



# Deep Learning, Machine Learning und Künstliche Intelligenz – SS 25

Gelesen von Prof. Dr. Wolfgang Konen und Prof. Dr. Daniel Gaida

# Projektteams



Gruppe SoSe2025

Gruppe WPF DLML im Sommersemester 2025

Anmeldungsende: Morgen, 05:40 Freie Plätze: 10

Veranstaltungszeitraum: 1. Mär 2025 - 31. Okt 2025

Nachname	Vorname	Studiengang	Thema1	Thema2	Thema3
Al Ahmady	Paiwand	Wirtschaftsinformatik (Bachelor)	-		
Bach	Jonathan Frederic	Wirtschaftsinformatik (Bachelor)	NLP	RL	
Boettcher	Malte Fabian	Informatik (Bachelor)			
Brylev	Kirill Igorevic	Informatik (Bachelor)	Embedded	NLP	KI Ethik
Dieper	Louis Henri	Wirtschaftsinformatik (Bachelor)	Unsupervised	NLP	
Faust	Max	Informatik (Bachelor)	AHE CNN	Gen AI	
Golbik	Jessica Victoria	Wirtschaftsinformatik (Bachelor)			
Gomer	Maximilian	Informatik (Bachelor)	NLP	RL	
Gürtler	Nico	Informatik (Bachelor)	Unsupervised		
Kern	Daniel	?? Technische Hochschule Köln			
Khamkaew	Phithaya	Medieninformatik (Bachelor)			
Khemiri	Karim	Medieninformatik (Bachelor)			
Klesen	Emma Jolien	Informatik (Bachelor)			
Kühnast	Marius	Informatik (Bachelor)	AHE CNN	Gen AI	
Kural	Ali-Enes	Maschinenbau (Bachelor)			
Murza	Kai	Informatik (Bachelor)			
Özkurt	Cihat	Informatik (Bachelor)			
Pivovar	Dimitrij	Informatik (Bachelor)	Embedded	Augmentatio	GenAI
Preußé	Jan	Informatik (Bachelor)	NLP	Embedded	AHE CNN
Schuldeis	Nick	Wirtschaftsinformatik (Bachelor)	NLP	RL	
Schulten	Frederic	Informatik (Bachelor)	KI Ethik	Augmentatio	KI Krieg
Struckmeyer	Nick	Informatik (Bachelor)	AHE CNN	Gen AI	
Uysal	Muhterem Zahide	Abschluss Ausland Wirtschaftsinformatik (Ba.) Wirtschaftsinformatik BA			
Vogt	Paul Julius	Wirtschaftsinformatik (Bachelor)			
Wedler	Kevin	Wirtschaftsinformatik (Bachelor)	Unsupervised	NLP	
Woithe	Adrian Richard	Wirtschaftsinformatik (Bachelor)	AHE CNN	Vision Tr	Augmentation

# Lernziele von heute und Fragen zur Überprüfung der Lernziele

Die Studierenden können ein Machine Learning Projekt selbstständig durchführen, indem sie gegebene Daten visualisieren und für Machine Learning aufbereiten, ein Machine Learning Modell trainieren und dessen Hyperparameter optimieren können, um später eigene Machine Learning Projekte umsetzen zu können.

Überprüfung des Lernziels: S. fortlaufende Übung in diesem Lernraum I.

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Ihre Beteiligung & Zeitplan

- Fragen gerne zwischendurch stellen.
- Zeitplan für heute
- 13-16 Uhr
- Alle 60 Minuten: 5-10 Minuten Pause
  
- Programmierübungen während der Vorlesung.
- Gibt es aktuell Fragen?
- Möchte jemand Protokoll führen?

# Ein Machine Learning Projekt von A bis Z

## Empfehlungen zur Vorgehensweise

- Checkliste für Machine-Learning Projekte aus Geron19 (Anhang B)
- CRISP-DM

## Checkliste für Machine-Learning-Projekte

Diese Checkliste begleitet Sie durch Ihre Machine-Learning-Projekte. Sie besteht aus acht wesentlichen Punkten:

1. Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation.
2. Beschaffen Sie sich Daten.
3. Erkunden Sie die Daten, um daraus Erkenntnisse zu gewinnen.
4. Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können.
5. Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl.
6. Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung.
7. Stellen Sie Ihre Lösung vor.
8. Starten, beobachten und warten Sie Ihr System.

Natürlich sollten Sie diese Checkliste bei Bedarf anpassen.

### Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation

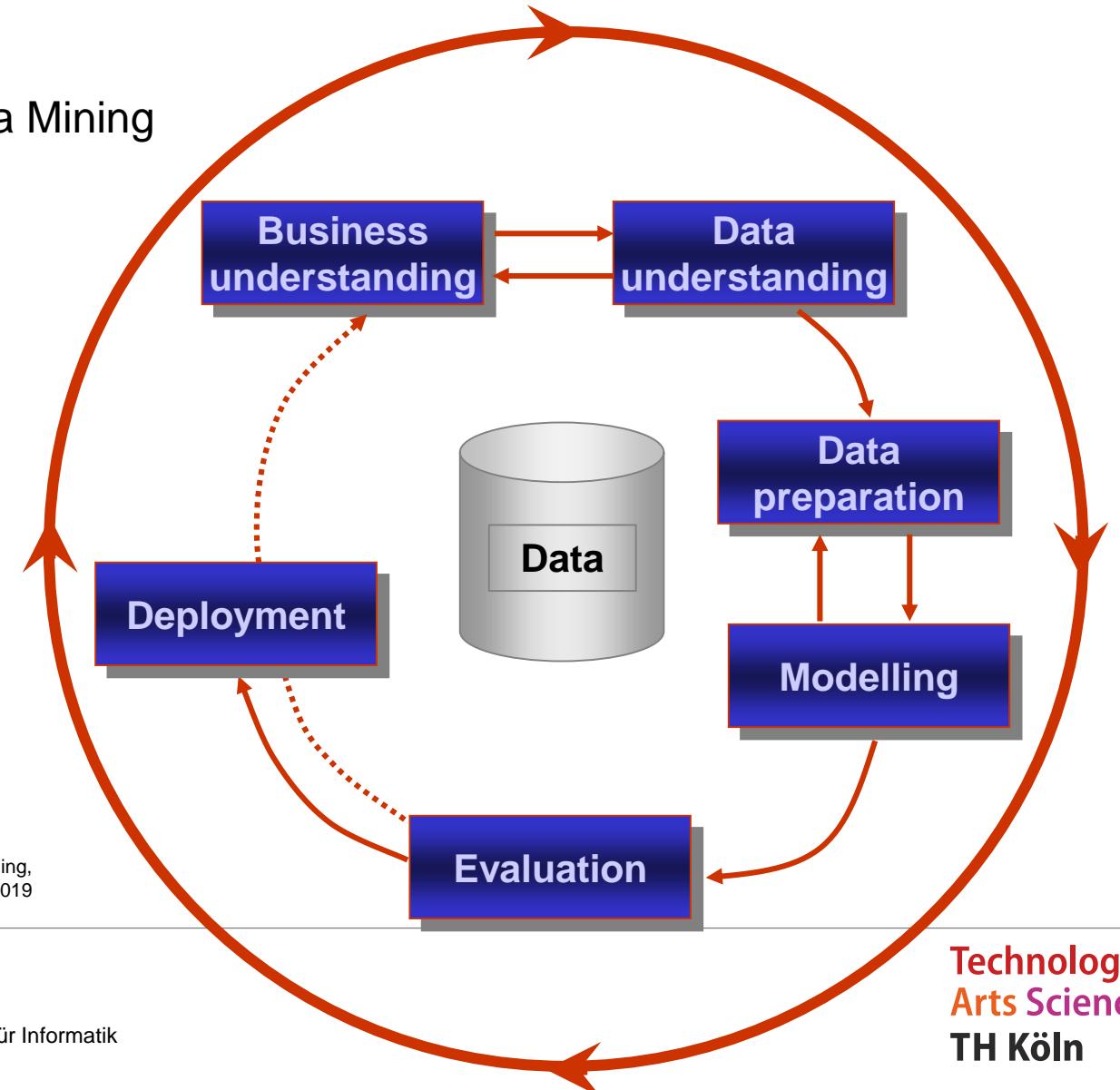
1. Definieren Sie das Geschäftsziel.
2. Wie wird Ihre Lösung eingesetzt werden?
3. Wie sehen die bisherigen Lösungen oder Übergangslösungen aus (falls vorhanden)?
4. In welchen Bereich fällt die Aufgabe (überwacht/unüberwacht, online/offline und so weiter)?
5. Wie wird die Qualität der Lösung gemessen?
6. Liefert das Qualitätsmaß einen Beitrag zum Geschäftsziel?

Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

# Ein Machine Learning Projekt von A bis Z

## CRISP-DM

- Cross-Industry Standard Process for Data Mining
- In dieser Vorlesung gehen wir nach dem CRISP-DM Standard vor, ohne diesen groß zu erwähnen



# Der Umgang mit realen Daten

## Datenquellen

Beliebte Archive mit frei verfügbaren Daten:

- PapersWithCode (<https://paperswithcode.com/datasets>)
- Datensätze von Kaggle (<https://www.kaggle.com/datasets>)
- UC Irvine Machine Learning Repository (<http://archive.ics.uci.edu/ml/>)
- Datensätze von Amazon AWS (<https://registry.opendata.aws/>)
- Roboflow (<https://universe.roboflow.com/>)

Metaseiten (Listen von Datenarchiven):

- Data Portals (<http://dataportals.org/>)
- OpenDataMonitor (<http://opendatamonitor.eu/>)

Andere Seiten, die beliebte offene Datenarchive auflisten:

- Die Wikipedia-Seite mit Machine-Learning-Datensätzen (<https://homl.info/9>)

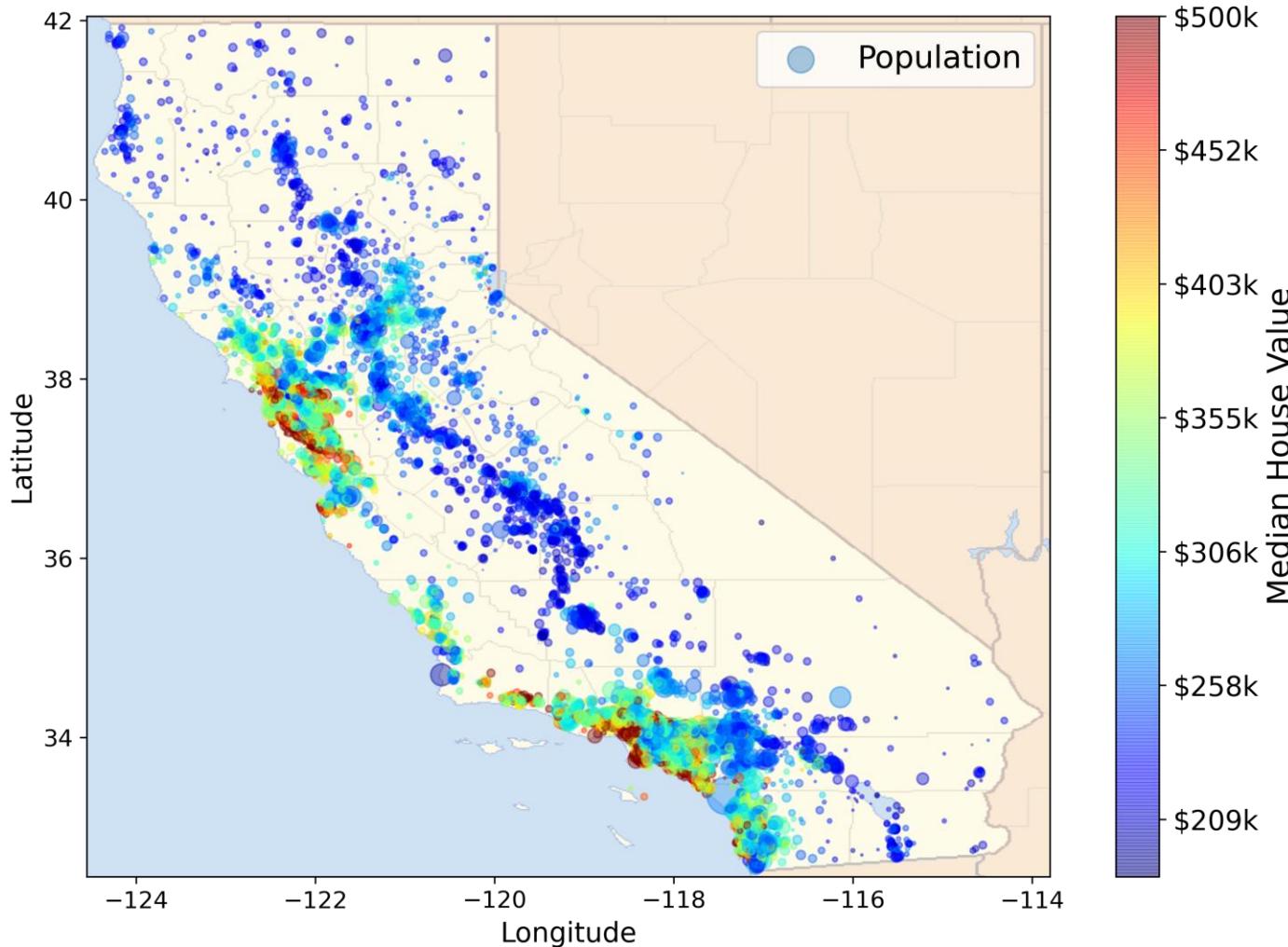
Daten Suchmaschine:

- <https://datasetsearch.research.google.com/>

# Der Umgang mit realen Daten

Datensatz für unser Projekt:

- Immobilienpreise in Kalifornien
  - Lage
  - Mittleres Einkommen
  - Anzahl Räume
  - Einwohnerzahl (Population)
- Alle Daten liegen für jeden Bezirk als Mittelwerte vor
  - Bezirk: ca.: 600 – 3000 Personen



# Der Umgang mit realen Daten

## Ihre freiwillige Aufgabe

- Wenn Sie Ihre Note um 0,3 – 0,7 Punkte verbessern möchten, können Sie sich wie folgt ein Badge holen:
  - Führen Sie die in diesem Lernraum I gezeigten Schritte an einem Datensatz Ihrer Wahl durch.
  - Sie erhalten ein Jupyter Notebook für den Datensatz: „Immobilienpreise in Kalifornien“
    - 02\_end\_to\_end\_machine\_learning\_project\_student.ipynb
  - Ihre Aufgabe: Schreiben Sie ein ähnliches Notebook für Ihren Datensatz
  - Während Vorlesung regelmäßig praktische Übungen:
    - Schritt 1: Verstehen der Schritte für „unseren“ Datensatz
    - Schritt 2: Umsetzen der Schritte für Ihren Datensatz (teilweise auch nach der Vorlesung)

# Der Umgang mit realen Daten

## Ihre freiwillige Aufgabe

Mögliche Datensätze für Ihr Projekt:

- <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/page/25/>
- <https://www.drivendata.org/competitions/57/nepal-earthquake/page/136/>
- <https://www.drivendata.org/competitions/66/flu-shot-learning/page/211/>
- <https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/page/82/> (evtl. schwierig)
  
- <https://www.kaggle.com/c/titanic/overview>
- <https://www.kaggle.com/c/demand-forecasting-kernels-only/overview> (evtl. schwierig)
- <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>
- <https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>
- <https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>
- <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>
  
- <https://archive.ics.uci.edu/ml/datasets/Covtype>
- <https://archive.ics.uci.edu/ml/datasets/breast+cancer>

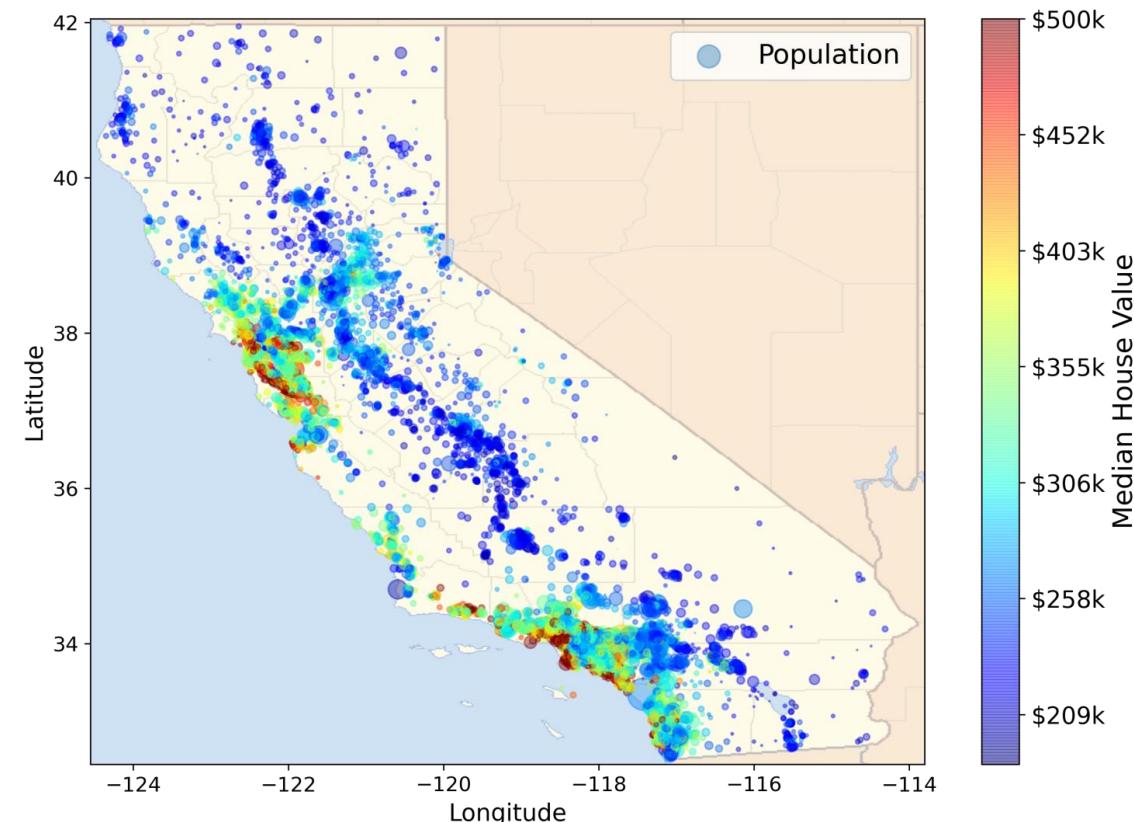
# Fragen?

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - **Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation**
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

- Aufgabe: Vorhersage des Immobilienpreises je Bezirk auf Basis der erhobenen Merkmale je Bezirk (Lage, mittleres Einkommen, Anzahl Räume, ...)
- Die Aufgabe abstecken
- Wähle ein Qualitätsmaß aus
- Überprüfe die Annahmen



# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Die Aufgabe abstecken

- Das Geschäftsziel verstehen
  - Lohnt sich die Investition in Immobilien in bestimmten Bezirken?
- Wie soll das Machine Learning Modell am Ende genutzt werden?
  - Teil eines größeren Systems?
- Wie wird es ausgerollt und mit welchen anderen Software-Komponenten soll es zusammen arbeiten?
  - Zugriff auf eine bestimmte Datenbank,
  - Integration des Modells in eine inhouse entwickelte Software, ...
  - Mobile Nutzung?
- Wie sieht die aktuelle Lösung aus (falls vorhanden)?
  - Liefert u.A. Hinweise auf aktuelle Vorhersagegenauigkeiten
- Handelt es sich bei der Aufgabe um überwachtes, unüberwachtes oder Reinforcement Learning?
- Ist es eine Klassifikationsaufgabe, eine Regressionsaufgabe oder etwas anderes?

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation Notation in dieser Vorlesung

- Unser Datensatz hat m Datenpunkte:  $i = 1, \dots, m$
- Jeder Datenpunkt besteht aus mehreren Merkmalen

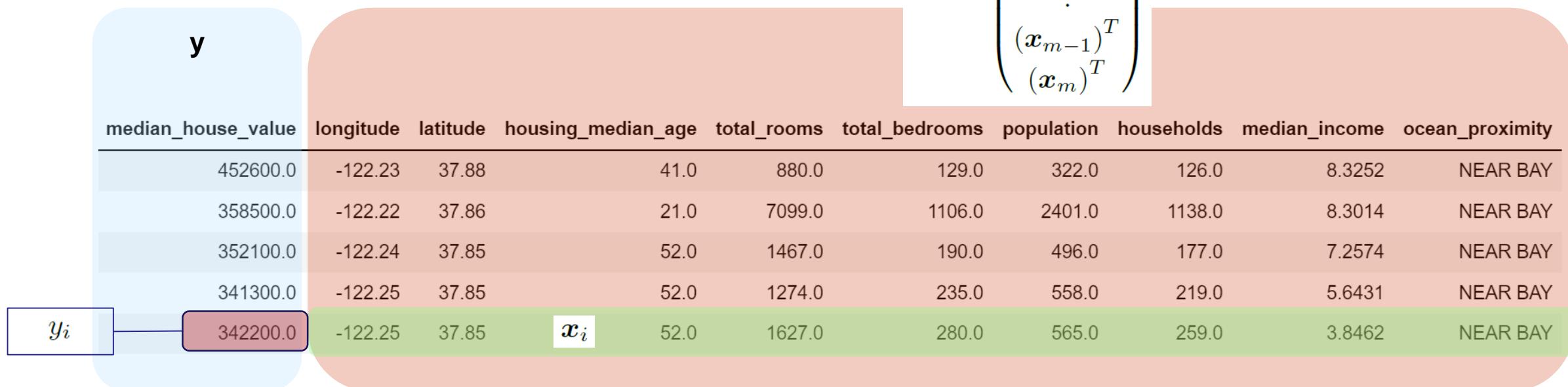
**Label** → **Merkmal**

	median_house_value	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
	452600.0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	NEAR BAY
<b>Label</b>	358500.0	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	NEAR BAY
	352100.0	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	NEAR BAY
<b>Datenpunkt</b>	341300.0	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	NEAR BAY
	342200.0	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	NEAR BAY

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation Notation in dieser Vorlesung

- Unser Datensatz hat m Datenpunkte:  $i = 1, \dots, m$

$$\mathbf{X} := \begin{pmatrix} (\mathbf{x}_1)^T \\ (\mathbf{x}_2)^T \\ \vdots \\ (\mathbf{x}_{m-1})^T \\ (\mathbf{x}_m)^T \end{pmatrix}$$



The diagram illustrates a dataset  $\mathbf{X}$  represented as a matrix with 5 columns and 5 rows. The columns are labeled: median\_house\_value, longitude, latitude, housing\_median\_age, total\_rooms, total\_bedrooms, population, households, median\_income, and ocean\_proximity. The rows represent individual data points. A specific row  $i$  is highlighted in green, and its value  $y_i$  is shown in a blue box.

$\mathbf{y}$	median_house_value	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
	452600.0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	NEAR BAY
	358500.0	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	NEAR BAY
	352100.0	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	NEAR BAY
	341300.0	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	NEAR BAY
$y_i$	342200.0	-122.25	37.85	$\mathbf{x}_i$	52.0	1627.0	280.0	565.0	3.8462	NEAR BAY

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation Notation in dieser Vorlesung

$m$  ist die Anzahl der Datenpunkte im Datensatz

$\mathbf{x}_i$  ist ein Vektor der Werte aller Merkmale (ohne das Label) des  $i$ ten Datenpunkts im Datensatz, und  $y_i$  ist das dazugehörige Label (der gewünschte Ausgabewert für diesen Datenpunkt).

$\mathbf{X}$  ist eine Matrix mit den Werten sämtlicher Merkmale (ohne Labels) für alle Datenpunkte im Datensatz. Pro Datenpunkt gibt es eine Zeile, und die  $i$ te Zeile entspricht der transponierten Form von  $\mathbf{x}_i$ , auch als  $(\mathbf{x}_i)^T$  geschrieben.

$$\mathbf{X} := \begin{pmatrix} (\mathbf{x}_1)^T \\ (\mathbf{x}_2)^T \\ \vdots \\ (\mathbf{x}_{m-1})^T \\ (\mathbf{x}_m)^T \end{pmatrix}$$

Formel 1-1: Ein einfaches lineares Modell

Zufriedenheit =  $\theta_0 + \theta_1 \times \text{BIP\_pro\_Kopf}$

$h$  ist die Vorhersagefunktion des Systems, auch Hypothese genannt. Wenn das System den Merkmalsvektor eines Datenpunkts  $\mathbf{x}_i$  erhält, gibt es den Vorhersagewert  $\hat{y}_i = h(\mathbf{x}_i)$  für diesen Datenpunkt aus.

