plot(x, y, marker='.', linestyle='none')

plt.show()
```

Scatterplot von 10000 Datenpunkten



Innerhalb der dichten Region kann man die Struktur der Daten nicht erkennen. Im obigen Beispiel könnten die Daten in der Region zum Beispiel gleichverteilt, normalverteilt oder eine Mischung aus mehreren Verteilungen mit unterschiedlichen Peaks sein. Man verliert also wegen der Dichte wertvolle Informationen über die Daten. Es gibt zwei Möglichkeiten, wie man dieses Problem lösen kann. Eine schnelle und sehr einfache Lösung besteht darin, die Datenpunkte transparent zu machen.

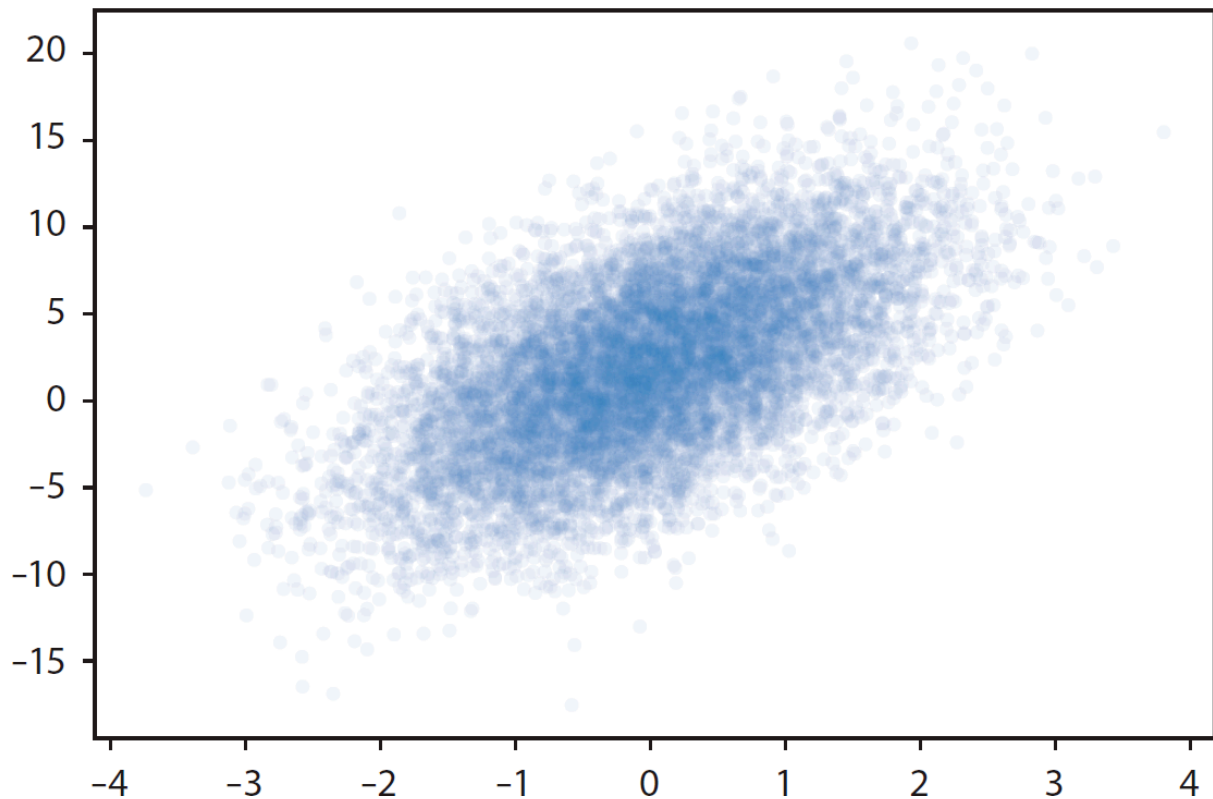
```
plt.figure()
```

```
plt.title('Scatterplot von 10000 transparenten Datenpunkten')
```

```
plt.plot(x, y, marker='.', linestyle='none', alpha=0.05)
```

```
plt.show()
```

Scatterplot von 10000 transparenten Datenpunkten



Durch die Transparenz kann man die Struktur erkennen und sieht, dass die Daten normalverteilt sind mit dem Mittelwert im Zentrum der dichten Region. Der Nachteil dieser Lösung ist, dass man immer noch nicht sieht, wie viele Daten genau in einer Region liegen, da es keine Farbskala gibt. Wenn dies ebenfalls wichtig ist, sollte man *Hexbinplots* verwenden. Hexbinplots sind mehr oder weniger zweidimensionale Histogramme: Die Daten sind in sechseckige Bins unterteilt und die Farbe zeigt, wie viele Daten in einer Region liegen.

```
fig, ax = plt.subplots()

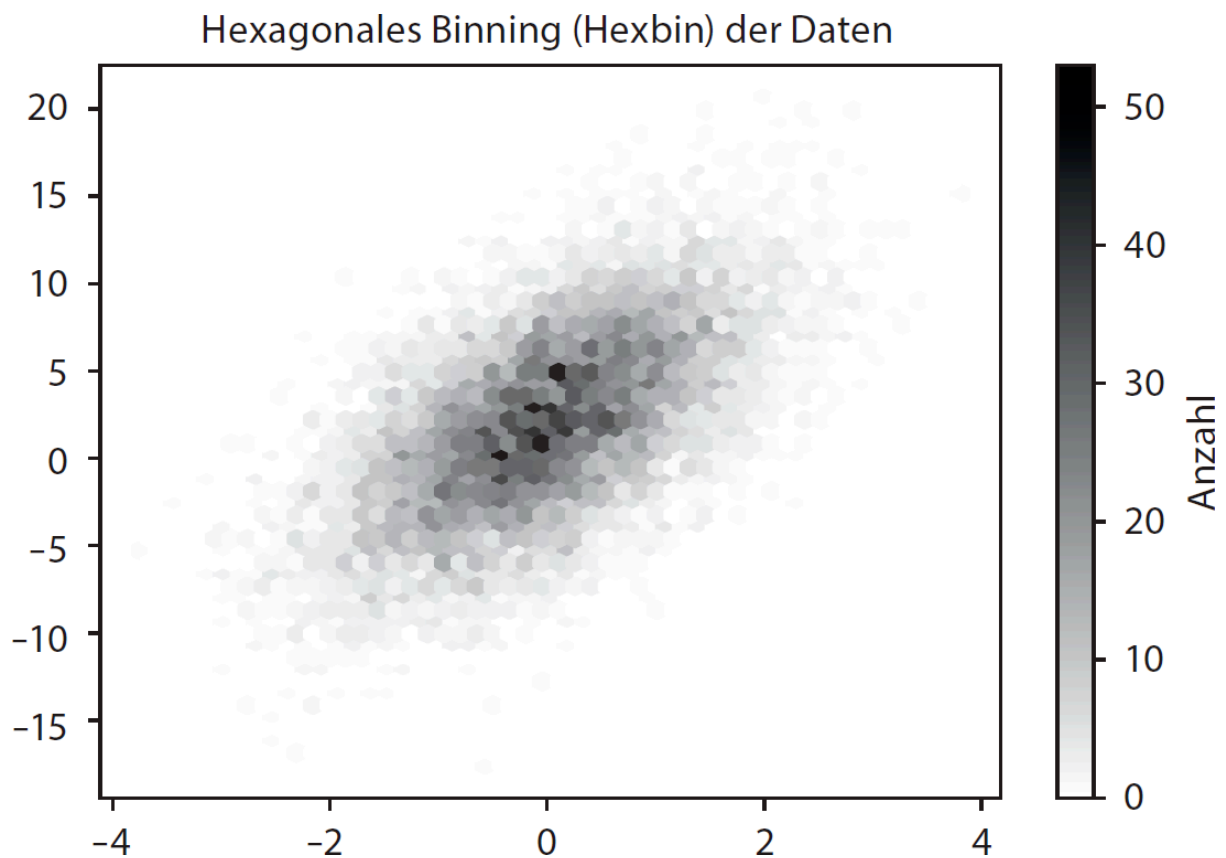
hb = ax.hexbin(x, y, gridsize=50, cmap='binary')

