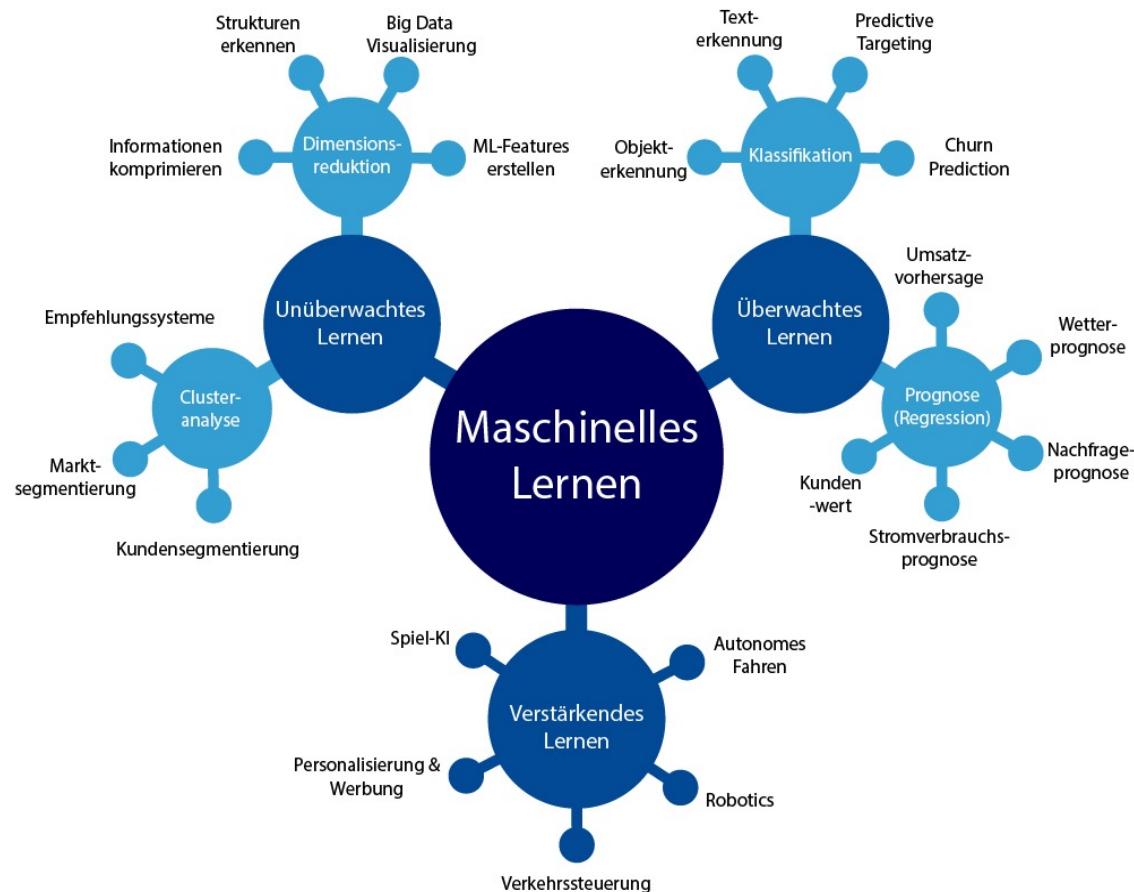


Artificial Intelligence / Machine Learning

02/2023

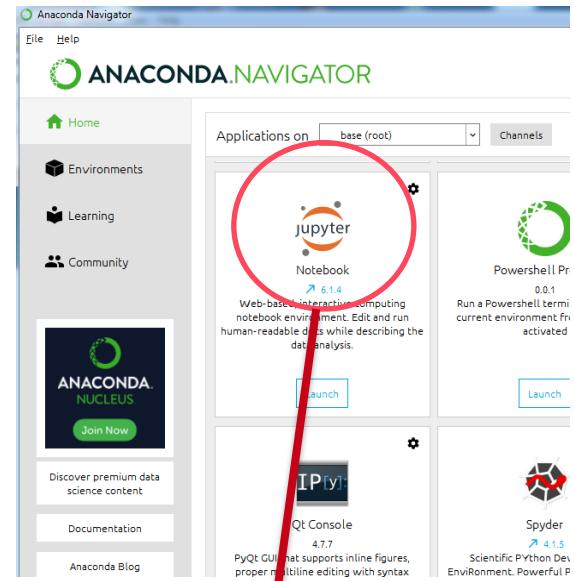


1. Einführung in das programmgesteuerte DataMining und das Machine-Learning



1.0 Python als Umgebung für das Machine Learning

- Wir arbeiten mit:
 - **Anaconda** – Ein Python Komplettpaket mit vielen Dingen, die normalerweise einzeln installiert werden müssten, wie z.B. APIs, Jupyter-Notebook usf.
 - **Jupyter-Notebook** – Einer flow-basierten Programmier- und Ausführungsumgebung
 - Läuft als lokaler Webserver
 - Einstieg: <http://localhost:8890>

A screenshot of a Jupyter Notebook browser window titled 'jupyter 03-NützlichesInPythonFürMachineLearning1 (autosaved)'. The notebook contains a single cell with the title 'Nützliches für Machine Leaning in Python I' and 'Numpy Arrays'. The code in the cell is:

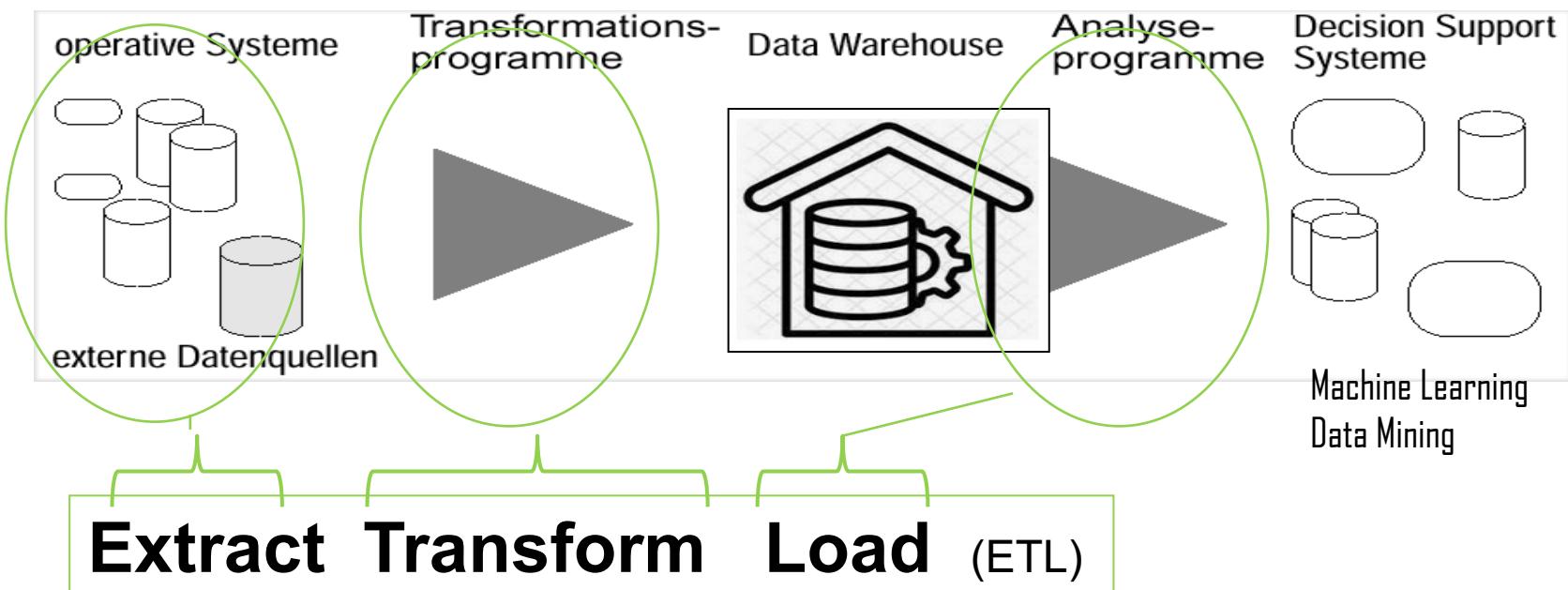
```
In [129]: #spezielle Arrays mit numpy
import numpy as np
numbers = np.array([1,2,3,4,5,6,7,8])
print(numbers)
print(numbers * 2)
print(numbers / 2)
print(numbers < 4)
# in anderen Programmiersprachen ist das viel komplizierter und i
# Python ist das auch speziell optimiert.

[1 2 3 4 5 6 7 8]
[ 2  4  6  8 10 12 14 16]
[-1  0  1  2  3  4  5  6]
[ True  True  True False False False False]
```

The cell output shows the results of the numpy operations.