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

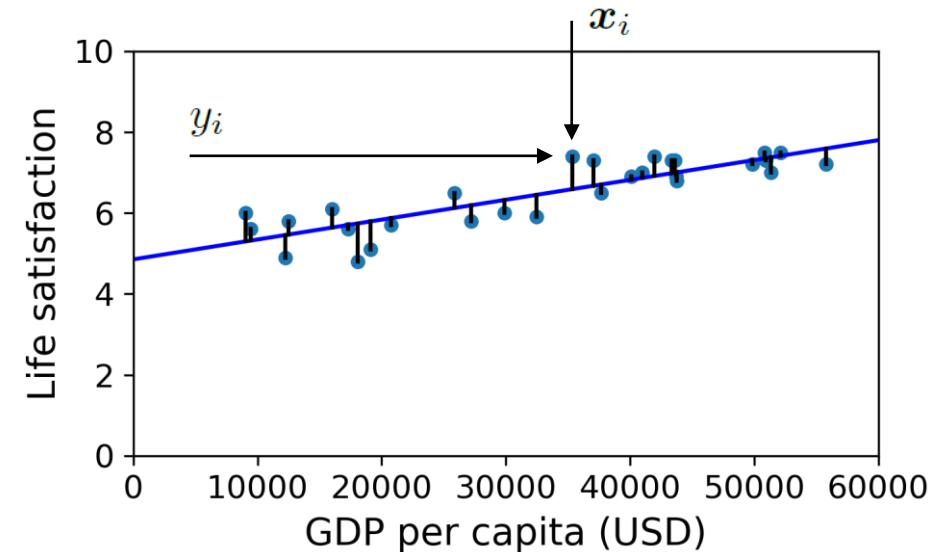
## Wähle ein Qualitätsmaß aus

- Wie soll die Qualität der Modellvorhersagen (Vorhersagefehler) gemessen werden?
- Stimmen diese mit dem Geschäftsziel überein?
- RMSE (root-mean-square error, Wurzel der mittleren quadratischen Abweichung)

$$\text{RMSE}(\mathbf{X}, h) := \sqrt{\frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}_i) - y_i)^2}$$

- MAE (mittlerer absoluter Fehler)
  - Robuster gegenüber Ausreißern

$$\text{MAE}(\mathbf{X}, h) := \frac{1}{m} \sum_{i=1}^m |h(\mathbf{x}_i) - y_i|$$



# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

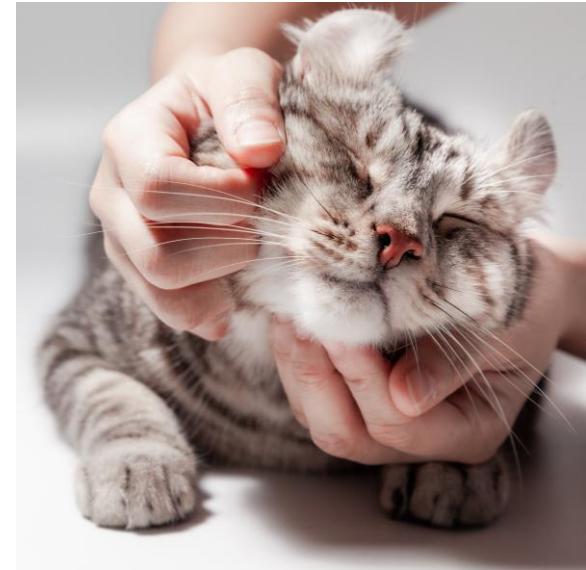
## Überprüfe die Annahmen

- Wie werden die Vorhersagen des ML Modells von nachgeschalteter Software genutzt?
  - Werden Preise genutzt oder nur Kategorien wie günstig, mittel und teuer?
  - Falls ja, dann ...
    - Problem als Klassifikationsproblem formulieren
- Stimmen Sie Ihre getroffenen Annahmen mit dem Auftraggeber ab.
  
- Was müssten wir bis hierhin ändern, wenn das Problem ein Klassifikationsproblem wäre?
  - Die Aufgabe abstecken?
  - Notation?
  - Wähle ein Qualitätsmaß aus?

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Binärer Klassifikator für Katzen

Klassifizieren Sie alle Katzen



# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Konfusionsmatrix

Perfekter Klassifikator:



		ML system says	
		Cat	No Cat
Truth	Cat	4 	0
	No Cat	0	2 

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Konfusionsmatrix



		ML system says	
		Cat	No Cat
Truth	Cat	Richtig	Falsch
	No Cat	Falsch	Richtig

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Konfusionsmatrix



		ML system says	
		Cat	No Cat
Truth	Cat	Positive	Negative
	No Cat	Positive	Negative

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Konfusionsmatrix



		ML system says	
		Cat	No Cat
Truth	Cat	Richtig Positive	Falsch Negative
	No Cat	Falsch Positive	Richtig Negative

# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus – Konfusionsmatrix

Genauigkeit = 4/6 = 66 %



		ML system says	
		Cat	No Cat
Truth	Cat	3 	1 
	No Cat	1 	1 

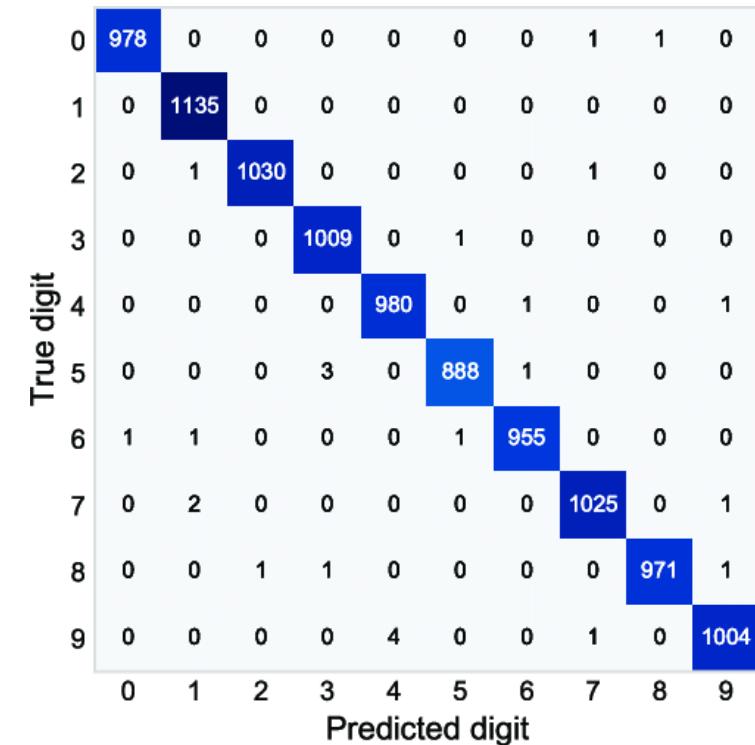
# Klären Sie Aufgabenstellung und betrachten Sie Gesamtsituation

## Wähle ein Qualitätsmaß aus

0 0  
1 1  
2 2  
3 3  
4 4  
5 5  
6 6  
7 7  
8 8  
9 9

- Binäres Klassifikationsproblem:
  - Genauigkeit =  $(RP + RN) / (RP + RN + FP + FN)$
- Multinomiales Klassifikationsproblem (mehrere Klassen):
  - Konfusionsmatrix
  - Genauigkeit pro Klasse =  
Anzahl richtig klassifiziert / Anzahl Datenpunkte in Klasse
  - Genauigkeit für Datensatz =  
Anzahl richtig klassifiziert / Anzahl Datenpunkte in Datensatz

RP	FN
FP	RN



# Fragen?

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - **Beschaffen Sie sich Daten**
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

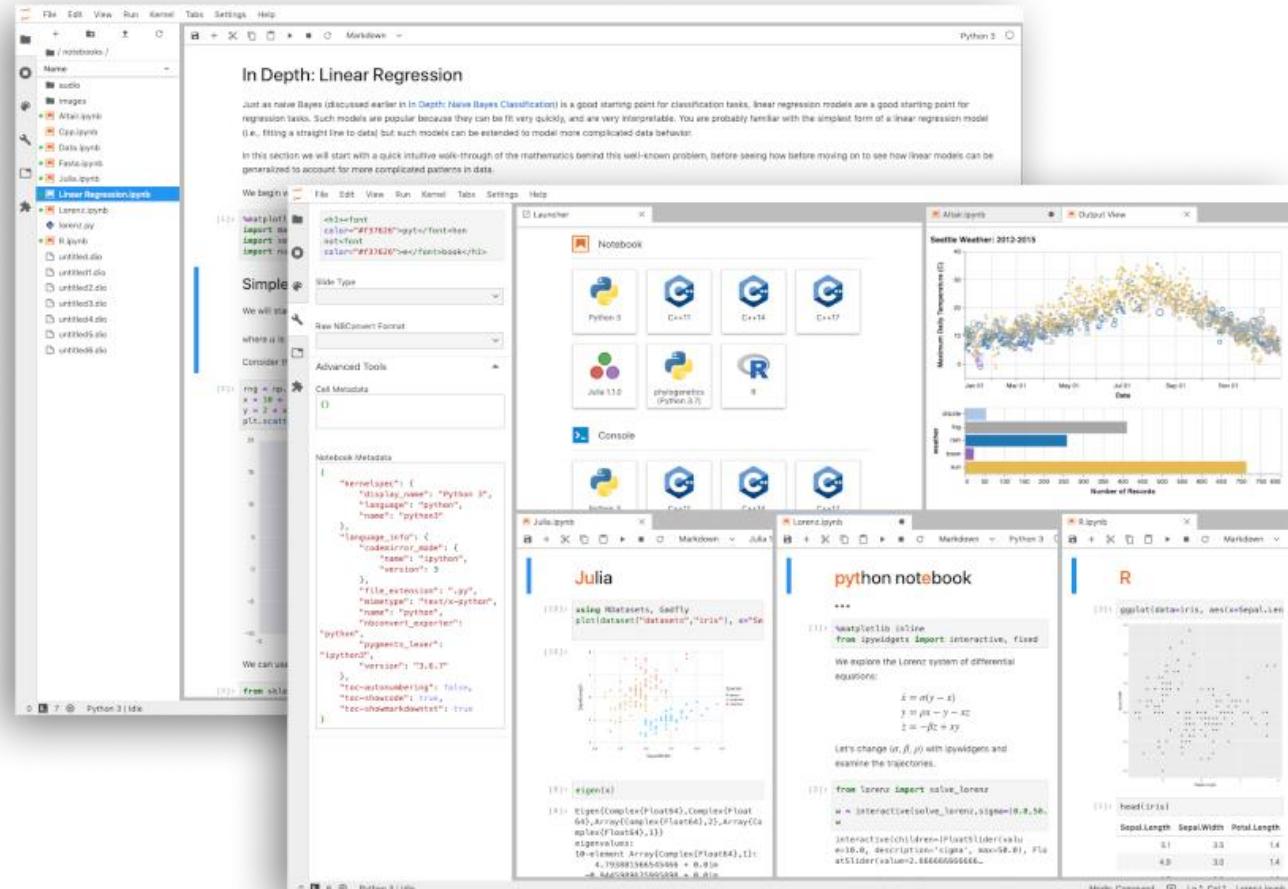
# Beschaffen Sie sich Daten

- Erstelle eine Arbeitsumgebung
- Die Daten herunterladen
- Einen kurzen Blick auf die Datenstruktur werfen
- Erstelle einen Testdatensatz

# Beschaffen Sie sich Daten

## Erstelle eine Arbeitsumgebung

- Lokale Installation:
  - Anaconda (oder Miniconda),  
<https://www.anaconda.com/products/individual>
  - Enthält Python und viele Standard Bibliotheken
  - Jupyter Notebook oder Jupyter Lab
  - Pandas, Scikit-learn, Matplotlib
  - Später: TensorFlow
  - environment.yml in ILU
- Cloud Umgebung:
  - Google Colab



# Beschaffen Sie sich Daten

## Erstelle eine Arbeitsumgebung

- Mit diesen Bibliotheken arbeiten wir in diesem Kurs:
  - Jupyter Notebook oder Jupyter Lab
    - Praktisch, um schnell und im Browser Python zu programmieren
  - Pandas
    - Der Standard, um strukturierte Daten (bspw. Tabellen) in Python zu verarbeiten
  - Matplotlib, Seaborn
    - Visualisierung in Python
  - Scikit-learn
    - Die Standardbibliothek für Machine Learning in Python
  - Später: TensorFlow
    - Neben PyTorch der Standard, um tiefe neuronale Netze zu trainieren

# Beschaffen Sie sich Daten

## Erstelle eine Arbeitsumgebung

- Es empfiehlt sich mit einer standardisierten und logischen Ordner- und Dateistruktur zu arbeiten
  - Übersichtlichkeit – Alle die die Struktur kennen, wissen wo sie etwas finden
  - Reproduzierbarkeit – Auch noch nach ein paar Jahren
  - Wiederverwendbarkeit – Klassen und Methoden können in neuen Projekten genutzt werden
- Cookiecutter kann solch eine Ordnerstruktur erstellen
  - <https://cookiecutter.readthedocs.io/en/1.7.2/>
- Gutes Template für Machine Learning Projekte:
  - <http://drivendata.github.io/cookiecutter-data-science/>
- Eine Empfehlung, kein Muss

```
src           <- Source code for use in this project.  
├── __init__.py    <- Makes src a Python module  
|  
└── data          <- Scripts to download or generate data  
    └── make_dataset.py  
  
    └── features      <- Scripts to turn raw data into features for modeling  
        └── build_features.py  
  
    └── models         <- Scripts to train models and then use trained models to make  
        |  
        |   predictions  
        └── predict_model.py  
        └── train_model.py  
  
    └── visualization <- Scripts to create exploratory and results oriented visualizations  
        └── visualize.py
```

# Beschaffen Sie sich Daten

## Erstelle eine Arbeitsumgebung

- Cookiecutter Template für ML Projekte:
  - <http://drivendata.github.io/cookiecutter-data-science/>
- Eine Empfehlung, kein Muss

```

LICENSE                                <- Makefile with commands like `make data` or `make train`
Makefile                               <- The top-level README for developers using this project.
README.md
data
  external                            <- Data from third party sources.
  interim                             <- Intermediate data that has been transformed.
  processed                           <- The final, canonical data sets for modeling.
  raw                                 <- The original, immutable data dump.

docs                                    <- A default Sphinx project; see sphinx-doc.org for details

models                                  <- Trained and serialized models, model predictions, or model summaries

notebooks                               <- Jupyter notebooks. Naming convention is a number (for ordering),
                                         the creator's initials, and a short ``-`` delimited description, e.g.
                                         `1.0-jqp-initial-data-exploration`.

references                            <- Data dictionaries, manuals, and all other explanatory materials.

reports
  figures                            <- Generated analysis as HTML, PDF, LaTeX, etc.
                                         <- Generated graphics and figures to be used in reporting

requirements.txt                         <- The requirements file for reproducing the analysis environment, e.g.
                                         generated with `pip freeze > requirements.txt`

setup.py                                <- Make this project pip installable with `pip install -e`
src
  __init__.py                          <- Source code for use in this project.
                                         <- Makes src a Python module

  data                                 <- Scripts to download or generate data
    make_dataset.py

  features                             <- Scripts to turn raw data into features for modeling
    build_features.py

  models                               <- Scripts to train models and then use trained models to make
                                         predictions
    predict_model.py
    train_model.py

  visualization <- Scripts to create exploratory and results oriented visualizations
    visualize.py

tox.ini                                 <- tox file with settings for running tox; see tox.readthedocs.io

```

# Beschaffen Sie sich Daten

## Die Daten herunterladen

- Am Besten alle Schritte automatisieren, um diese auf neue Daten anwenden zu können
- Funktion schreiben, zum Herunterladen der Daten
  - `urllib.request.urlretrieve()`
- Daten in pandas einlesen
- Was ist pandas (Python and Data Analysis)?
  - Pandas ist ein leistungsstarkes Python Datenanalyse Toolkit, das auf NumPy aufbaut und als Bibliothek in Python verfügbar ist
  - Im Mittelpunkt von Pandas stehen DataFrame-Objekte. Als Excel-Tabelle vorstellbar.

# Beschaffen Sie sich Daten

## Die Daten in Python importieren

Im Folgenden: **Python's Datenanalyse Toolkit Pandas (Python and Data Analysis)**

```
1 import pandas as pd  
2  
3 df_csv = pd.read_csv("../DeinPfad/Datei.csv")  
4 df_excel = pd.read_excel("../DeinPfad/Datei.xlsx")
```

```
1 import pandas as np  
2 from pymongo import MongoClient  
3  
4 connect_string='mongodf://USER://:PW@DBHOST/DB'  
5 conn = MongoClient(connect_string)  
6 db = conn['DB'] # äquivalent zu conn.DB  
7 collection = db['test_collection'] # äquivalent zu db.test_collection  
8 df.nosql = pd.DataFrame(list(collection.find()))
```

```
1 import pandas as pd  
2 import sqlalchemy as sql  
3  
4 connect_string = 'mysql://USER:PW@DBHOST/DB'  
5 sql_engine = sql.create_engine(connect_string)  
6  
7 df_sql_query = pd.read_sql_query(query, sql_engine)  
8 dt_sql_table = pd.read_sql_table("name", con=sql_engine)
```

# Beschaffen Sie sich Daten

## Übung – Die Daten in Python importieren

- Laden Sie die Daten herunter und lesen Sie die Daten mit pandas in Python ein.
- 10 Min.

### 1.1. Download the Data

```
from pathlib import Path
import pandas as pd
import tarfile
import urllib.request
import numpy as np

def load_housing_data():
    tarball_path = Path("datasets/housing.tgz")
    if not tarball_path.is_file():
        Path("datasets").mkdir(parents=True, exist_ok=True)
        url = "https://github.com/ageron/data/raw/main/housing.tgz"
        urllib.request.urlretrieve(url, tarball_path)
        with tarfile.open(tarball_path) as housing_tarball:
            housing_tarball.extractall(path="datasets")
    return pd.read_csv(Path("datasets/housing/housing.csv"))

housing = load_housing_data()
```

# Kurze Pause?

- Ja, 5 Minuten
  - Ja, 10 Minuten
  - Nein
- 
- In ILU Kurs finden sich Handreichungen zu TH-KI-GPT-Lab vom ZLE



[THKI\\_GPT-Lab\\_Handout\\_Studierende\\_eng\\_v01](#)

pdf 400,1 KB Heute, 20:00 Anzahl Seiten: 3



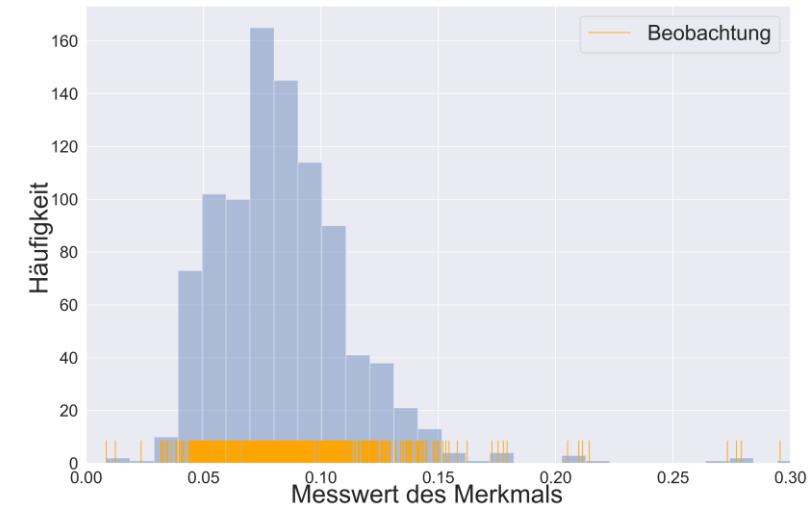
[THKI\\_GPT-Lab\\_Handreichung\\_Studierende\\_v03](#)

pdf 402,2 KB Heute, 20:00 Anzahl Seiten: 3

# Beschaffen Sie sich Daten

## Einen kurzen Blick auf die Datenstruktur werfen

- Nützliche Methoden eines DataFrame Objekts
  - head() → Gibt die ersten n Zeilen des DataFrames aus
  - info() → Gibt Anzahl Zeilen, Datentyp des Attributs und Anzahl Werte ungleich null aus
  - value\_counts() → Gibt Anzahl unterschiedlicher Kategorien eines kategorischen Merkmals aus
  - describe() → Gibt numerische Zusammenfassung über alle numerischen Merkmale (mean, std, min, max, ...) aus
- Histogramme (hist())
  - Stellt die Häufigkeiten eines Merkmals grafisch dar
  - Gibt Aufschluss über die tatsächliche Verteilung (gleichverteilt, gaußverteilt, ...)



# Beschaffen Sie sich Daten

## Übung - Einen kurzen Blick auf die Datenstruktur werfen

- Machen Sie sich mit dem Datensatz vertraut unter Nutzung der Methoden:

- head()
- info()
- value\_counts()
- describe()
- hist()

```
In [5]: housing = load_housing_data()
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

- 10 Min.

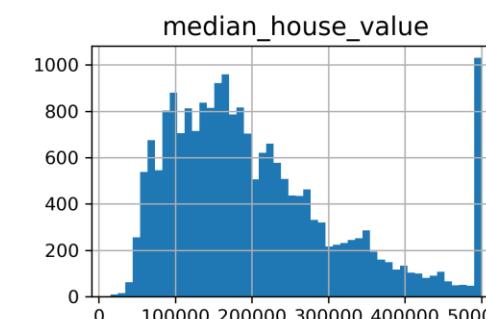
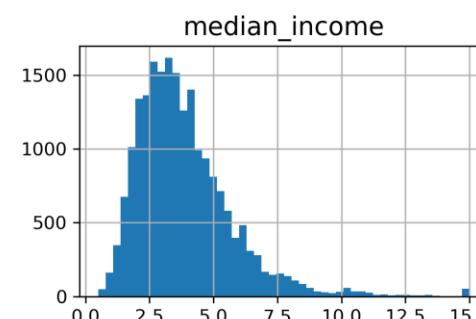
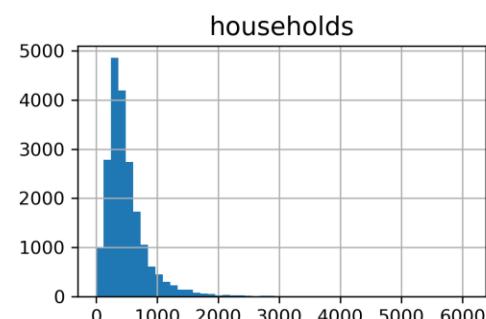
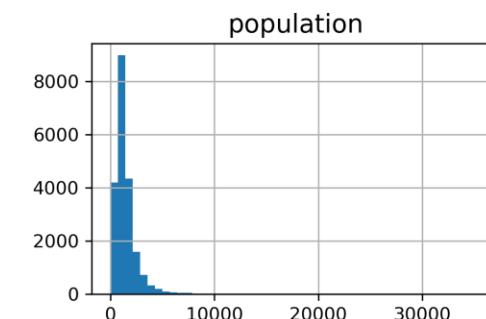
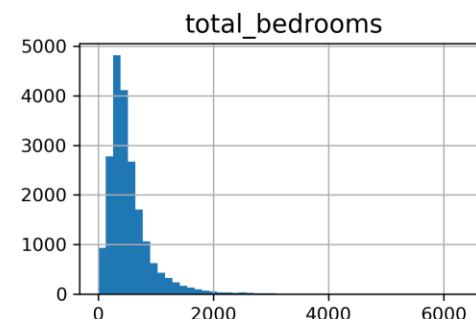
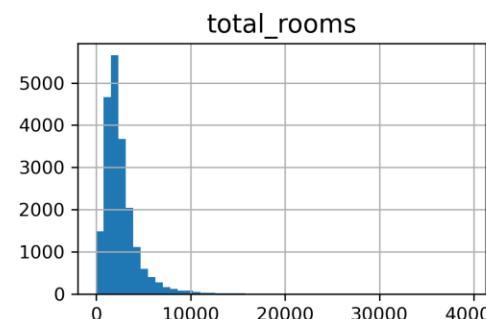
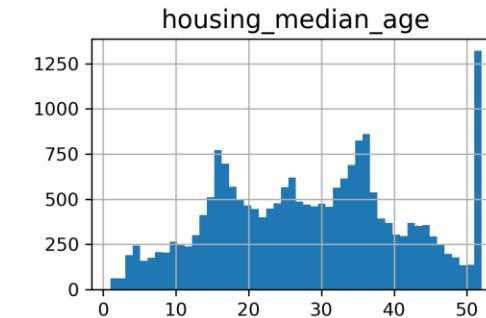
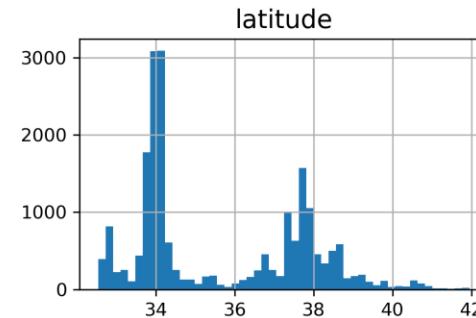
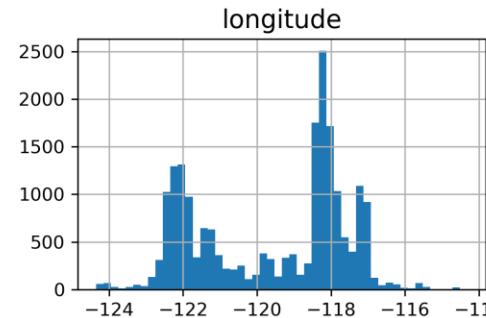
```
In [6]: housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   longitude        20640 non-null   float64
 1   latitude         20640 non-null   float64
 2   housing_median_age 20640 non-null   float64
 3   total_rooms      20640 non-null   float64
 4   total_bedrooms   20433 non-null   float64
 5   population       20640 non-null   float64
 6   households       20640 non-null   float64
 7   median_income    20640 non-null   float64
 8   median_house_value 20640 non-null   float64
 9   ocean_proximity  20640 non-null   object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

# Beschaffen Sie sich Daten

## Einen kurzen Blick auf die Datenstruktur werfen

- Was ist Ihnen bei den Histogrammen aufgefallen?



# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz

### Wichtig!

- Bevor Sie sich den Datensatz im Detail anschauen, trennen Sie diesen in zwei disjunkte Teile:
  - Trainingsdatensatz (ca. 70 % der Daten) trainiert das Machine Learning Modell
  - Testdatensatz (ca. 30 % der Daten) testet das Machine Learning Modell
- Bei sehr großen Datensätzen (> 1 Mio. gelabelte Daten), darf der Testdatensatz auch kleiner ausfallen (5 %)

### Wichtig!

- Testdatensatz nicht weiter anschauen, sondern wegschließen und vergessen... ;-)
- Test auf Testdaten darf nur **einmal** erfolgen, da man sonst auf dem Testdatensatz optimiert

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz

### Merke:

- Testdatensatz muss den erwarteten Datensatz möglichst gut repräsentieren
- Wenn Machine Learning Modelle in der Praxis nicht den erwünschten Erfolg liefern, liegt dieses meistens an der **falschen Wahl des Testdatensatzes!**

### Einfache Fehler:

- Testdatensatz einer Bohrmaschine enthält nur Daten mit niedrigen Drehzahlen, Trainingsdatensatz nur mit hohen Drehzahlen
- Trainingsdatensatz für ein Hotel enthält nur Daten aus dem Sommer
- Das funktioniert schon beim Testen des Machine Learning Modells nicht

**ABER ACHTUNG: ES GIBT AUCH WEITERE FEHLERQUELLEN!**

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz - Beispiel

- Ein Maschinenhersteller möchte für seine Kunden den Ausfall seiner Maschinen vorhersagen (vorausschauende Wartung)
- Hersteller hat Daten von 8 seiner Kunden, welche je 4-5 seiner (baugleichen) Maschinen betreiben
- Seine Kunden möchten mit einem ML-Modell für alle deren Maschinen (und zukünftige Maschinen) zukünftige Ausfälle vorhersagen.
- Wie könnten Sie die Daten in Trainings- und Testdaten aufteilen?
- Bitte gehen Sie zu [www.menti.com](http://www.menti.com) und nutzen den Code 43 75 16

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz - Beispiel



- Seine Kunden möchten mit einem ML-Modell für alle deren Maschinen (und zukünftige Maschinen) zukünftige Ausfälle vorhersagen. Wie könnten Sie die Daten in Trainings- und Testdaten aufteilen?

Testdaten: Von einem Kunden randomisiert ausgewählte Maschinendaten (ca. 30%). Trainingsdaten: Verbliebene Maschinendaten des gleichen Kunden

Als Testdaten nehmen Sie von einem Kunden die Daten von 2 Maschinen und als Trainingsdaten die Daten der verbliebenen 3 Maschinen des gleichen Kunden

Als Testdaten nehmen Sie von jedem Kunden die Daten von 1-2 Maschinen und als Trainingsdaten die Daten der verbliebenen 2-3 Maschinen jedes Kunden

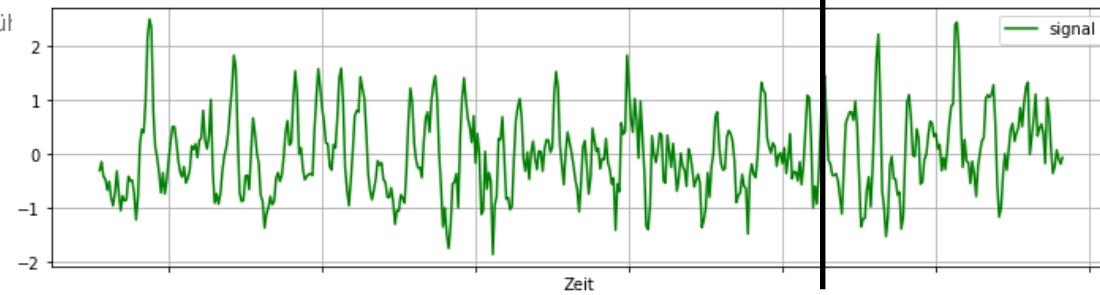
Testdaten: Von jedem Kunden nehmen Sie die neuesten 30% aller Maschinendaten. Trainingsdaten: Verbliebene Maschinendaten aller Kunden

Auf keinen Fall

Ja, auf jeden Fall

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz - Beispiel



- Seine Kunden möchten mit einem ML-Modell für alle deren Maschinen (und zukünftige Maschinen) zukünftige Ausfälle vorhersagen. Wie könnten Sie die Daten in Trainings- und Testdaten aufteilen?

Testdaten: Von einem Kunden randomisiert ausgewählte Maschinendaten (ca. 30%). Trainingsdaten: Verbliebene Maschinendaten des gleichen Kunden

Als Testdaten nehmen Sie von einem Kunden die Daten von 2 Maschinen und als Trainingsdaten die Daten der verbliebenen 3 Maschinen des gleichen Kunden

Als Testdaten nehmen Sie von jedem Kunden die Daten von 1-2 Maschinen und als Trainingsdaten die Daten der verbliebenen 2-3 Maschinen jedes Kunden

Testdaten: Von jedem Kunden nehmen Sie die neuesten 30% aller Maschinendaten. Trainingsdaten: Verbliebene Maschinendaten aller Kunden

Auf keinen Fall

Ja, auf jeden Fall

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz

Warum sollten bei Zeitreihen die Testdaten neuer sein als die Trainingsdaten?

- Trainiertes Modell soll im Einsatz funktionieren, d.h. für zukünftige Daten.
- Deshalb sollten die Testdaten sicherheitshalber neuer als die Trainingsdaten sein, um während des Tests zu simulieren, dass das Modell gute Vorhersagen für Daten machen kann, die neuer als die Trainingsdaten sind

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz

- Wenn Ihre Daten
  - keine Zeitabhängigkeit und
  - keine Objektzugehörigkeit (Maschine, Patient, Sprecher, Kamera, Tier) haben und diese Information wichtig ist (ML-Modell soll auch für unbekannte Maschine, Patient, ... gute Ergebnisse liefern),
  - Oder sich die Grundgesamtheit nicht verändert (Bspw. kommen zu Kalifornien keine weiteren Bezirke hinzu, d.h. Trainings-/Testdatensatz decken gesamten Wertebereich ab)
- können Sie diese auch zufällig in Trainings- und Testdaten einteilen
- `train_test_split`
  - Teilt Datensatz zufällig in Trainings- und Testdatensatz auf

```
from sklearn.model_selection import train_test_split
```

```
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```
- S. Jupyter Notebook

# Beschaffen Sie sich Daten

## Erstelle einen Testdatensatz

Warum können wir bei unserem Datensatz die Daten zufällig in Training- und Testdaten aufteilen?

- Wir wollen das Modell nur in Kalifornien nutzen und nicht in anderen Bundesländer
  - Falls doch, dann müssen im Testdatensatz Daten von mind. einem anderen Bundesland existieren, wobei das Bundesland nicht im Trainingsdatensatz enthalten war
- Wir wollen nicht prüfen ob mit nur den Daten aus San Francisco Vorhersagen für ganz Kalifornien gemacht werden können.
  - Falls doch, dann
    - Trainingsdatensatz: San Francisco
    - Testdatensatz: alle anderen Daten
    - Oder Testdatensatz: Daten aus Los Angeles, wenn wir prüfen möchten, ob mit Daten aus SF Vorhersagen für LA (bzw. andere Großstädte) gemacht werden können

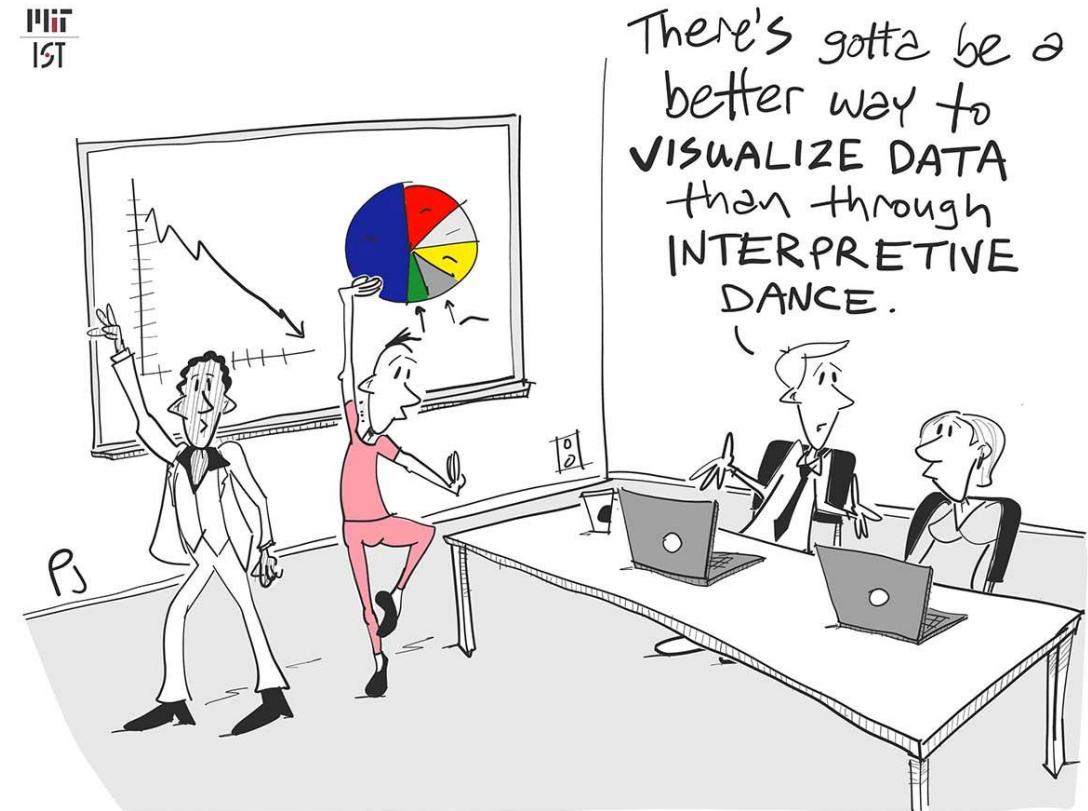
# Fragen?

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - **Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen**
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

- Visualisieren
- Suche nach Korrelationen
- Experimentieren mit Kombinationen von Merkmalen

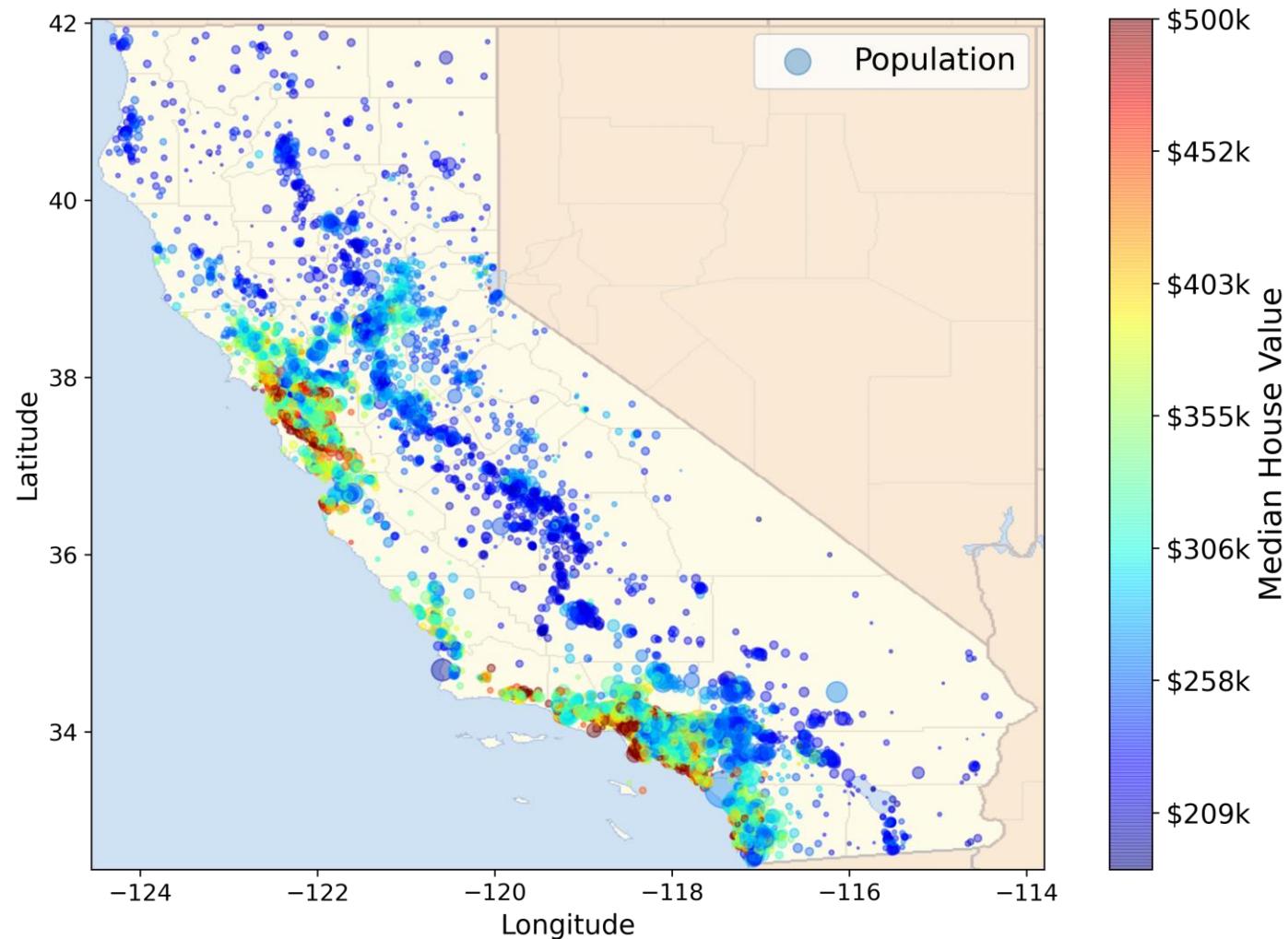


# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Visualisieren

Die Datenvisualisierung:

- verschafft einen Überblick
- dient zur Datenanalyse
- hilft bei der Dokumentation
- sowie der Kommunikation

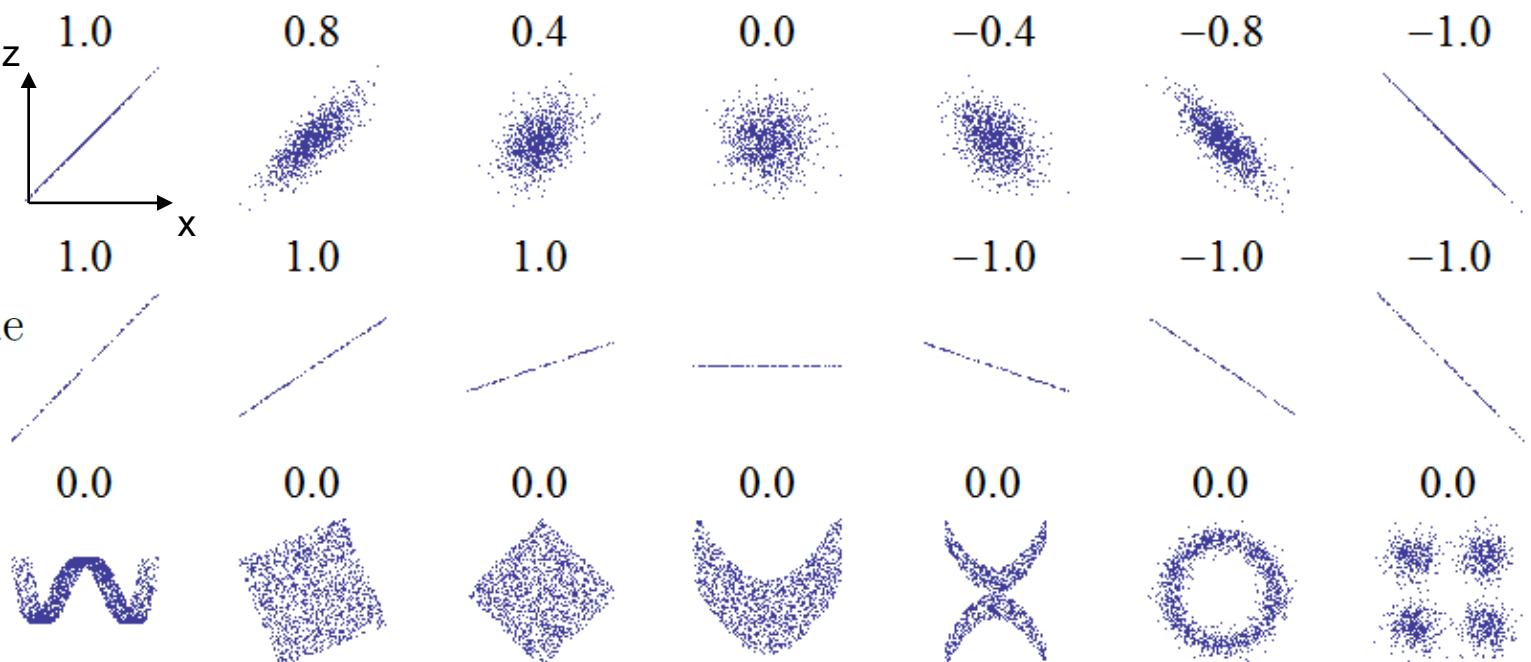


# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Suche nach Korrelationen

- Berechnung des Korrelationskoeffizienten zwischen allen Merkmalen, insbesondere zur Zielgröße
  - Der Pearson-Korrelationskoeffizient liegt zwischen -1 und 1.
  - Der Korrelationskoeffizient erfasst ausschließlich lineare Korrelationen

$$r_{x,z} := \frac{\sum_{i=1}^m (x_i - \bar{x})(z_i - \bar{z})}{\sqrt{\sum_{i=1}^m (x_i - \bar{x})^2 \sum_{i=1}^m (z_i - \bar{z})^2}}$$



$x_i$  und  $z_i$  sind einzelne Werte der Merkmale  $x$  und  $z$ .  $\bar{x}$  und  $\bar{z}$  sind die Mittelwerte der beiden Merkmale.

# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Suche nach Korrelationen

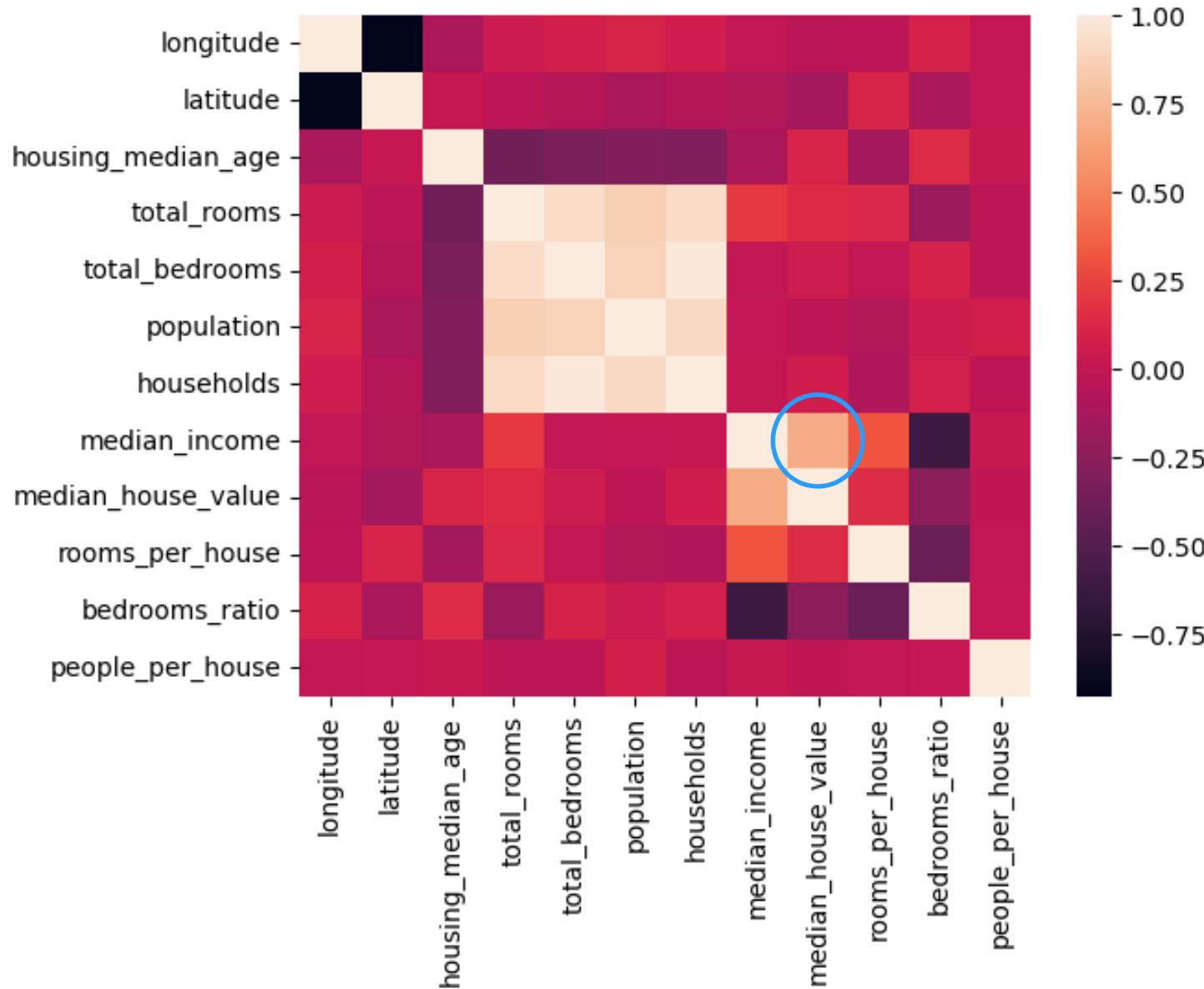
Pandas Methode: corr()

```
corr_matrix = housing.corr()
```

Seaborn:

- <https://seaborn.pydata.org/>
- Basiert auf matplotlib