cb = fig.colorbar(hb)

cb.set_label('Anzahl')

ax.set_title("Hexagonales Binning (Hexbin) der Daten")
```

```
plt.show()
```



Im Hexbinplot können wir jetzt nicht nur die Struktur der Daten erkennen, sondern auch ablesen, wie viele Datenpunkte sich in der dichten Region befinden.

Die letzte Visualisierung, die wir uns bezüglich des Zusammenhangs von Merkmalen anschauen wollen, zeigt die Korrelationen zwischen den Merkmalen. Wir haben uns Korrelationen bereits im Zusammenhang mit Scatterplots durch die Regressionsgeraden angesehen. Dies ist jedoch relativ aufwendig, insbesondere wenn es viele Merkmale gibt. Der Aufwand steigt quadratisch mit der Anzahl der Merkmale. Eine effizientere Methode, einen Überblick über die Korrelationen zu bekommen, ist eine *Korrelations-Heatmap*.

```
# Compute the correlation matrix that contains the correlation coefficients
```

```
corr = iris_df.corr()
```

```
# we use customized colors
```

```

fig, ax = plt.subplots()

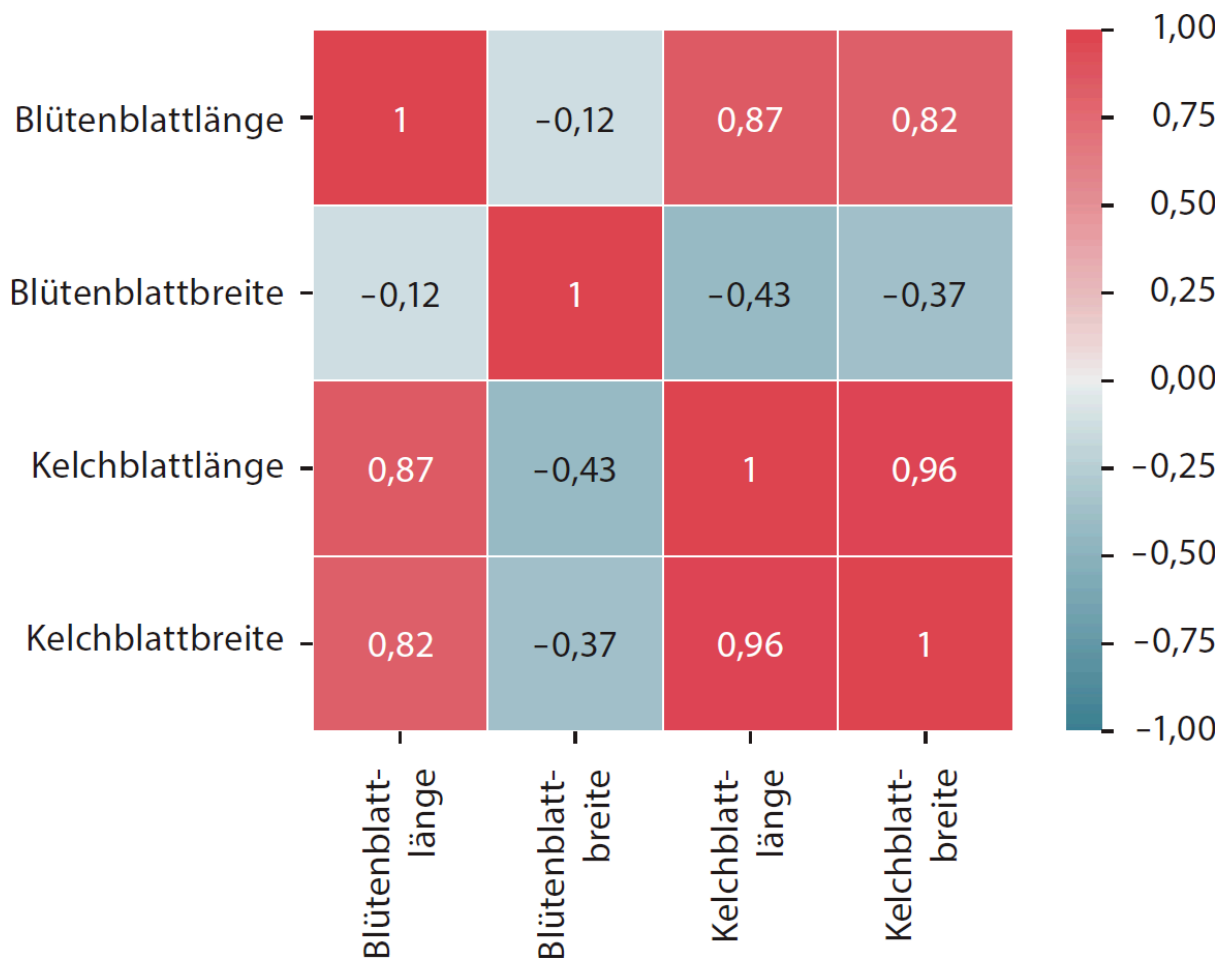
cmap = sns.diverging_palette(220, 10, as_cmap=True)

sns.heatmap(corr, square=True, linewidths=.5,

            vmin=-1.0, vmax=1.0, cmap=cmap, annot=True, ax=ax)