1. 1 Grundsätzliches Vorgehen im Machine Learning

- **Datengewinnung nach dem ETL-Prinzip**
 - Eine Datenanalyse wird häufig in separaten Systemen, sog. **Data-Warehouses** realisiert.
 - Aus operativen Systemen müssen Daten zu Analysezwecken entnommen, umgeformt und adäquat gespeichert werden. Prinzip: Extract -> Transform -> Load (ETL)



1.1 Grundsätzliches Vorgehen im Machine Learning

Extract Transform Load (ETL)

– Extract-Beispiel

Hier ein Beispiel zur Gewinnung von zeitaktuellen Daten aus einer SQL-Datenbank.
Die Ausgabe als CSV-Datei kann per HTTP-Aufruf erzeugt werden.

weatherSurvey.ipynb Python-Programm holt Daten

```
import requests
import numpy as np
import pandas as pd
df = pd.read_csv("http://windmesser.irpfel.org/weatherCSV.php?lastDays=5")
```

station	temperature	QFE	QNH	dewpoint	ceiling	humidity	speed_avg	speed_min	speed_max
EDNG	12.19	950.80	1011.17	8.09	3431.83	75.99	2	2	
EDNG	12.27	950.77	1011.14	8.10	3461.55	75.62	3	3	
EDNG	12.38	950.81	1011.18	8.13	3493.39	75.23	3	3	
EDNG	12.52	950.77	1011.14	8.14	3505.70	75.74	^	^	

HTTP

Extraktion aus einer SQL-Datenbank und Umwandlung in CSV-Format durch ein PHP-Programm

```
<?php
//Ausbabe einer CSV-Datei, um die Daten der EDNG-Sensorbeobachtung in Python-Skripten darzustellen
//R. Assfalg 04/2022
//
if (isset($_GET['lastDays'])) {
    $lastDays = intval($_GET['lastDays']);
    if ($lastDays < 1)
        exit("<html>\nUngültiger Aufrufparameter!\n</html>\n");
    $sql = "SELECT * FROM weather where DATEDIFF(NOW(), createTime) < $lastDays order by createTime DESC";
} else
    $sql = "SELECT * FROM weather order by createTime DESC LIMIT 1000";
$pdo = new PDO('mysql:host=rdbms.strato.de;dbname=XXXXX', 'XXXXX', 'XXXXX');

//Header der CSV-Datei
$columnsquery = $pdo->query("SHOW columns from weather");
$columns = array();
foreach ($columnsquery as $k => $v) {
    $header .= $v[0].',';
}
echo substr($header, 0, -1)."\n"; //letztes Komma am Zeilenende muss weg, dann <CRLF>

//Daten der CSV-Datei
$result = $pdo->query($sql);
while($rows = $result->fetch(PDO::FETCH_OBJ)) { //Hole anonyme Objekte
    $csvline = '';
    foreach($rows as $key => $value)
        $csvline .= $value.';';
    echo substr($csvline, 0, -1)."\n"; //letztes Komma am Zeilenende muss weg, dann <CRLF>
}?

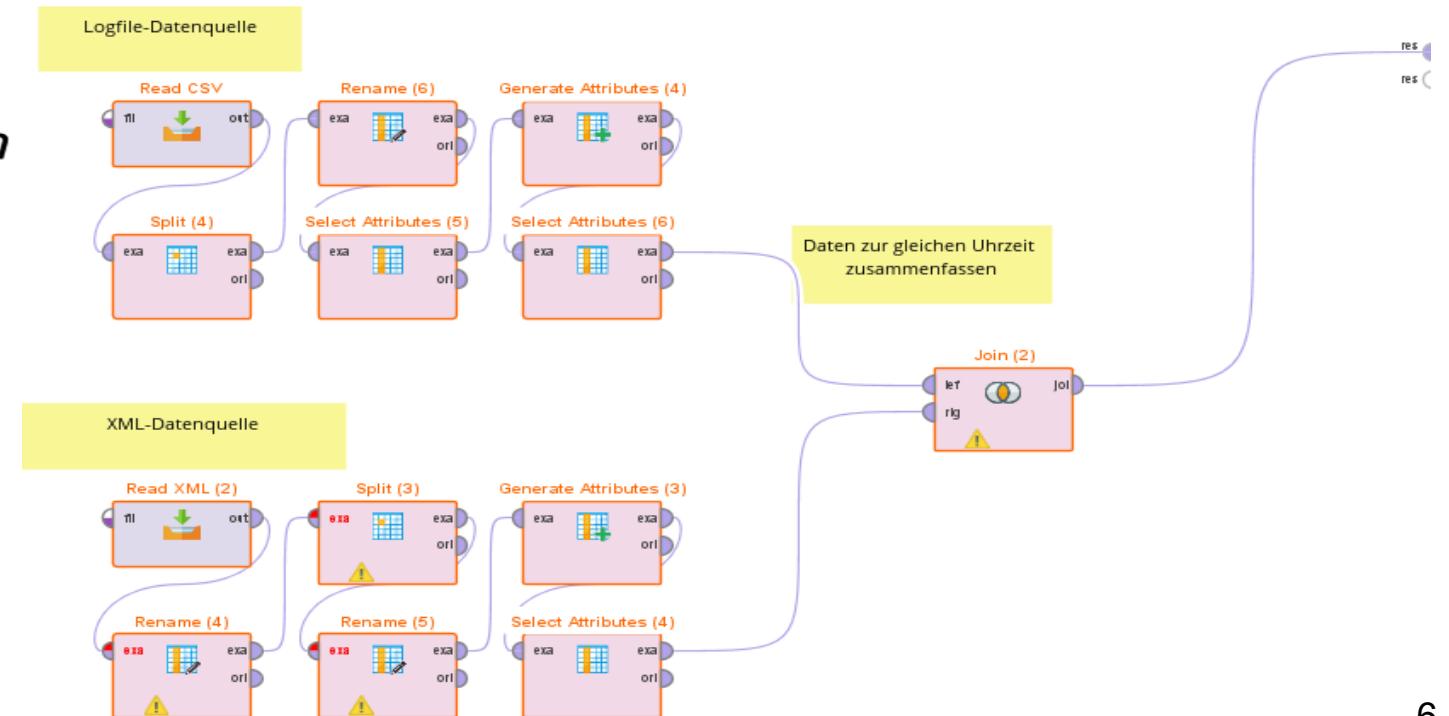
```

SQL

- **Typische Datenvorverarbeitungsschritte im DataMining**

- **Extraktionsmöglichkeiten bei XML**

- *Projektion (π) und Selektion (σ): Gesucht sind z.B.*
 - Einzelne Spalten aus CSV
 - Einzelne Tag-Inhalte oder Tag-Parameter aus XML
- *Umwandlung von XML per XSLT*
- *Extraktion mit XPATH-Ausdrücken*
- *Zusammenführung (joining)*

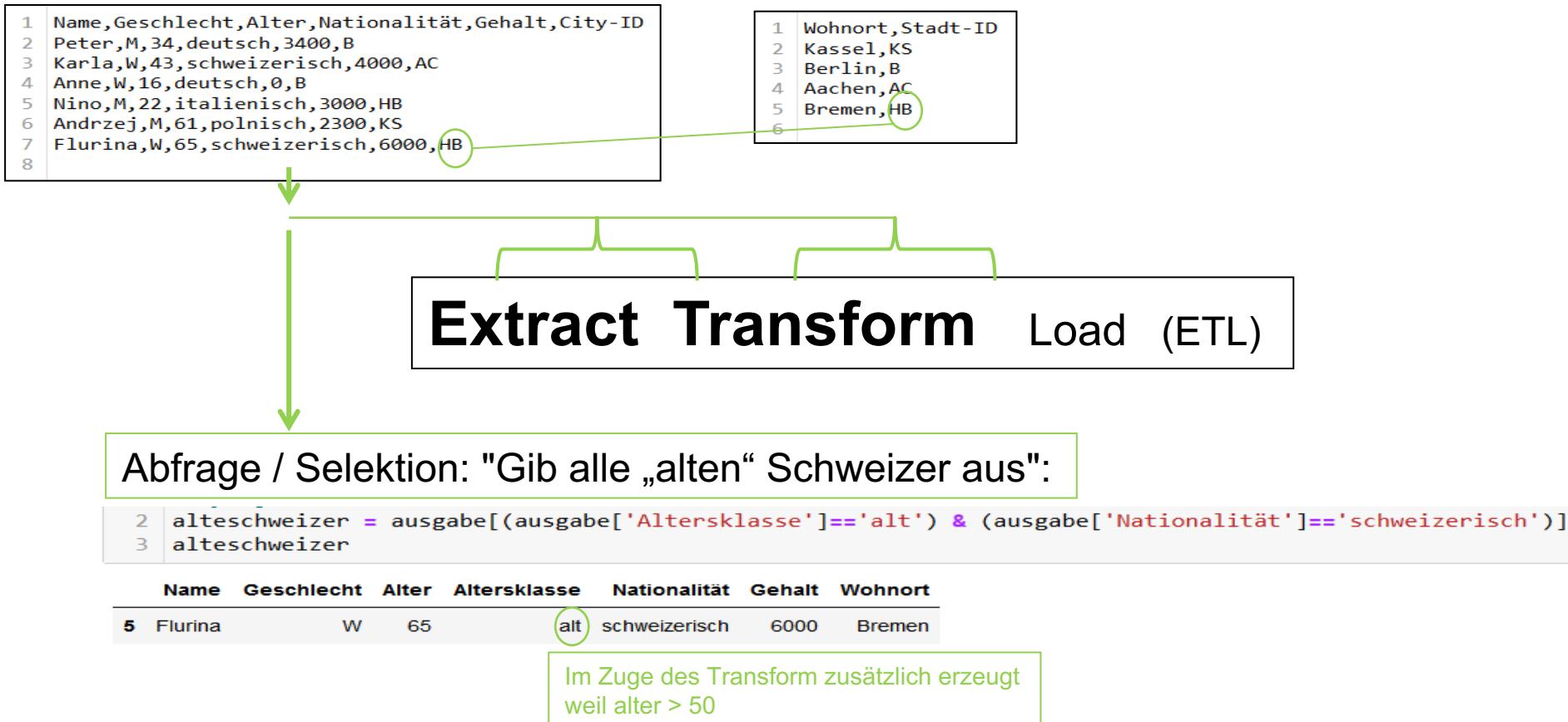


1.1 Grundsätzliches Vorgehen im Machine Learning

– Extract-Transform-Beispiel in Python:

Transform-BespielZuJoinMerge.ipynb

Python-Programm holt Daten aus CSV-Dateien, führt die Daten zusammen, ändert diese und filtert sie

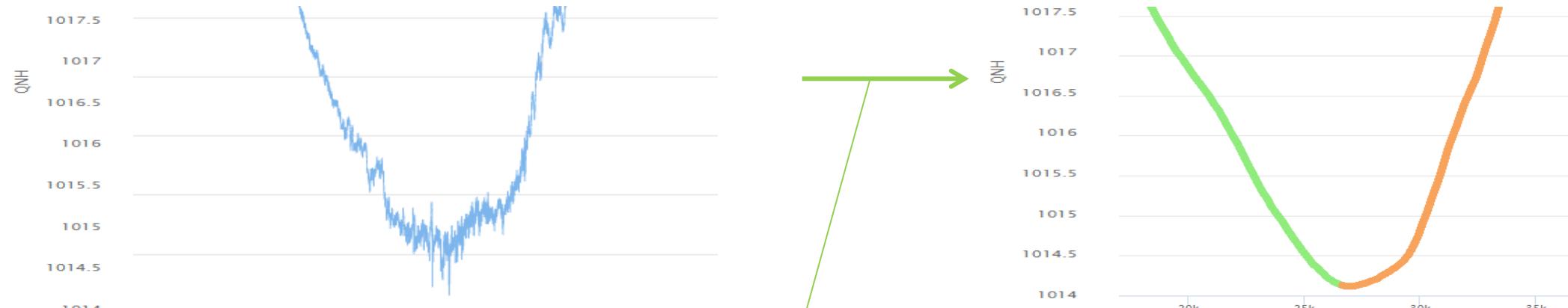


Extract **Transform** Load (ETL)

– Konvertierung

- *Filtern / Mathematische Operationen
(Integration, Differentiation, ...)*

Beispiel: Luftdruck bei Frontdurchgang



Extract **Transform** Load (ETL)

Extract **Transform** Load (ETL)

– **Cell Replacing**

– Numerical to Binomial



– Binomial to Numerical



– **Partinioning**

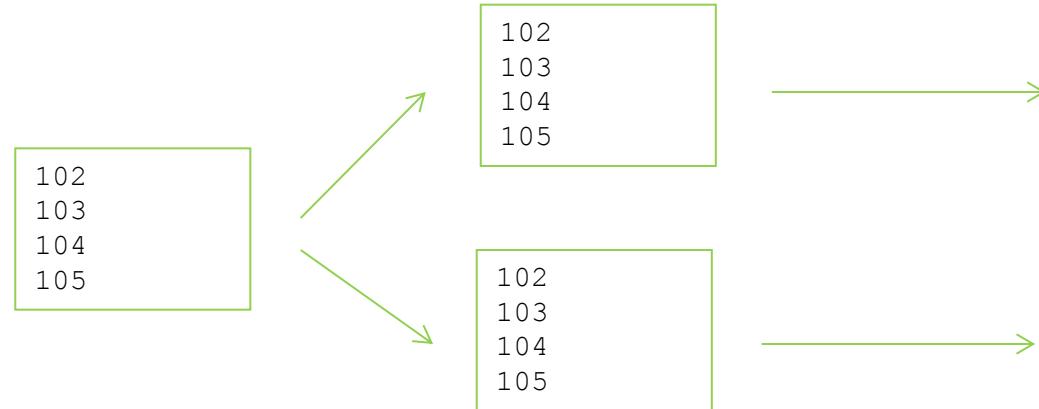
– Binning

(z.B. 3 Bereiche)



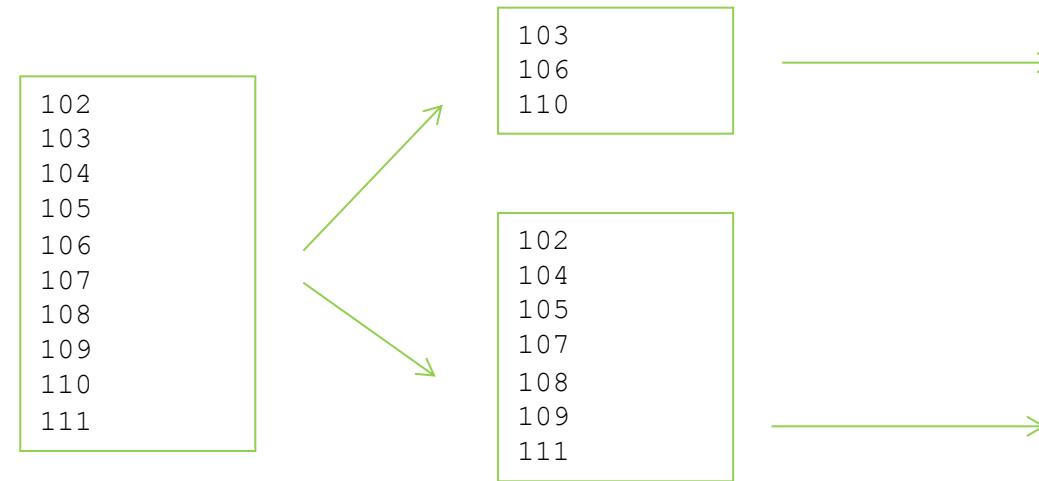
Extract **Transform** Load (ETL)

- **Multiply**



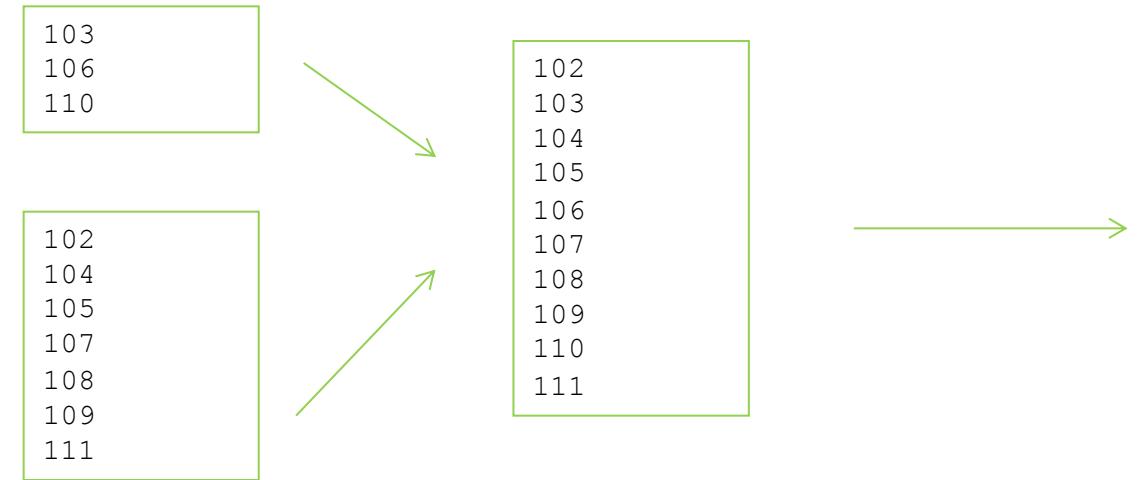
- **Split**

(z.B. 30% - 70 %)

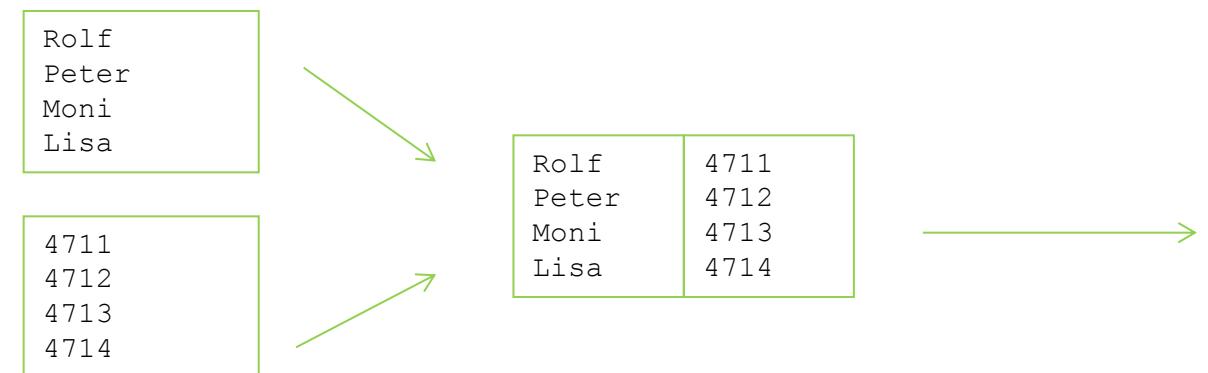


Extract **Transform** Load (ETL)

– **Concatenating**



– **Joining / Merging**



Extract **Transform** Load (ETL)

- Beispiel zu Joining/Merging und Cell Placing

- Beispieldatei: **Transform-BeispielZuJoinMerge.ipynb**

	Name	Geschlecht	Alter	Nationalität	Gehalt	City-ID
0	Peter	M	34	deutsch	3400	B
1	Karla	W	43	schweizerisch	4000	AC
2	Anne	W	16	deutsch	0	B
3	Nino	M	22	italienisch	3000	HB
4	Andrzej	M	61	polnisch	2300	KS
5	Flurina	W	65	schweizerisch	6000	HB

	Wohnort	Stadt-ID
0	Kassel	KS
1	Berlin	B
2	Aachen	AC
3	Bremen	HB

```
person.merge(right=wohnort, how='left', left_on='City-ID', right_on='Stadt-ID')
```