```
seaborn.heatmap(corr_matrix)
```



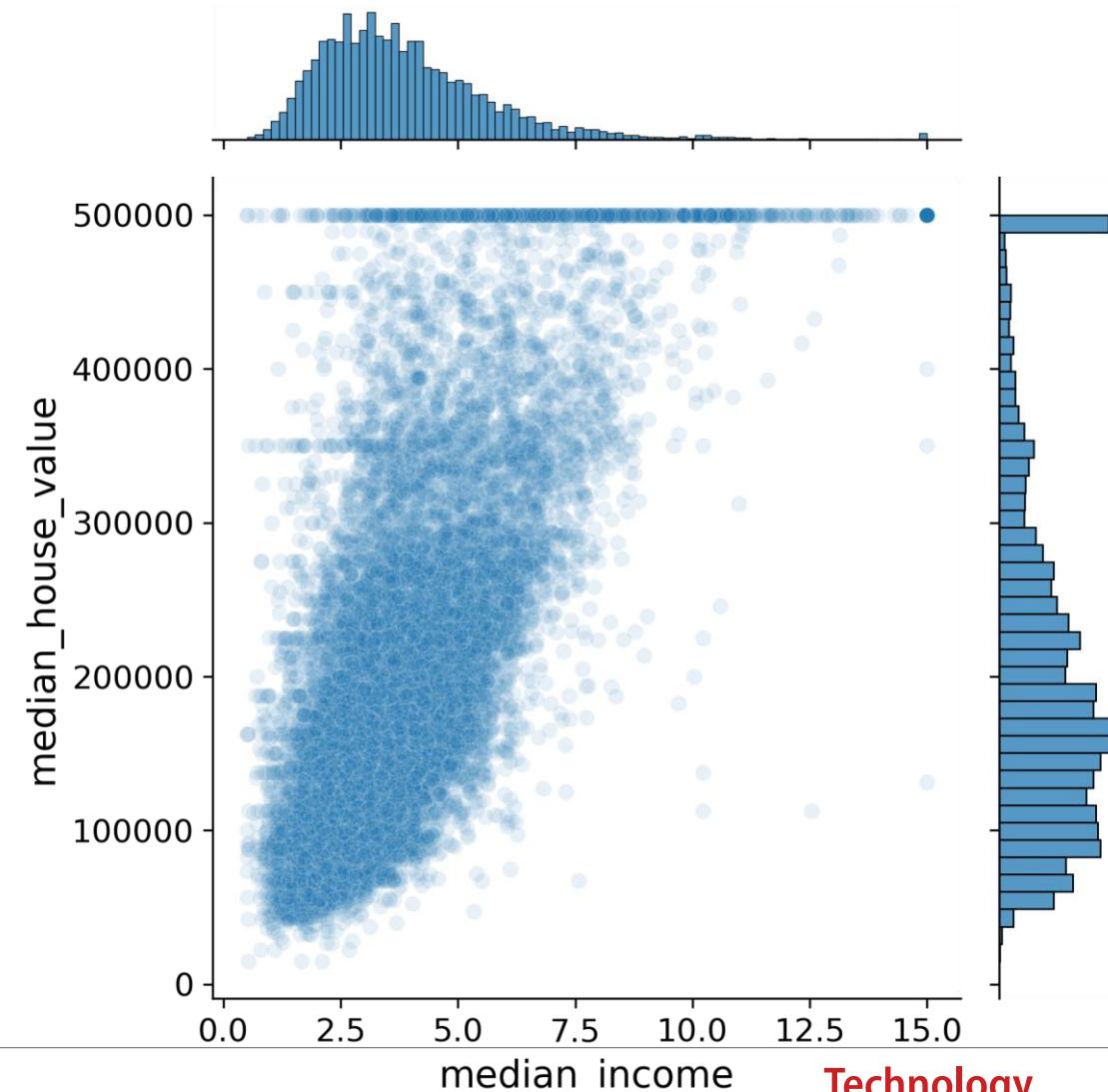
# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Suche nach Korrelationen

- **Streudiagramme** vergleichen zwei Merkmale miteinander
- Zusätzliche **Randdiagramme** veranschaulichen Beziehungen zwischen den zwei Merkmalen

### Bild

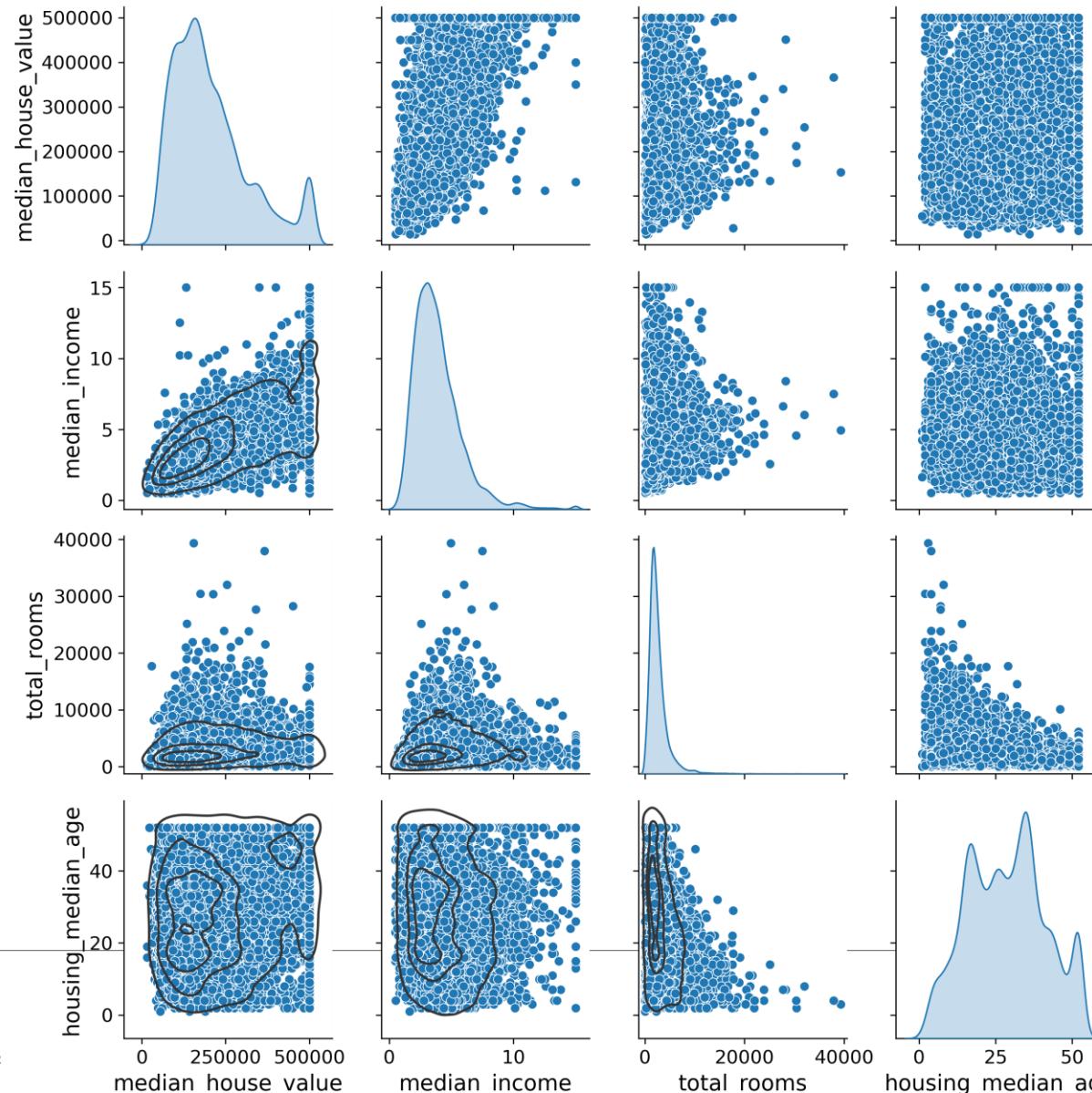
- Bildmitte zeigt ein Streudiagramm
- Randdiagramme zeigen Projektionen der zwei Merkmale in Form von Histogrammen
- `seaborn.jointplot()`



# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Suche nach Korrelationen

seaborn.pairplot()



# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Übung - Suche nach Korrelationen

```
In [25]: corr_matrix = housing.corr()
```

```
In [26]: corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
Out[26]: median_house_value    1.000000
median_income      0.687160
total_rooms        0.135097
housing_median_age 0.114110
households         0.064506
total_bedrooms     0.047689
population        -0.026920
longitude          -0.047432
latitude           -0.142724
Name: median_house_value, dtype: float64
```

```
attributes = ["median_house_value", "median_income", "total_rooms",
              "housing_median_age"]
g = sns.pairplot(housing[attributes], diag_kind="kde")
g.map_lower(sns.kdeplot, levels=4, color=".2")
save_fig("scatter_matrix_plot")
```

# Erkunde und visualisiere die Daten, um Erkenntnisse zu gewinnen

## Experimentieren mit Kombinationen von Merkmalen

- Feature Engineering (Teil 1)

```
housing["rooms_per_house"] = housing["total_rooms"] / housing["households"]
housing["bedrooms_ratio"] = housing["total_bedrooms"] / housing["total_rooms"]
housing["people_per_house"] = housing["population"] / housing["households"]
```

- Jupyter Notebook

# Fragen?

# Kurze Pause?

- Ja, 5 Minuten
- Ja, 10 Minuten
- Nein

The screenshot shows the DataCamp platform interface for the "Associate Data Scientist in Python" career track. At the top, it displays the track title "CAREER TRACK Associate Data Scientist in Python" with a "Continue Track" button. Below this, it lists the track's requirements: "Python, Theory", "90 hours", "23 Courses", "3 skill assessments", "11 Projects", and "373,194 participants". A progress bar indicates "TRACK COMPLETION" at 56%. On the right, there is a section titled "BONUS MATERIAL - 0 OF 14" showing a row of 14 small circular icons. The main content area is titled "Track Description" and includes a brief overview: "Learn data science in Python, from data manipulation to machine learning. This track provides the skills needed to succeed as a data scientist!". Below this, two course sections are listed under "COURSE": "Introduction to Python" and "Intermediate Python", each preceded by a green circular icon with a checkmark.

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - **Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können**
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

- Aufbereiten der Daten
- Bearbeiten von Text und kategorischen Merkmalen
- Skalieren von Merkmalen
- Pipelines zur Transformation
  - Eigene Transformer

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

- Trennung der Zielvariable (den Labels) von den Merkmalen

```
housing = train_set.drop("median_house_value", axis=1) # drop labels for training set  
housing_labels = train_set["median_house_value"].copy()
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

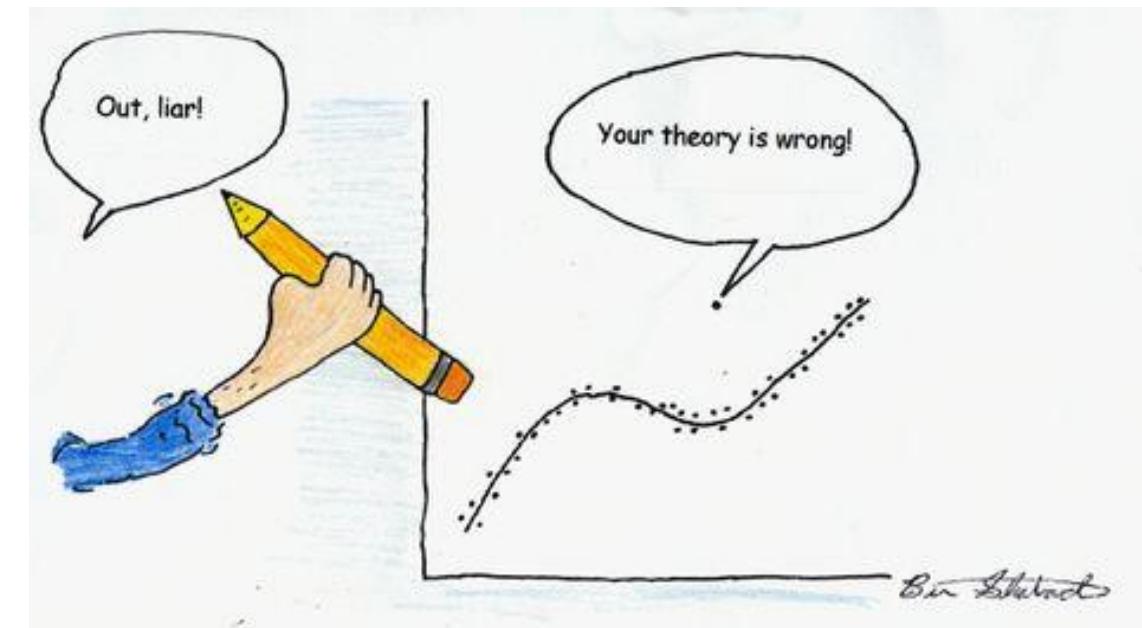
## Aufbereiten der Daten

Rohdaten können **fehlerbehaftet** sein und müssen vor einer Analyse aufbereitet werden!

Dies wird als **Data Cleaning** bezeichnet

**Wichtig!**

- **Rohdaten roh halten!**
- Datenaufbereitung muss **reproduzierbar** sein



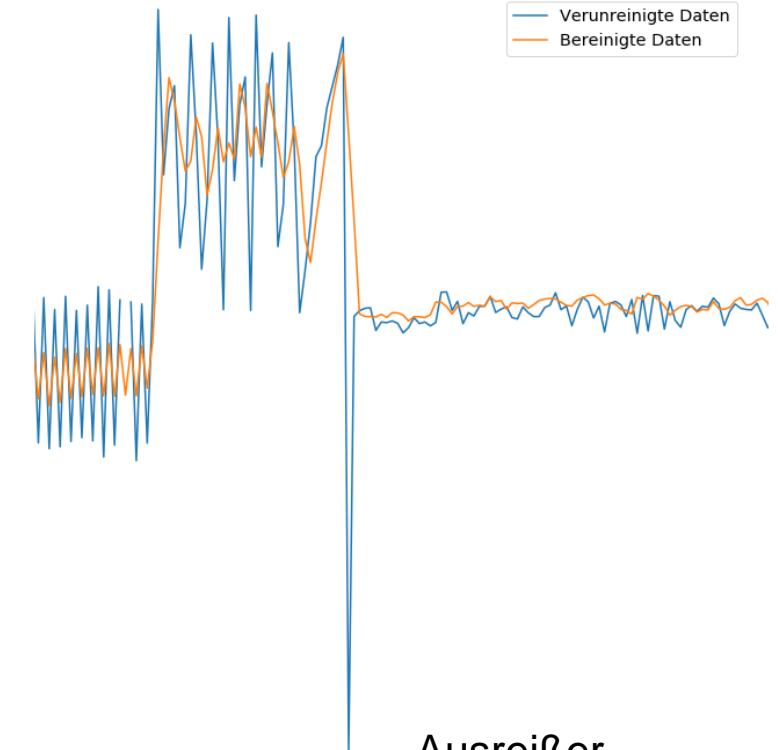
pinterest.es

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

Zu den möglichen Fehlern zählen:

- **Ausreißer:** systematisch oder zufällig, z.B.  
Messfehler, falsche manuelle Eingabe, Sensordefekt
- **Rauschen:** systematisch, z.B. Übertragungsfehler  
(weißes Rauschen, thermisches Rauschen,...)

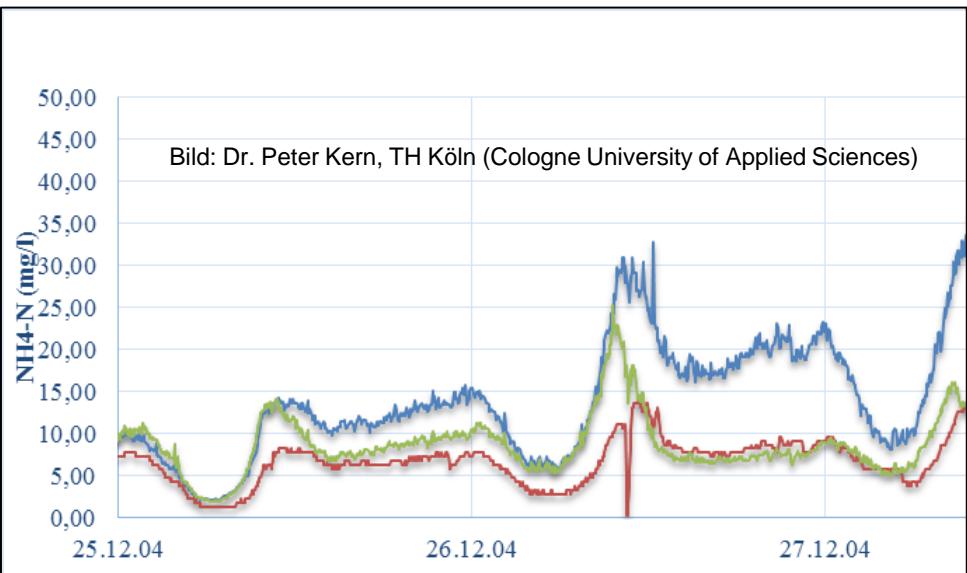
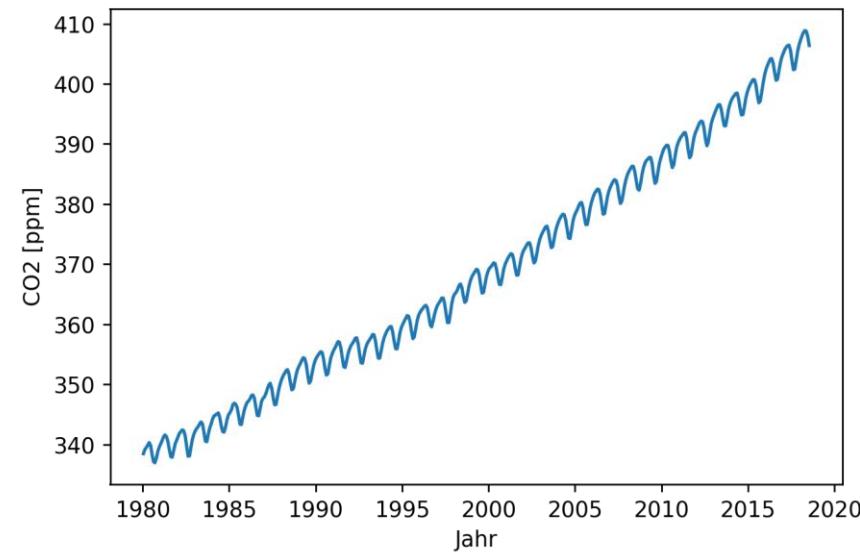


# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

Zu den möglichen Fehlern zählen:

- **Ausreißer:** systematisch oder zufällig, z.B. Messfehler, falsche manuelle Eingabe, Sensordefekt
- **Rauschen:** systematisch, z.B. Übertragungsfehler (weißes Rauschen, thermisches Rauschen,...)
- **Drift, Saisonalität und Bias:** Sensoren, Kultureller Bias



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

Zu den möglichen Fehlern zählen:

- **Ausreißer:** systematisch oder zufällig, z.B.  
Messfehler, falsche manuelle Eingabe, Sensordefekt
- **Rauschen:** systematisch, z.B. Übertragungsfehler  
(weißes Rauschen, thermisches Rauschen,...)
- **Drift, Saisonalität und Bias:** Sensoren,  
Kultureller Bias (Geschlecht, Rasse, Hautfarbe, Religion)

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

Zu den möglichen Fehlern zählen:

- **Ausreißer:** systematisch oder zufällig, z.B.  
Messfehler, falsche manuelle Eingabe, Sensordefekt
- **Rauschen:** systematisch, z.B. Übertragungsfehler  
(weißes Rauschen, thermisches Rauschen,...)
- **Drift, Saisonalität und Bias:** Sensoren, Kultureller Bias
- **Fehlende Daten:** meist zufällig, vergessene manuelle Eingabe,  
Datenverlust über Kommunikationskanal
- **Ungültige Datensätze:** z.B. falscher Datentyp, außerhalb des  
gültigen Wertebereichs, unvollständig
- **Redundante Informationen:** doppelte oder hochkorrelierte Daten

### → Data Cleaning

- Identifizierung und Behandlung von fehlerhaften Daten

Date	Humidity	Pressure
2013-01-01 00:00:00	NaN	1024.0
2013-01-01 01:00:00	64.0	1022.0
2013-01-01 02:00:00	69.0	1022.0
2013-01-01 03:00:00	NaN	1021.0
2013-01-01 04:00:00	68.0	1021.0
2013-01-01 05:00:00	68.0	1020.0
2013-01-01 06:00:00	NaN	NaN
2013-01-01 07:00:00	68.0	1018.0
2013-01-01 08:00:00	NaN	1018.0
2013-01-01 09:00:00	NaN	1018.0
2013-01-01 10:00:00	69.0	1017.0
2013-01-01 11:00:00	64.0	1017.0
2013-01-01 12:00:00	55.0	1017.0
2013-01-01 13:00:00	59.0	1017.0
2013-01-01 14:00:00	59.0	1017.0

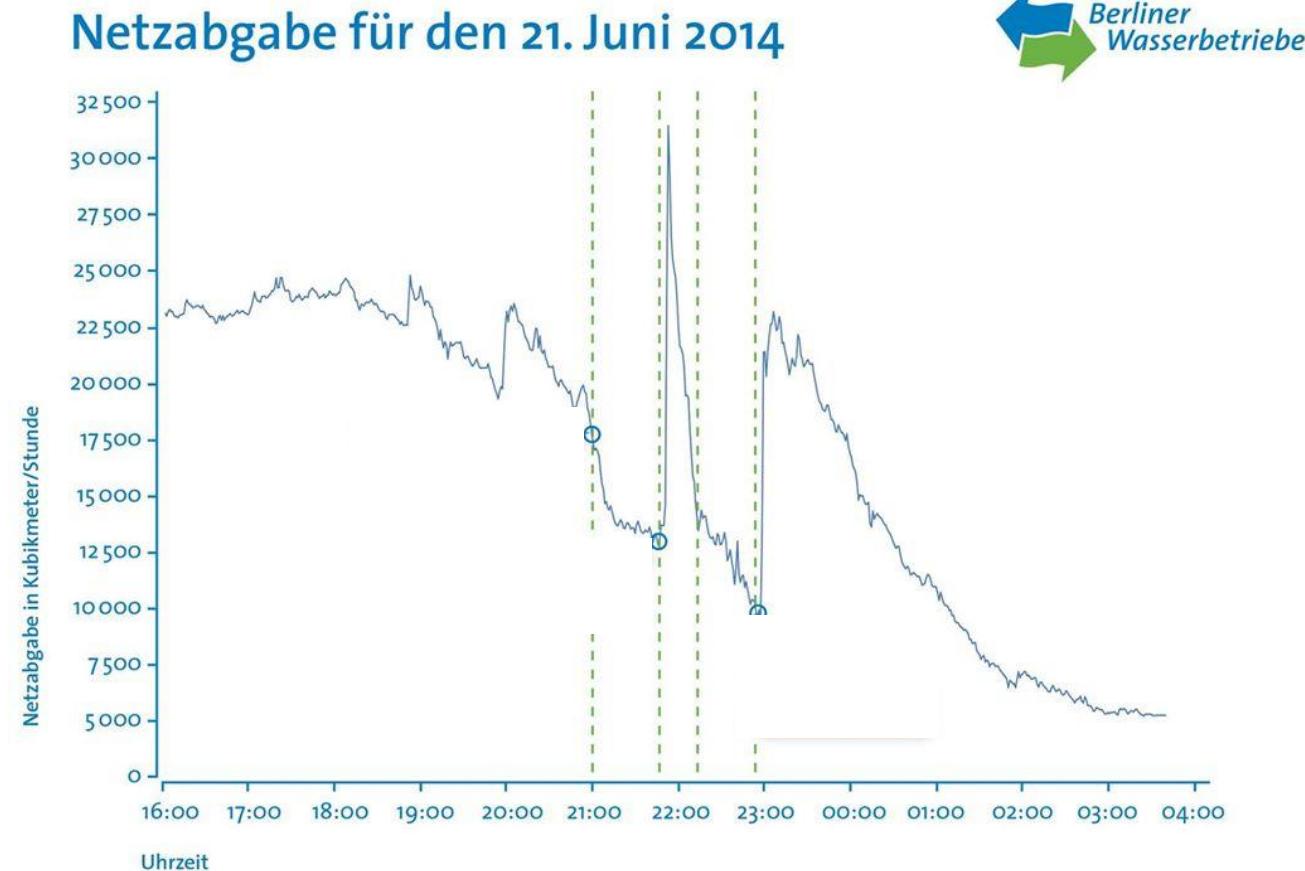
# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

Fehler	Identifizierung	Behebung
<b>Ausreißer</b>	Sigma-Regel (Z-Score), Boxplot, Max-Min-Intervall	Ersetzen, Feature / Datensatz ignorieren
<b>Fehlende Daten</b>	Pandas isnull(), isnull().values.any(), info()	Ersetzen fillna(), Feature / Datensatz ignorieren dropna()
<b>Ungültige Daten</b>	Boxplot, Pandas where()	Pandas where(), ersetzen von Hand
<b>Rauschen</b>	Scatterplot, Signal-to-Noise-Verhältnis	Pandas rolling_median() Messung wiederholen

**Beachte:** Entfernen falscher Ausreißer, sogenannte **Exoten**, heißt **Verlust von wichtigen Informationen!**

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Exot oder Ausreißer?



