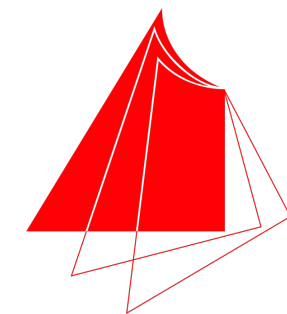


KI Summer School

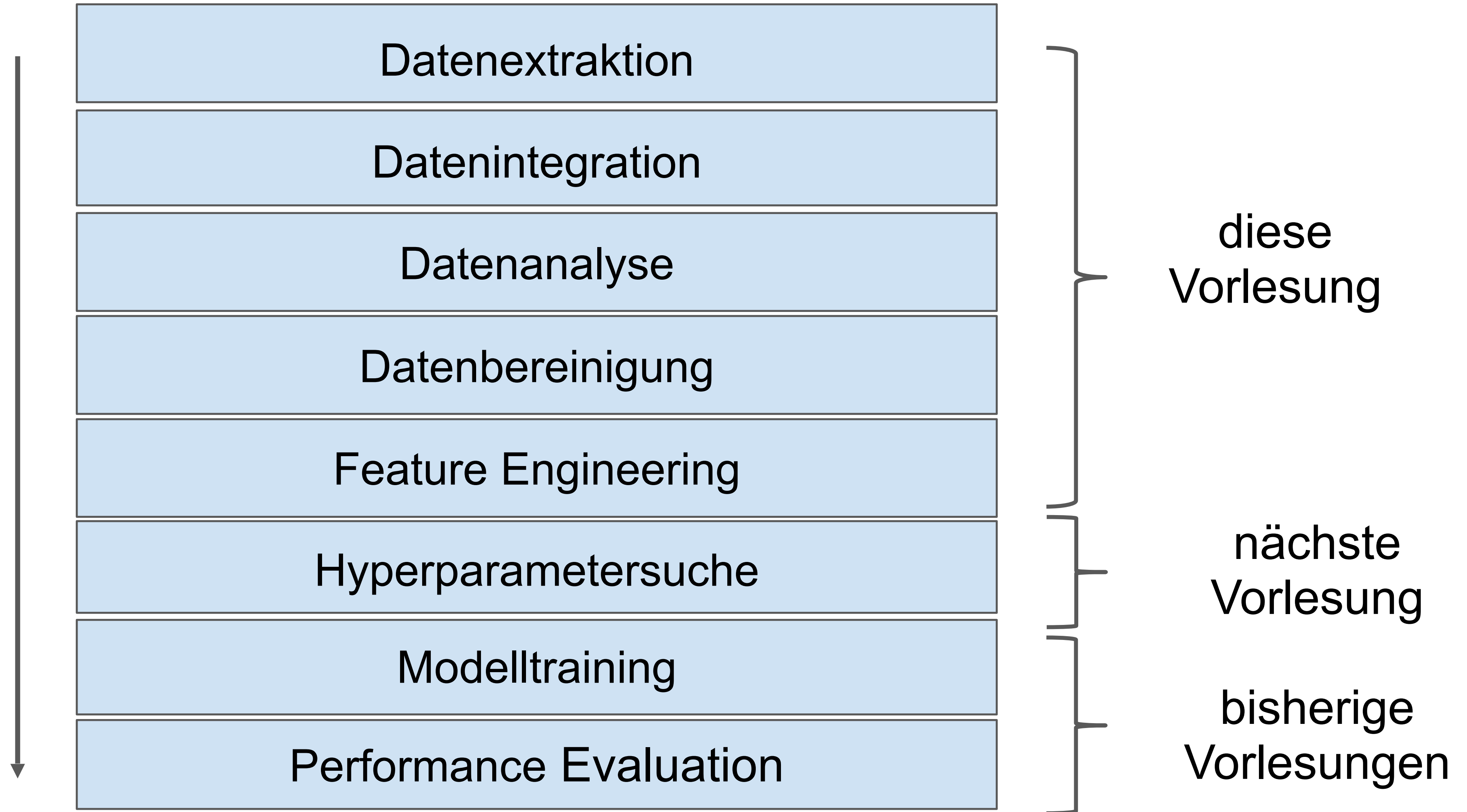
ML Workflow



Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Patrick Baier

ML-Workflow

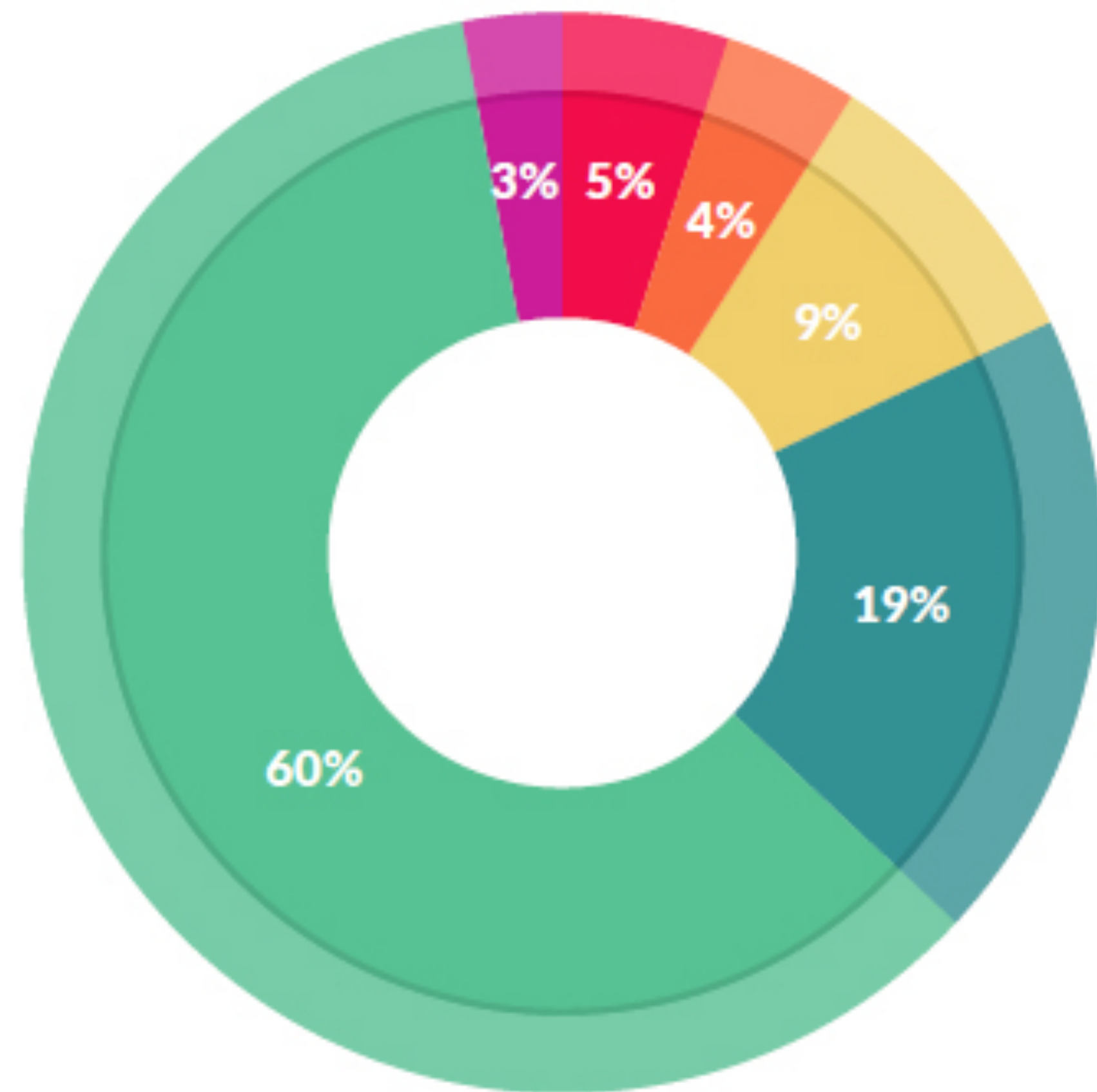


ML-Workflow

Bevor das Training des ML-Modells beginnen kann, muss folgendes erfolgen:

- **Datenextraktion**: Extrahiere Daten für das Modelltraining. Allgemein gilt: Je mehr Trainingsdaten vorhanden sind, desto besser wird die Performance des ML-Modells.
- **Datenintegration**: Daten müssen aus verschiedenen Teilsystemen zusammengeführt werden (z.B. Logdateien mit Datenbankeinträgen).
- **Datenanalyse**: Die Daten werden grob analysiert um ein Verständnis für die einzelnen Attribute zu entwickeln .
- **Datenbereinigung**: Auffüllen von Nullwerten, Deduplizieren.
- **Feature Engineering**: Aus den vorhandenen Daten müssen Feature gebaut werden welche gut das Label vorhersagen können.

Zeitliche Aufteilung

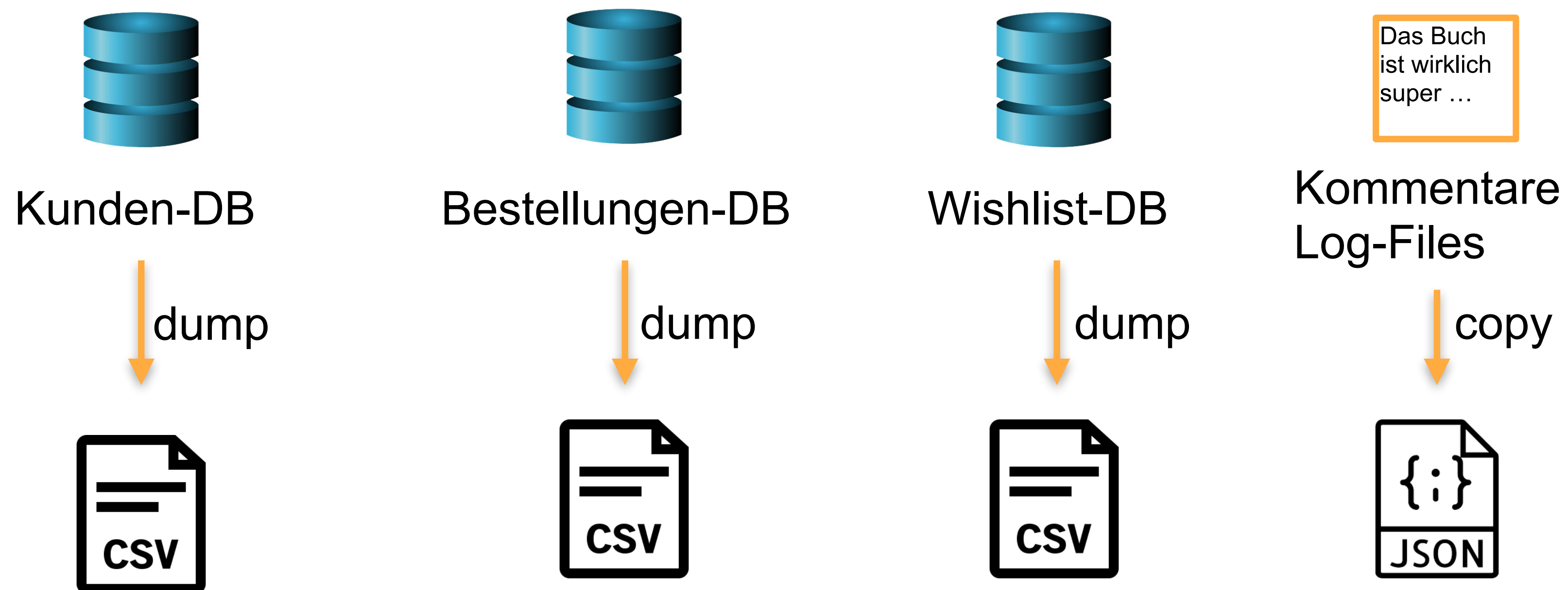


What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets; 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

Datenextraktion

- Ziel der Datenextraktion ist es alle für das Modelltraining relevanten Daten aus den unterschiedlichen Quelldatensystemen zu extrahieren.
- Beispiel: Buchempfehlung für Onlineshop



Datenextraktion

- Bei der Datenextraktion müssen i.d.R. Daten aus einem Datenspeicher extrahiert werden (z.B. Datenbank oder Filesystem).
- Je nach Datenquelle stehen dazu unterschiedliche Methoden zur Verfügung.
- Die häufigsten Zielformate für eine Extraktionen sind CSV- und JSON-Dateien.