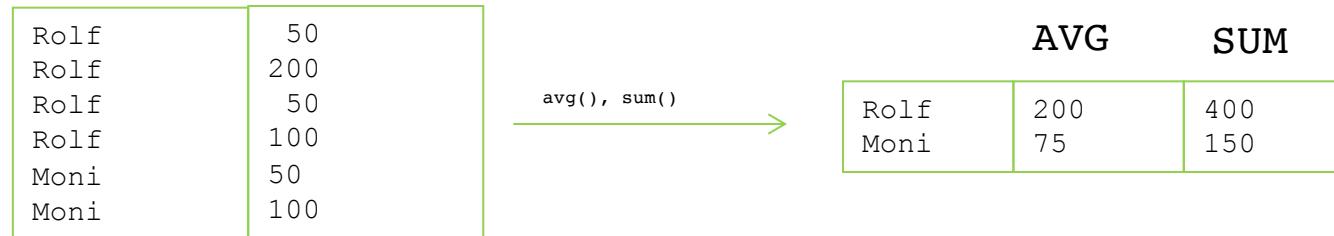
	Name	Geschlecht	Alter	Nationalität	Gehalt	City-ID	Wohnort	Stadt-ID
0	Peter	M	34	deutsch	3400	B	Berlin	B
1	Karla	W	43	schweizerisch	4000	AC	Aachen	AC
2	Anne	W	16	deutsch	0	B	Berlin	B
3	Nino	M	22	italienisch	3000	HB	Bremen	HB
4	Andrzej	M	61	polnisch	2300	KS	Kassel	KS
5	Flurina	W	65	schweizerisch	6000	HB	Bremen	HB

```
for index, row in dataframe.iterrows():
    if (int(row["Alter"]) > 50):
        dataframe.at[index, "Altersklasse"] = "alt"
    else:
        dataframe.at[index, "Altersklasse"] = "jung"
```

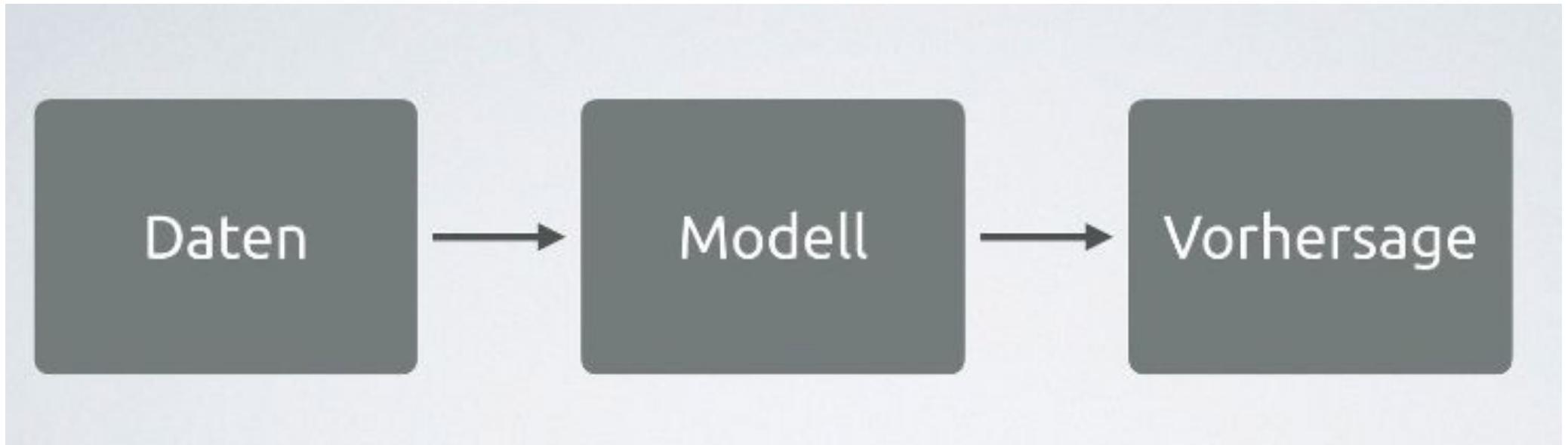
	Name	Geschlecht	Alter	Nationalität	Gehalt	City-ID	Wohnort	Stadt-ID	Altersklasse
0	Peter		34	deutsch	3400	B	Berlin	B	jung
1	Karla		43	schweizerisch	4000	AC	Aachen	AC	jung
2	Anne		16	deutsch	0	B	Berlin	B	jung
3	Nino		22	italienisch	3000	HB	Bremen	HB	jung
4	Andrzej		61	polnisch	2300	KS	Kassel	KS	alt
5	Flurina		65	schweizerisch	6000	HB	Bremen	HB	alt

- **Grouping / Aggregating**

Es muss immer eine Aggregatfunktion angegeben sein. Z.B.: avg(), sum(), max(), min(), ...



1. 1 Grundsätzliches Vorgehen im Machine Learning

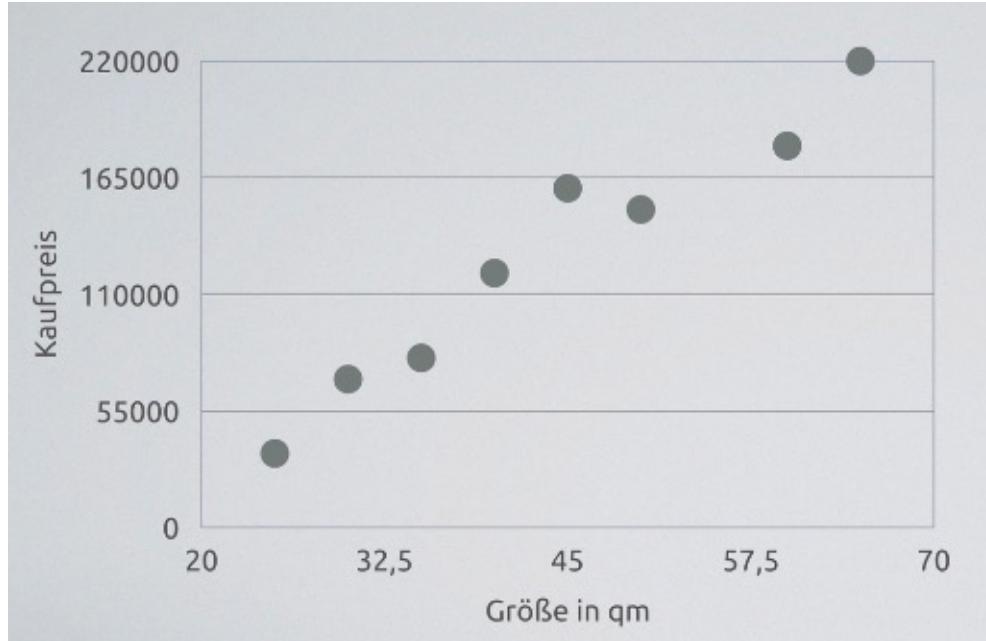


1. 2 Grundlegende Problemtypen des Machine Learning

- **Regression**
- **Klassifizierung**
- **Clustering**
- **Natural Language Processing**
- **Reinforcement Learning**

1.3 Regression

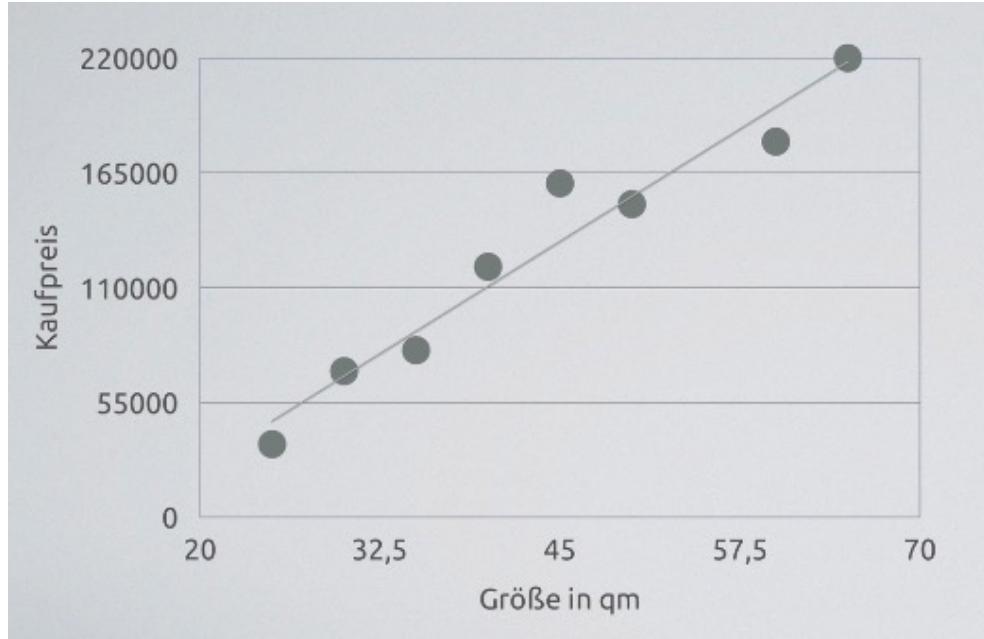
- **Gegeben:** Daten (hier Punkte), die vermutlich eine Anhängigkeit (z.B. linear) zueinander haben



- **Gesucht:** Beschreibung der Abhängigkeit mit einer möglichst gut angenäherten Formel (hier als Gerade), die z.B. für unbekannte x-Werte die y-Werte prognostizieren kann.