plt.show()

```



Eine Korrelations-Heatmap visualisiert die Werte von Korrelationskoeffizienten zwischen den Merkmalen. Rot heißt, dass es eine positive Korrelation gibt, Blau steht für negative Korrelation. Je schwächer die Farbe, desto schwächer die Korrelation. Konsequenterweise heißt Weiß also, dass es keine Korrelation gibt. Auf der Diagonale gibt es immer eine perfekte positive Korrelation, da selbstverständlich jedes Merkmal mit sich selbst korreliert ist. Außerdem sehen

wir die Bestätigung unserer Erkenntnisse über die Korrelationen aus den paarweisen Scatterplots: Es gibt eine starke positive Korrelation zwischen der Breite und Länge der Kelchblätter und keine Korrelation bei den Blütenblättern. Man sieht jedoch noch weitere Korrelationen, zum Beispiel eine negative Korrelation zwischen der Kelchblattlänge und der Blütenblattbreite.

Bemerkung:

Selbstverständlich kann man auch andere Farben zur Darstellung der Korrelationskoeffizienten nutzen. Wichtig ist jedoch, dass man eine Farbskala wählt, bei der man die Stärke der Korrelation gut erkennen kann und man gut zwischen positiver und negativer Korrelation unterscheiden kann. Das bedeutet insbesondere auch, dass man Werte in der Nähe von null gut erkennt.

4.3.4 Scatterplots für hochdimensionale Daten

Häufig hat man sehr viele Dimensionen, sodass die Analyse von paarweisen Scatterplots schwierig wird. In diesem Fall kann man Methoden zur *Dimensionsreduktion* anwenden. Die Idee hierbei ist es, die Korrelationen zwischen den Merkmalen auszunutzen und neue Merkmale zu berechnen, die unkorreliert sind. Bei den neuen Merkmalen beinhalten dann möglichst wenige Merkmale möglichst viel der Informationen aus den ursprünglichen Daten. Ein beliebtes Verfahren hierfür ist die *Hauptkomponentenanalyse* (engl. *principal component analysis/PCA*). Bei der PCA wird der Raum von reellwertigen Merkmalen so transformiert, dass das erste Merkmal möglichst viel der Varianz der Daten erklärt, das zweite Merkmal den zweit größten Anteil und so fort. Die neuen Merkmale nennt man auch die *Hauptkomponenten* (engl. *principal components*) der Daten. Da die Varianz eines Merkmals direkt mit der vorhandenen Information verwandt ist, beinhaltet also die erste Hauptkomponente die meiste Information, die zweite Hauptkomponente die zweitmeiste Information und so fort. Das liegt ganz einfach daran, dass konstante Merkmale keine Informationen beinhalten. Für die Irisdaten können wir mit der PCA die vier Merkmale auf die ersten zwei Hauptkomponenten reduzieren und mit der Farbe die Art der Iris darstellen.

```
from sklearn import decomposition
```

```
pca = decomposition.PCA()
```

```
pca.fit(iris.data)
```

```

iris_pca = pca.transform(iris.data)

fig, ax = plt.subplots()

for i, label in enumerate(iris.target_names):

    # we need to plot each type of iris on its own to get
    colors with a legend

    ax.scatter(iris_pca[iris.target==i,0],
               iris_pca[iris.target==i,1],

label=label)

ax.set_xlabel('1. Hauptkomponente (%.2f Varianz)' %
pca.explained_variance_ratio_[0])

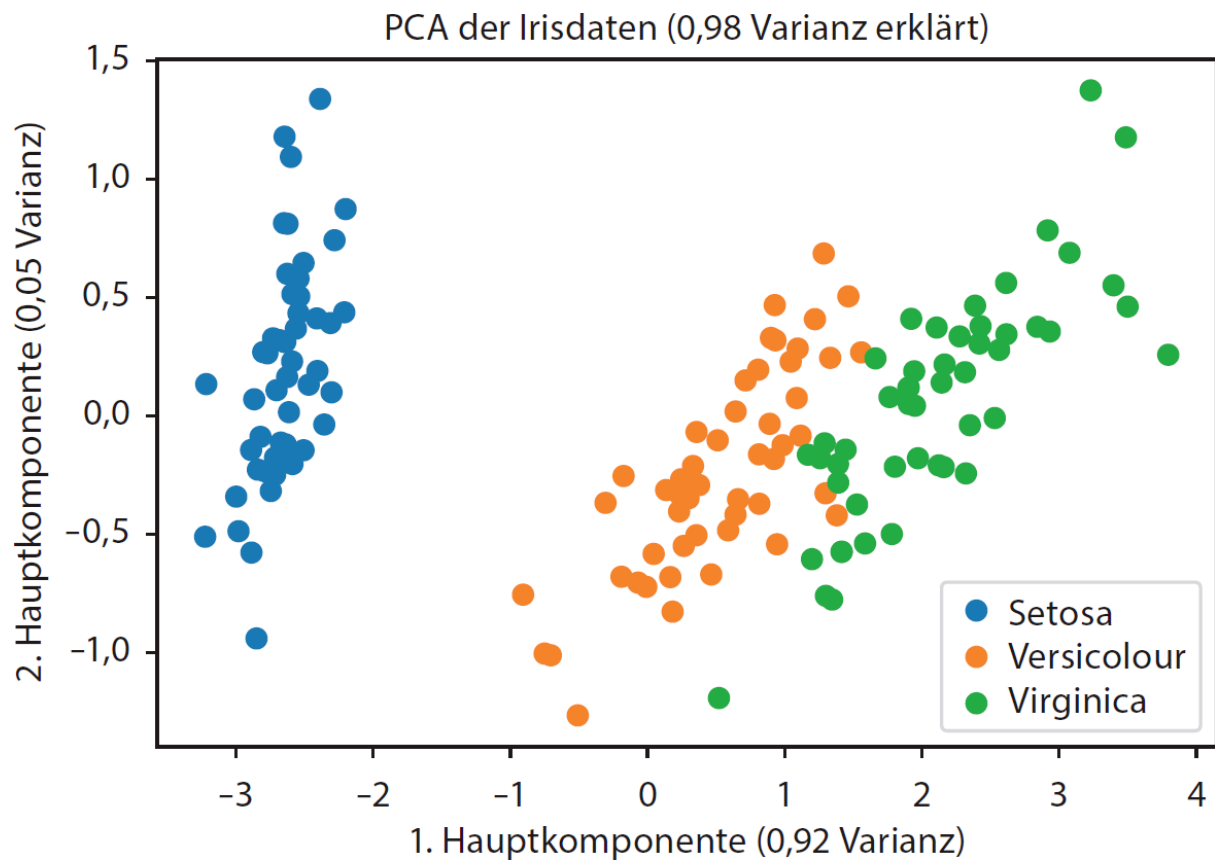
ax.set_ylabel('2. Hauptkomponente (%.2f Varianz)' %
pca.explained_variance_ratio_[1])

ax.set_title('PCA der Irisdaten (%.2f Varianz erklärt)' %
sum(pca.explained_variance_ratio_[:2]))

plt.legend()