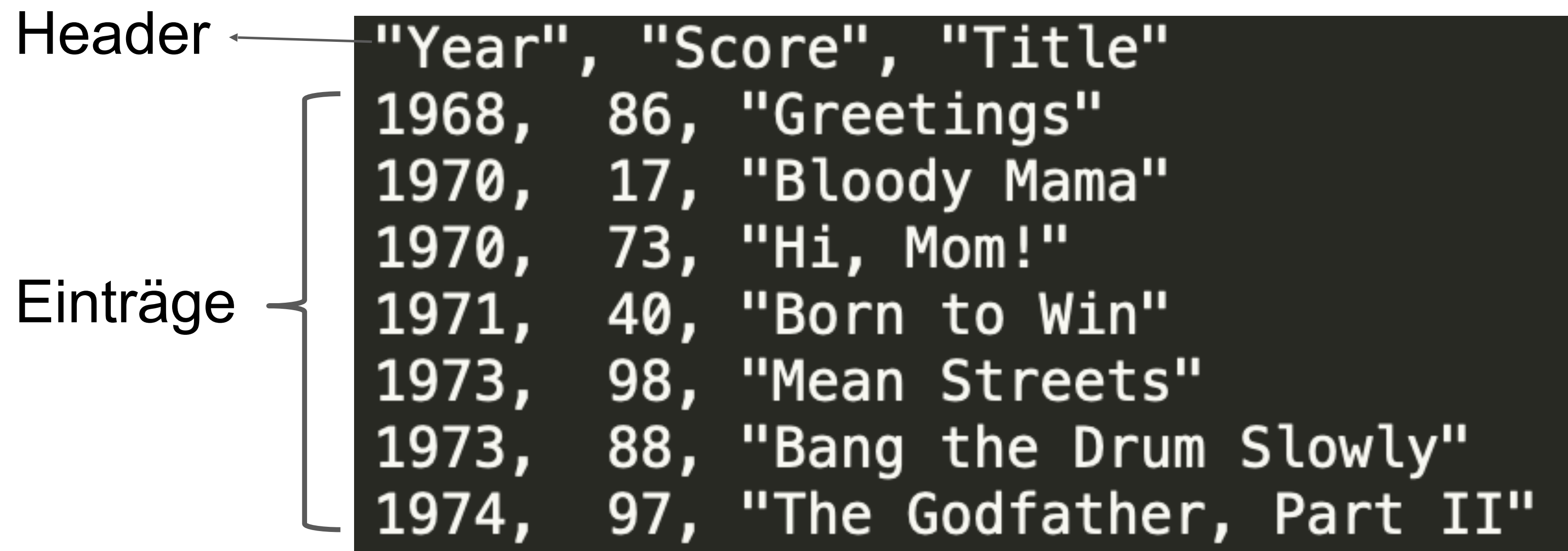
Beispiel: Extraktion von Daten aus Postgres SQL Datenbank:

```
COPY Kunden(Kundennr, Alter, Wohnort)
TO 'C:\tmp\persons_partial_db.csv' DELIMITER
',' CSV HEADER;
```

Vorsicht bei Extraktion aus einer Live-Datenbank! (Extraktion kann Latenz der Datenbank für Requests enorm steigen).

Datenformate - CSV

Eine CSV-Datei ("Comma separated values") ist eine Tabelle in Textform. Die erste Zeile bildet den Header und enthält die Spaltennamen. Jede weitere Zeile entspricht einem Eintrag in der Tabelle.



The diagram illustrates the structure of a CSV file. It shows a list of rows with labels and arrows indicating their function. The first row is labeled 'Header' and contains the column names: "Year", "Score", and "Title". The subsequent seven rows are grouped by a bracket labeled 'Einträge' (Entries) and represent individual data points. Each entry row contains a year, a score, and a movie title, separated by commas.

Header	"Year"	"Score"	"Title"
Einträge	1968,	86,	"Greetings"
	1970,	17,	"Bloody Mama"
	1970,	73,	"Hi, Mom!"
	1971,	40,	"Born to Win"
	1973,	98,	"Mean Streets"
	1973,	88,	"Bang the Drum Slowly"
	1974,	97,	"The Godfather, Part II"

Datenformate - JSON

Eine Json-Datei (“JavaScript Object Notation”) ist eine Zusammensetzung aus:

- Elementen der Form:
“Name”: Wert
- Strukturen von mehreren Elementen der Form:
{ ..., ..., ... }
- Listen aus mehreren Elementen oder Strukturen der Form:
“Listenname” : [..., ..., ...]

```
{  
  "book": [  
    {  
      "id": "444",  
      "language": "C",  
      "author": "Dennis Ritchie "  
    },  
    {  
      "id": "555",  
      "language": "C++",  
      "author": " Bjarne Stroustrup "  
    }  
  ]  
}
```


Datenformate

CSV- und Json-Dateien beinhalten eine Strukturdefinition (welche Felder existieren und welche Werte haben diese).

→ Beide können direkt in Pandas als DataFrames geladen werden.

```
df1 = pandas.read_csv("/Users/pbaier/file1.csv")  
df2 = pandas.read_json("/Users/pbaier/file2.json")
```

Datenintegration

Ziel der Datenintegration ist es alle Daten in einheitliches Format (z.B. Pandas DataFrame) zu überführen, bei dem für jeden Datenpunkt alle verfügbaren Informationen verknüpft sind.

Kundennr.	Alter	Wohnort	Bestellte Artikel	Wishlist-Artikel	Kommentare
1234	35	Karlsruhe	[243, 567, 888, 9]	[67, 34]	["Ich empfehle ...", ...]
1235	61	Esslingen	[45]	-	["Ich mag keine ..."]



Kunden-DB



Bestellungen-DB



Wishlist-DB



Kommentare-Logs

Datenintegration

Typisches Vorgehen:

1. Laden aller extrahierten Daten (csv/json-Dateien) in Pandas DataFrames.
2. Join aller Datenquelle zu einem DataFrame.

```
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],  
                    'A': ['A0', 'A1', 'A2', 'A3'],  
                    'B': ['B0', 'B1', 'B2', 'B3']})  
  
right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],  
                     'C': ['C0', 'C1', 'C2', 'C3'],  
                     'D': ['D0', 'D1', 'D2', 'D3']})  
  
result = pd.merge(left, right, on='key')
```

left				right			
	key	A	B		key	C	D
0	K0	A0	B0	0	K0	C0	D0
1	K1	A1	B1	1	K1	C1	D1
2	K2	A2	B2	2	K2	C2	D2
3	K3	A3	B3	3	K3	C3	D3

Result					
	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Terminologie

Für den Rest der Vorlesung nutzen wir folgende Terminologie:

Kundennr.	Alter	Wohnort	Bestellte Artikel	Wishlist-Artikel	Kommentare
1234	35	Karlsruhe	[243, 567, 888, 9]	[67, 34]	["Ich empfehle ...", ..]
1235	61	Esslingen	[45]	-	["Ich mag keine ..."]

- Datenpunkt: Zeile in Dataframe
- Datenattribut: Spalte in Dataframe
- Datenfeld: Zelle in Dataframe

Feature: Input-Spalte für das ML-Training (abgeleitet aus Attribut)

Big Data Engineering

Die Datenextraktion und -integration von großen Datenbeständen aus heterogenen Quellen erfordert oft spezielle Techniken und Tools, welche auch als Big Data Engineering bezeichnet werden.