1.3 Regression

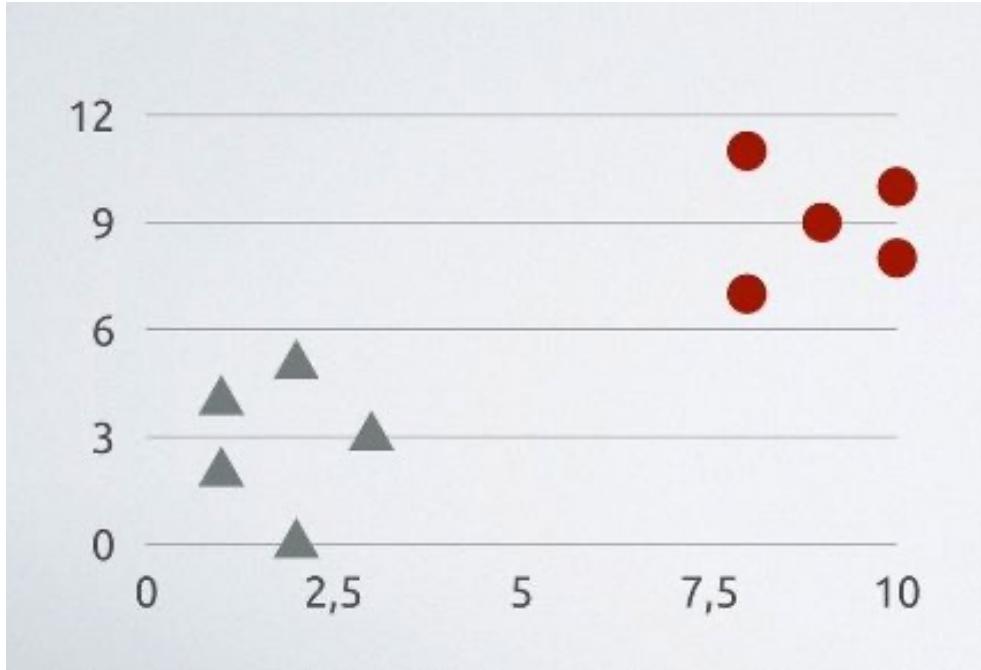
- **Gegeben:** Daten (hier Punkte), die vermutlich eine Anhängigkeit (z.B. linear) zu einander haben



- **Gesucht:** Beschreibung der Abhängigkeit mit einer möglichst gut angenäherten Formel (hier als Gerade), die z.B. für unbekannte x-Werte die y-Werte prognostizieren kann.
- Das **Modell** ist die Formel der Gerade

1.4 Klassifizierung

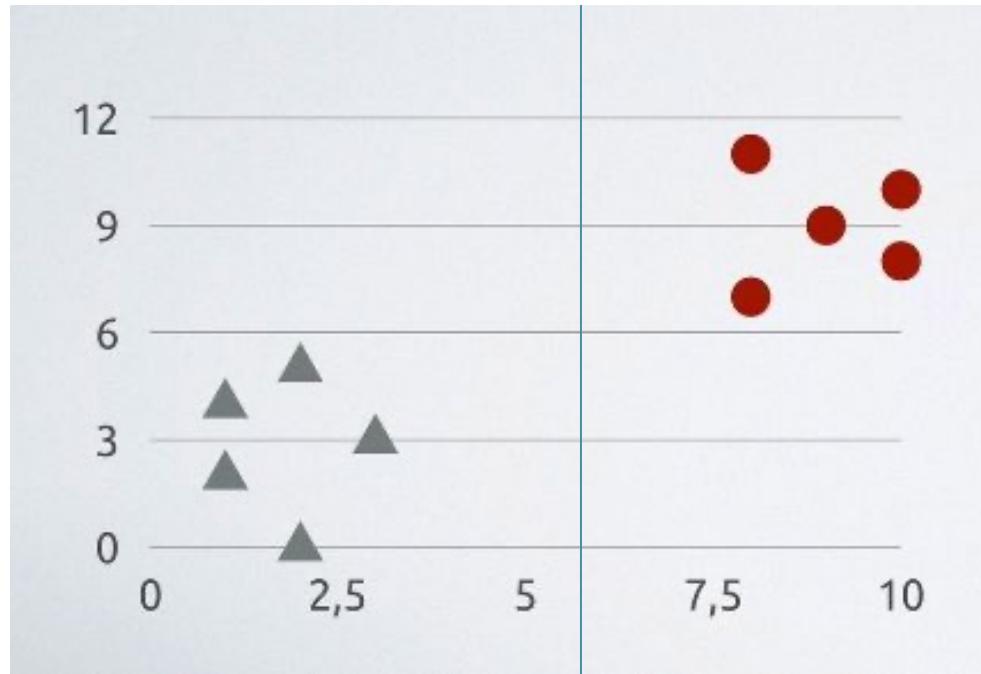
- **Gegeben:** Mehrere verschiedene (hier Dreiecke und Punkte).



- **Gesucht:** Welche Eigenschaften können zur Unterscheidung herangezogen werden (hier wäre bei $x=6$ eine Art Trennlinie). Für neue Datenpunkte könnte man das Kriterium $x < 6$ oder $x \geq 6$ nehmen, um zwischen Dreieck und Punkt zu unterscheiden.

1.4 Klassifizierung

- **Gegeben:** Mehrere verschiedene (hier Dreiecke und Punkte).



- **Gesucht:** Welche Eigenschaften können zur Unterscheidung herangezogen werden (hier wäre bei $x=6$ eine Art Trennlinie). Für neue Datenpunkte könnte man das Kriterium $x < 6$ oder $x \geq 6$ nehmen, um zwischen Dreieck und Punkt zu unterscheiden.
 - Das **Modell** ist also die **Trennlinie** bei $x=6$

1.4 Klassifizierung

- **Weitere Beispiele:**

- Kauft unser Produkt: Ja / Nein
- Brustkrebs: Ja / Nein
- Email: Spam / Nicht Spam
- Welche Zahl: 1, 2, 3, ... 9, 0



1.4 Klassifizierung

- **Beispiel:** Gegeben sind «gelabelte» Daten (rote Spalte ist das *Label* bzw. das *Learning Target* bzw. *Y*).

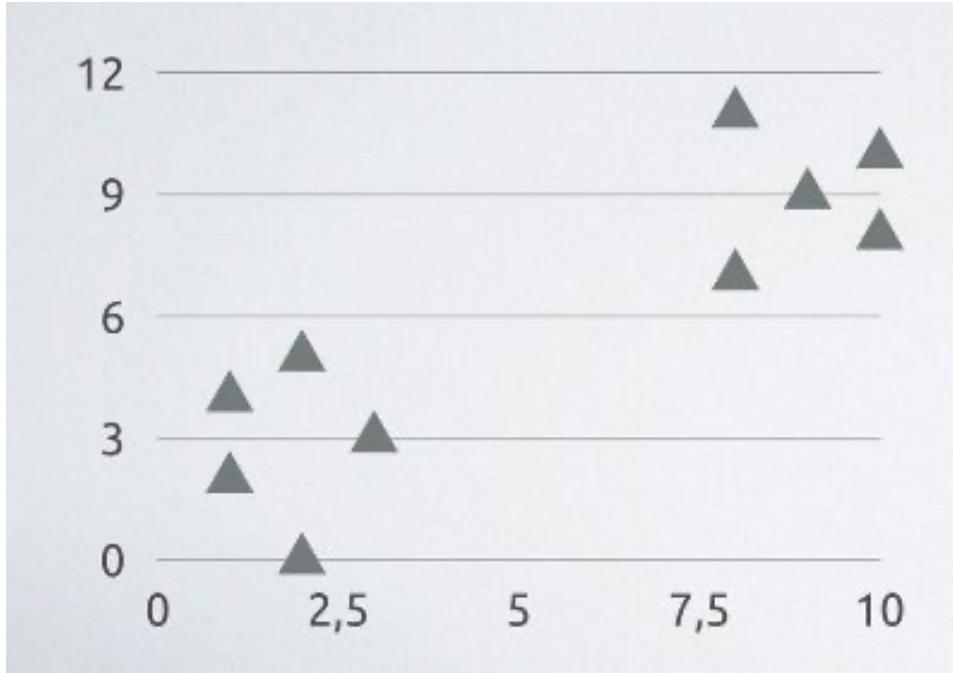
Y	X ₁	X ₂	X ₃	X ₄
Diabetes?	Müdigkeit	Gewicht	Blutdruck	Haarfarbe
Ja	Häufig	120kg	Hoch	Blond
Nein	Selten	120kg	Mittel	Blond
Nein	Gelegentlich	70kg	Niedrig	Dunkel
Ja	Gelegentlich	100kg	Hoch	Dunkel

- Wie sieht das **Modell** aus? - Ein Patient stellt sich beim Arzt vor, ist häufig müde, wiegt 130kg, hat hohen Blutdruck und ist Brünett. Diabetes???