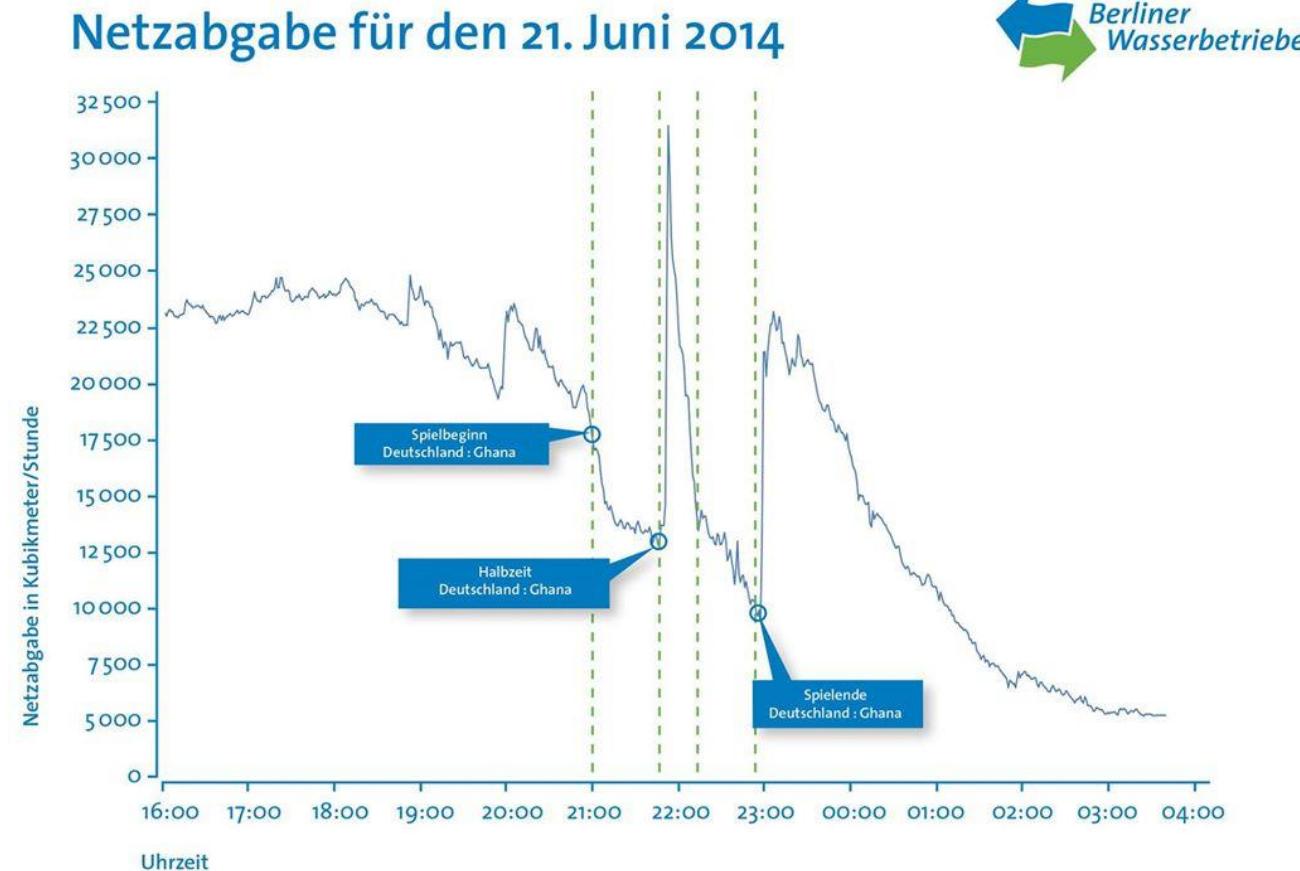
# Ihre Vorträge - Termine

- 1.7.: AHE CNN, Data Augmentation, Embedded KI
- 8.7.: NLP, Unsupervised ML

Wer möchte Protokoll führen?

- Protokolle liegen in „Gruppe SoSe 25“ / Protokolle

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Exot oder Ausreißer?



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Aufbereiten der Daten

- Fehlerhaften Wert des Datenpunkts durch NaN (Not a Number) ersetzen
  
- Daten Imputation in pandas
  - Dropna() → Entfernt die Zeilen in denen Werte fehlen (damit auch die Werte anderer Merkmale (NA → NaN, None))
  - Drop() → Entfernt das Merkmal vollständig
  - Fillna() → Ersetzt die fehlenden Werte durch einen Standardwert (0, Median, ...)
  
- SimpleImputer
  - Klasse in scikit-learn für Daten Imputation (Alternative zu fillna())
  
- SimpleImputer ist nur für numerische Merkmale anwendbar, d.h. kategorische Merkmale müssen separat verarbeitet werden

Date	Humidity	Pressure
2013-01-01 00:00:00	NaN	1024.0
2013-01-01 01:00:00	64.0	1022.0
2013-01-01 02:00:00	69.0	1022.0
2013-01-01 03:00:00	NaN	1021.0
2013-01-01 04:00:00	68.0	1021.0
2013-01-01 05:00:00	68.0	1020.0
2013-01-01 06:00:00	NaN	NaN
2013-01-01 07:00:00	68.0	1018.0
2013-01-01 08:00:00	NaN	1018.0
2013-01-01 09:00:00	NaN	1018.0
2013-01-01 10:00:00	69.0	1017.0
2013-01-01 11:00:00	64.0	1017.0
2013-01-01 12:00:00	55.0	1017.0
2013-01-01 13:00:00	59.0	1017.0
2013-01-01 14:00:00	59.0	1017.0

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Übung - Aufbereiten der Daten

- Schauen Sie sich SimpleImputer, drop(), ... an

### Data Cleaning

```
In [35]: ┏━▶ from sklearn.impute import SimpleImputer  
      imputer = SimpleImputer(strategy="median")
```

Remove the text attribute because median can only be calculated on numerical attributes:

```
In [36]: ┏━▶ housing_num = housing.drop("ocean_proximity", axis=1)  
      # alternatively: housing_num = housing.select_dtypes(include=[np.number])
```

```
In [37]: ┏━▶ imputer.fit(housing_num)
```

```
Out[37]: SimpleImputer(strategy='median')
```

```
In [38]: ┏━▶ imputer.statistics_
```

```
Out[38]: array([-118.51 ,  34.26 ,  29. ,  2119.5 ,  433. ,  1164. ,  
                 408. ,  3.5409])
```

Transform the training set:

```
In [40]: ┏━▶ x = imputer.transform(housing_num)
```

```
In [42]: ┏━▶ housing_tr = pd.DataFrame(x, columns=housing_num.columns,  
      index=housing_num.index)
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Bearbeiten von Text und kategorischen Merkmalen

- Kategoriale Merkmale müssen Sie vorverarbeiten, da für ML-Methoden zwischen zwei Werten eines Merkmals eine numerische Metrik (ein Abstand) definiert sein muss
- Bsp.:
  - Das Merkmal ocean\_proximity in unserem Datensatz

ocean_proximity	
<1H OCEAN	
<1H OCEAN	
NEAR OCEAN	
INLAND	
<1H OCEAN	
INLAND	
<1H OCEAN	
INLAND	
<1H OCEAN	
<1H OCEAN	

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Bearbeiten von Text und kategorischen Merkmalen

- Kategorische Merkmale müssen Sie vorverarbeiten, da für ML-Methoden zwischen zwei Werten eines Merkmals eine numerische Metrik (ein Abstand) definiert sein muss
  - Bsp.:
    - Das Merkmal ocean\_proximity in unserem Datensatz

## Klassen in scikit-learn:

- **OrdinalEncoder** → wandelt Kategorien in die Zahlen 0, 1, 2, ... um
    - Nur nutzen, wenn es eine Metrik für die Kategorien gibt, bspw. die Kategorien „schlecht“, „durchschnittlich“ und „gut“
  - **OneHotEncoder** → Jede Kategorie wird durch einen K-dimensionalen Vektor dargestellt, der nur an einer Stelle eine 1 hat, Rest 0. K ist die Anzahl der Kategorien.
    - Abstand zwischen jeder Kategorie ist damit gleich

```
array([[0., 0., 0., 0., 1.],  
       [0., 0., 0., 0., 1.],  
       [0., 0., 0., 1., 0.],  
       ...,  
       [0., 1., 0., 0., 0.],  
       [0., 1., 0., 0., 0.],  
       [1., 0., 0., 0., 0.]])
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Bearbeiten von Text und kategorischen Merkmalen

- OneHotEncoder → Jede Kategorie wird durch einen K-dimensionalen Vektor dargestellt, der nur an einer Stelle eine 1 hat, Rest 0. K ist die Anzahl der Kategorien.
  - Abstand zwischen jeder Kategorie ist damit gleich
  - Jede Kategorie wird zu einem eigenen binären Merkmal

	<b>ocean_proximity_&lt;1H OCEAN</b>	<b>ocean_proximity_INLAND</b>	<b>ocean_proximity_ISLAND</b>	<b>ocean_proximity_NEAR BAY</b>	<b>ocean_proximity_NEAR OCEAN</b>
<b>14196</b>	0.0	0.0	0.0	0.0	1.0
<b>8267</b>	0.0	0.0	0.0	0.0	1.0
<b>17445</b>	0.0	0.0	0.0	1.0	0.0
<b>14265</b>	1.0	0.0	0.0	0.0	0.0
<b>2271</b>	0.0	1.0	0.0	0.0	0.0

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Übung - Bearbeiten von Text und kategorischen Merkmalen

- Schauen Sie sich die Klassen OneHotEncoder, ... an

```
In [45]: ┏━▶ from sklearn.preprocessing import OneHotEncoder  
cat_encoder = OneHotEncoder()  
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)  
housing_cat_1hot  
  
Out[45]: <16512x5 sparse matrix of type '<class 'numpy.float64'>'  
with 16512 stored elements in Compressed Sparse Row format>
```

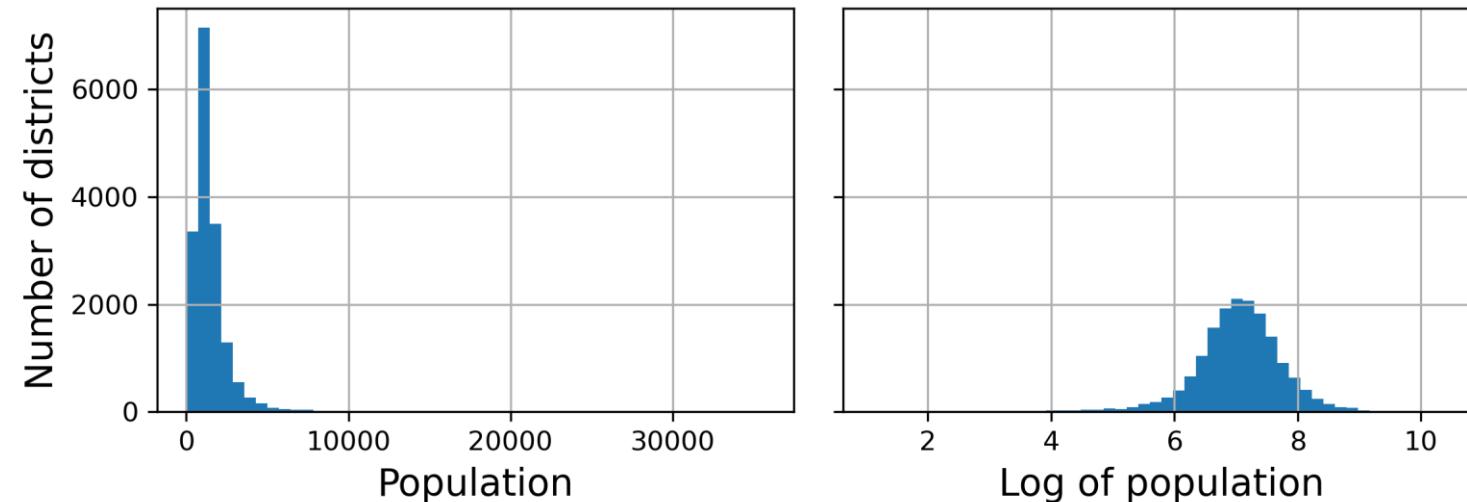
By default, the `OneHotEncoder` class returns a sparse array, but we can convert it to a dense array if needed by calling the `toarray()` method:

```
In [46]: ┏━▶ housing_cat_1hot.toarray()  
  
Out[46]: array([[1., 0., 0., 0., 0.],  
[1., 0., 0., 0., 0.],  
[0., 0., 0., 0., 1.],  
...,  
[0., 1., 0., 0., 0.],  
[1., 0., 0., 0., 0.],  
[0., 0., 0., 1., 0.]])
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Feature Engineering (Teil 2)

- Normalisierung einer linksschiefen Verteilung eines Merkmals durch Logarithmieren
  - Warum macht man das?
    - Manche ML-Algorithmen gehen von Gauß-verteilten Merkmalen aus
    - Hilft bei der Konvergenz der ML-Algorithmen



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Feature Engineering (Teil 2)

- Logarithmisch transformiertes Merkmal:

```
from sklearn.preprocessing import FunctionTransformer

log_transformer = FunctionTransformer(np.log, inverse_func=np.exp)
log_pop = log_transformer.transform(housing[["population"]])
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

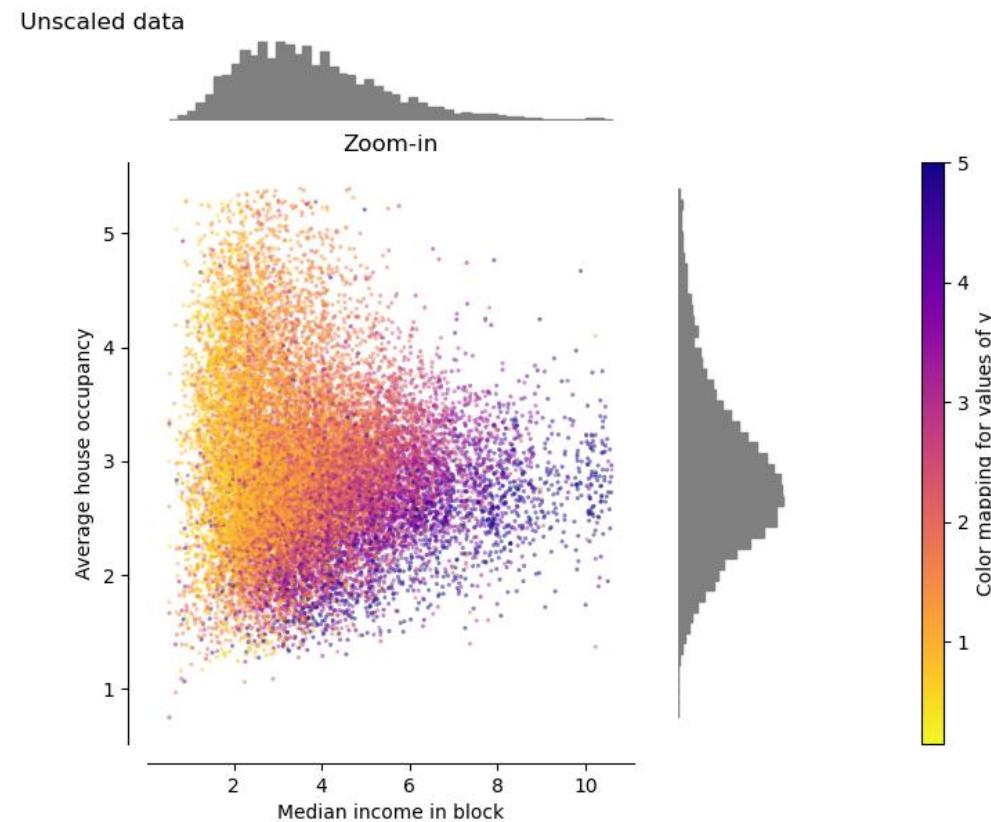
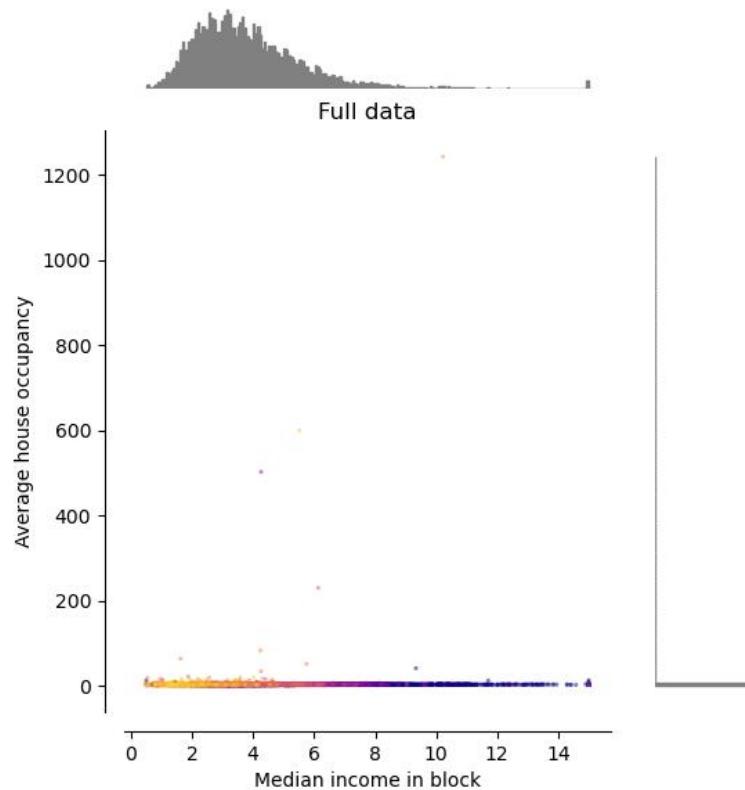
## Skalieren von Merkmalen

- Die meisten Machine Learning Methoden können nicht mit beliebig (vor allem sehr unterschiedlich) skalierten Merkmalen arbeiten
- Deshalb sollten Sie fast immer die Merkmale in einen Wertebereich von bspw. 0 bis 1 oder -1 bis 1 bringen
- MinMaxScaler → Min-Max-Skalierung (Normalisieren)
  - Merkmale sind dann zwischen 0 und 1 skaliert
  - Anfällig gegenüber Ausreißer

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Skalieren von Merkmalen

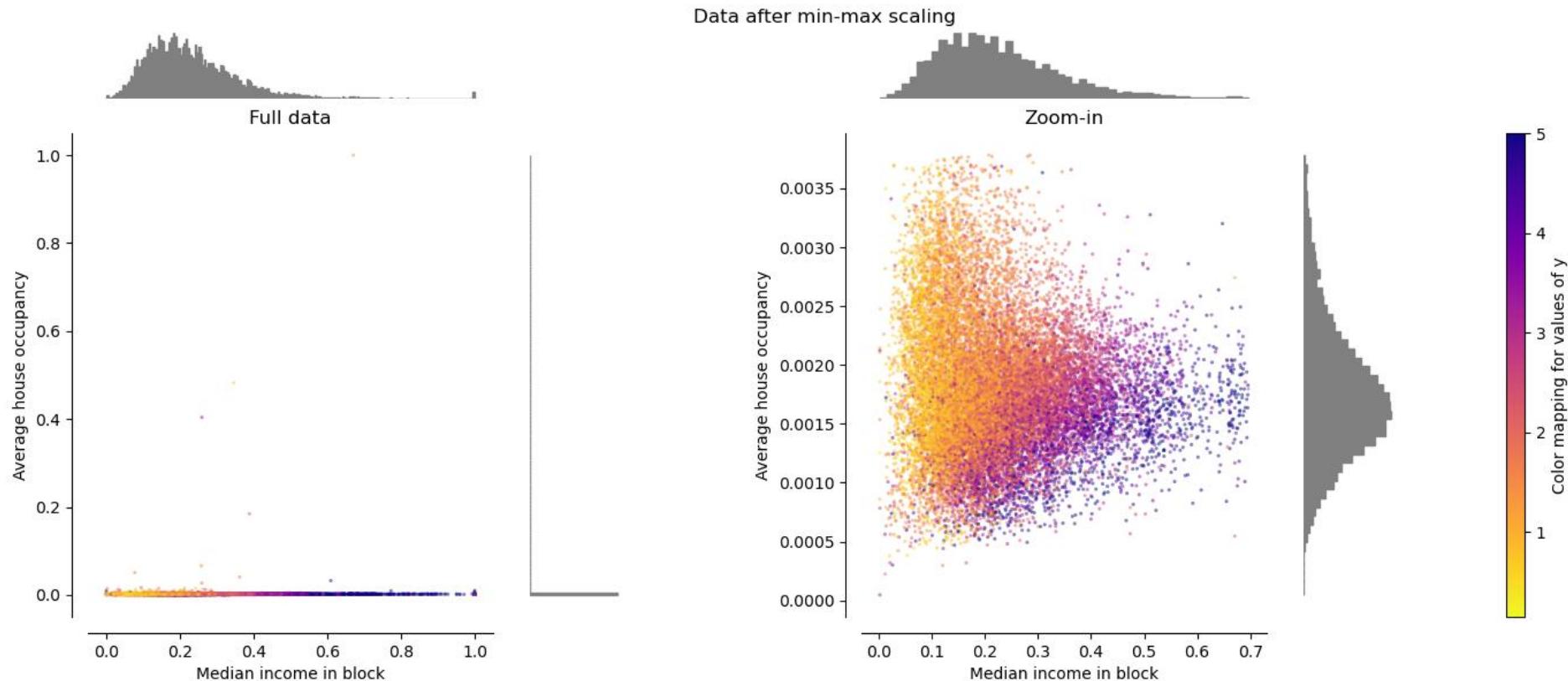
- Originaldaten



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Skalieren von Merkmalen

- MinMaxScaler



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Skalieren von Merkmalen

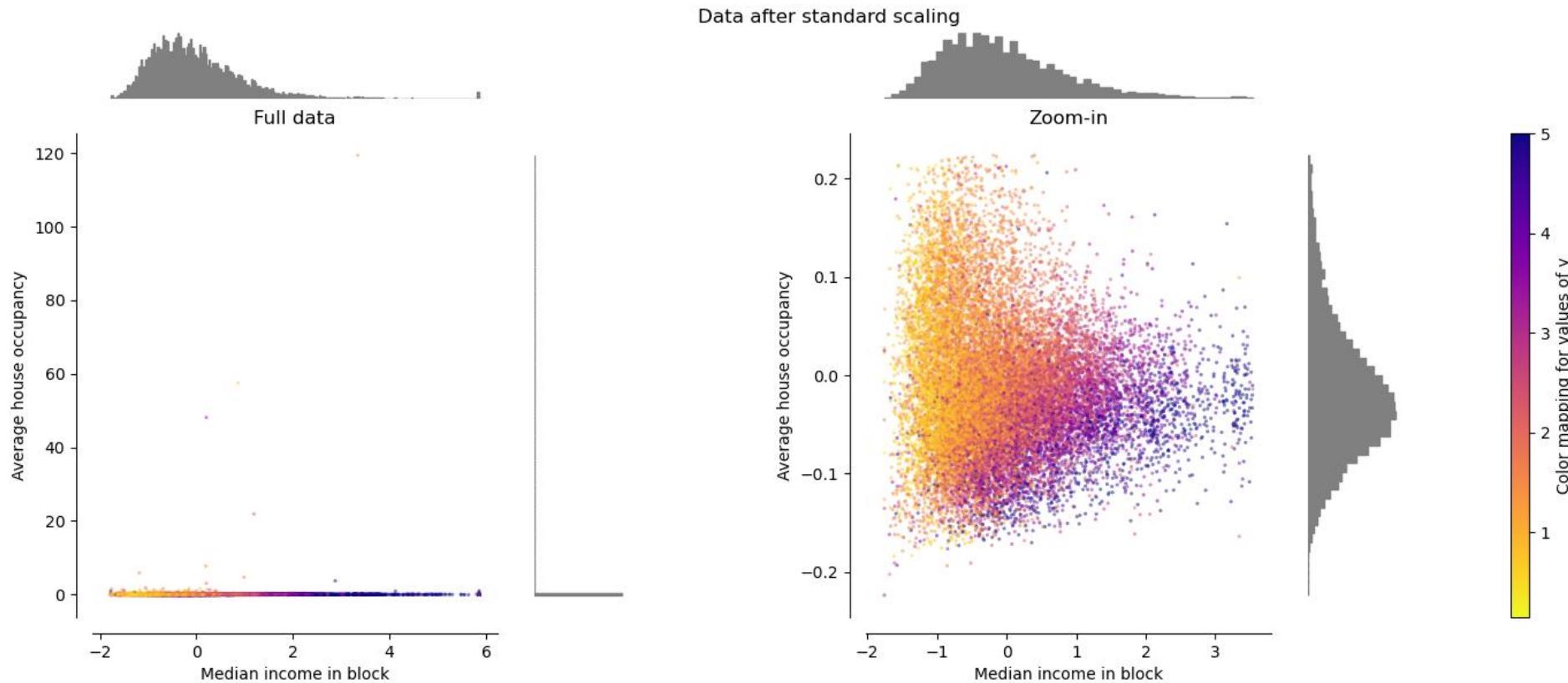
- Die meisten Machine Learning Methoden können nicht mit beliebig (vor allem sehr unterschiedlich) skalierten Merkmalen arbeiten
- Deshalb sollten Sie fast immer die Merkmale in einen Wertebereich von bspw. 0 bis 1 oder -1 bis 1 bringen
- MinMaxScaler → Min-Max-Skalierung (Normalisieren)
  - Merkmale sind dann zwischen 0 und 1 skaliert
  - Anfällig gegenüber Ausreißer
- StandardScaler → Standardisierung
  - Mittelwert abziehen und durch Standardabweichung dividieren
  - Mittelwert ist 0, Varianz ist 1
- Scikit-learn hat weitere Scaler