→ Mittlerweile eigenes Berufsbild

→ Weitere Details out-of-scope für diese Vorlesung



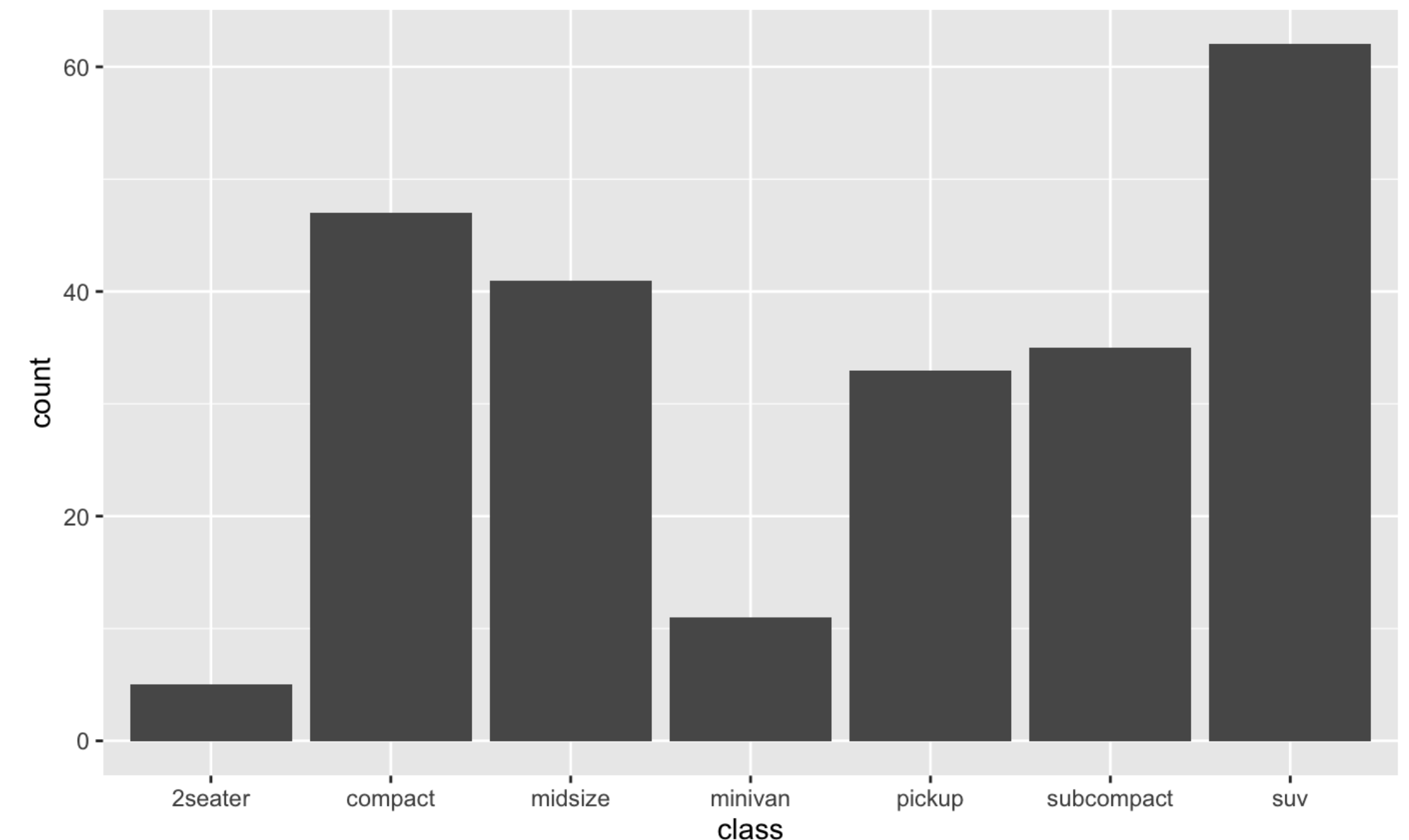
Datenanalyse

- Ziel der Datenanalyse ist es ein Verständnis für die Daten zu entwickeln. Dazu werden verschiedene Statistiken und Visualisierungen erstellt.
- Dieser Schritt wird oft auch als „explorative Datenanalyse“ (EDA) bezeichnet.
- EDA ist ein riesiges Gebiet, daher beschränken wir uns auf die wichtigsten Analysen:
 - **Datenverteilungen (in Form von Histogrammen)**
 - **Deskriptive Statistiken („Summary Statistics“)**
 - **Boxplot**

Datenanalyse

Ein Histogramm verschafft einen schnellen Überblick über:

- Den Wertebereich des Attributes (Minimum und Maximum).
- Wie sind die Werte verteilt?
- Welche Werte kommen am öftesten vor.
- Welche Werte sind eher selten.



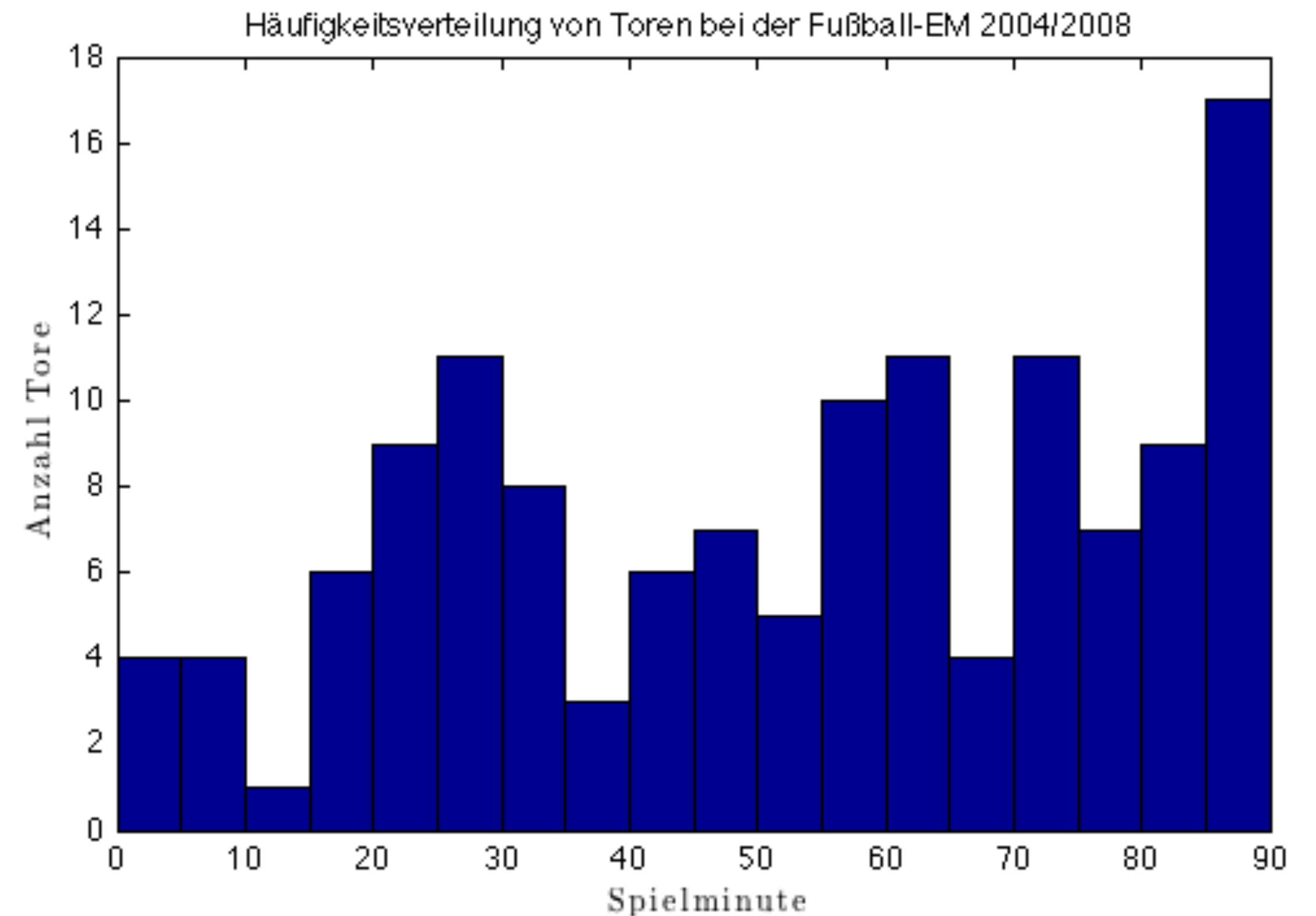
Konstruktion bei kategorischem Attribut:

- x-Achse: Ausprägungen des Attributs
- y-Achse: Anzahl Datenpunkte in Datensatz pro Ausprägung („count“).

Datenanalyse

Konstruktion bei numerischem Attribut:

- x-Achse:
 - Unterteile Wertebereich in k Intervalle (sog. „bins“).
 - Intervalle sind typischerweise (aber nicht zwingend) gleich groß.
 - Bestimme für jeden Datenpunkt zugehöriges Intervall.
- y-Achse:
 - Anzahl der Datenpunkte in dem jeweiligen Intervall.



Datenanalyse

Zusätzlich ist es oft hilfreich sich deskriptive Statistiken zu einem Attribute anzeigen zu lassen:

```
1 v = [20, 10, 25, 15, 12]
2 s = pd.Series(v)
3 s.describe()
```

```
count      5.000000
mean       16.400000
std         6.107373
min        10.000000
25%        12.000000
50%        15.000000
75%        20.000000
max        25.000000
dtype: float64
```

25% der Werte sind kleiner als 12

50% der Werte sind kleiner als 15 (Median)

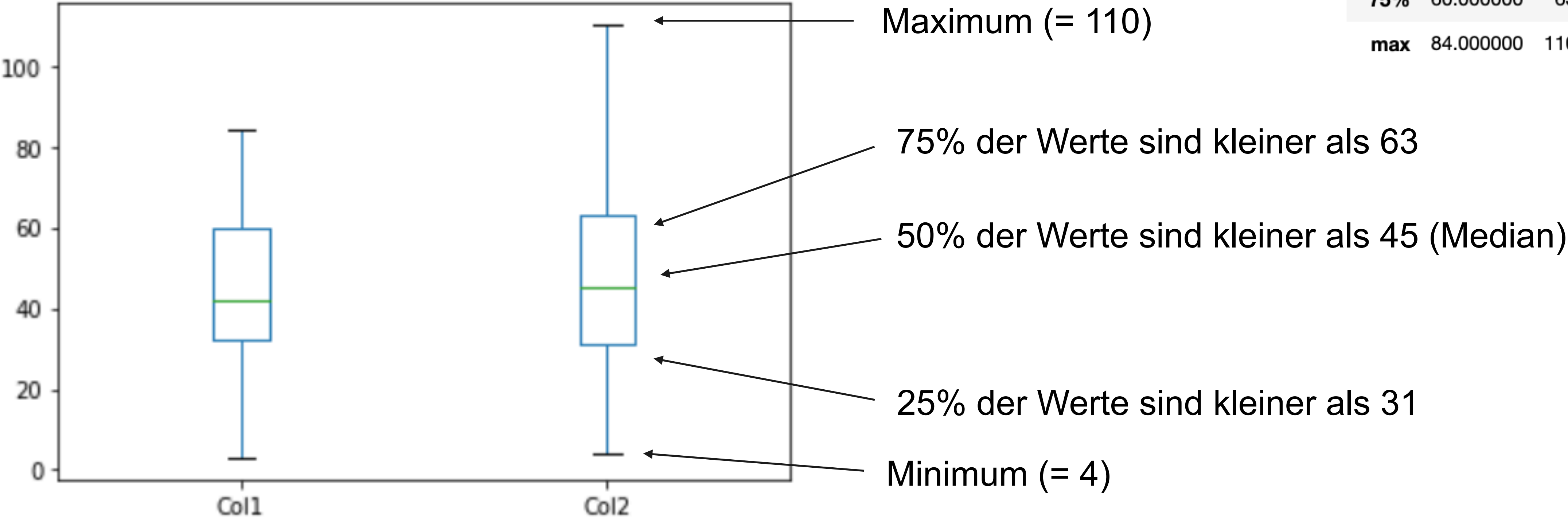
75% der Werte sind kleiner als 20

```
1 df.describe()
```