???	Häufig	130kg	Hoch	Brünett
-----	--------	-------	------	---------

1.5 Clustering

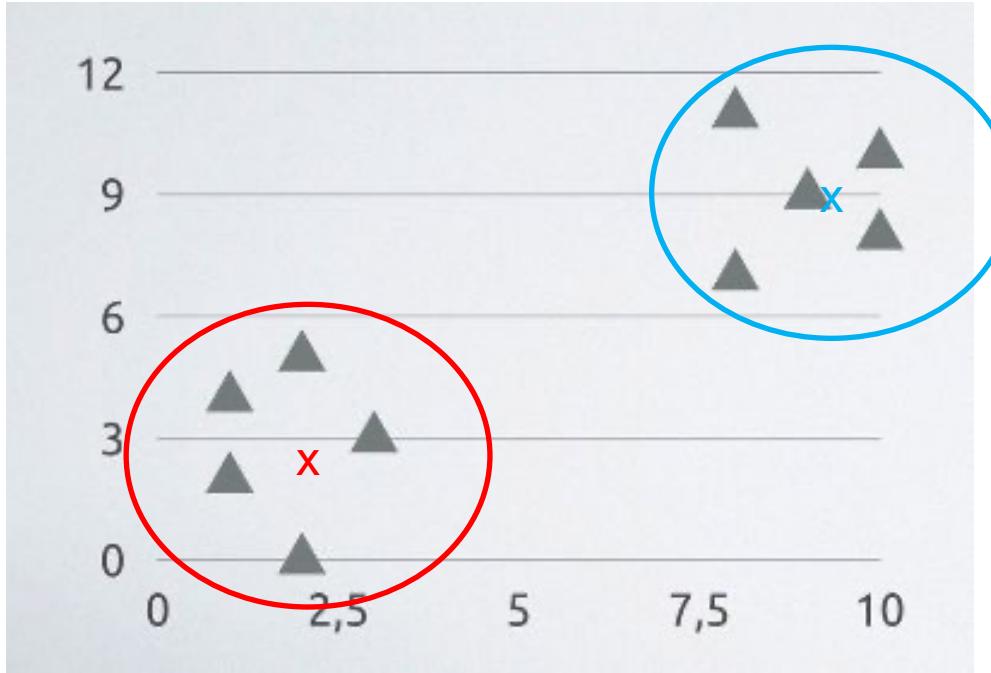
- **Gegeben:** Daten, die untereinander mehr oder weniger ähnlich sind. Es sollen z.B. zwei Gruppen gefunden werden ($k=2$)



- **Gesucht:** Wo befinden sich die beiden Gruppen?
- Das **Modell** sind die Bereiche um die Punkte A(2,4|3) und B(9|9) herum.
- Neue unbekannte Punkte (wie z.B. (1|1)) können dann dem nächstliegenden Cluster (A) zugeordnet werden.

1.5 Clustering

- **Gegeben:** Daten, die untereinander mehr oder weniger ähnlich sind. Es sollen z.B. zwei Gruppen gefunden werden ($k=2$)



- **Gesucht:** Wo befinden sich die beiden Gruppen?
- Das **Modell** sind die Bereiche um die Punkte A(2,4|3) und B(9|9) herum.
- Neue unbekannte Punkte (wie z.B. (1|1)) können dann dem nächstliegenden Cluster (A) zugeordnet werden.

1.6 Natural Language Processing

- **Allgemein:** Information aus Text extrahieren
- **Typische Aufgaben** sind:
 - E-Mail-Adressen aus einem Text extrahieren
 - Bedeutung von Adjektiven erlernen
 - Fragen zu einem Text beantworten
 - Mails nach Spam / Nicht Spam klassifizieren
 - Themen in einem Textkorpus finden (Anzahl vorgegeben)
 - Sentimentanalyse auf einem Textkorpus: Stimmungen bewerten / einordnen.

1.7 Generative versus Diskriminative Modelle

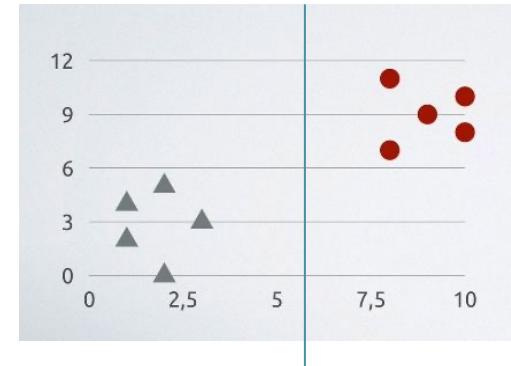
– Diskriminative Modelle:

Das Modell gibt eine Entscheidungsgrenze vor, um die Daten zu unterscheiden.

Beispiele: Entscheidungsbaum, SVM,
Logistische Regression,...

Vorteile: Anwendung einfach und braucht
wenig Rechenzeit

$X > 5.5$

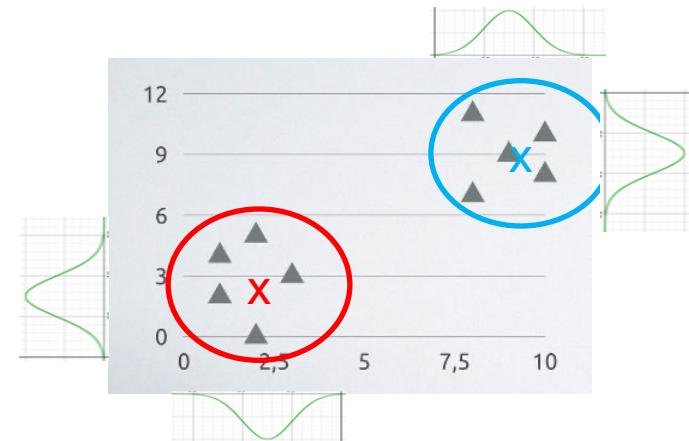


– Generative Modelle :

Das Modell versucht die Daten anhand von Werten wie Durchschnitt, Verteilungsmaß/Varianz zu beschreiben.

Beispiele: k-Means, Naive Bayes,...

Vorteile: Auf unbekannten Daten sind Ausreißer leicht erkennbar.



2. Lineare Regression

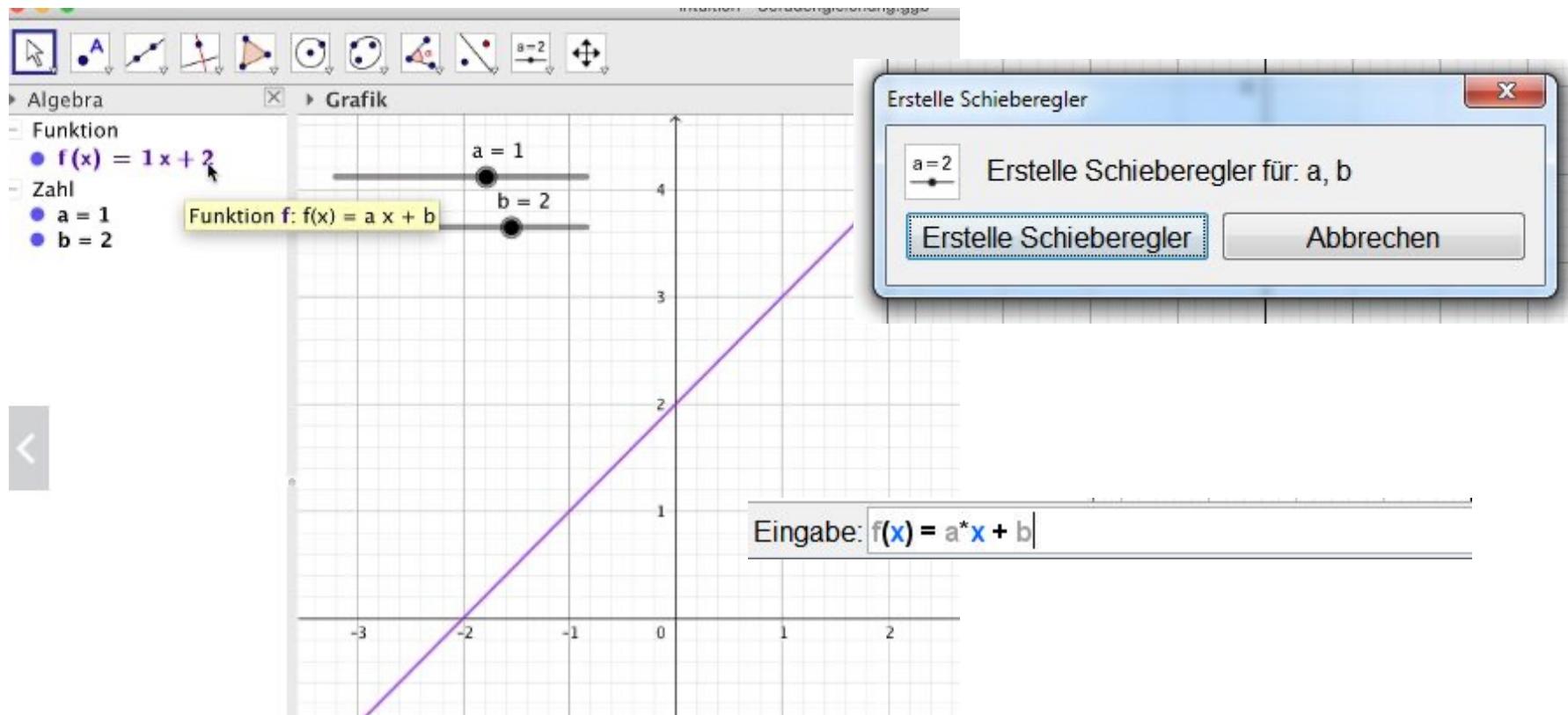


2.1 Theoretischer Hintergrund der Linearen Regression

- Grundlegende Idee der Linearen Regression:
 - Man zeichnet eine Linie durch Datenwerte
 - Mit der Linie (und ihrer Fortsetzung) kann man Werte vorhersagen

2.1 Theoretischer Hintergrund der Linearen Regression

- Zunächst: Veranschaulichung einer Geradengleichung:

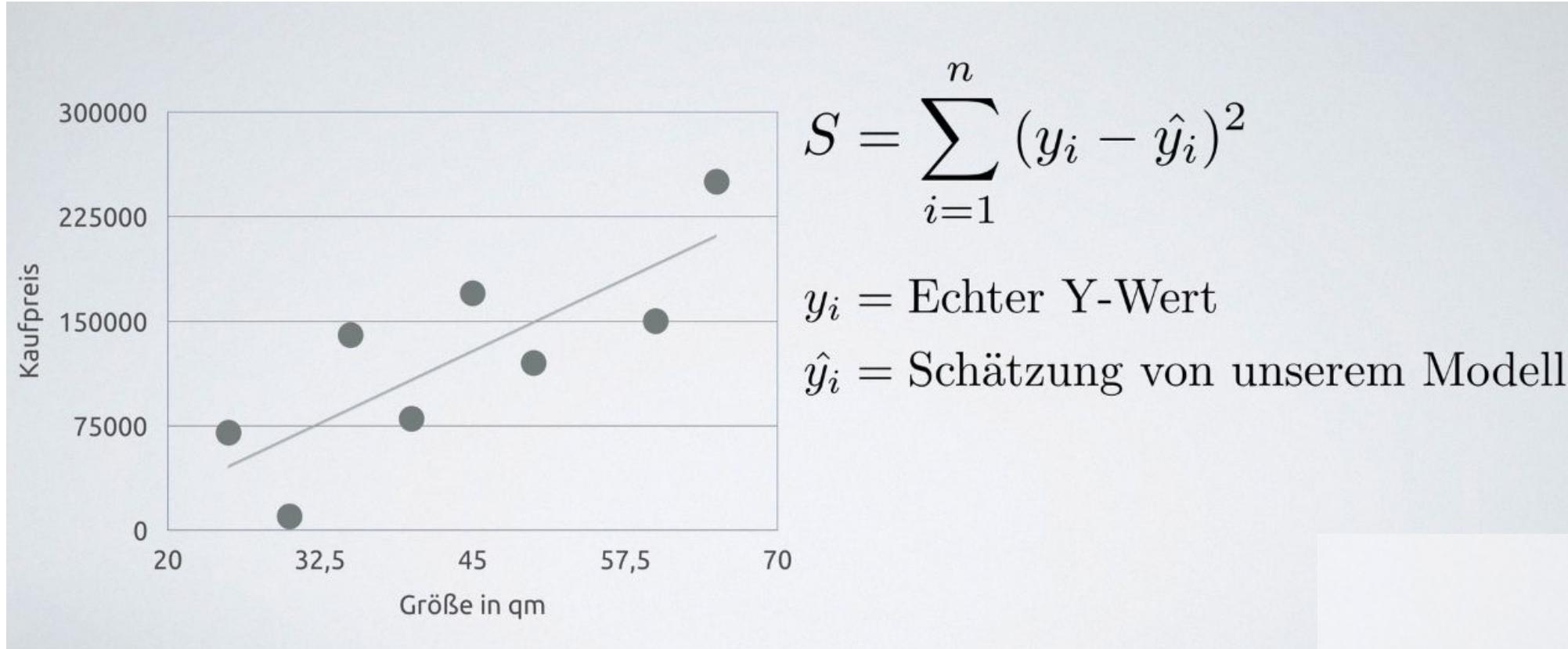


Software: GeoGebra – Files auf Moodle

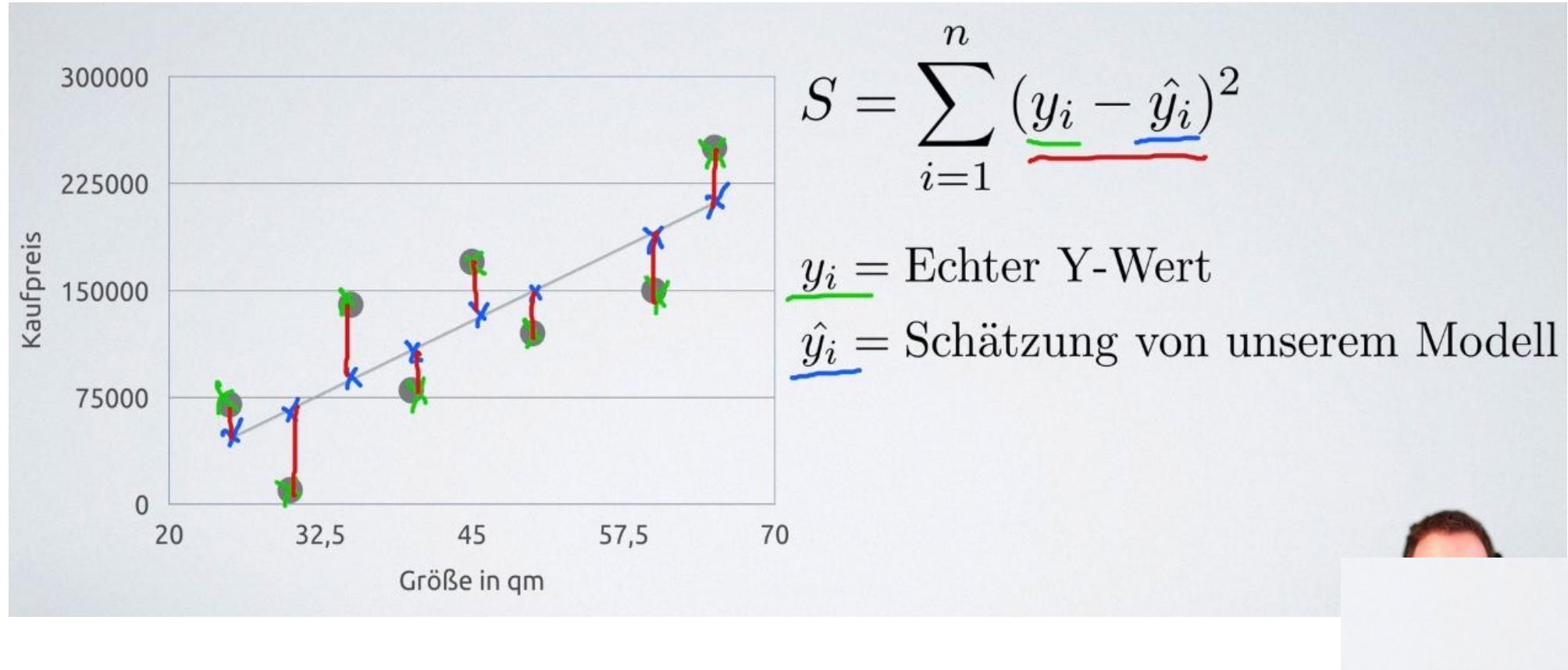
- File: [Geradengleichung.ggb](#)

2.1 Theoretischer Hintergrund der Linearen Regression

- Die Fehlerquadratsumme S:



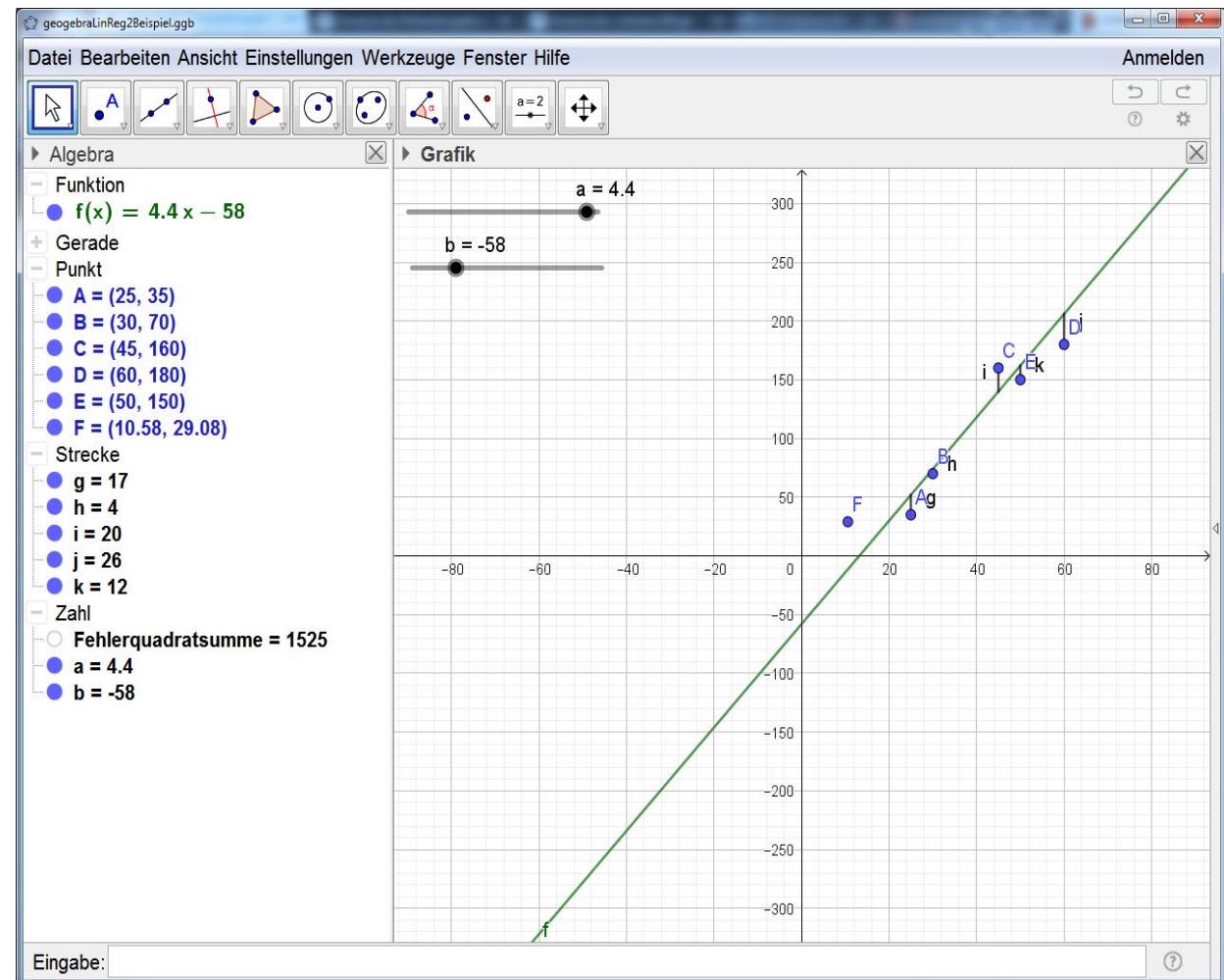
2.1 Theoretischer Hintergrund der Linearen Regression