$$x_{\text{norm},i} := \frac{x_i - \bar{x}}{\sigma}$$

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Skalieren von Merkmalen

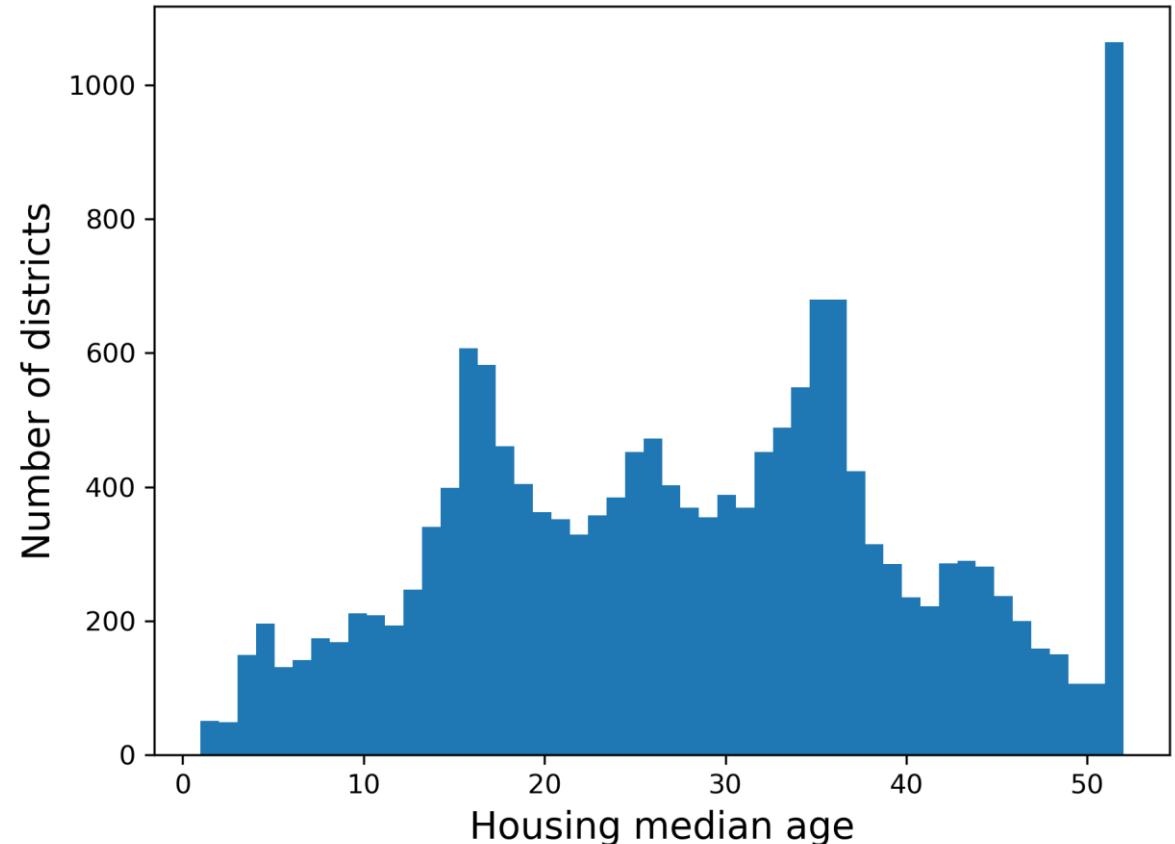
- StandardScaler



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Feature Engineering (Teil 2)

- Multimodale Verteilung
- Es gibt viele Häuser, die ca. 35 Jahre alt sind (und die ca. 17 Jahre alt sind)
- Aus multimodaler Verteilung mehrere Gauß-verteilte Merkmale erstellen, wobei die neuen Merkmale ihr Maximum bei den Modi haben



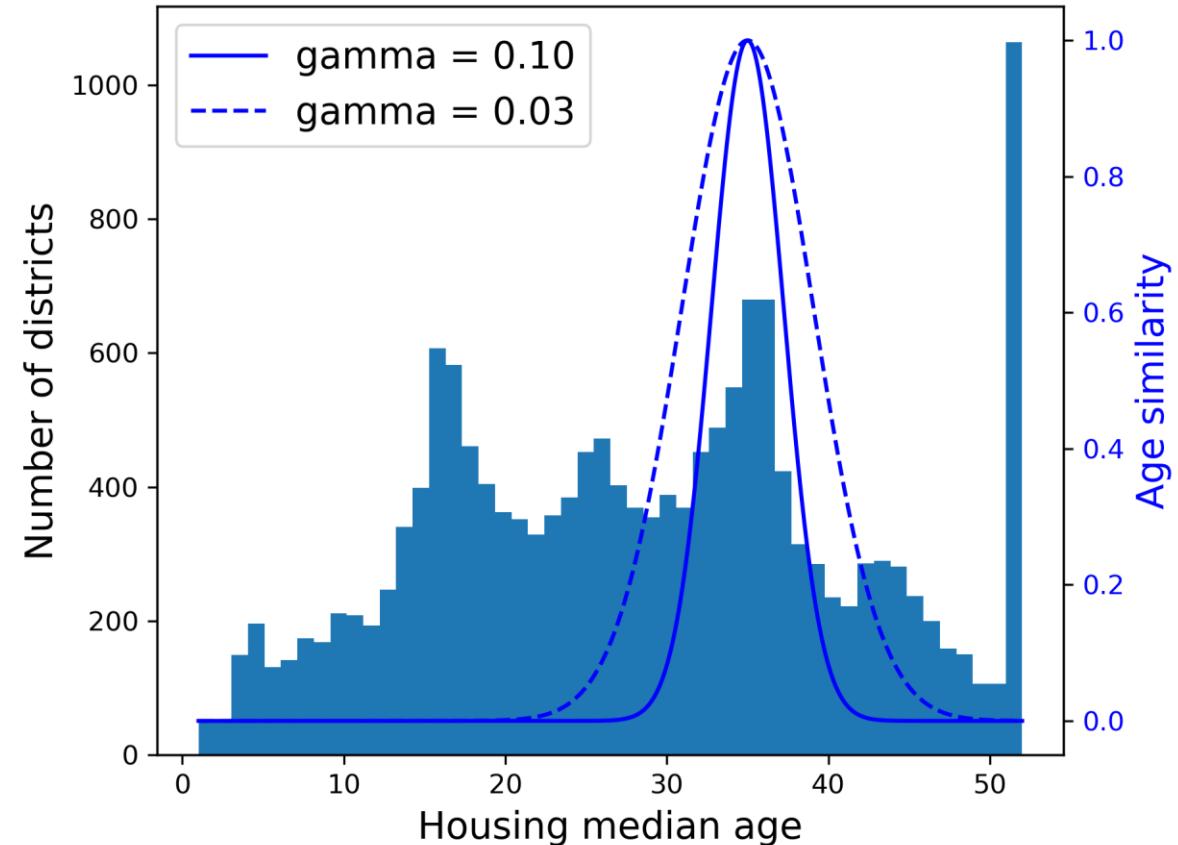
# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Feature Engineering (Teil 2)

- Multimodale Verteilung
- Feature: Age similarity 35 years
  - Wertebereich: 0 ... 1
  - Wenn Haus 35 Jahre alt ist: 1
  - Wenn Haus ein ganz anderes Alter hat: 0

$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$

```
age_simil_35 = rbf_kernel(housing[["housing_median_age"]],  
[[35]], gamma=0.1)
```



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Feature Engineering (Teil 2)

- Logarithmisch transformiertes Merkmal:

```
from sklearn.preprocessing import FunctionTransformer

log_transformer = FunctionTransformer(np.log, inverse_func=np.exp)
log_pop = log_transformer.transform(housing[["population"]])
```

- Ähnlichkeit zu Hausalter: 35:

```
rbf_transformer = FunctionTransformer(rbf_kernel,
                                      kw_args=dict(Y=[[35.]], gamma=0.1))
age_simil_35 = rbf_transformer.transform(housing[["housing_median_age"]])
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Das Design von Scikit-Learn

- Estimatoren
  - Jedes Objekt, das Parameter anhand eines Datensatzes abschätzen kann
  - Bspw.: Imputer, Encoder, Scaler
  - Besitzt die Methode fit()
- Transformer
  - Ein Estimator, der den übergegebenen Datensatz außerdem transformieren kann
  - Bspw.: Imputer, Encoder, Scaler
  - Besitzt die Methoden fit() und transform(); fit\_transform() führt beide hintereinander aus
- Prädiktoren
  - Estimator, der auf dem gegebenen Datensatz Vorhersagen treffen kann
  - Bspw.: LinearRegression
  - Besitzt die Methoden fit(), predict() und score()

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Pipelines zur Transformation

- Zur sequentiellen Ausführung der verschiedenen Transformationen gibt es in Scikit-Learn die Klasse Pipeline
  - Benötigt eine Liste von Name-Estimator-Paaren, die die Abfolge der einzelnen Schritte definiert
- 
- Aufruf von `fit_transform()` ruft `fit_transform()` für alle Transformer sequentiell auf.
  - Ausgabe ist Eingabe des nächsten Schritts.
- 
- Vorteil:
    - Datenvorverarbeitung wird transparenter und kann direkt auf neue Daten angewendet werden
- ```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
    ("impute", SimpleImputer(strategy="median")),
    ("standardize", StandardScaler()),
])

housing_num_tr = num_pipeline.fit_transform(housing_num)
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Pipelines zur Transformation

- Pipeline für kategorische Merkmale:
- Imputation: Ersetze fehlende Daten mit der am häufigsten auftretenden Kategorie des Merkmals

```
cat_attribs = ["ocean_proximity"]

cat_pipeline = make_pipeline(
    SimpleImputer(strategy="most_frequent"),
    OneHotEncoder(handle_unknown="ignore"))
```

```
cat_pipeline.fit_transform(housing_cat)
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Pipelines zur Transformation

- Zusammenführung von numerischen und kategorischen Merkmalen:
- ColumnTransformer

X  
[[1., 0., 0., 0., 0.],  
 [1., 0., 0., 0., 0.],  
 [0., 0., 0., 0., 1.],  
 ...,  
 [0., 1., 0., 0., 0.],  
 [1., 0., 0., 0., 0.],  
 [0., 0., 0., 1., 0.]]]

```
from sklearn.compose import ColumnTransformer

num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]

preprocessing = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", cat_pipeline, cat_attribs),
])

housing_prepared = preprocessing.fit_transform(housing)
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Pipelines zur Transformation

- Zusammenführung von numerischen und kategorischen Merkmalen:

| num_population | num_households | num_median_income | cat_ocean_proximity_<1H<br>OCEAN | cat_ocean_proximity_INLAND |
|----------------|----------------|-------------------|----------------------------------|----------------------------|
| 1.081011       | 1.507507       | 0.379698          | 0.0                              | 0.0                        |
| -0.643842      | -0.878707      | 0.420068          | 0.0                              | 0.0                        |

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Übung - Pipelines zur Transformation

- Programmieren Sie eine Pipeline für Ihren Datensatz
- 10 Min.

```
from sklearn.compose import make_column_selector

def column_ratio(X):
    return X[:, [0]] / X[:, [1]]

def ratio_name(function_transformer, feature_names_in):
    return ["ratio"] # feature names out

def ratio_pipeline():
    return make_pipeline(
        SimpleImputer(strategy="median"),
        FunctionTransformer(column_ratio, feature_names_out=ratio_name),
        StandardScaler())

log_pipeline = make_pipeline(
    SimpleImputer(strategy="median"),
    FunctionTransformer(np.log, feature_names_out="one-to-one"),
    StandardScaler())
cluster_simil = ClusterSimilarity(n_clusters=10, gamma=1., random_state=42)
default_num_pipeline = make_pipeline(SimpleImputer(strategy="median"),
                                      StandardScaler())
preprocessing = ColumnTransformer([
    ("bedrooms", ratio_pipeline(), ["total_bedrooms", "total_rooms"]),
    ("rooms_per_house", ratio_pipeline(), ["total_rooms", "households"]),
    ("people_per_house", ratio_pipeline(), ["population", "households"]),
    ("log", log_pipeline, ["total_bedrooms", "total_rooms", "population",
                           "households", "median_income"]),
    ("geo", cluster_simil, ["latitude", "longitude"]),
    ("cat", cat_pipeline, make_column_selector(dtype_include=object)),
],
    remainder=default_num_pipeline) # one column remaining: housing_median_age
```

# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Feature Engineering (Teil 2)

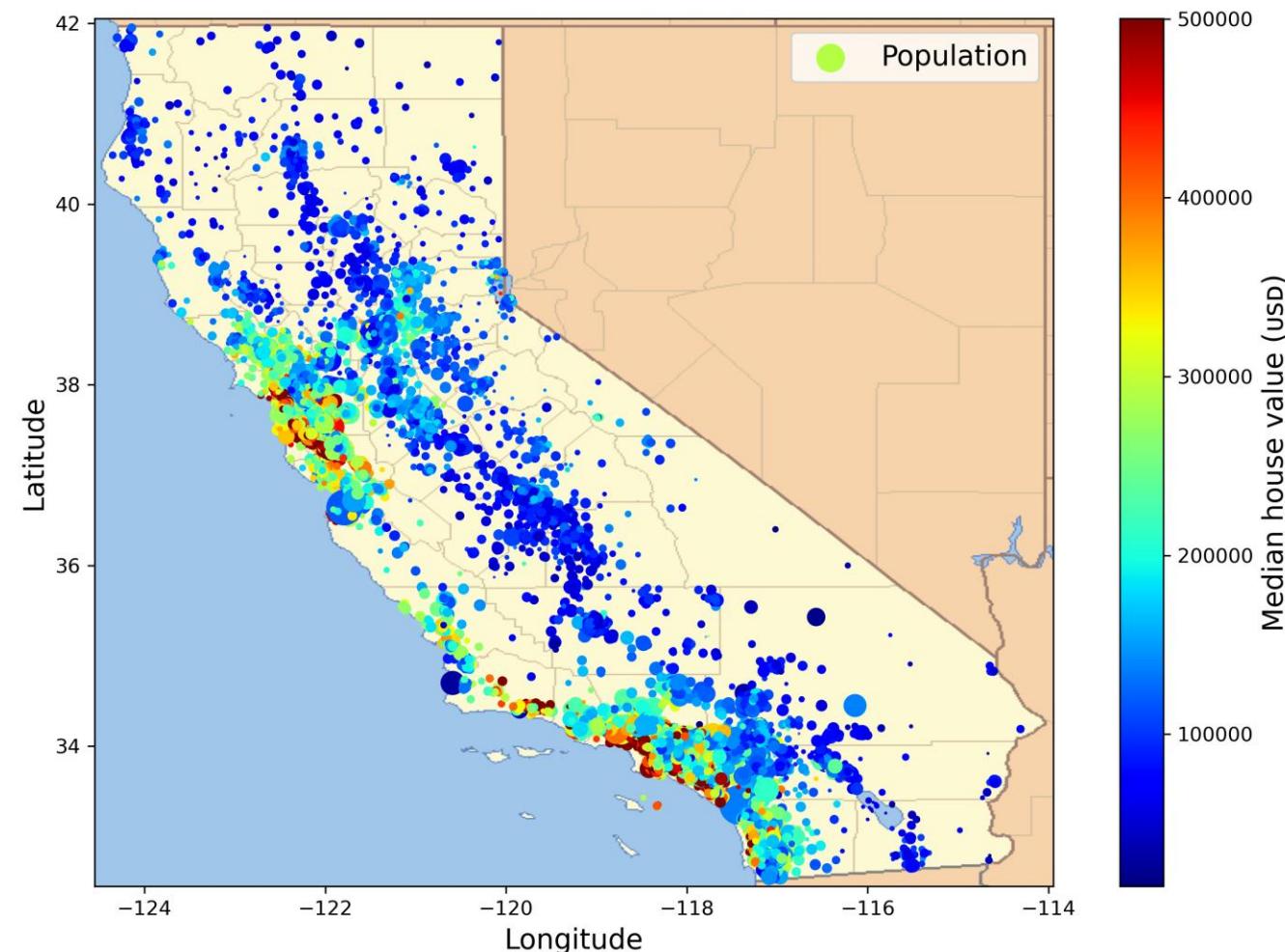
- Ähnlichkeit (bzw. Abstand zu) San Francisco:

```
# Lat and Long coordinates of San Francisco
sf_coords = 37.7749, -122.41
sf_transformer = FunctionTransformer(rbf_kernel,
                                     kw_args=dict(Y=[sf_coords], gamma=0.1))
sf_simil = sf_transformer.transform(housing[["latitude", "longitude"]])
```

- Bezirke in der Nähe von San Francisco: 1
- Bezirke weit weg von San Francisco: 0
- Allgemein: Merkmal erstellen, das misst wie nah ein Bezirk an einer Metropole ist

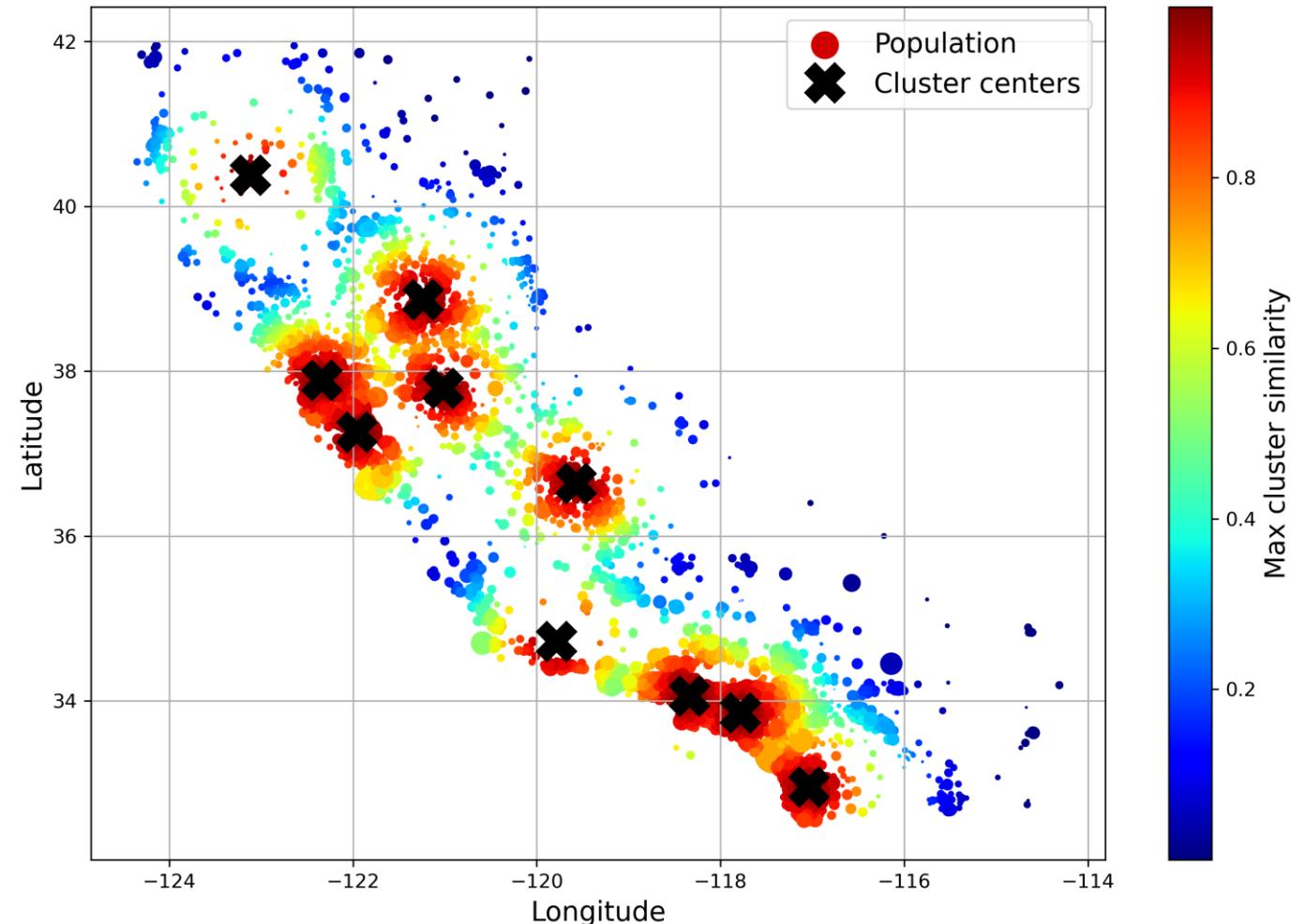
# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Feature Engineering (Teil 2)

- Allgemein: Merkmal erstellen, das misst wie nah ein Bezirk an einer Metropole (einem Ballungsraum) ist



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor Feature Engineering (Teil 2)

- Allgemein: Merkmal erstellen, das misst wie nah ein Bezirk an einer Metropole (einem Ballungsraum) ist



# Bereiten Sie die Daten für Machine-Learning-Algorithmen vor

## Eigene Transformer

```
class ClusterSimilarity(BaseEstimator, TransformerMixin):
    def __init__(self, n_clusters=10, gamma=1.0, random_state=None):
        self.n_clusters = n_clusters
        self.gamma = gamma
        self.random_state = random_state

    def fit(self, X, y=None, sample_weight=None):
        self.kmeans_ = KMeans(self.n_clusters, n_init=10,
                             random_state=self.random_state)
        self.kmeans_.fit(X, sample_weight=sample_weight)
        return self # always return self!

    def transform(self, X):
        return rbf_kernel(X, self.kmeans_.cluster_centers_, gamma=self.gamma)

    def get_feature_names_out(self, names=None):
        return [f"Cluster {i} similarity" for i in range(self.n_clusters)]
```

# Fragen?

# Kurze Pause?

- Ja, 5 Minuten
- Ja, 10 Minuten
- Nein

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - **Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl**
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

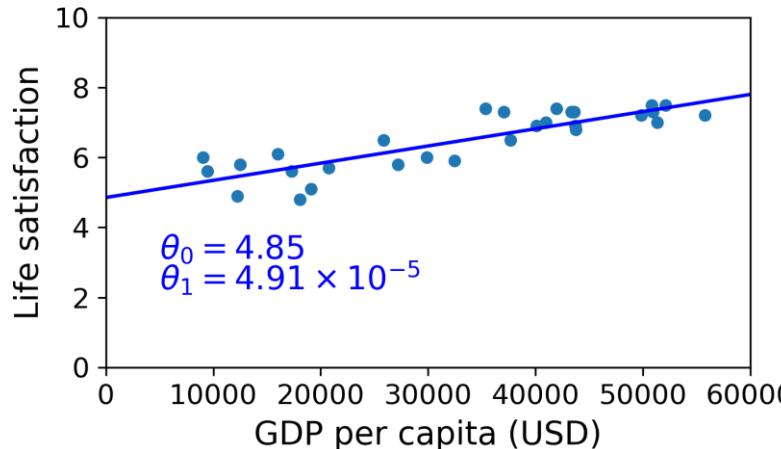
# Wähle ein Modell aus und trainiere es

- Trainieren und Auswerten auf dem Trainingsdatensatz
- Was ist ein Validierungsdatensatz?
- Kreuzvalidierung