Boxplot

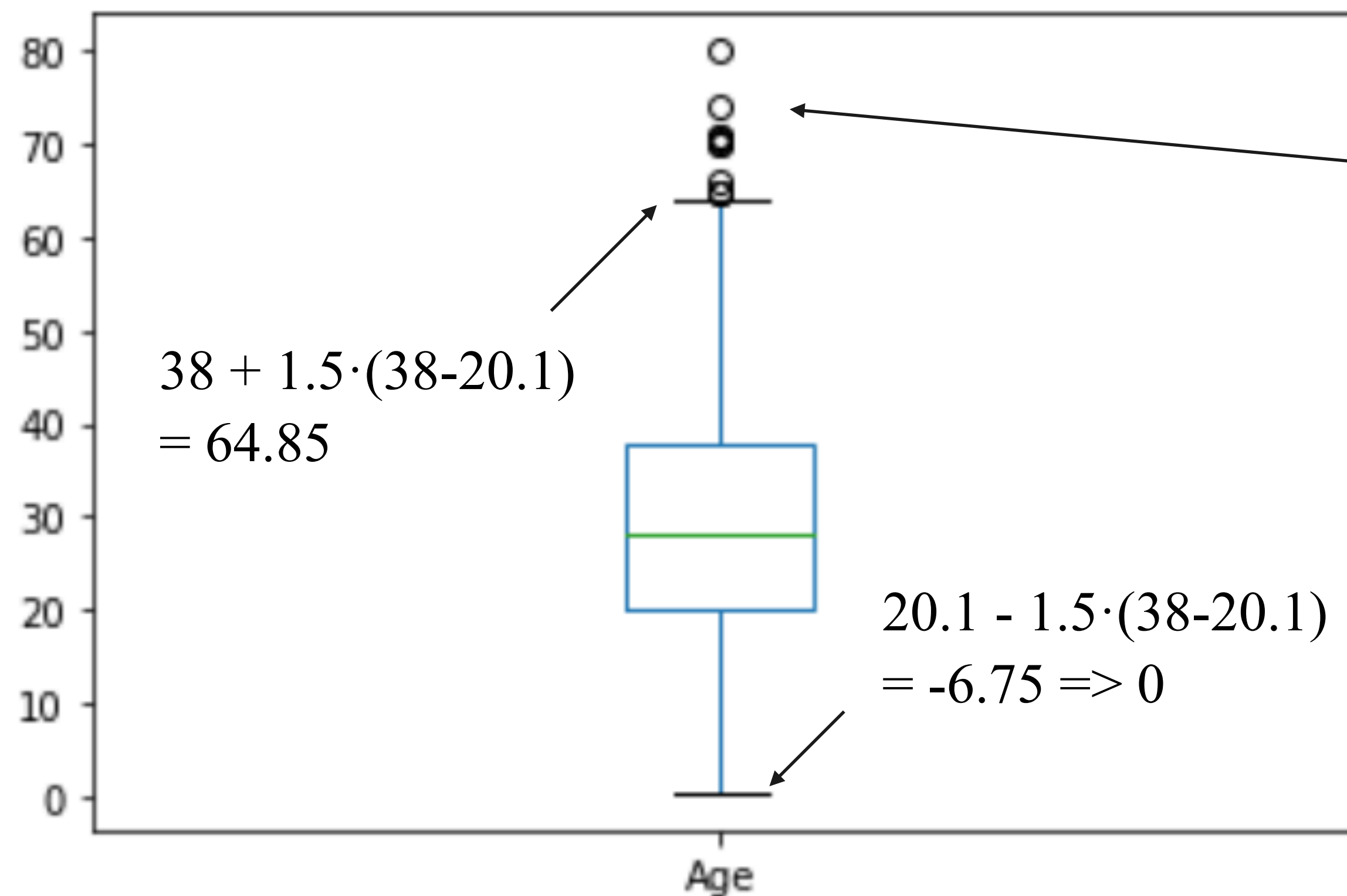
```
1 col1 = [42, 84, 41, 3, 55, 68, 31, 32, 60]
2 col2 = [12, 4, 31, 63, 45, 58, 70, 110, 40]
3 df = pd.DataFrame({"Col1": col1, "Col2": col2})
4 boxplot = df.boxplot(column=['Col1', 'Col2'], grid=False)
```

	Col1	Col2
count	9.000000	9.000000
mean	46.222222	48.111111
std	23.758040	32.107026
min	3.000000	4.000000
25%	32.000000	31.000000
50%	42.000000	45.000000
75%	60.000000	63.000000
max	84.000000	110.000000



Boxplot mit Ausreißer

```
df = pd.read_csv("data/titanic.csv")
df.boxplot(column=['Age'], grid=False)
```



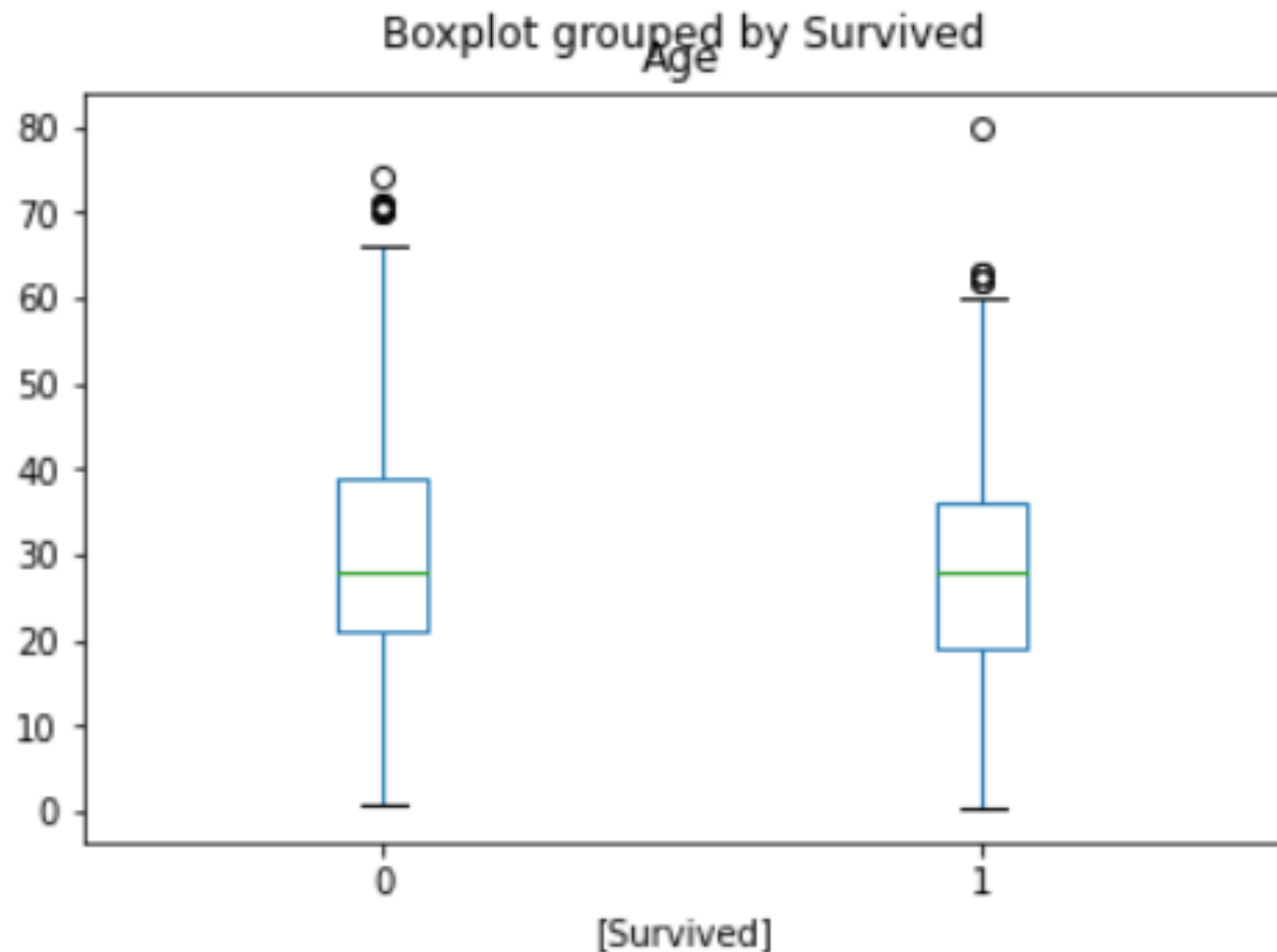
1	df["Age"].describe()
count	714.000000
mean	29.699118
std	14.526497
min	0.420000
25%	20.125000
50%	28.000000
75%	38.000000
max	80.000000
Name: Age, dtype: float64	

Ausreißer

- Wenn der Datensatz Ausreißer enthält, werden diese als Punkte ober- bzw. unterhalb der Antennen („whiskers“) geplottet.
- Die Whiskers enden in diesem Fall nicht am Minimum bzw. Maximum, sondern an den Punkten:
 - $25\text{-Quantil} - 1.5 \cdot (75\text{-Quantil} - 25\text{-Quantil})$
 - $75\text{-Quantil} + 1.5 \cdot (75\text{-Quantil} - 25\text{-Quantil})$
- Die Whiskers gehen jedoch nie über das Minimum bzw. Maximum hinaus

Boxplot nach Label

```
df = pd.read_csv("data/titanic.csv")  
df.boxplot(column=['Age'], by=["Survived"], grid=False)
```



- Ein Boxplot nach Label erlaubt es uns Rückschlüsse über die Aussagekraft des Features auf das Label.
- Je mehr die Boxen versetzt sind, desto besser kann das Feature diskriminieren.
- Aber: Indirekte Zusammenhänge über andere Features sind so nicht sichtbar. Feature kann trotz identischer Boxplots hilfreich sein.

Datenbereinigung

Ziel der Datenbereinigung ist es Daten so aufzubereiten, dass keine Duplikate, fehlerhafte oder fehlende Datenfelder vorhanden sind.

Solche Probleme können verschiedene Ursachen haben (je nachdem wie die Daten erstellt wurden):

- Fehlende Messungen von Sensordaten
- Optionale Felder in Eingabeformularen
- Duplikation nach Einsatz von Recovery-Mechanismen
- Outage im System oder Datenbanken
- Bugs in Programmlogik
- ...

Duplikate

- Duplikate sind Datenpunkte welche mehrfach in den Daten vorkommen, d.h. alle Attribute haben die genau gleichen Werte.
- Meistens entstehen Duplikate aus fehlerhafter Datenverarbeitung.
- Da Duplikate keine neuen Information bieten, können diese aus dem Datensatz entfernt ohne dass sich die Performance des ML-Systems verschlechtert.

Kundennr.	Alter	Wohnort	Bestellte Artikel	Wishlist-Artikel	Kommentare
1234	35	Karlsruhe	[243, 567, 888, 9]	[67, 34]	["Ich empfehle ...", ...]
1235	61	Esslingen	[45]	-	["Ich mag keine ..."]
1234	35	Karlsruhe	[243, 567, 888, 9]	[67, 34]	["Ich empfehle ...", ...]

Fehlende Datenfelder

Fehlende Datenfelder werden (je nach Programmiersprache) durch folgende Werte dargestellt: NaN, null, None, 0, ...

In Pandas werden fehlende Datenfelder durch “NaN”-Werte dargestellt:

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

Fehlende Datenfelder

Es gibt verschiedene Wege fehlende Datenfelder zu bereinigen:

1. Ganzen Datenpunkt **entfernen** (= ganze Zeile in DataFrame löschen).
 - Kann zu erheblicher Reduktion der Datenmenge führen.
 - Kann zu Einbußen bei der Performance des Modells führen.
 - Nur ratsam wenn wenige Datenpunkte betroffen sind.
2. Ganzes Attribut entfernen (= ganze Spalte in DataFrame löschen).
 - Führt zu Verlust von Informationen => Modell ggf. schlechter,
 - Macht nur Sinn wenn großer Prozentsatz der Datenfelder im Attribut fehlen.