plt.show()

```



Die zwei Hauptkomponenten erklären bereits 98% der Varianz der Daten, also beinahe alle Informationen. Man sieht auch, dass sich die Arten der Iris sehr gut unterscheiden lassen und separate Gruppen in den Daten darstellen, auch wenn die Versicolor und die Virginica sehr nah beieinander liegen. Wenn die PCA weniger Varianz der Daten erklärt, kann es sein, dass eine Trennung von Gruppen vorhanden ist, auch wenn man sie nicht sieht: Man bräuchte dann nur die restliche Information, um dies zu erkennen.

Die PCA ist ein mächtiges Werkzeug, um die Dimensionen von Daten zu reduzieren, ohne den Informationsgehalt zu verringern, und insbesondere auch um hochdimensionale Daten darzustellen. Dennoch gibt es zwei wesentliche Nachteile, die auch ähnliche Verfahren betreffen. Das erste Problem ist, dass nicht klar ist, *wofür* die Hauptkomponenten stehen. Beim paarweisen Scatterplot war klar, welche Beziehungen gezeigt werden und somit auch welche Information. Die Hauptkomponenten sind Linearkombinationen der Merkmale, wie diese kombiniert werden, ist jedoch nicht offensichtlich. Hier könnte man höchstens noch die Transformation der PCA genauer betrachten, um an diese Information zu kommen. Das ist aber sehr komplex. Daher eignet sich die PCA in der Regel nur dafür, zu analysieren, ob Daten entsprechender Eigenschaften so aussehen wie erwartet oder erhofft (zum Beispiel ob die Arten von Iris gruppiert sind), es ist jedoch nicht ohne Weiteres möglich, die Verteilung der Datenpunkte zu verstehen. Der zweite Nachteil ist, dass die

Information möglicherweise nicht mit dem Ziel der Analyse zusammenhängt. Wenn ein Fragebogen zur Kreditwürdigkeit die Lieblingszahl abfragen würde, hätte dies keinen Bezug zum eigentlichen Ziel der Analyse und die Varianz dieser Information wäre entsprechend auch irrelevant. Daher sollte man bei der PCA vorher sicherstellen, dass nur Merkmale benutzt werden, die mit dem Ziel der Analyse zusammenhängen.

Bemerkung:

Für die PCA interpretiert man die Daten als Matrix A , wobei jede Zeile der Matrix eine Instanz ist. Anschließend muss man die folgenden Schritte durchführen:

- Zuerst müssen die Spalten der Matrix A so skaliert werden, dass der Erwartungswert null ist (A_0). Hierdurch werden Skaleneffekte zwischen den Merkmalen vermieden und die folgenden Berechnungen vereinfacht.
- Anschließend wird die Kovarianzmatrix Cov_A von A_0 berechnet. Wir berechnen also die Kovarianzen, um die Korrelation zwischen den Merkmalen auszudrücken.
- Nun kann man die Eigenwerte und Eigenvektoren von Cov_A berechnen. Die Eigenvektoren müssen anhand ihrer Eigenwerte absteigend sortiert werden, sodass jeder Vektor die Spalte einer Matrix Eig ist. Wenn man die Eigenwerte so reskaliert, dass ihre Summe eins ergibt, erhält man den Anteil der erklärten Varianz der jeweiligen Hauptkomponente.
- Durch die Matrixmultiplikation von Eig und A_0 erhält man die Hauptkomponenten von A .

4.3.5 Zeitliche Trends

Die bisherigen Visualisierungen sind alle davon ausgegangen, dass die einzelnen Datenpunkte nicht direkt miteinander zusammenhängen. Häufig gibt es jedoch Zusammenhänge, insbesondere bei Daten, die über einen gewissen Zeitraum erfasst worden sind. Als Beispiel betrachten wir jetzt Fluggastdaten einer US Airline aus den Jahren 1940 bis 1960. Die Werte geben die monatliche Anzahl von Passagieren an.

```
!head data/AirPassengers.csv
```

```
Month, #Passengers
```

```
1949-01, 112
```

1949-02,118

1949-03,132

1949-04,129

1949-05,121

1949-06,135

1949-07,148

1949-08,148

1949-09,136

Wir könnten solche Daten auch unabhängig von der Zeit betrachten, zum Beispiel als Histogramm.

```
air_passengers_df = pd.read_csv("data/AirPassengers.csv",
header = 0,

parse_dates = [0], names = ['Month', 'Passengers'], index_col
= 0,

squeeze=True)

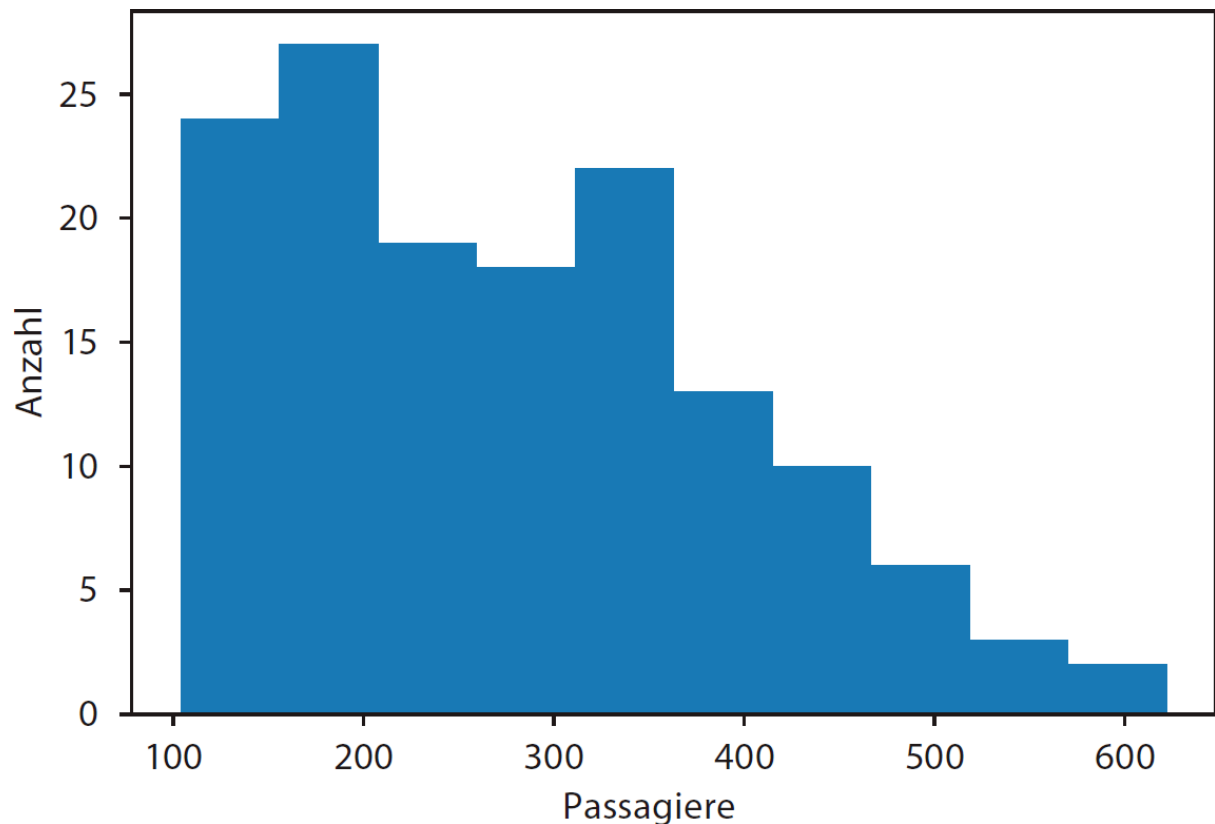
fig, ax = plt.subplots()