# Wähle ein Modell aus und trainiere es

## Trainieren und Auswerten auf dem Trainingsdatensatz

- LinearRegression



```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)

from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
```

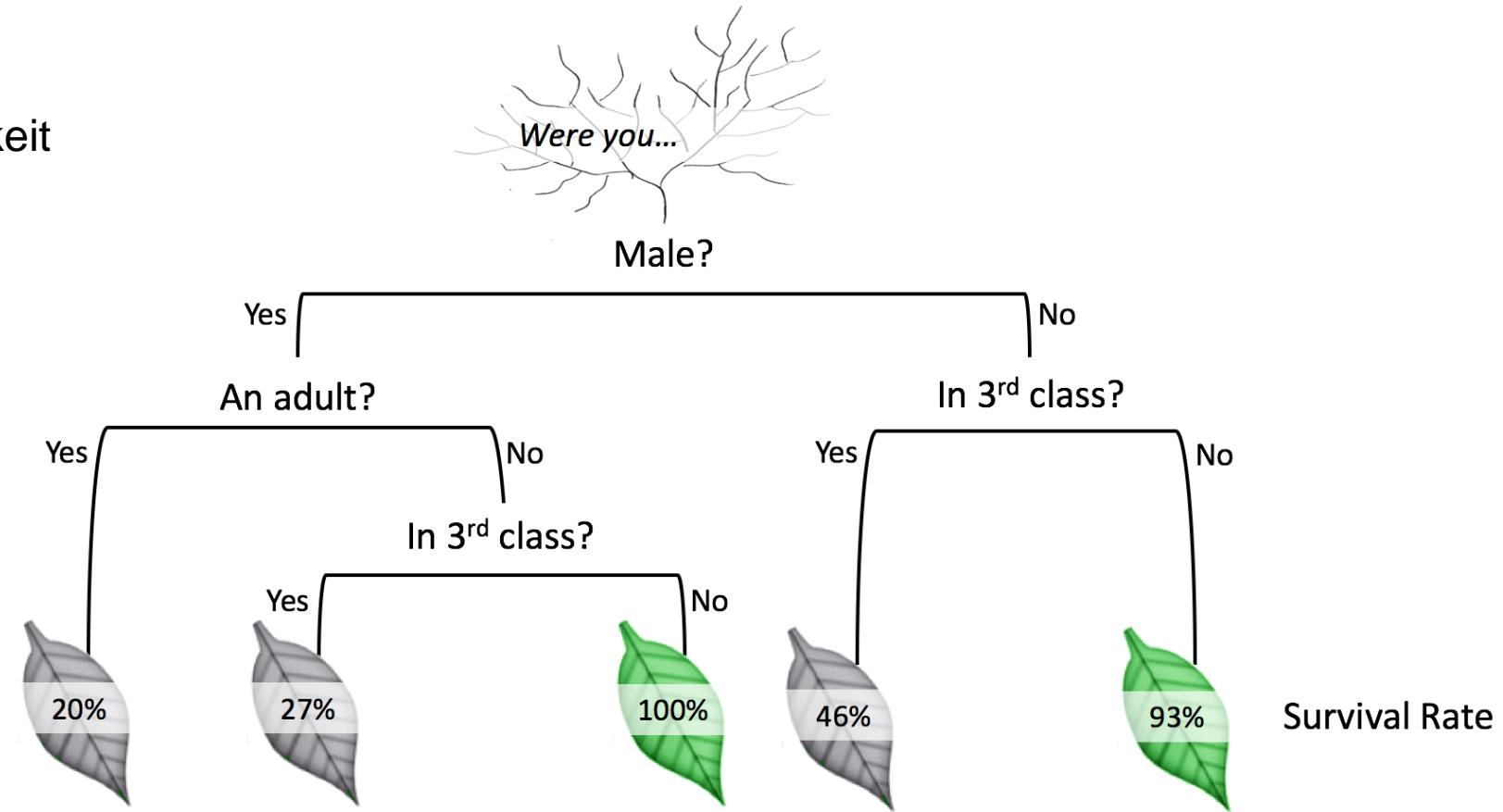
- DecisionTreeRegressor

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(housing_prepared, housing_labels)
```

# Wähle ein Modell aus und trainiere es

## Trainieren und Auswerten auf dem Trainingsdatensatz - Entscheidungsbaum

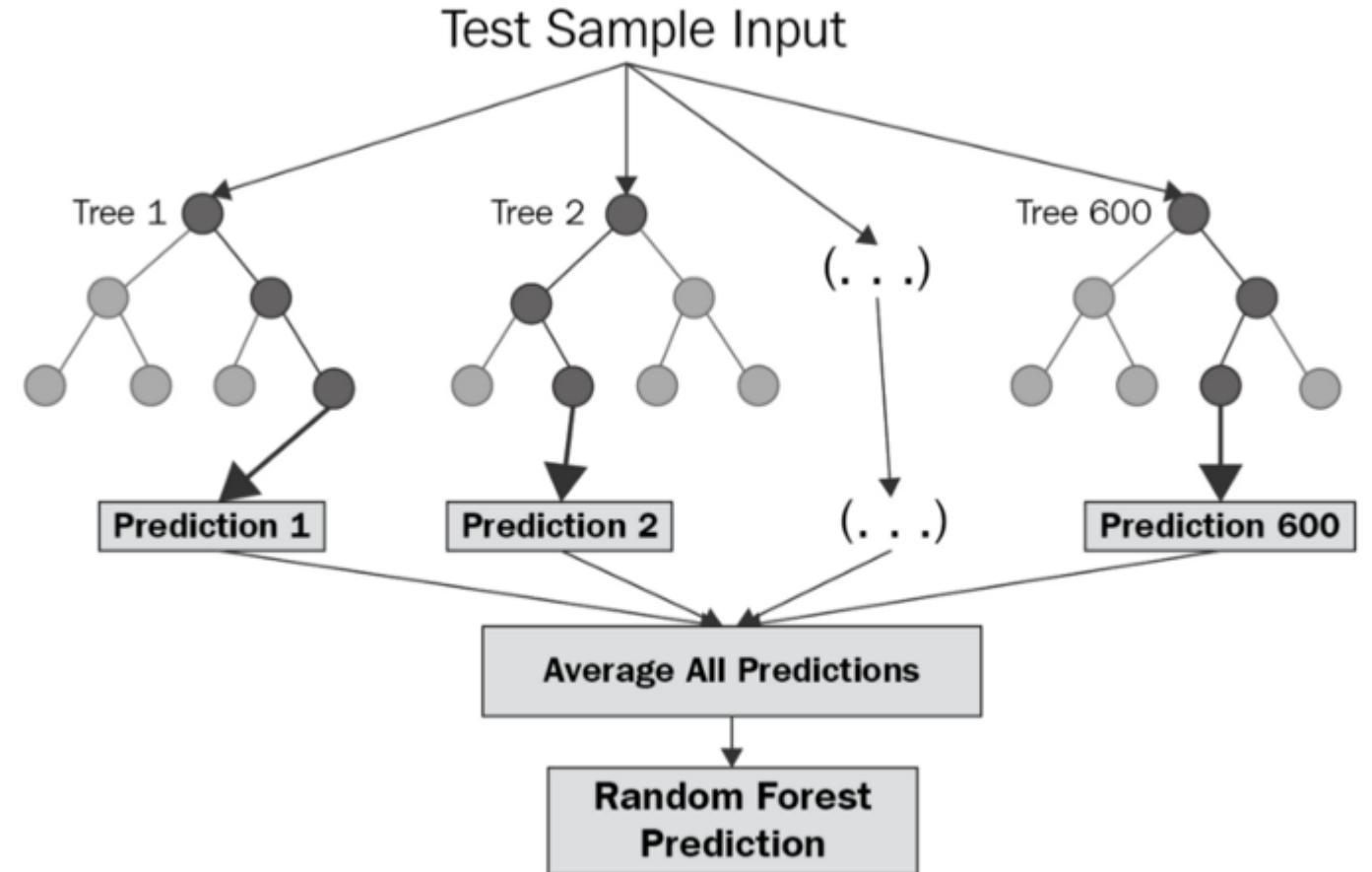
- Decision Tree
- Überlebenswahrscheinlichkeit auf der Titanic
- Modell wird aus Trainingsdaten gelernt



# Wähle ein Modell aus und trainiere es

## Trainieren und Auswerten auf dem Trainingsdatensatz – Random Forest

- RandomForestRegressor



# Wähle ein Modell aus und trainiere es

## Übung

- Trainieren Sie ein RandomForestRegressor auf Ihrem Trainingsdatensatz
- Wie groß ist der Prädiktionsfehler auf den Trainingsdaten?
- 10 Min.

```
In [63]: from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor(random_state=42)
tree_reg.fit(housing_prepared, housing_labels)
```

```
Out[63]: DecisionTreeRegressor(random_state=42)
```

```
In [64]: housing_predictions = tree_reg.predict(housing_prepared)
tree_rmse = mean_squared_error(housing_labels, housing_predictions,
                               squared=False)
tree_rmse
```

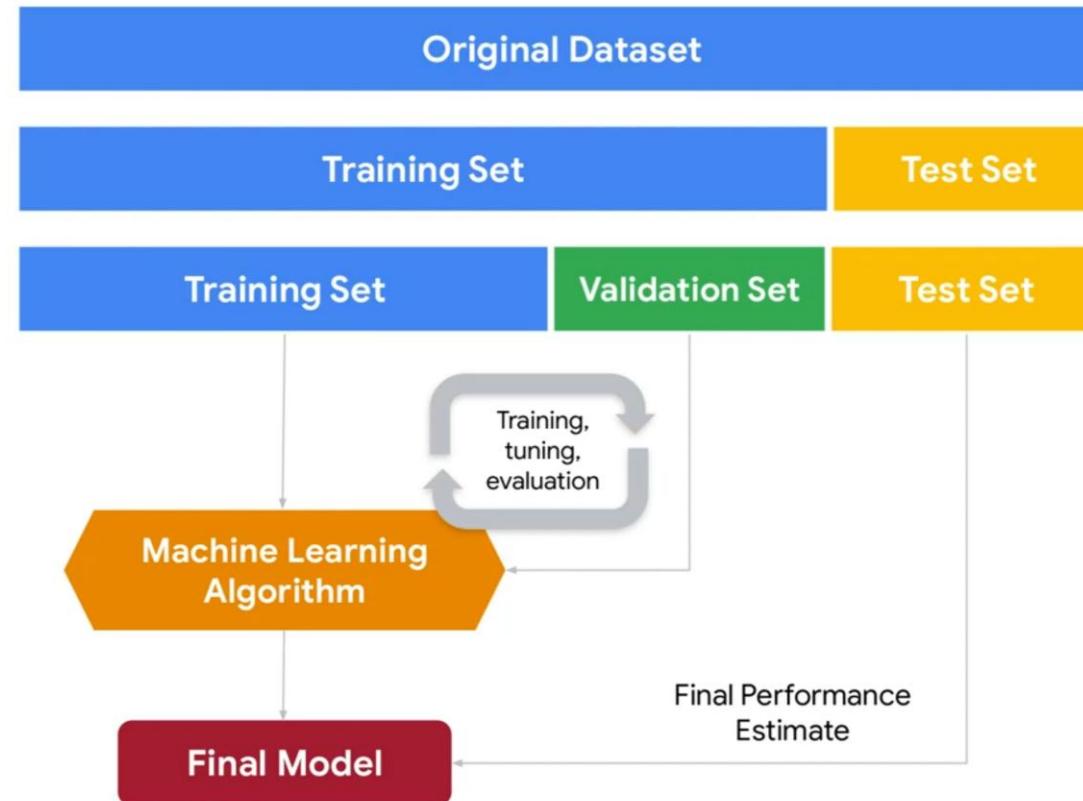
```
Out[64]: 0.0
```

```
# TODO: importieren Sie RandomForestRegressor und trainieren Sie einen Random Forest auf den vorverarbeiteten Daten
```

```
# TODO: machen Sie Vorhersagen auf den Trainingsdaten und berechnen Sie deren RMSE, nennen Sie den RMSE forest_rmse.
# Vergleichen Sie forest_rmse mit tree_rmse
# forest_rmse
```

# Wähle ein Modell aus und trainiere es

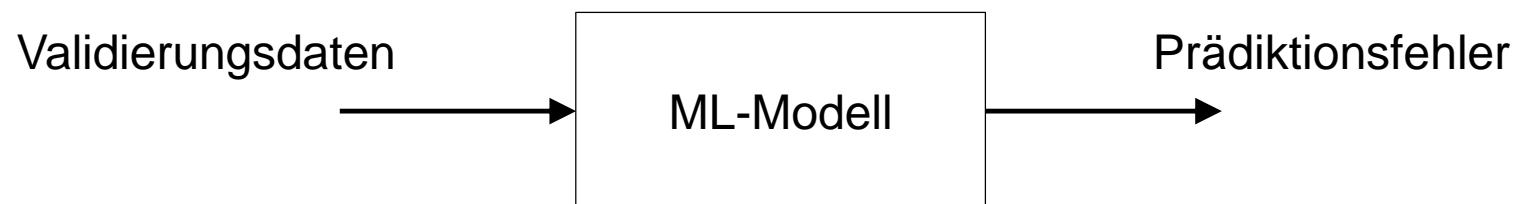
## Was ist der Validierungsdatensatz?



# Wähle ein Modell aus und trainiere es

## Was ist der Validierungsdatensatz?

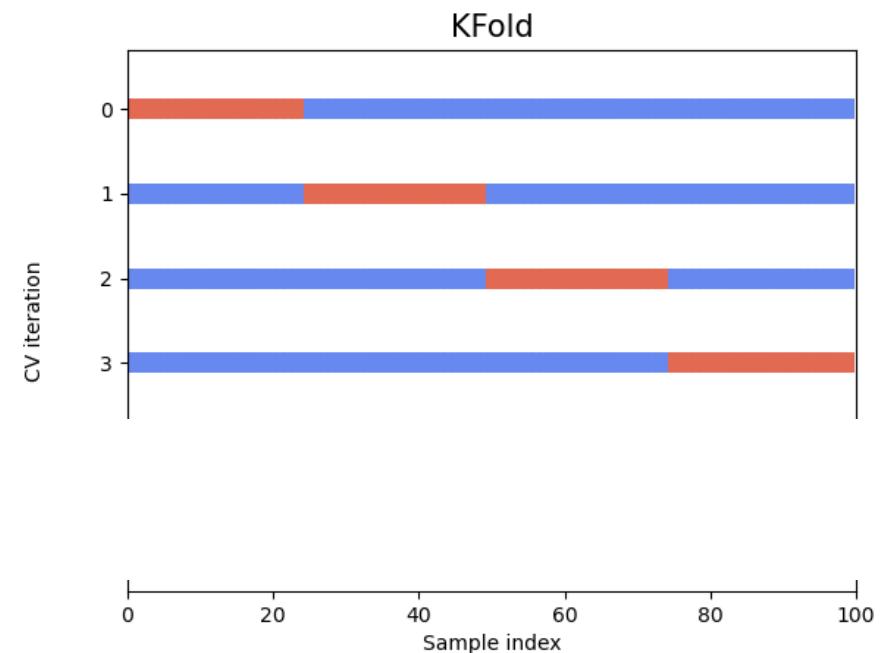
- Der Validierungsdatensatz wird aus dem Trainingsdatensatz erstellt
- Validierungsdatensatz und Trainingsdatensatz (nach der Abtrennung des Validierungsdatensatzes) sind disjunkt
- Die Berechnung des Prädiktionsfehlers der Modellvorhersagen auf dem Validierungsdatensatz, gibt eine etwas bessere Schätzung über den zu erwartenden Testfehler ab



# Wähle ein Modell aus und trainiere es

## Kreuzvalidierung

- Um eine robustere Schätzung für den zu erwartenden Testfehler des Machine Learning Modells zu erhalten, macht man den Trainings-/Validierungssplit nicht nur einmal, sondern k-mal.
- k-fache Kreuzvalidierung (k-fold Cross Validation):
  - k disjunkte Teilmengen aus dem Trainingsdatensatz
  - Verwende k-1 für Training und 1 für Validierung
  - k Trainingsdurchläufe



Bsp.: 4-fache Kreuzvalidierung

# Wähle ein Modell aus und trainiere es

## Kreuzvalidierung

- cross\_val\_score

```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()

from sklearn.model_selection import cross_val_score
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                        scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
```

# Wähle ein Modell aus und trainiere es

## Kreuzvalidierung - Übung

- Trainieren und validieren Sie den RandomForestRegressor mittels Kreuzvalidierung
  - Vergleichen Sie den mittleren Validierungsfehler mit dem Trainingsfehler

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = make_pipeline(preprocessing,
                           RandomForestRegressor(random_state=42))
forest_rmses = -cross_val_score(forest_reg, housing, housing_labels,
                                scoring="neg_root_mean_squared_error", cv=2)
```

```
pd.Series(forest_rmses).describe()
```

|        |              |
|--------|--------------|
| count  | 2.000000     |
| mean   | 49128.717554 |
| std    | 748.944859   |
| min    | 48599.133565 |
| 25%    | 48863.925560 |
| 50%    | 49128.717554 |
| 75%    | 49393.509548 |
| max    | 49658.301543 |
| dtype: | float64      |

```
from sklearn.ensemble import RandomForestRegressor
```

# TODO: Führen Sie eine Kreuzvalidierung für RandomForestRegressor aus und speichern Sie die RMSE-Werte der einzelnen Forests  
# in forest\_rmses

# Wähle ein Modell aus und trainiere es

- Abspeichern eines trainierten Machine Learning Modells

```
import joblib

joblib.dump(my_model, "my_model.pkl")
# und später ...
my_model_loaded = joblib.load("my_model.pkl")
```

# Fragen?

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - **Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung**
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Optimiere das Modell

- Optimierung der Hyperparameter
- Gittersuche
- Random Search
- Analysiere die besten Modelle und ihre Fehler
- Evaluation auf dem Testdatensatz

# Optimiere das Modell

## Wähle optimale Hyperparameter auf Basis des Validierungsdatensatzes

- Hyperparameter bei einem Machine Learning Algorithmus können sein:
  - Random Forest: Anzahl Entscheidungsbäume, maximale Tiefe der Entscheidungsbäume
  - Optimierungsalgorithmus, Schrittweite (Lernrate)
  - Art der Datenvorverarbeitung (bspw. Skalierung der Merkmale, Daten Imputation)
  - ...
- Validierungsdatensatz wird zur Bestimmung optimaler Hyperparameter des Machine Learning Algorithmus genutzt

# Optimiere das Modell

## Wähle optimale Hyperparameter auf Basis des Validierungsdatensatzes

**Parameters:**
**`n_estimators : int, default=100`**

The number of trees in the forest.

*Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.*

**`criterion : {"squared_error", "absolute_error", "friedman_mse", "poisson"}, default="squared_error"`**

The function to measure the quality of a split. Supported criteria are “squared\_error” for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, “friedman\_mse”, which uses mean squared error with Friedman’s improvement score for potential splits, “absolute\_error” for the mean absolute error, which minimizes the L1 loss using the median of each terminal node, and “poisson” which uses reduction in Poisson deviance to find splits. Training using “absolute\_error” is significantly slower than when using “squared\_error”.

*New in version 0.18: Mean Absolute Error (MAE) criterion.*

*New in version 1.0: Poisson criterion.*

**`max_depth : int, default=None`**

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

**`min_samples_split : int or float, default=2`**

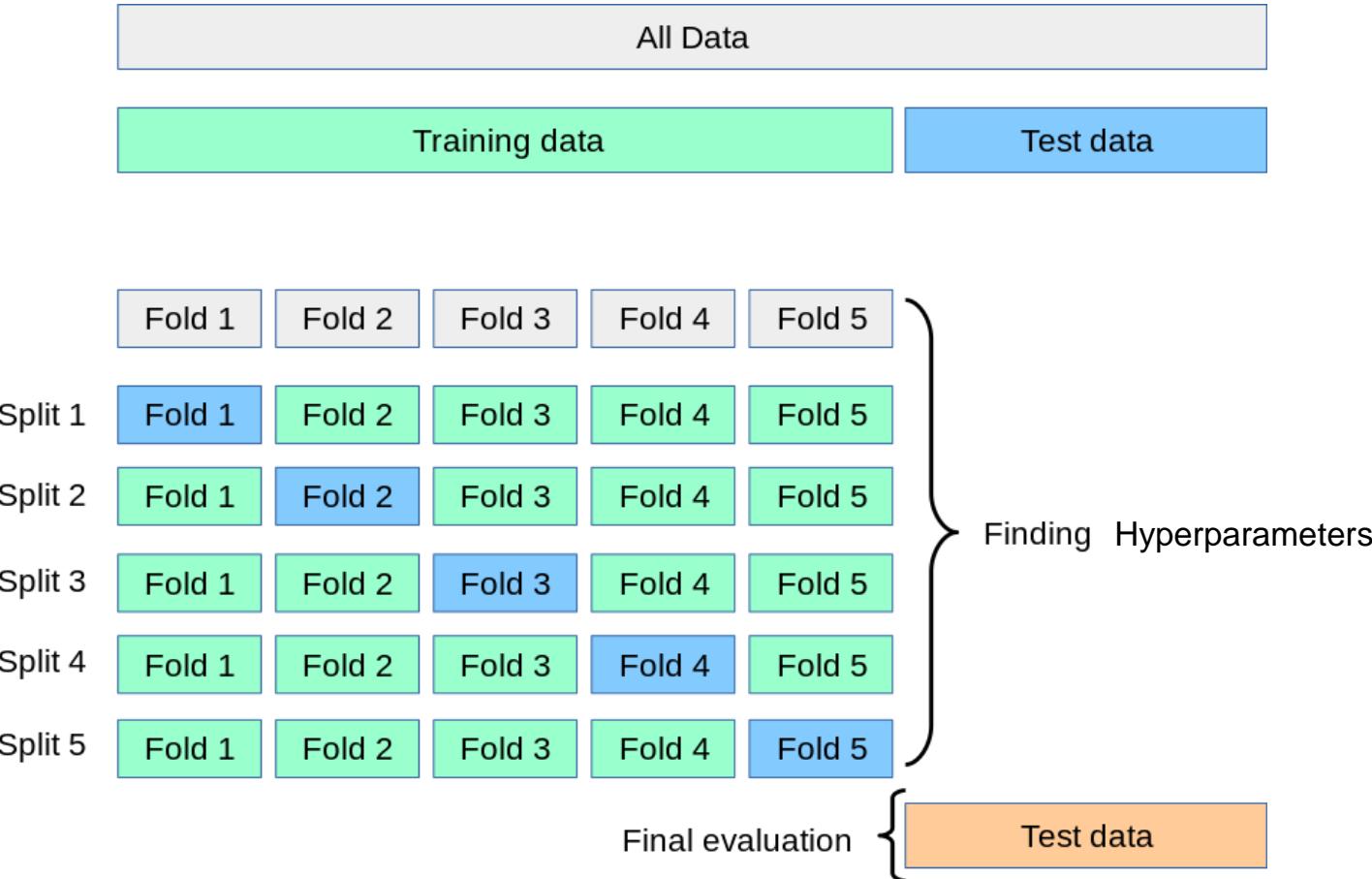
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