Fehlende Datenfelder

Es gibt verschiedene Wege fehlende Datenfelder zu bereinigen:

3. Fehlende Datenfelder mit **neutralem Wert** auffüllen:

- Numerischen Features: Arithmetischer Durchschnitt oder Median
- Kategorischen Features:
 - Häufigster Wert
 - Platzhalter, z.B. Farben: „gelb“, „rot“, „unbekannt“

Fehlende Datenfelder - Beispiel

1. Ganzen Datenpunkt **entfernen**.

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

Fehlende Datenfelder - Beispiel

2. Ganzes Attribut **entfernen**.

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

Fehlende Datenfelder - Beispiel

3. Fehlende Datenfelder mit **neutralem Wert** auffüllen.

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	116.7	33.6	0.627	50	1
1	1	85.0	66.0	29.0	116.7	26.6	0.351	31	0
2	8	183.0	64.0	30.8	116.7	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	30.8	116.7	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1

Fehlende Datenfelder

Werden fehlende Datenfelder auch in den Testdaten aufgefüllt?

Testdaten sollen als Proxy für die Daten dienen, welche wir später im Live-System sehen werden. Daher kommt es darauf an, wie wir später im Live-System fehlende Daten behandeln wollen:

- **Option 1: Kein Auffüllen.** In den Testdaten werden Datenpunkte mit fehlenden Datenfeldern verworfen, d.h. es wird keine Vorhersage gemacht. („Aus unvollständigen Daten können wir keine zuverlässige Vorhersage machen“).
- **Option 2: Auffüllen.** Wir füllen die fehlenden Daten auch in den Testdaten auf, und zwar mit den gleichen Werten (Mean, Median, etc.) welche wir auf den Trainingsdaten verwendet haben. („Besser eine schlechte Vorhersage wie gar keine Vorhersage“).

Welche Option besser passt hängt vom jeweiligen Use-Case ab.

Für den Rest dieser Vorlesung nehmen wir Option 1.

Ausreißer

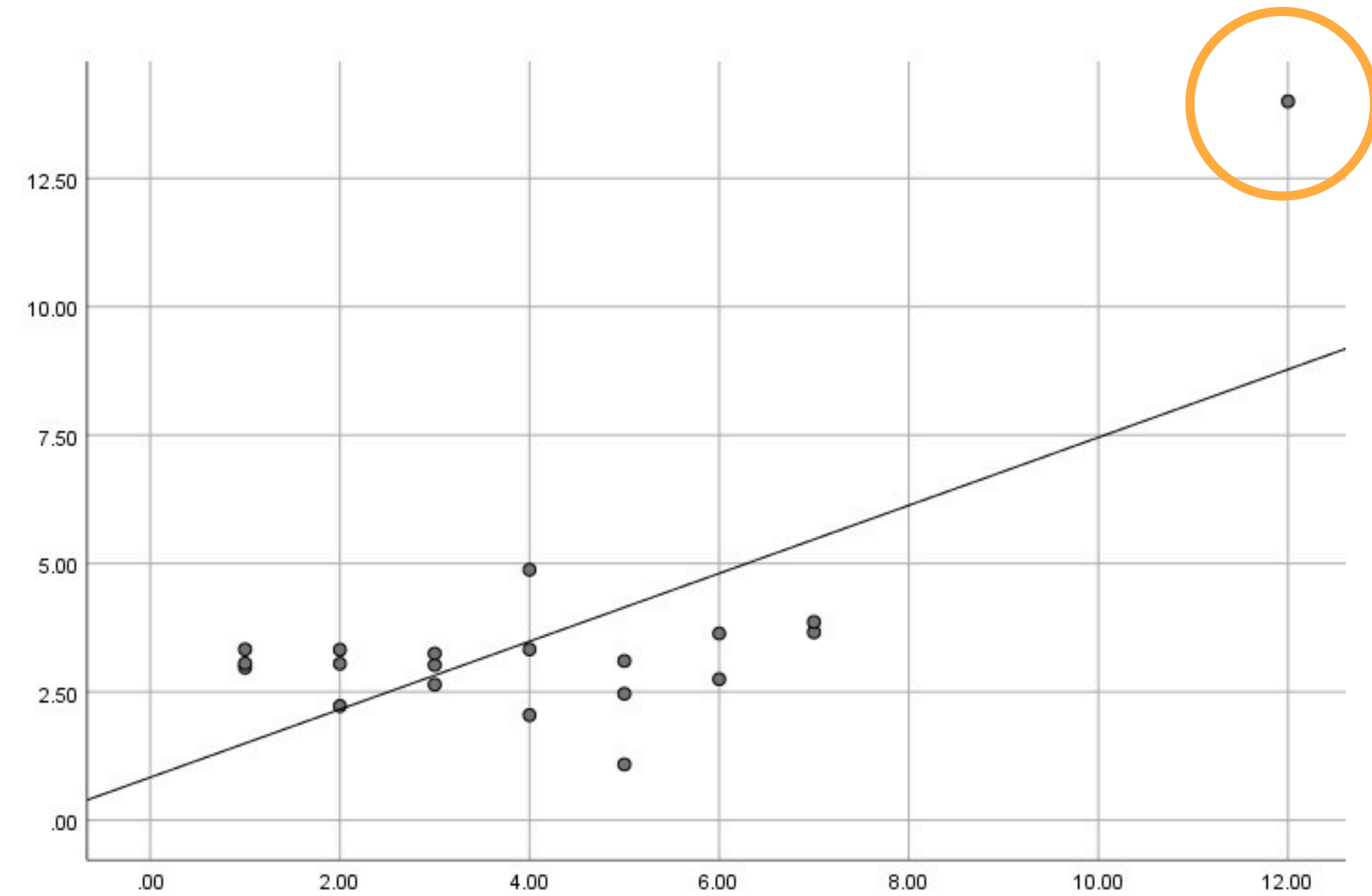
Ein Ausreißer ist ein Datenpunkt der von den übrigen Datenpunkten in auffälliger Weise abweicht.

Müssen Ausreißer entfernt werden?

→ Nur wenn der entsprechende Datenpunkt offensichtlich fehlerhaft ist.

Beispiele:

- Attribut „Außentemperatur in Celsius“, Datenpunkt hat Wert „500“ → entfernen
- Attribut „Auswärtstore“, Datenpunkt hat Wert „14“ → nicht entfernen



Feature Engineering

„Feature engineering is the process of using domain knowledge to extract features from raw data [...] Features can be used to improve the performance of machine learning algorithms “

https://en.wikipedia.org/wiki/Feature_engineering

„...some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.“

Pedro Domingos, “A Few Useful Things to Know about Machine Learning”


Feature Engineering

Nachdem die Daten aufbereitet wurden, folgt das Feature Engineering, welches selbst aus mehrere Teilaufgaben besteht:

1. Entfernen unnötiger oder redundanter Datenattribute.
2. Transformieren der Datenattribute in numerische Werte mittels “One-hot-encoding” oder semantischer Transformation.
3. Umwandlung der Datenattribute in vorhersagekräftige Features.
4. Skalierung der Features.

Datenattribute Eliminieren

Datenattribute die offensichtlich nicht mit der Zielvariablen korrelieren (z.B. fortlaufender Index oder Zufallsnummer) werden vor dem Training eliminiert.



Bestellnummer	Anzahl Artikel	Summe	Artikel < 30€	Teuerster Artikel
12341234	4	546	1	300
23452345	3	77	2	34
3456456	4	80	3	50

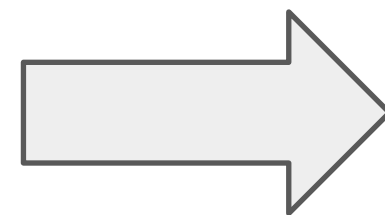
Numerische Transformation

- Generell gibt es zwei Arten von Datenattributen:
- **Numerisch**: Wertebereich ist numerisch.
 - Größe einer Person in cm (182, 165, 191, ...)
 - Alter einer Person in Jahren (44, 13, 77, ...)
- **Kategorisch**: Wertebereich ist diskret.
 - Farbe des Autos (rot, grün, schwarz, weiß, ...)
 - Nationalität (deutsch, englisch, französisch, ...)
- (Die meisten) ML-Algorithmen benötigen numerische Features.
- Umwandlung mittels:
 - One-hot Encoding
 - Semantische Transformation

One-hot Encoding

- Kategorische Datenattribute können in numerische mittels “One-hot Encoding” umgewandelt werden:

Gewicht	Farbe
123	rot
345	grün
222	blau
100	blau
501	rot



Gewicht	rot	grün	blau
123	1	0	0
345	0	1	0
222	0	0	1
100	0	0	1
501	1	0	0

- Idee: Führe eigene Spalte für jede Ausprägung des Attributs ein.

Semantische Transformation

- Einige Datenattribute können auch durch eine Umwandlung direkt in numerische Werte transformiert werden:
 - “12h 30min” → 750 min
 - “Packung Schokolade ” → 500 kCal
- Für diese Form der Umwandlung ist oft Kreativität oder Expertenwissen der Anwendungsdomäne erforderlich.

Umwandlung in Features

- Das Erstellen von Features mit hoher Vorhersagekraft ist eine der wichtigsten (und interessantesten) Aufgaben im Machine Learning.
- Vorgehensweise:
Transformiere Datenattribute zu Features welche Rückschlüsse auf das Label des Datenpunkts ermöglichen.
- Feature Engineering erfordert:
 - Expertenwissen der Anwendungsdomäne
 - Kreativität
 - Erfahrung

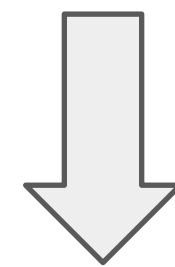
Beispiel

- Datenattribute: Bestellnummer und Warenkorb einer Onlinebestellung
- Vorhersage: Werden Teile der Bestellung zurückgeschickt?