ax.hist(air_passengers_df)

ax.set_xlabel('Passagiere')

ax.set_ylabel('Anzahl')
```

```
plt.show()
```



Es ist nicht so, dass Histogramme bei über die Zeit verteilten Daten keine Informationen liefern würden. Man erkennt, dass es maximal 600 Passagiere in einem Monat gab und dass es circa 20 Monate mit jeweils 100–150, 150–200, ..., bzw. 300–350 Passagieren gab. Was man jedoch nicht erkennt ist die Entwicklung der Passagierzahlen über die Zeit. Hier könnten wir zum Beispiel einfach einen Scatterplot betrachten, bei dem die y-Achse die Passagierzahlen und die x-Achse die Monate anzeigt.

```
fig, ax = plt.subplots()
```

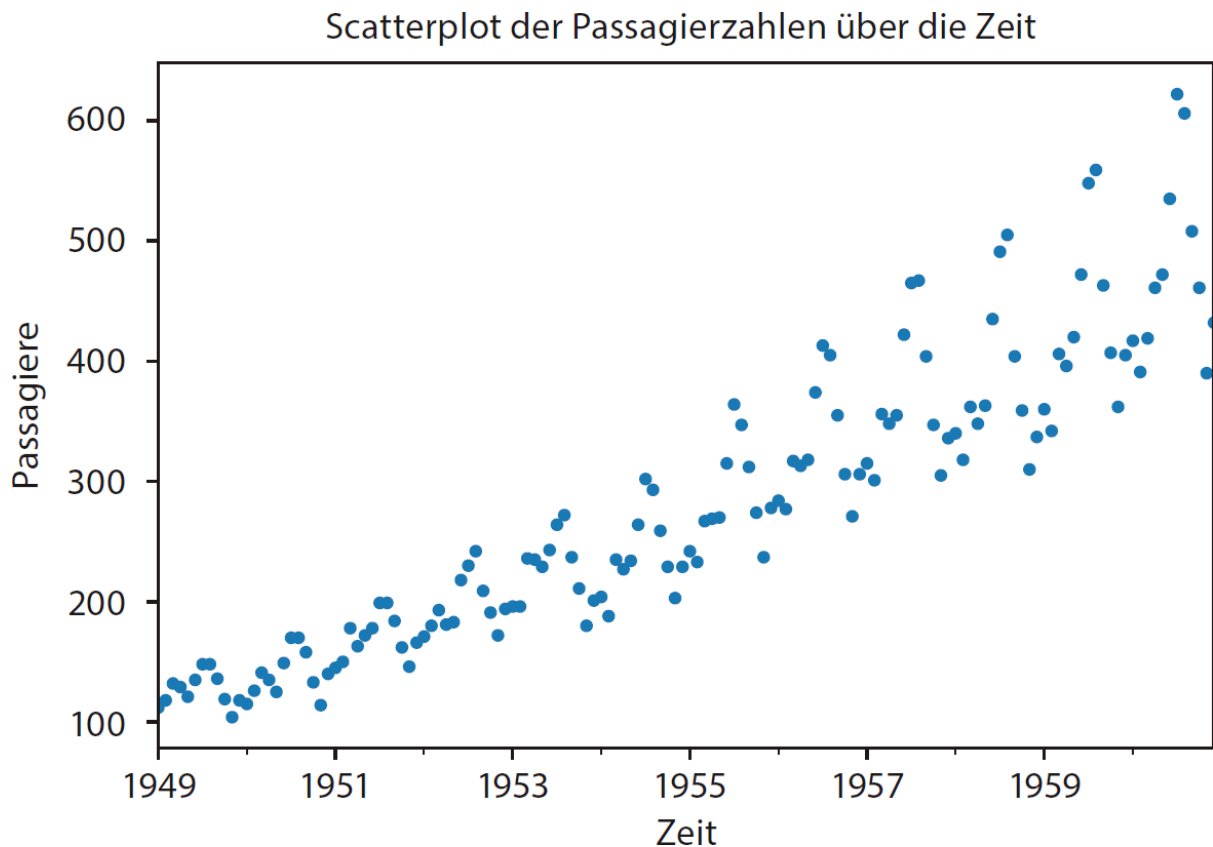
```
air_passengers_df.plot(linestyle='', marker='.', ax=ax)
```

```
ax.set_title("Scatterplot der Passagierzahlen über die Zeit")
```

```
ax.set_xlabel('Zeit')
```

```
ax.set_ylabel('Passagiere')
```

```
plt.show()
```



In diesem Scatterplot erkennt man den zeitlichen Verlauf der Daten. Man sieht einen klaren Trend, dass die Passagierzahlen mit der Zeit steigen. Wie sich die Passagierzahlen von Monat zu Monat verändern, erkennt man jedoch nicht ohne Weiteres. Hierfür müssen wir – im wahrsten Sinne des Wortes – die Punkte miteinander verbinden, um ein *Liniendiagramm* zu erhalten.