# Optimiere das Modell

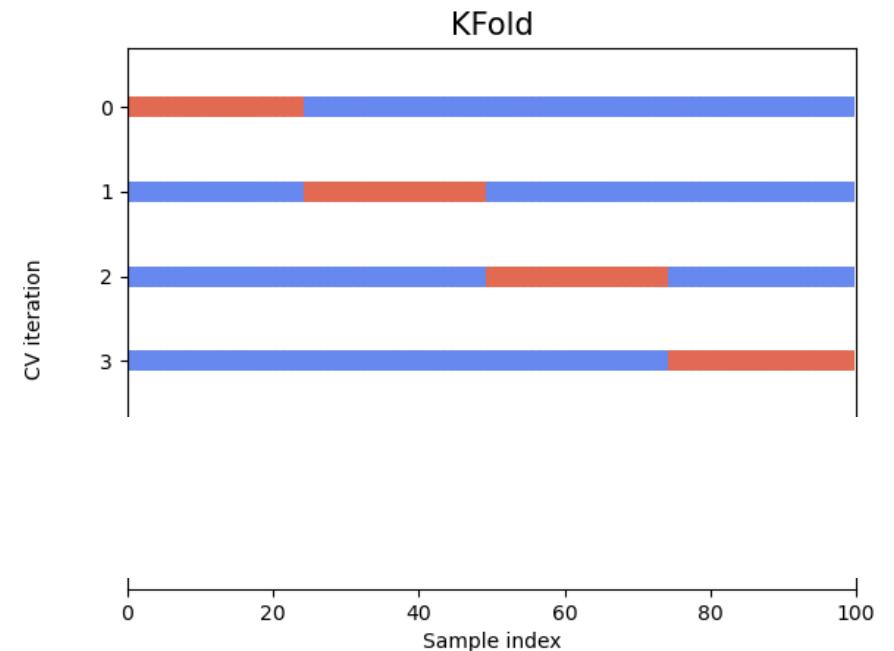
Wähle optimale Hyperparameter auf Basis des Validierungsdatensatzes



# Optimiere das Modell

## Kreuzvalidierung

- Um ein robusteres Machine Learning Modell zu erhalten, macht man den Trainings-/Validierungssplit nicht nur einmal, sondern k-mal.
- k-fache Kreuzvalidierung (k-fold Cross Validation):
  - k disjunkte Teilmengen aus dem Trainingsdatensatz
  - Verwende  $k-1$  für Training und 1 für Validierung
  - k Trainingsdurchläufe
  - Pro Hyperparameter erhält man k Ergebnisse
    - Mittelwert, Standardabweichung, etc. über die k Ergebnisse (Genauigkeit, ...) sind ein robustes Maß für die Qualität des Hyperparameters



Bsp.: 4-fache Kreuzvalidierung

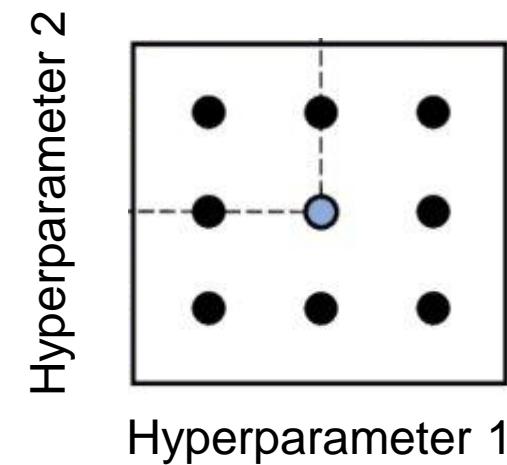
# Optimiere das Modell

## Optimierung der Hyperparameter - Gittersuche

- Angabe der zu optimierenden Hyperparameter und deren Wertebereiche

```
param_grid = [  
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},  
]
```

- GridSearchCV
  - Evaluiert alle möglichen Kombinationen der Hyperparameter über eine Kreuzvalidierung



# Optimiere das Modell

## Gittersuche

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
]

forest_reg = RandomForestRegressor()

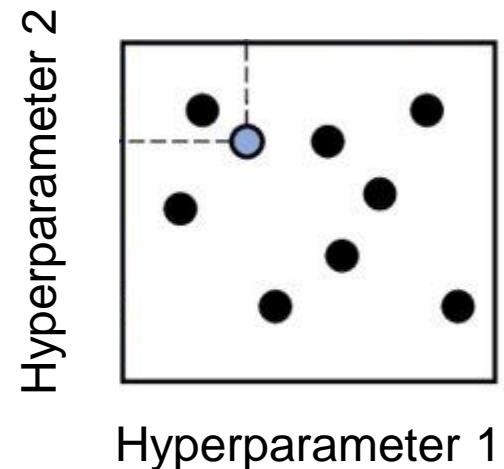
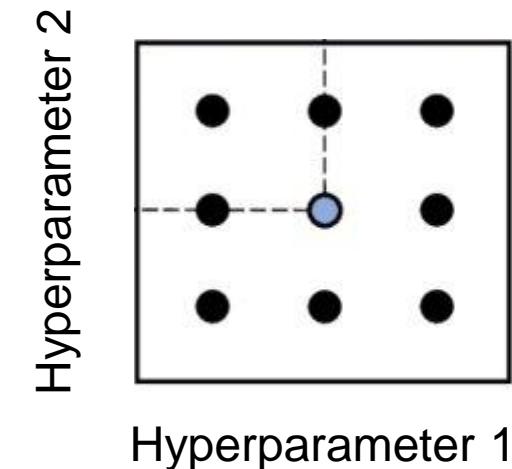
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

# Optimiere das Modell

## Optimierung der Hyperparameter - Zufällige Suche

- RandomizedSearchCV
  - Zufällig gewählte Werte für die Hyperparameter werden evaluiert
  - Man kann angeben wie viele Durchläufe gemacht werden sollen
  
- Werte der besten Hyperparameter:
  - `grid_search.best_params_`
  
- Das beste Machine Learning Modell
  - `grid_search.best_estimator_`



# Optimiere das Modell

## Optuna: A hyperparameter optimization framework

- <https://github.com/optuna/optuna>

[https://github.com/optuna/optuna-examples/blob/main/sklearn/sklearn\\_simple.py](https://github.com/optuna/optuna-examples/blob/main/sklearn/sklearn_simple.py)

```
def objective(trial):
    iris = sklearn.datasets.load_iris()
    x, y = iris.data, iris.target

    classifier_name = trial.suggest_categorical("classifier", ["SVC", "RandomForest"])
    if classifier_name == "SVC":
        svc_c = trial.suggest_float("svc_c", 1e-10, 1e10, log=True)
        classifier_obj = sklearn.svm.SVC(C=svc_c, gamma="auto")
    else:
        rf_max_depth = trial.suggest_int("rf_max_depth", 2, 32, log=True)
        classifier_obj = sklearn.ensemble.RandomForestClassifier(
            max_depth=rf_max_depth, n_estimators=10
        )

    score = sklearn.model_selection.cross_val_score(classifier_obj, x, y, n_jobs=-1, cv=3)
    accuracy = score.mean()

    return accuracy
```

---

```
if __name__ == "__main__":
    study = optuna.create_study(direction="maximize")
    study.optimize(objective, n_trials=100)
    print(study.best_trial)
```

# Optimiere das Modell

## Übung

- Führen Sie eine Hyperparameteroptimierung für den DecisionTreeRegressor durch
- 10 Min.

```
In [69]: from sklearn.model_selection import GridSearchCV
param_grid = [
    # try 4 values of max_features at once
    {'max_features': [2, 4, 6, 8]},
]
tree_reg = DecisionTreeRegressor(random_state=42)
# train across 5 folds, that's a total of (12+6)*5=90 rounds of training
grid_search = GridSearchCV(tree_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(housing_prepared, housing_labels)

Out[69]: GridSearchCV(cv=5, estimator=DecisionTreeRegressor(random_state=42),
                      param_grid=[{'max_features': [2, 4, 6, 8]}],
                      return_train_score=True, scoring='neg_mean_squared_error')
```

The best hyperparameter combination found:

```
In [70]: grid_search.best_params_
```

```
Out[70]: {'max_features': 8}
```

```
In [71]: grid_search.best_estimator_
```

```
Out[71]: DecisionTreeRegressor(max_features=8, random_state=42)
```

# Optimiere das Modell

## Analysiere die besten Modelle und ihre Fehler

- feature\_importances

```
final_model = rnd_search.best_estimator_ # includes preprocessing
feature_importances = final_model["random_forest"].feature_importances_
feature_importances.round(2)

array([0.08, 0.06, 0.08, 0.01, 0.01, 0.01, 0.01, 0.24, 0.03, 0.02, 0.05,
       0.04, 0.03, 0.06, 0.03, 0.02, 0.03, 0.02, 0.12, 0. , 0. , 0.01,
       0.03])
```

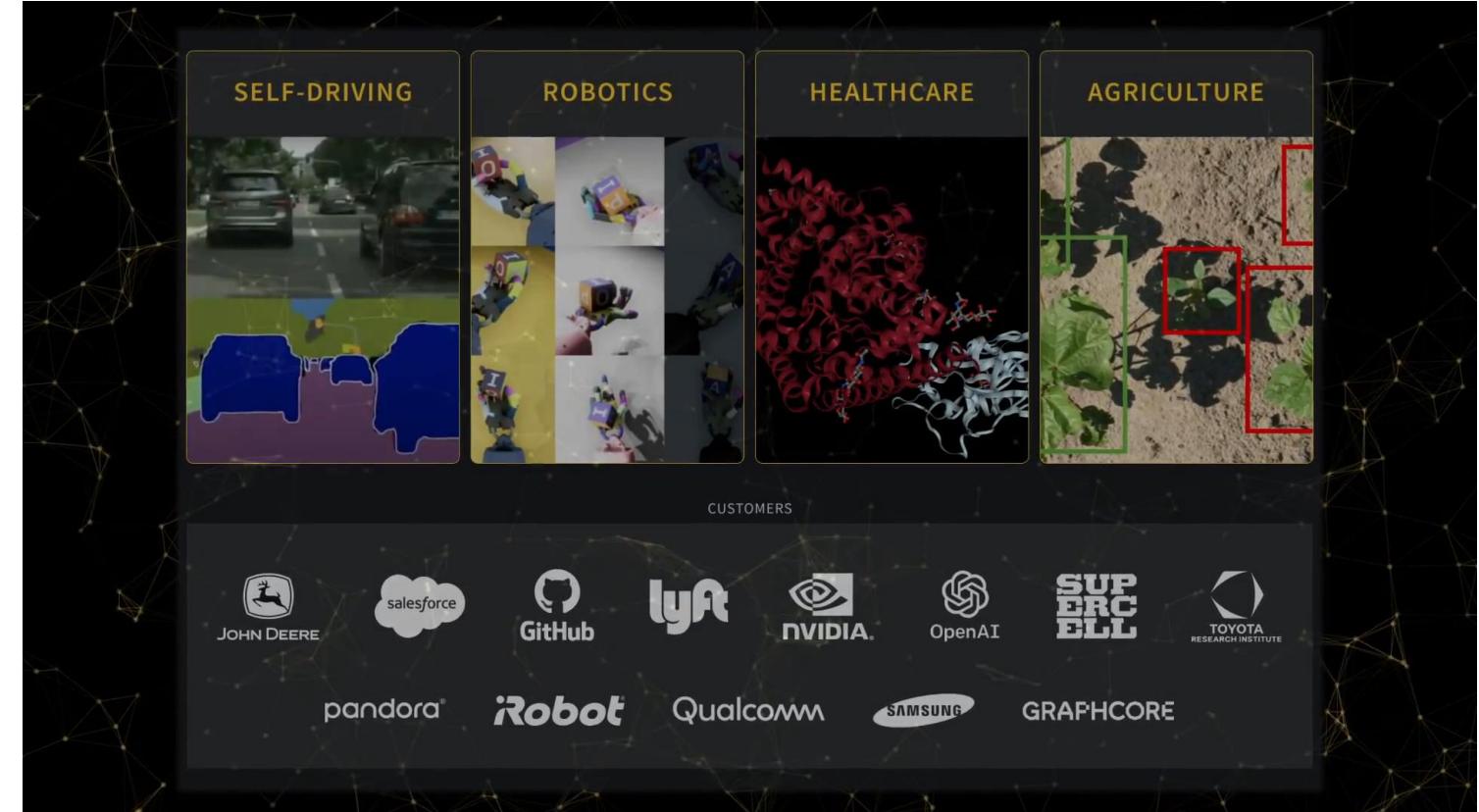
```
sorted(zip(feature_importances,
           final_model["preprocessing"].get_feature_names_out()),
       reverse=True)
```

```
[(0.24109295782313162, 'log__median_income'),
 (0.12230764758860226, 'cat__ocean_proximity_INLAND'),
 (0.08359848144981241, 'bedrooms__ratio'),
 (0.07794622064394396, 'people_per_house__ratio'),
 (0.06010110969683507, 'geo__Cluster 5 similarity'),
 (0.06007479778992901, 'rooms_per_house__ratio'),
 (0.047244619197197385, 'geo__Cluster 2 similarity'),
 (0.03554999761549535, 'geo__Cluster 3 similarity'),
 (0.0349051266051341, 'geo__Cluster 4 similarity'),
 (0.031174653614035774, 'geo__Cluster 6 similarity'),
 (0.030080451852985277, 'geo__Cluster 0 similarity'),
 (0.029693615131634588, 'geo__Cluster 8 similarity'),
 (0.02544740787138902, 'remainder__housing_median_age'),
 (0.02167295418290811, 'geo__Cluster 1 similarity'),
 (0.01825909802674457, 'geo__Cluster 7 similarity')]
```

# Optimiere das Modell

## Analysiere die besten Modelle und ihre Fehler

- Weights & Biases
- <https://wandb.ai/site>



# Optimiere das Modell

## Evaluiere das Machine Learning System auf dem Testdatensatz

- Wenn Sie glücklich mit Ihrem Modell sind, dann können Sie jetzt **einmal**:
  - Testdatensatz mit Pipeline transformieren und dann mit **finalem** Modell Vorhersagen machen
  - Hier ist Datentransformation und finales Modell bereits in der Pipeline `final_model` gespeichert

```
X_test = test_set.drop("median_house_value", axis=1)
y_test = test_set["median_house_value"].copy()

final_predictions = final_model.predict(X_test)

final_rmse = mean_squared_error(y_test, final_predictions, squared=False)
print(final_rmse)
```

# Optimiere das Modell

## Übung - Evaluiere das Machine Learning System auf dem Testdatensatz

- Transformieren Sie den Testdatensatz mit Pipeline und machen Sie mit **finalem** Modell Vorhersagen
- 10 Min.

```
X_test = test_set.drop("median_house_value", axis=1)
y_test = test_set["median_house_value"].copy()

final_predictions = final_model.predict(X_test)

final_rmse = mean_squared_error(y_test, final_predictions, squared=False)
print(final_rmse)
```

# Fragen?

# Kurze Pause?

- Ja, 5 Minuten
- Ja, 10 Minuten
- Nein

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - **Stellen Sie Ihre Lösung vor**
  - Starten, beobachten und warten Sie Ihr System

# Stellen Sie Ihre Lösung vor

- Dokumentieren Sie Ihr Werk
- Erstellen Sie eine ansprechende Präsentation
  - Heben Sie zu Beginn das Gesamtbild hervor
  - Stimmen Sie die Inhaltstiefe auf Ihre Zielgruppe ab
- Erklären Sie, warum Ihre Lösung zum Geschäftsziel beiträgt
- Vergessen Sie nicht, auf dem Weg gewonnene interessante Erkenntnisse zu erwähnen
  - Beschreiben Sie, was funktioniert hat und was nicht
  - Zählen Sie Ihre Annahmen und die Beschränkungen Ihres Systems auf
- Stellen Sie sicher, dass Ihre wichtigsten Erkenntnisse durch eine ansprechende Visualisierung oder eingängige Aussagen untermauert werden (z.B. »das mittlere Einkommen hat den stärksten Einfluss bei der Vorhersage von Immobilienpreisen«)

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - **Starten, beobachten und warten Sie Ihr System**

# Starten, beobachten und warten Sie Ihr System

## Was kann ich mit dem Modell machen?

- Wo soll das Modell laufen?
  - Webserver
    - Google Cloud AI Platform
    - JSON-Requests und Responses
    - TFX, TensorFlow Serving
  - Mikrocontroller
    - Kompression des Modells (Quantisierung)
    - Für neuronale Netze → TensorFlow Lite, Edge Impulse, ...
  - Browser
    - TensorFlow.js

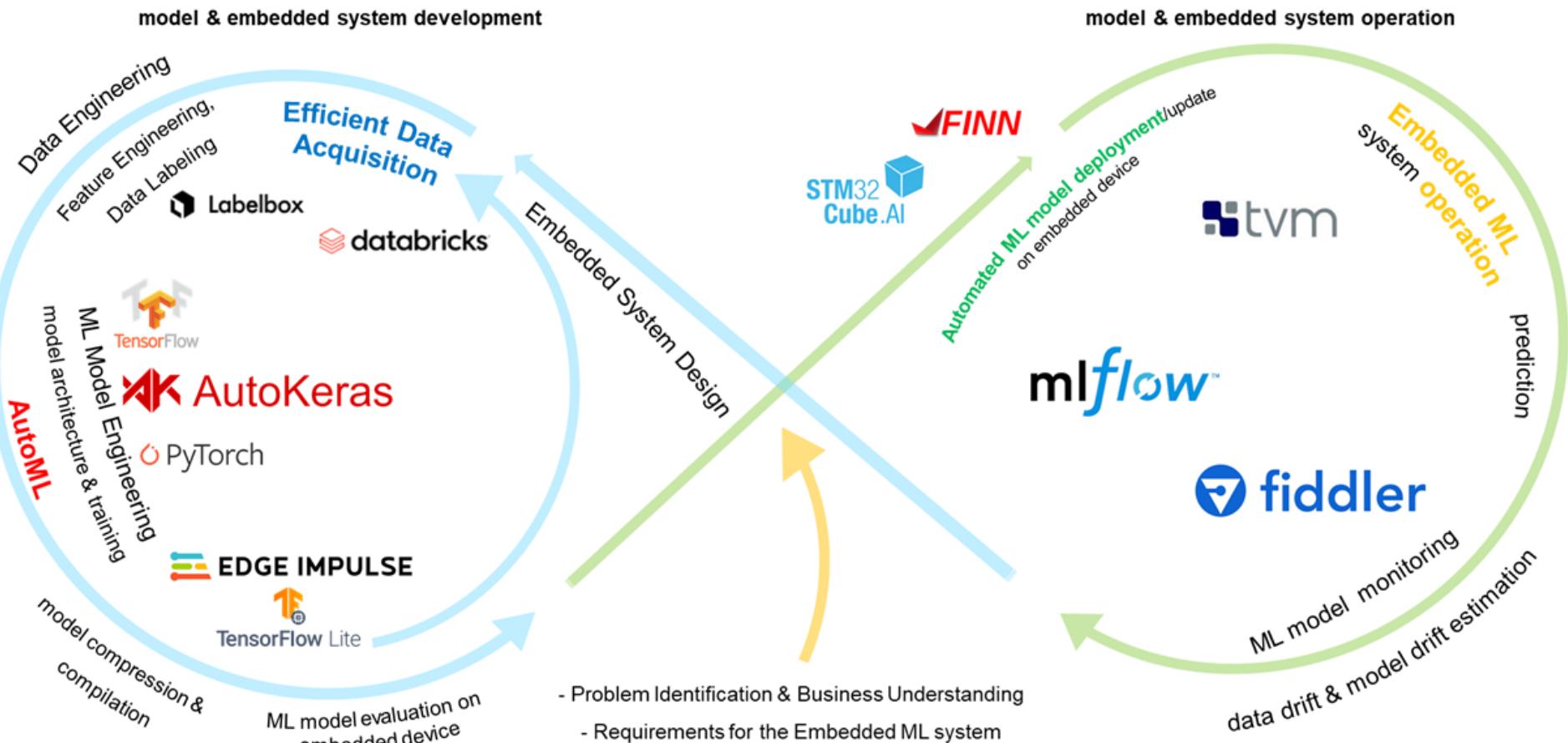


# Starten, beobachten und warten Sie Ihr System

- Überwachen der Qualität des Modells
  - Verändern sich die Eingangsdaten?
  - Modell regelmäßig Nachtrainieren, wenn mehr Daten zur Verfügung stehen
- MLOps            → DevOps für Machine Learning

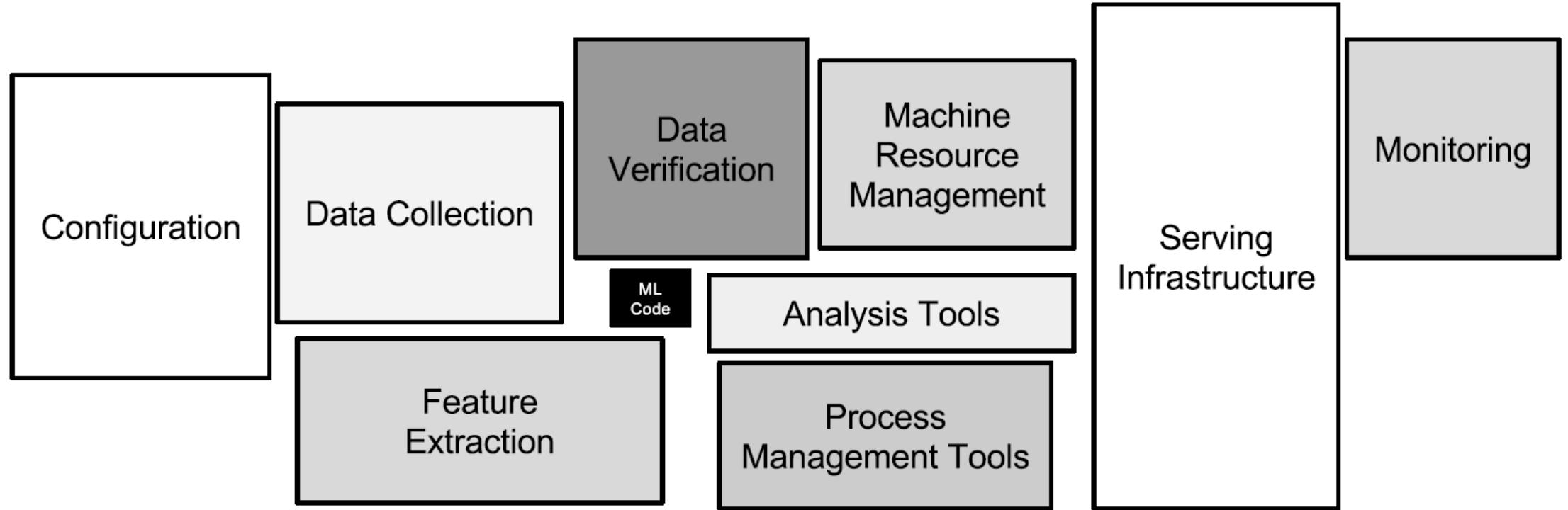
# Starten, beobachten und warten Sie Ihr System

## MLOps



# Starten, beobachten und warten Sie Ihr System

## Hidden Technical Debt in Machine Learning Systems



# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt?

## Zusammenfassung und Fragen

- Ein Machine Learning Projekt von A bis Z
  - Klären Sie die Aufgabenstellung und betrachten Sie die Gesamtsituation
  - Beschaffen Sie sich Daten
  - Erkunden und visualisieren Sie die Daten, um daraus Erkenntnisse zu gewinnen
  - Bereiten Sie die Daten so auf, dass Machine-Learning-Algorithmen die Muster darin leichter erkennen können
  - Probieren Sie viele unterschiedliche Modelle aus und treffen Sie eine engere Auswahl
  - Optimieren Sie Ihre Modelle und kombinieren Sie diese zu einer guten Lösung
  - Stellen Sie Ihre Lösung vor
  - Starten, beobachten und warten Sie Ihr System

# Lernraum I: Wie wird ein Machine Learning Projekt durchgeführt? Ausblick

- Geht das auch, ohne viel zu programmieren?
- PYCARET IS DEMOCRATIZING MACHINE LEARNING
- PyCaret empowers anyone to build low-code, powerful, end-to-end machine learning solutions. There are tons of ways for you to get started.
- <https://pycaret.org/>
- <https://pycaret.gitbook.io/docs/get-started/tutorials>

# Lernziele von heute und Fragen zur Überprüfung der Lernziele

Vorlesung am 3.6. (offen für alle)

Die Studierenden können kleine KI-Apps in Python entwickeln, indem sie

- für eine gegebene Anwendung ein geeignetes vortrainiertes Deep Learning Modell (LLM, Bilderkennung, Text2Speech, Speech2Text) auswählen,
- dieses herunterladen,
- mit Gradio eine App unter Einbindung des Deep Learning Modells in Python programmieren,
- das Deep Learning Modell testen und moderne Techniken nutzen, um die Performance des Modells zu steigern,

um später eigene Apps unter Nutzung von Deep Learning Modellen umsetzen und optimieren zu können.

# Nächste Vorlesung und Fragen

- Neuronale Netze und Deep Learning mit Prof. Dr. W. Konen
- Fragen?