Bestellnummer	Warenkorb [€]
12341234	[12, 200, 34, 300]
23452345	[34, 23, 20]
34564356	[10, 10, 50, 10]

Beispiel

Bestellnummer	Warenkorb [€]
12341234	[12, 200, 34, 300]
23452345	[34, 23, 20]
34564356	[10, 10, 50, 10]



Transformiere Warenkorb in vier Features

Bestellnummer	Anzahl Artikel	Summe	Artikel < 30€	Teuerster Artikel
12341234	4	546	1	300
23452345	3	77	2	34
34564356	4	80	3	50

Feature Scaling

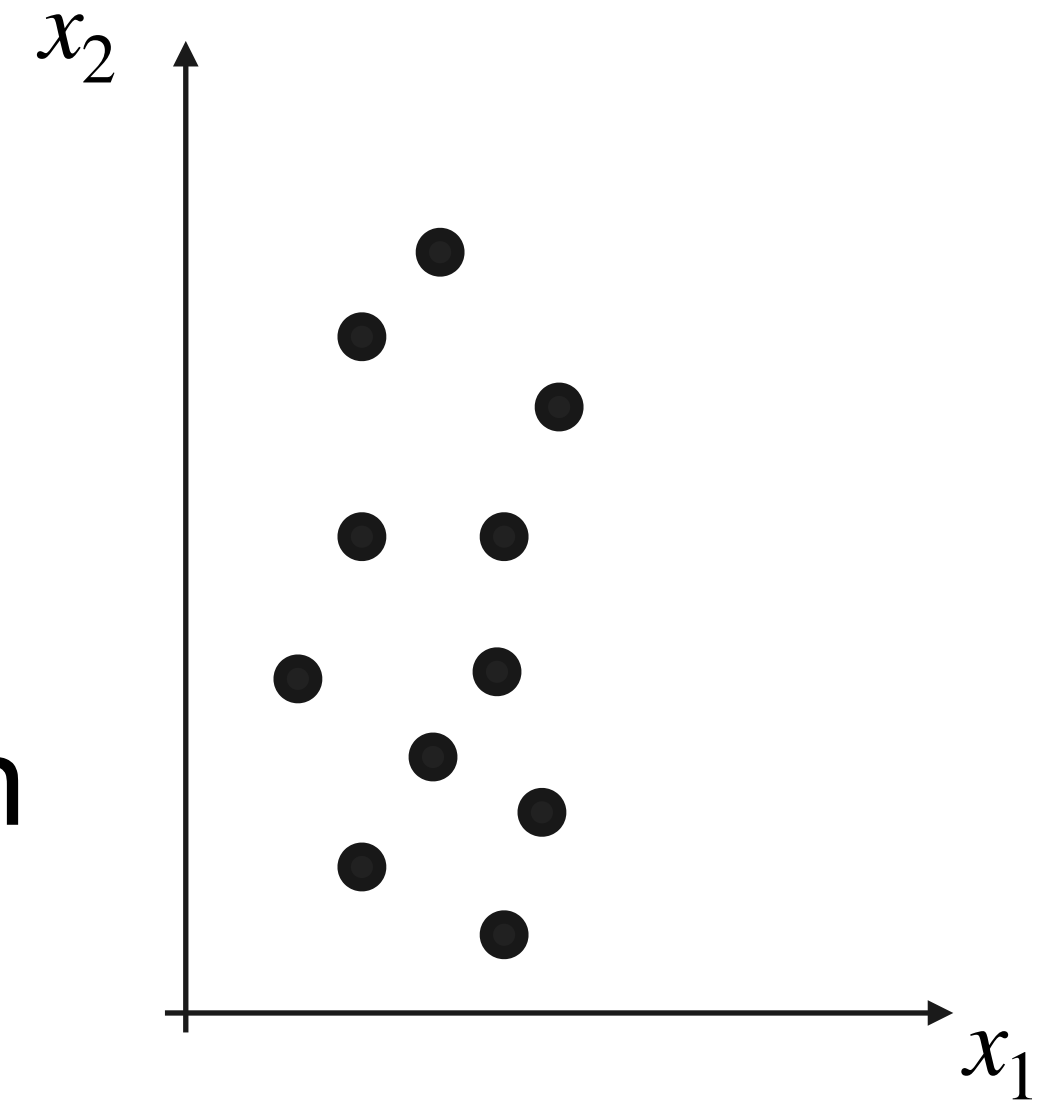
Features können generell sehr unterschiedliche Wertebereiche aufweisen.

Beispiel: Vorhersage des Mietpreises einer Wohnung

x_1 Entfernung zum Zentrum $\in [0, 30]$ km

x_2 Größe der Wohnung $\in [30, 300]$ m²

Wertebereich von Feature x_2 ist um Faktor 10 größer als von Feature x_1 .



Feature Scaling

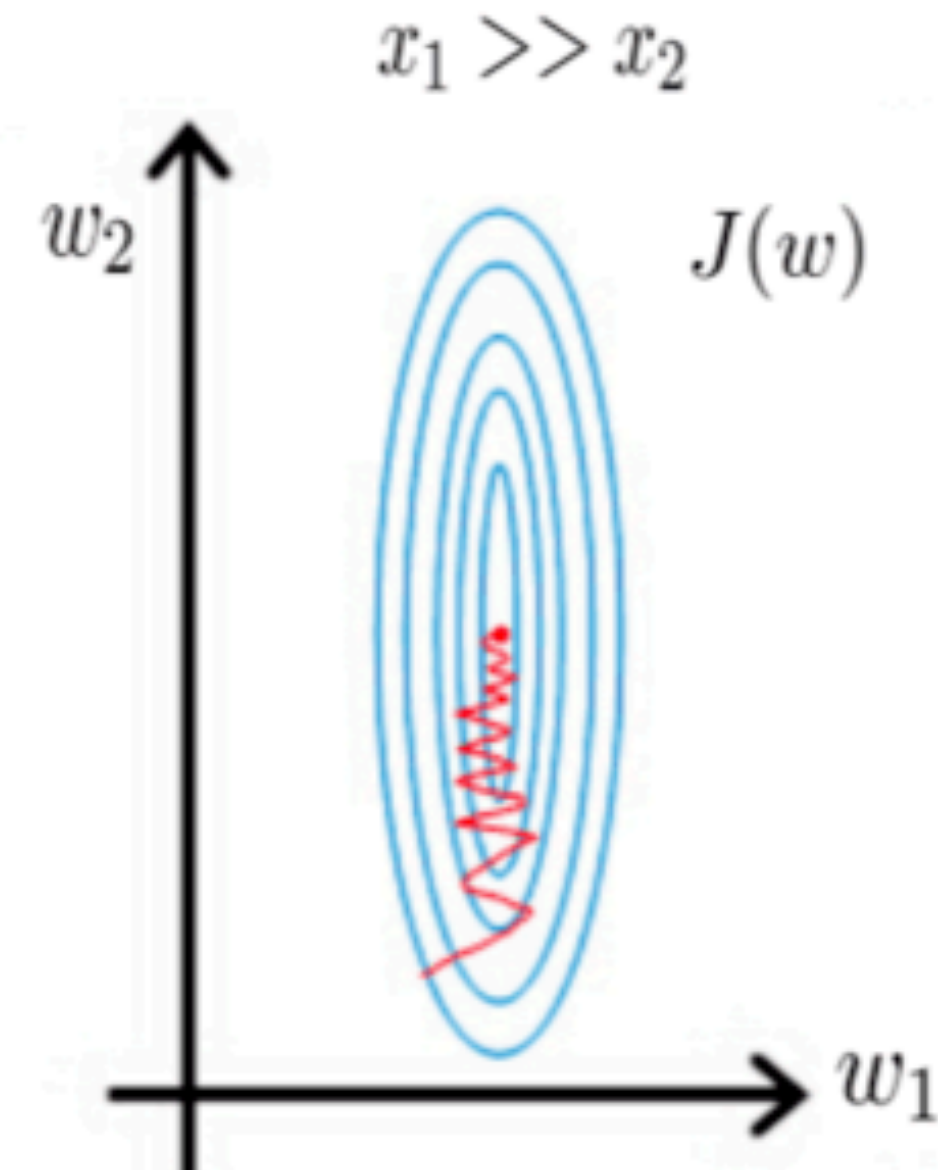
- Feature mit größerem Wertebereich dominiert den Gradienten.
- Gradient Descent springt vor und zurück.
- Verfahren konvergiert langsamer.
→ Features müssen in den gleichen Wertebereich skaliert werden.

Zwei wichtigste Methoden:

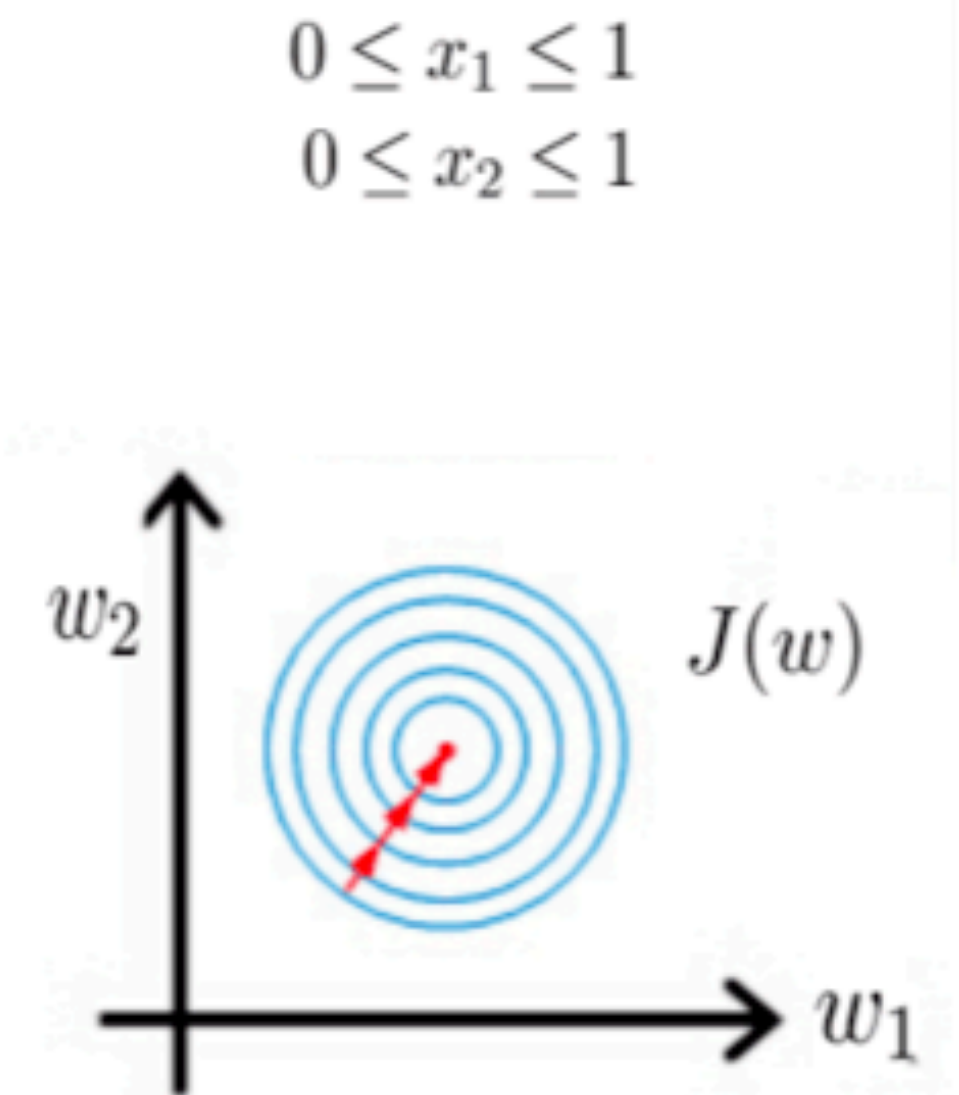
1. Standardisierung

2. Min-Max-Normalisierung

Gradient descent
without scaling



Gradient descent
after scaling variables



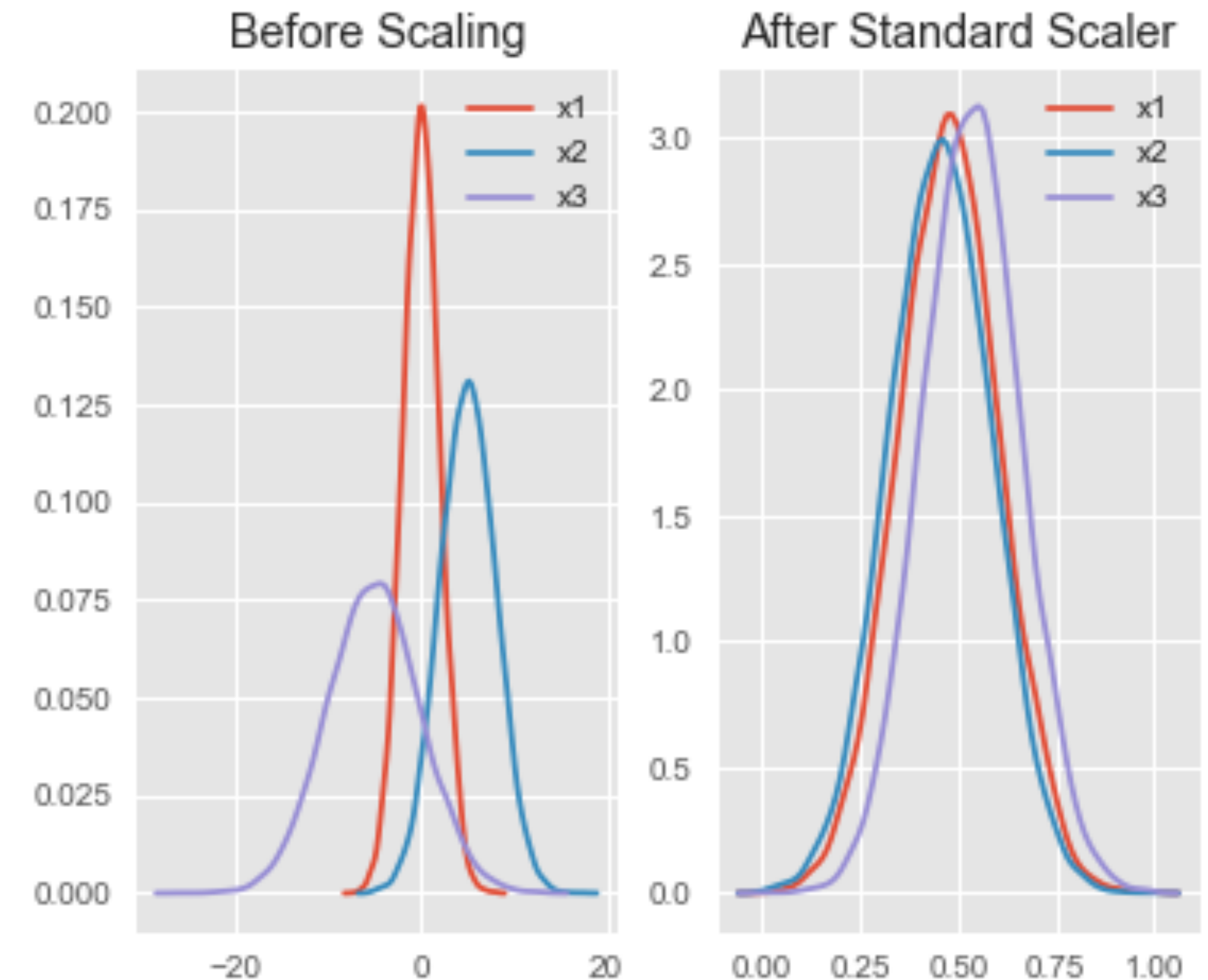
Feature Scaling - Standardisierung

Ziel: Jedes Feature hat

1. im arithmetischen Mittel den Wert “Null”.
2. eine Standardabweichung von “Eins”.

→ Transformiere Feature x zu x' wie folgt:

$$x' = \frac{x - \bar{x}}{\sigma} = \frac{x - \frac{1}{N} \sum_{i=0}^N x_i}{\sqrt{\frac{1}{N} \sum_{i=0}^N (x_i - \bar{x})^2}}$$



Feature Scaling - Standardisierung

	Größer Wohnung	Entfernung Zentrum
	120	0,5
	80	5
	110	8
	50	1
	200	30
	94	14
$\bar{x} =$	109,0	9,75
$\sigma =$	46,5	10,13



Größer Wohnung	Entfernung Zentrum
0,24	-0,91
-0,62	-0,47
0,02	-0,17
-1,27	-0,86
1,96	2,00
-0,32	0,42

Feature Scaling - Min-Max-Normalisierung

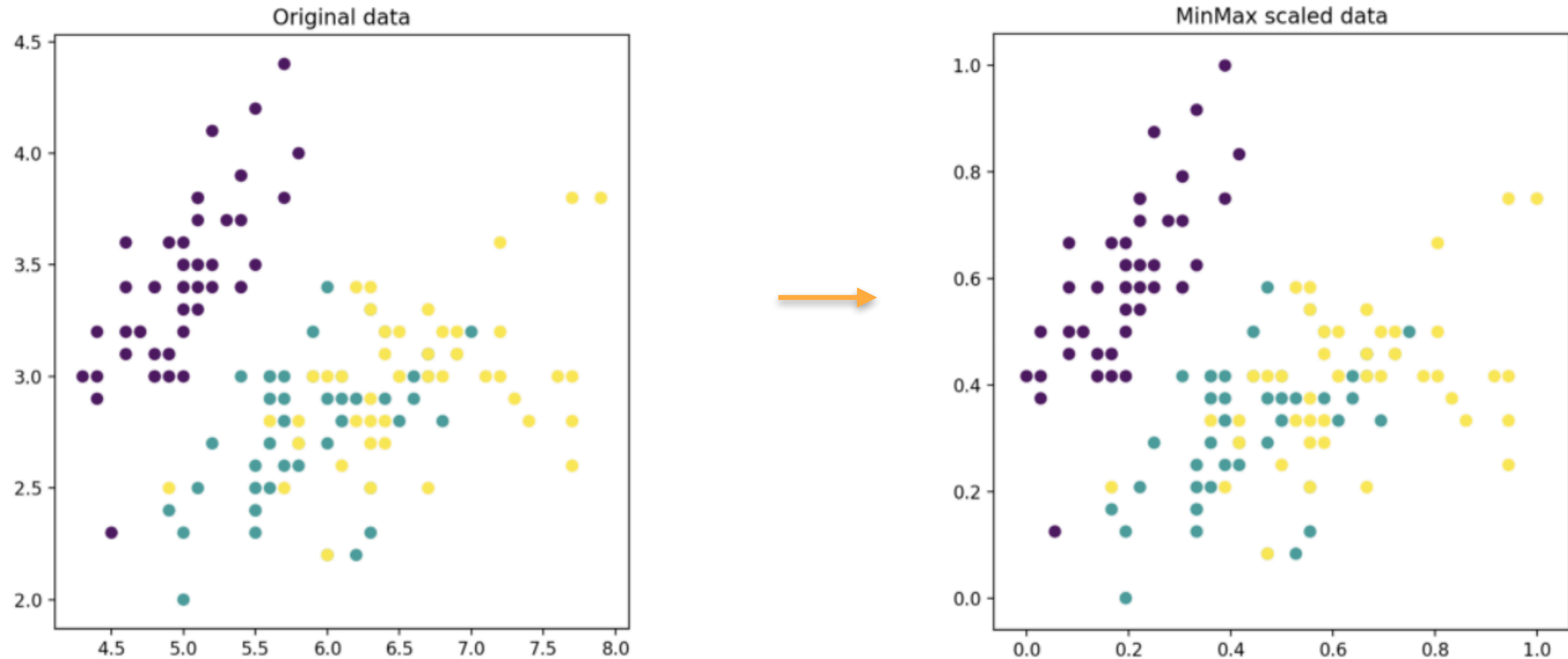
Transformiere Feature x zu x' wie folgt:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Effekt:

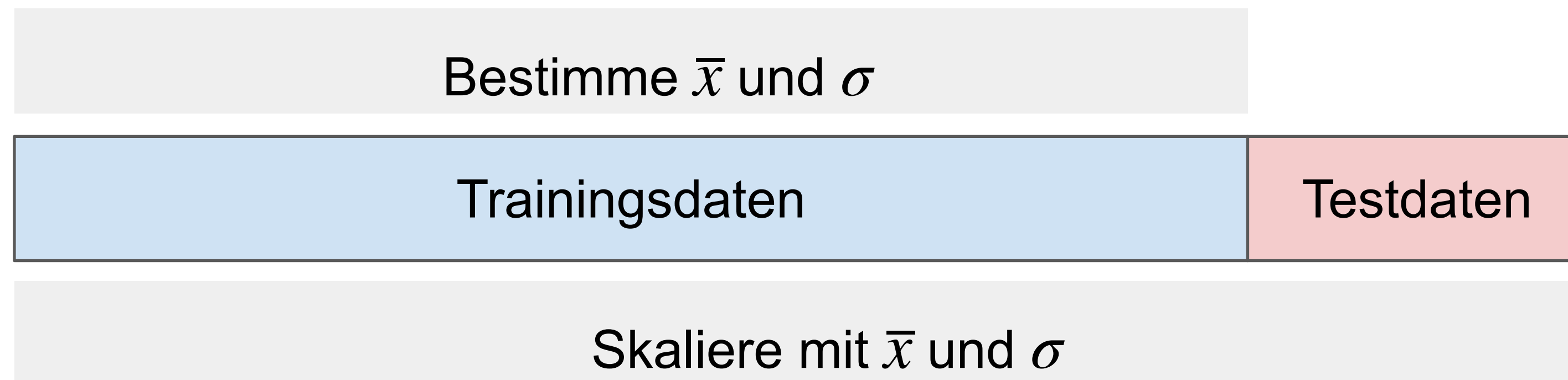
- Das Minimum des Features wird zu einer 0.
- Das Maximum des Features wird zu einer 1.
- Min-Max-Normalisierung ist nicht sehr robust gegen Outlier.
- Daher wird oft Standardisierung bevorzugt.

Feature Scaling - Min-Max-Normalisierung



Feature Scaling

Die Parameter \bar{x} und σ (bzw. $\min(x)$ und $\max(x)$) werden auf den Trainingsdaten bestimmt und werden auf Trainingsdaten und Testdaten zur Skalierung verwendet:



Erinnerung: Testdaten dürfen während des Modelltrainings (= bestimmten der Parameter \bar{x} und σ) nicht verwendet werden.