```
fig, ax = plt.subplots()
```

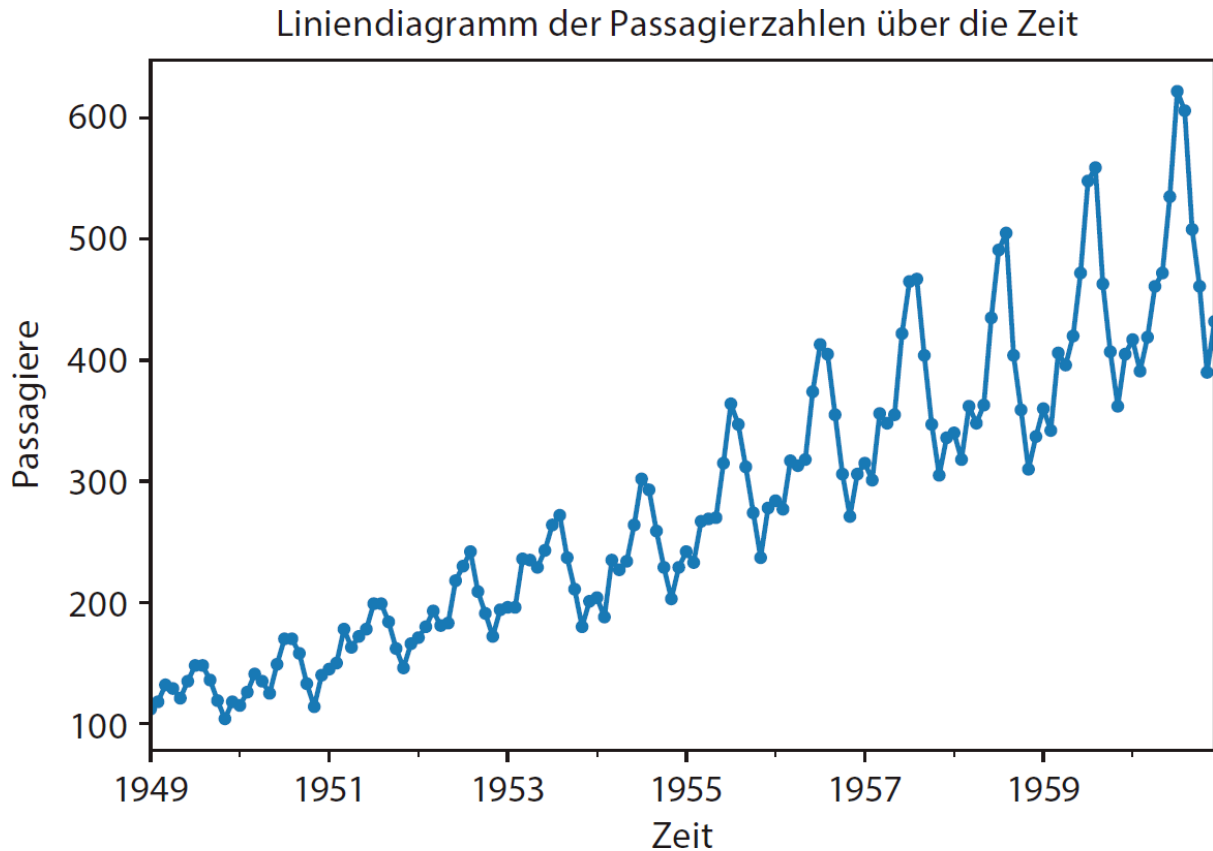
```
air_passengers_df.plot(marker='.', ax=ax)
```