Datenmatrix

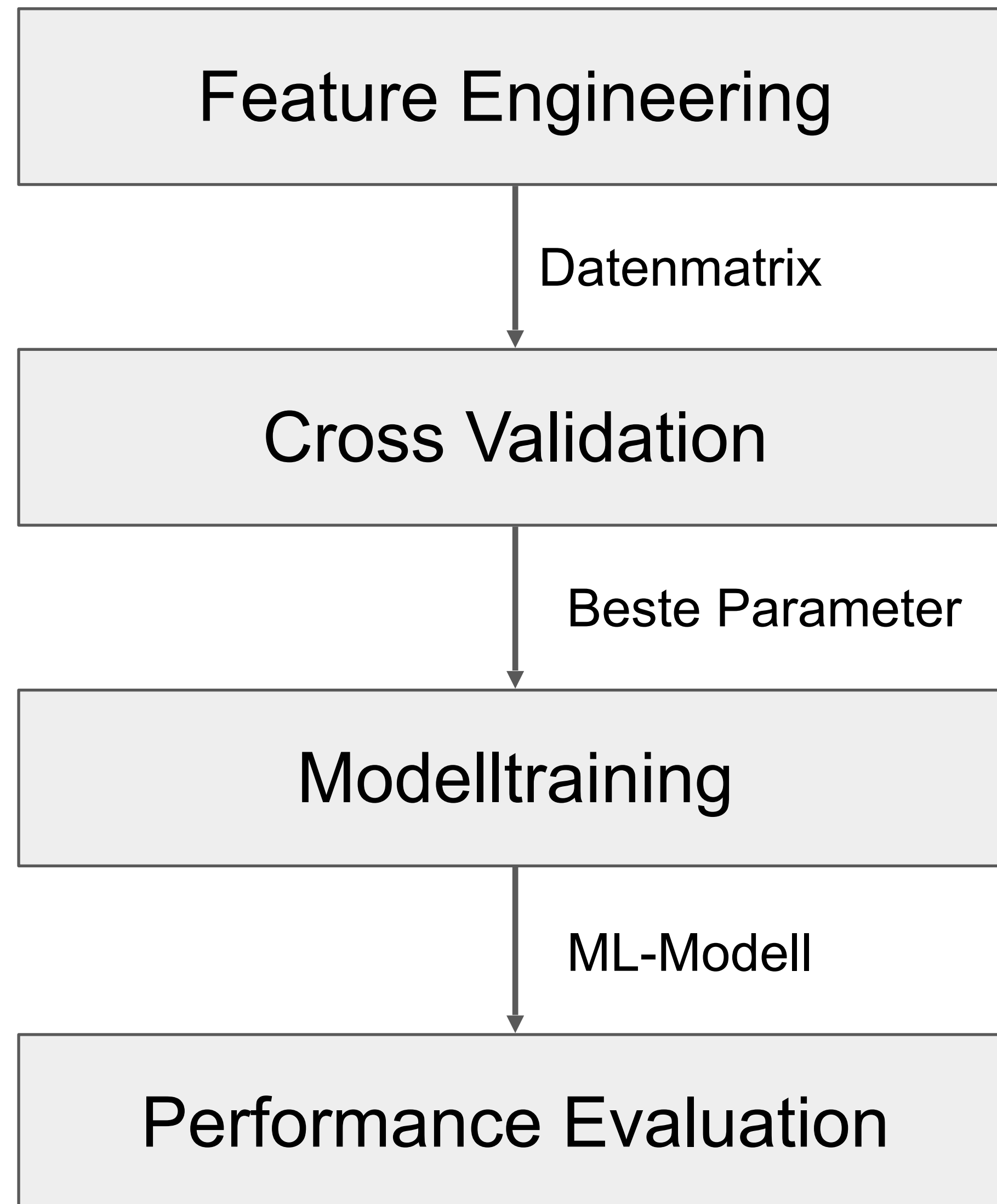
- Das Ergebnis des Feature Engineering ist die Datenmatrix, diese besteht aus den normalisierten Features.
- Die Datenmatrix ist zusammen mit dem Label der direkte Input für das Training des ML-Modells:

Anzahl Artikel	Summe	Artikel < 30€	Teuerster Artikel	Label
0.23	-1.27	-0.91	-0.86	0
-0.62	1.95	0.46	1.99	0
0.01	-0.30	-0.17	0.41	1
...

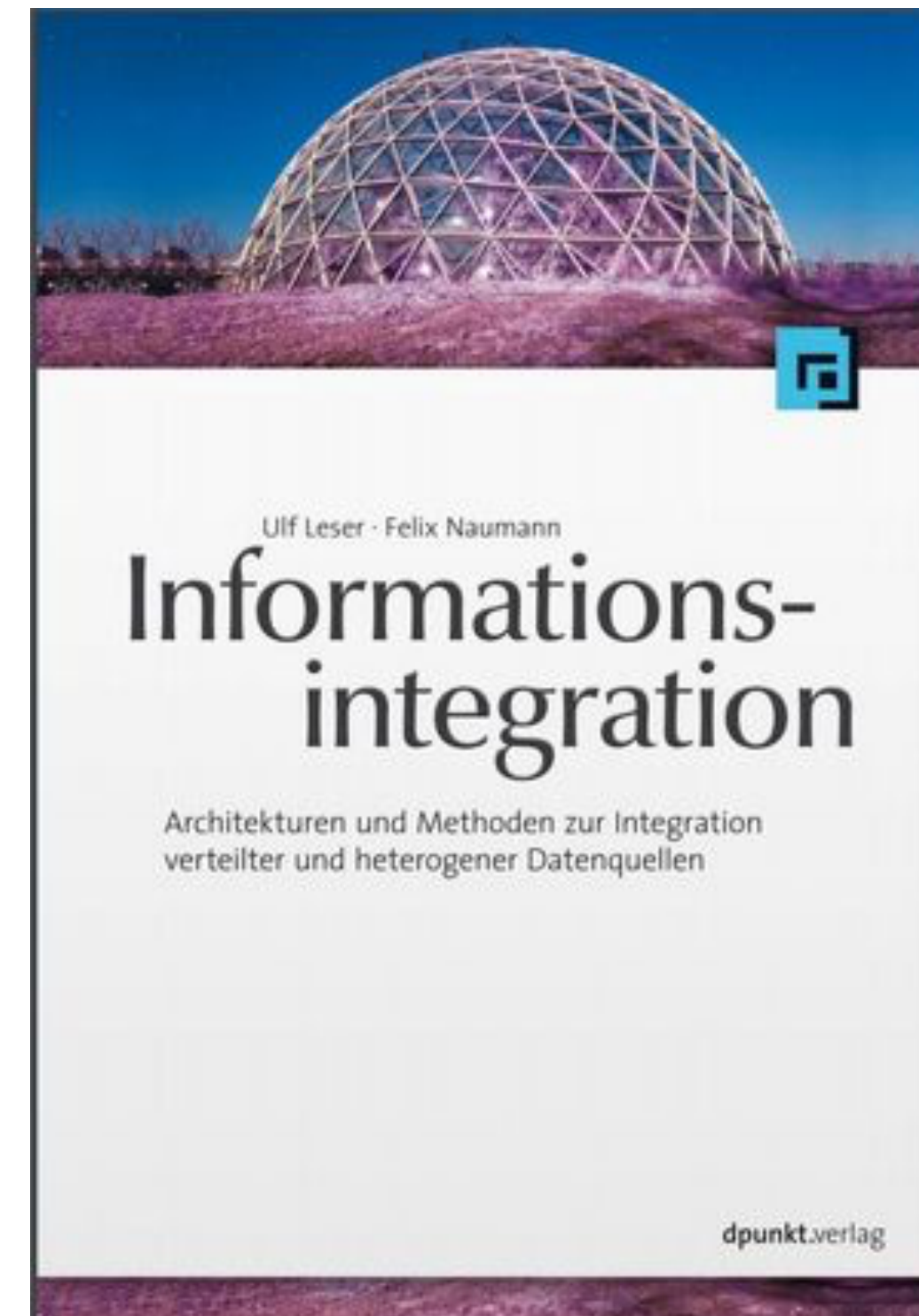
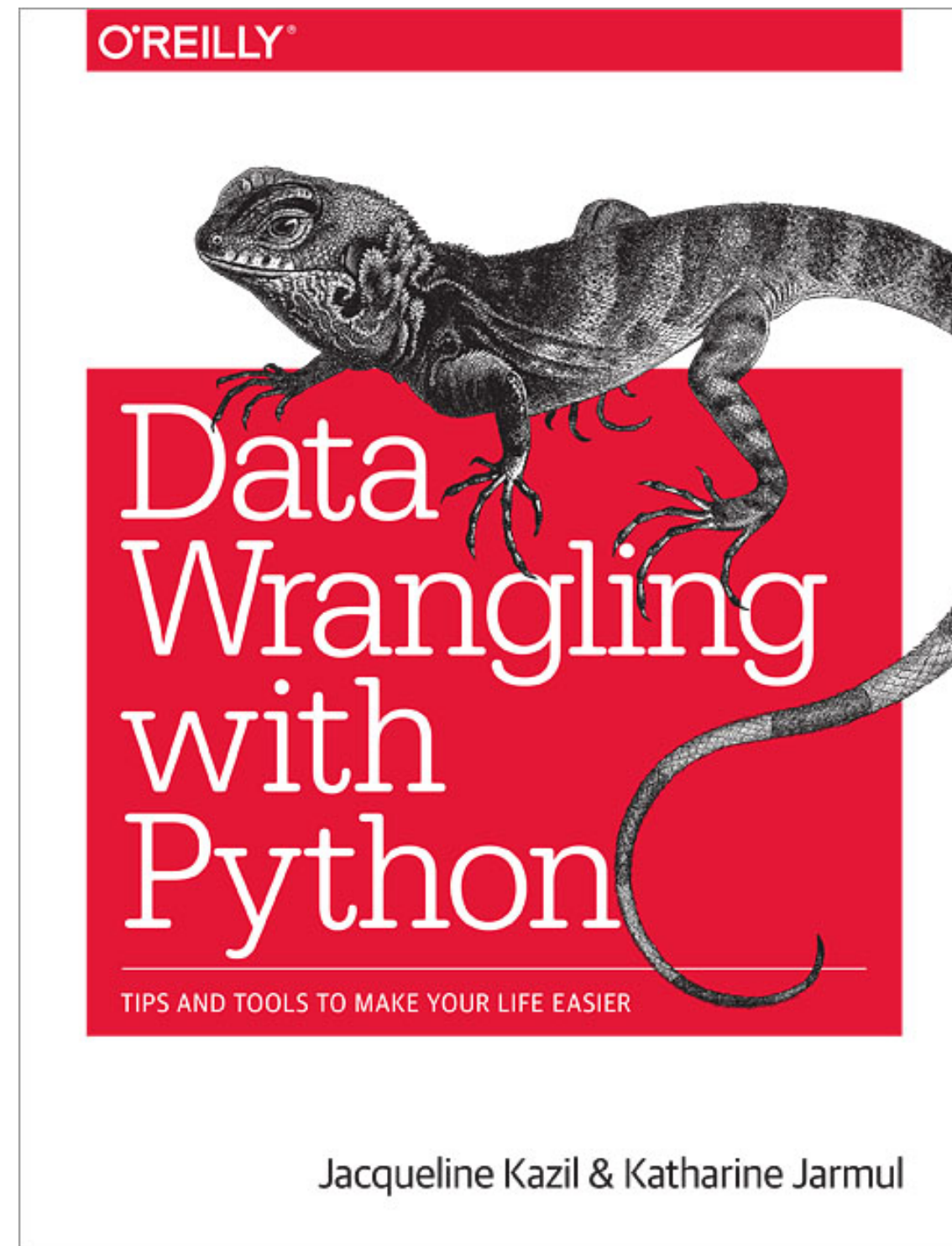
X_{train}

y_{train}

Überblick



Weiterführende Literatur



Fragen?