```
ax.set_title("Liniendiagramm der Passagierzahlen über die  
Zeit")
```

```
ax.set_xlabel('Zeit')
```

```
ax.set_ylabel('Passagiere')
```

```
plt.show()
```



Im Liniendiagramm können wir jetzt sowohl den Trend als auch die Veränderungen von Monat zu Monat sehen. Man erkennt auch ein Muster, das sich jährlich wiederholt.

4.4 Übung

In der ersten praktischen Übung geht es darum, selbst einige Daten zu erkunden. Hierzu benutzen wir wieder die Bostondaten, die wir vorher schon in diesem Kapitel verwendet haben.

Deskriptive Statistiken

Untersuchen Sie die Bostondaten mithilfe von deskriptiven Statistiken. Berechnen Sie die Lage durch das arithmetische Mittel und den Median, die Variabilität durch die Standardabweichung und den IQR sowie die Datenbereiche der Merkmale. Das Ziel dieser Aufgabe ist aber nicht die Berechnung dieser Werte, sondern ihre Interpretation. Überlegen Sie sich zum

Beispiel, was man über das Merkmal CRIM über das arithmetische Mittel und den Median lernt.

Visualisierungen einzelner Merkmale

Schauen Sie sich die Merkmale **zn** und **indus** mithilfe von Histogrammen und Densityplots (mit und ohne Rug) an. Was lernen Sie über diese Daten? Was sind die Vor- und Nachteile der verschiedenen Visualisierungen?

Paarweise Beziehungen

Betrachten Sie die paarweisen Beziehungen der Bostondaten. Benutzen Sie hierfür Scatterplots und eine Korrelations-Heatmap. Was lernen Sie über die Beziehungen aus den Daten? Beachten Sie hierbei insbesondere auch, ob Sie ethisch problematische Eigenschaften finden.

Index

A

Abhängige Variable 215

Add-One Glättung 64

Ähnlichkeitsmaß 96

Akitake Information Criterion 117

Algorithmen, Kategorien 41

Alpha Go 10

Alternativhypothese 268

ANOVA (F-Test) 273

Anscombes Quartett 48, 55

Apache Hadoop 292

 Architektur 292

Apache Spark 305

 Architektur 306

Apriori-Algorithmus 85

Apriori-Eigenschaft 91

Area Under the Curve 154

ARIMA-Modell 248

Arithmetisches Mittel 48

ARMA-Modell 246, 247, 248

AR-Modell 246

Artificial Intelligence *siehe Künstliche Intelligenz*

Assoziation 83

Assoziationsanalyse 83, 84

Assoziationsregeln 21, 41, 83, 84

 Bewertung von 93

Ausführungszeit 137, 206

Autokorrelation 234, 235, 242

 mit ARMA 242

 partielle 242

Autonomes Fahren 10

AutoRegressives-Modell 246

B

Bagging 174

Bag-of-Words 261, 264, 290

Bandbreitenparameter 61

Bayesian Information Criterion 115, 117

Bayesian Statistics 276, 282

Bedingte Entropie 167

Bedingung 84

Bewertung von Assoziationsregeln 93

Big Data 1, 7, 286, 288

 Definition 2

 drei Vs 6

 innovative Informationsverarbeitungsmethoden 6

 und Data Science 1

 Value 7

 Variety 4

 Velocity 3

 Veracity 7

Big Data Processing 285

Big-Data-Infrastruktur 288

Bimodale Daten 50

Bin 59, 63

Binäre Gütemaße 150

Binäre Klassifikation 145

Boosting 197

Bootstrap Aggregating 174

Bootstrap Sample 173

Bootstrap Sampling 310
Bostondaten 59
Box-Jenkins-Verfahren 235, 249
Boxplot 67, 69
Box-Whisker-Plot 68
Brückenpunkte 127

C

Centroid 98, 139
Centroidbasierte Clusteralgorithmen 98
Chance 174, 176
Chebychev-Ungleichung 52
Chebyshev-Abstand 97
Class Level Imbalance 309
Cluster 95, 96, 103, 119, 131
 Varianz innerhalb eines 104
Clusteralgorithmen 98, 100, 140
 centroidbasierte 98
 dichtebasierte 118
 hierarchische 129
Clusteranalyse 21, 95
Clusterformen 134
Clustering *siehe auch Clustern* 95
 dichtebasiertes 118
 Konzept 96
Clustern 41, 95, 310
 hierarchisches 128
Comma-Separated Values 45
Complete Linkage Clustering 129
Compute Layer 286
Confidence 87, 88, 93
Confusion Matrix 148, 157
 binäre 149, 156
Convolutional Layer 191, 193
Convolutional Neural Network 191

D

Data Parallelism 286

Data Privacy 31

Data Science

als mathematische Modellierung 8

Beispielanwendungen 10

Konzepte 7

Data Scientist 15, 16, 20, 26, 29

Fähigkeiten 11

Kategorisierung 11

Data-Science-Projekt 13

Discovery Phase 15

Modellerstellung 24

Operationalisierung 25

Phase der Datenvorbereitung 18

Data-Science-Projekt (Fortsetzung)

Rollen 26

Anwenderin 27

Data Scientist 29

Datenbankadministratorin 29

Dateningenieurin 28

Projektmanagerin 28

Projektsponsorin 27

Data-Science-Prozess 13

generischer 14

Modellplanung 21

Phasen 14

Tailoring 14

Daten 38

bimodale 50

Erkundung von 45

Hauptkomponenten 77

hochdimensionale 77, 97

kategorische 50

Lage der 47

- Lernen der 45
- Merkmale der 57
- Mittelwert 48
- Modus der 50
- multimodale 50
- quasistrukturierte 4
- semistrukturierte 4
- strukturierte 4
- unstrukturierte 4
- Variabilität 50, 51
- Varianz der 78
- Verständnis der 53
 - zeitlicher Verlauf 79
- Daten mit Ausreißer 49
- Datenanalyse 33
- Datenbereich 52
 - Maximum 52
 - Minimum 52
- Datenformat 46
- Datenlokalität 6, 288
- Daten-Parallelismus 287, 288
- Datenpunkt 49, 62, 63, 73, 79
- Datenpyramide 4
- Datenvolumen 2, 22
- Datenvorbereitung 18
- DBSCAN-Algorithmus 118, 119
- Decision Boundary 160, 191
- Decision Surface 158, 166, 191, 197
- Deep Learning 1, 192, 195
- Deep Neural Network *siehe Tiefes Neuronales Netz*
- Deliverables 29
 - Analystenpräsentation 30
 - Daten 31
 - Quelltext 31
 - Sponsorenpräsentation 30

Technische Spezifikation 31

Dendrogramm 130

Dense Layer 194

Densityplot 60, 63, 72

Deskriptive Statistik 47

Dichte 61

Dichte Nachbarschaft 119, 123

Dichtebasierte Clusteralgorithmen 118

Dichtebasiertes Clustering 118

Dichter Bereich 73

Differencing 240

Differenz erster Ordnung 240

Differenz zweiter Ordnung 240

Dimensionalität 265

Dimensionsreduktion 77

Discovery Phase 15

Domänenexpertin 11, 15, 94, 165

Domänenwissen 11, 15, 19, 104

Dropout Layer 191, 194

E

Effektstärke 277

Elastic Net 226

Ellenbogen 105

Ellipse 108

ELT 19

EM-Algorithmus 110

EM-Clustering 107, 110, 113, 117

Ensemble Learning 169

Entropie 167

 bedingte 167

Entscheidungsbaum 164, 197, 200, 209, 309

Epoche 195

ETL 19

Euklidische Norm 96

Euklidischer Abstand 96

Expectation-Maximization-(EM-)Algorithmus *siehe EM-Algorithmus* Exponentielles Wachstum 90
eXtensible Markup Language 45

F

FAIR-Prinzipien 31
Feature Expansion 182
Feature Importance Map 209
Feature Map 36
Fehlendes Merkmal 139, 210
Fläche unter der Kurve *siehe Area Under the Curve*
Flatten Layer 194
Folgerung 84, 117
Freiheitsgrad 50
Frequent Itemset 85, 86, 87, 90, 91
Fully Connected Layer 186, 194

G

Gaussian Mixture Model 108
Gaussian Naive Bayes 178
Gegenstände 83, 84
Gemeinsame Information 167
Generalisierung 39
Generative Adversarial Network 192
Gepoolte Standardabweichung 277
Gewichte 310
Glockenform 59, 108, 268
Grenzwert 145, 146
Großschreibung 255
Güte

- visuelle Bewertung 218
- von Klassifikationsalgorithmen 148
- von Regressionen 216

Gütemaße 87, 90, 93, 148, 309

- binäre 150
- für die Bewertung von Regressionen 221

H

Hadoop Distributed File System 292, 293

Harmonisches Mittel 152

Häufigkeitsbasierte Statistik 268

Hauptkomponenten 77

Hauptkomponentenanalyse 77

Hexbinplot 75

Hierarchische Clusteralgorithmen 129

Hierarchisches Clustern 128

Histogramm 59, 80, 267

Hochdimensionale Daten 77, 97

Holdout-Daten 40

Homonym 265

Hypothese 16, 144, 209

triviale 151

Hypothesentest 268, 272, 275

I

Imputation 139, 210

Induktive Statistik 47

Informatik 9

Informationsgewinn 167

Informationsverarbeitungsmethoden innovative 6, 285, 288

In-Memory-Datenverarbeitung 306

Innovative

Informationsverarbeitungsmethoden 6, 285, 288

Instanz 38, 131

Instanzbasiertes Verfahren 162

Interessante Beziehung 84

Internet der Dinge 1

Internet of Things *siehe Internet der Dinge*

Interquartilsabstand 51, 68

Inverse Document Frequency 262, 264

Irisdaten 65

Itemset *siehe auch Gegenstände* 84

J

Jackknife 310
JavaScript Object Notation 45
Jupyter Lab 313
Jupyter Notebook v, 45, 46, 314

K

Kategorische Daten 50
Kategorisches Merkmal 70, 139, 210
Kategorisierung von Objekten 143
Kernel Density Functions 61
Kernel-Trick 184
Kernpunkte 119, 123
Klassen 143
Klassifikation 21, 41, 143, 144, 310
 binäre 145
Klassifikationsalgorithmen 197, 211
Klassifikationsmodell schwaches 171
k-Means 98, 106, 110, 113
k-Means-Algorithmus 98, 100, 104, 106, 107, 113, 117
k-Means-Clustering 117
k-Means-Clustern 113
k-Nearest Neighbor 126, 160, 162, 197
k-Nearest-Neighbor-Algorithmus 160, 161, 200
Kolmogorow-Smirnow-Test 273
Kommandozeilenbefehl 45, 46
Kommunikation der Projektergebnisse 25
Konfidenzintervall 279
Konvergenz 113
Konzept der Korrelation 72
Konzept des Supports 85
Konzept für Objekte 143
Korpus von Dokumenten 253
Korrelation
 Konzept der 72
 lineare 72
Korrelations-Heatmap 76

Korrelationskoeffizient 72
Korreliertes Merkmal 140, 210
Kovarianz 113
Kovarianzmatrix 108
Kreuzvalidierung 40, 310
Kullback-Leibler-Divergenz 117
Künstliche Intelligenz 1, 10
 EU-Richtlinie 22
 klassische 9

L

Label 39
Lage der Daten 47
Lagemaße 47
Laplace Glättung 63, 64
Lasso 225, 309
Leave-One-Out 310
Lemmatisierung 257
Lernen
 überwachtes 39
 unüberwachtes 39
Lernen der Daten 45
Level 130, 131
Levene-Test 273
Leverage 87, 89, 93
Lift 87, 88, 93
Likelihood-Funktion 115
Lineare Regression 56, 215, 223, 231
Liniendiagramm 81
Log Odds Ratio 176
Logistische Regression 174, 197, 209
Lognormal-Verteilung 65
Long Short-Term Memory 192

M

Macro Averaging 156

MA-Modell 246
Manhattan-Distanz 97
Manhattan-Norm 97
Mann-Whitney-U-Test 273
MapReduce 288
map() 289
Margin 181
Marketing 10
Maschinelles Lernen 9
 Definition 34
Mathematik 8
Maximierung der Margin 181
Maximumsmetrik 97
Maximumsnorm 97
Median 48
Medizin 10
Merkmal 23, 35, 38, 67, 209, 309
 Beziehungen zwischen 69
 der Daten 57
 fehlendes 139, 210
 kategorisches 37, 70, 139, 210
 korreliertes 140, 210
 nominales 37
 normalisiertes 140
 ordinales 37
Merkmalsauswahl 309
Merkmalsraum 36
Message Passing 285, 287
Metadaten 20, 46, 47
Micro Averaging 156
Mittelwert der Daten 48
Modellerstellung 24
Modellplanung 21
Modus der Daten 50
Moving-Average-Modell 246

Multilayer Perceptron 186
Multimodale Daten 50
Multinomial Naive Bayes 178
Multivariate Normalverteilung 108
Mustererkennung 42
Mythical Man-Month 17

N

Nachbarschaft 123
 dichte 119, 123
Naive Assumption 177
Naive Bayes 177, 197
Natürliche Sprache 251
 Mehrdeutigkeit 265
Neuronales Netz 197, 264
 tiefes 1, 22
Neuronales Netzwerk 185
 tiefes 192
Nicht parametrische Statistik 48
No-free-Lunch-Theorem 33
NOIR-Skalen 36
Nominalskala 37
Normalverteilung 52, 139, 268
 multivariate 108
Nullhypothese 268

O

Ockhams Rasiermesser 115
Odd 174
Odds Ratio 176, 209
One-Hot Encoding 37
Operationalisierung 25
Optimierungsalgorithmen 33
Ordinalskala 37
Ordinary Least Squares 224
Overfitting 39, 115, 168, 191, 200

Overprediction 219

Oversampling 310

P

Paarweiser Scatterplot 70

Parallelisierung 285, 287, 291

Parallelität 285

Parametrische Statistik 48

Partielle Autokorrelation 242

Peak 60

p-Hacking 276

Polynomieller Kern 183

Pooling Layer 191, 194

Population 268

Preprocessing 253

 Visualisierung 259

Principal Components Analysis 309

Problembeschreibung 15

Punktsteigungsformel 240

Q

Quartil 51

Quelltext 46

R

Radiale Basisfunktion 183

Random Forest 169, 197

Random Tree 169

Rauschen 119, 125

 weißes 246

Receiver Operator Characteristic 152

Rechencluster 286

 Architektur 287

Rechenknoten 286

Rechtsschief 60

Rectified Linear Unit 188

Recurrent Neural Network 192

- reduce() 290
- Regression 41, 197, 215, 233, 236
 - Güte von 216
 - Gütemaße für die Bewertung 221
 - lineare 215, 223, 231
- Regressionsanalyse 215
- Regressionsmodelle 21
- Regularisierung
 - Auswirkungen 226
- Regularisierungsverfahren 225
- Reinforcement Learning 310
- Resampling 310
- Residuum einer Instanz 219
- Resilient Distributed Dataset 306
- Ressourcen 16
- Ridge 225
- Ridge-Regularisierung 225
- Risiken 16
- Robotik 10
- ROC-Kurve 152
- Rug 62, 63

S

- Satz von Bayes 177
- Scatterplot 56, 73, 80
 - für hochdimensionale Daten 77
 - paarweiser 70
- Schwaches Klassifikationsmodell 171
- Scores 145
- Scoring 209, 276
- Scoring-Funktion 145, 146, 209
- Shapiro-Wilk-Test 273
- Shared Memory 285, 287
- shuffle() 290
- Signifikanzniveau 272
- Single Linkage Clustering 128, 129

Skalen für Merkmale 36
Skalierungsparameter 61
SLINK-Algorithmus 129
Sneaker Net 3
Soft Clustering 111, 113, 145
Softmax Layer 192
Spark SQL 306
Stakeholderanalyse 15
Standardabweichung 50
 gepoolte 277
Statistik 8, 267
 häufigkeitsbasierte 268
Statistische Signifikanz 272
Stemming 257
Steven's Levels of Measurements 36
Stichprobe 45, 271, 273, 279
Stochastic Gradient Descent 310
Stoppwörter 256
Storage Area Network 286
Storage Layer 286
Stratified Sampling 310
Streaming Mode 302
Streamingdaten 3
Support 85
Support Vector Machine 179, 181, 182, 183, 185, 197, 206
Support-Vektor 182

T

Tail 60
Tensor 194
Term Frequency 261
Testdaten 23, 38, 39, 93, 148
Teststatistik 270
Text Mining 41, 251, 265
Texteditor 45
Tiefes neuronales Netz 1, 22

Tiefes neuronales Netzwerk 192
Trainingsdaten 23, 38, 39, 93, 195
Transaktion 83, 84
Transformer 192
Triviale Hypothese 151
t-Test 269
Typischer Wert 47

U

Überwachtes Lernen 39
Unabhängige Variable 215
Underprediction 219
Undersampling 310
Ungleichgewicht der Klassen 151
Unüberwachtes Lernen 39

V

Validierungsdaten 23, 40, 195
Variabilität der Daten 50, 51
Variable
 abhängige 215
 unabhängige 215
Varianz der Daten 78
Varianz innerhalb eines Clusters 104
Verständnis der Daten 53
Verteiltes Rechnen 286
Verteilungsbasiertes Clustern 107
Visualisierung 53, 79, 104, 132, 245
Voronoi-Zellen 113

W

Wahrscheinlichkeit 33, 108, 110, 111, 113, 197
 bedingte 177
 exakte 178
 gemeinsame 177
Wahrscheinlichkeitsrechnung 197
Wahrscheinlichkeitsverteilung 20, 60, 145, 188, 209

kontinuierliche 178
Weißes Rauschen 246
Welchs t-Test 269, 273
Whisker 68
Wilcoxon-Ranksum-Test 273
Within-Sum-of-Squares 104
Worteinbettung 264
Wortwolken 259

Y

Yet Another Resource Negotiator 292, 295

Z

Zeichensetzung 255
Zeitliche Trends 79
Zeitlicher Verlauf der Daten 79
Zeitreihenanalyse 21, 41, 233, 248
Zielkonzept für Objekte 145
Z-Score-Standardisierung 177