- Die Fehlerquadratsumme S ist also die Summe aller **roten** Linien

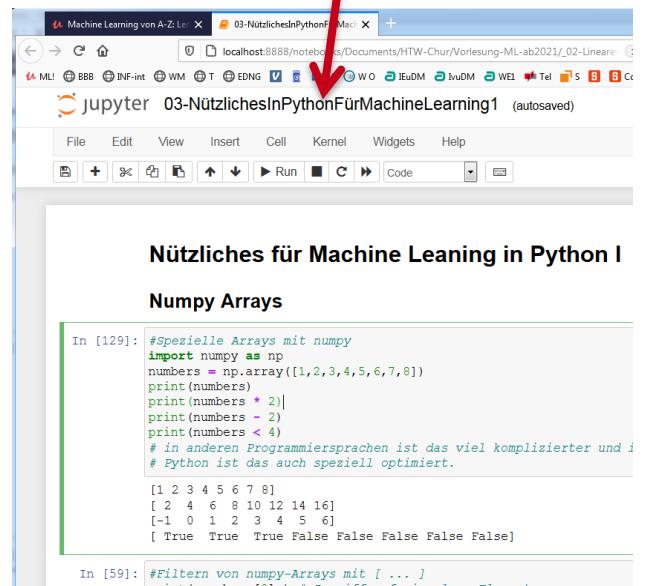
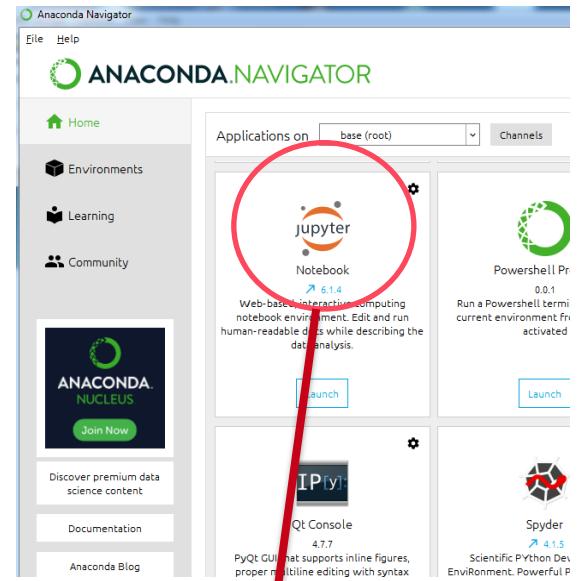
2.1 Theoretischer Hintergrund der Linearen Regression

- Gegeben sind Punkte A-F:
 - Gerade bitte so legen, dass diese **möglichst gut** durch das Punktefeld geht.
 - Welches **a** und welches **b** muss gelten, um die Fehlerquadratsumme zu **minimieren**?
 - File:
LineareRegression.ggb



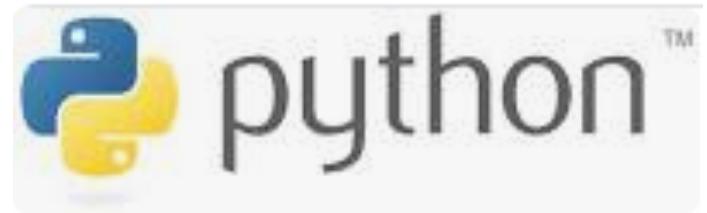
2.2 Python als Umgebung für das Machine Learning

- Wir arbeiten mit:
 - **Anaconda** – Ein Python Komplettpaket mit vielen Dingen, die normalerweise einzeln installiert werden müssten, wie z.B. APIs, Jupyter-Notebook usf.
 - **Jupyter-Notebook** – Einer flow-basierten Programmier- und Ausführungsumgebung
 - Läuft als lokaler Webserver
 - Einstieg: <http://localhost:8890>



2.2 Python als Umgebung für das Machine Learning

- Wichtige Python Zusätze die bereits installiert sind:
 - **Numpy**: Mathe
 - **Pandas**: Dataframes, also Tabellen mit Spaltennamen manipulieren/filtern
 - **Scikit-learn**: Verschiedene Klassen für verschiedene ML-Aufgaben



A screenshot of the official scikit-learn website at https://scikit-learn.org/stable/. The page features a navigation bar with links for Install, User Guide, API, Examples, and More. The main content area is titled "scikit-learn Machine Learning in Python". It highlights "Getting Started" and "Release Highlights for 0.24". Below this, there are six sections: "Classification" (with a scatter plot of digits), "Regression" (with a line plot of house prices), "Clustering" (with a scatter plot of digits grouped into four clusters), "Dimensionality reduction" (with a 3D scatter plot of digits), "Model selection" (with a line plot of cross-validation scores), and "Preprocessing" (with a grid of images showing different data transformations). Each section includes a brief description of its applications and algorithms.

2.2 Python als Umgebung für das Machine Learning

- Nachschlagen von Python-Ressourcen:
 - **Python**: <https://www.w3schools.com/python/>
 - **Google**: z.B. ein Suchbegriff wie «Python Array slice» hilft schnell weiter. Z.B.: Stackoverflow, w3schools
 - **In Jupyter-Notebook**: <Klassen/Variablenname>? zeigt häufig Einblicke in die Dokumentation

The screenshot shows a Jupyter Notebook interface. In the top cell (In [2]), the code `import numpy as np` is run, followed by `numbers = np.array([1,2,3,4,5,6,7,8])`. Below this, the user types `np?`, which is highlighted with a red circle. A red arrow points from this question mark to the resulting documentation in the bottom cell (In [2.0]).

In [2]:
import numpy as np
numbers = np.array([1,2,3,4,5,6,7,8])
np?

In [2.0]:
Type: module
String form: <module 'numpy' from 'C:\\\\ProgramData\\\\Anaconda3\\\\lib\\\\site-packages\\\\numpy__init__.py'>
File: c:\\programdata\\anaconda3\\lib\\site-packages\\numpy__init__.py
Docstring:
NumPy
=====

Provides
1. An array object of arbitrary homogeneous items
2. Fast mathematical operations over arrays
3. Linear Algebra, Fourier Transforms, Random Number Generators

How to use the documentation

2.2 Python als Umgebung für das Machine Learning

- Start der Demos:
 1. Anaconda starten
 2. Jupyter Notebook aufrufen.
 3. Datei suchen und anklicken.
- File auf Moodle: **NützlichesInPythonFürMachineLearning.ipynp**
 - Numpy-Arrays, Matrizen, Dataframes, Plotten in Diagrammen

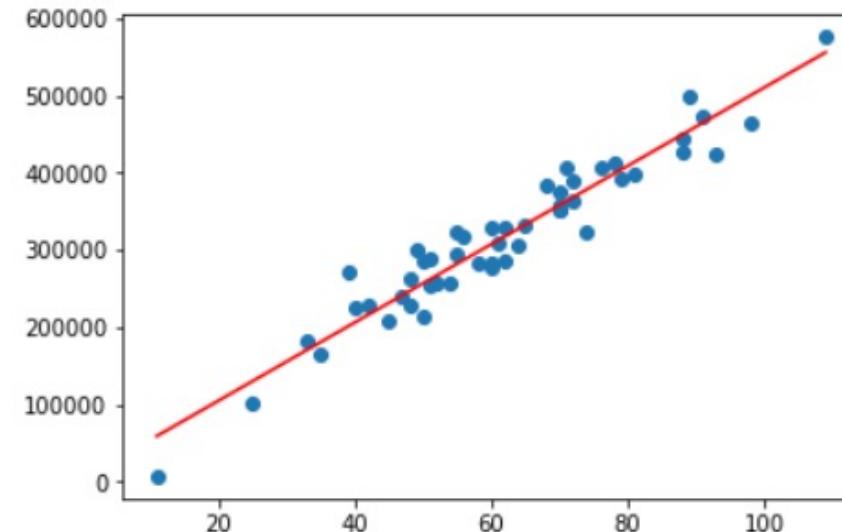
2.3 Lineare Regression in Python

[Übung anschauen](#)

- Gegebene Beispieldaten:
 - File auf Moodle: **wohnungspreise.csv**
 - Frage: Gibt es einen Zusammenhang zwischen den Quadratmetern und dem Verkaufspreis einer Wohnung?
 - Ziel: Das **a** und das **b** der Formel der **roten Linie** bestimmen.

	Quadratmeter	Verkaufspreis
0	70	351000
1	72	390000
2	91	473000
3	58	282000
4	49	300000

⋮



2.3 Lineare Regression in Python

- Demonstration der Lösung einer solchen Aufgabe:
 - File auf Moodle: **LineareRegression.ipynb**
 - Benötigte Komponente: https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares
 - Gefundene Lösung – Das Modell:
Verkaufspreis = 3143.28481869 + 5071.35242619 * Quadratmeter
 - Wenn meine Wohnung **61qm** hat, dann würde sie kosten:
312.495,78281643

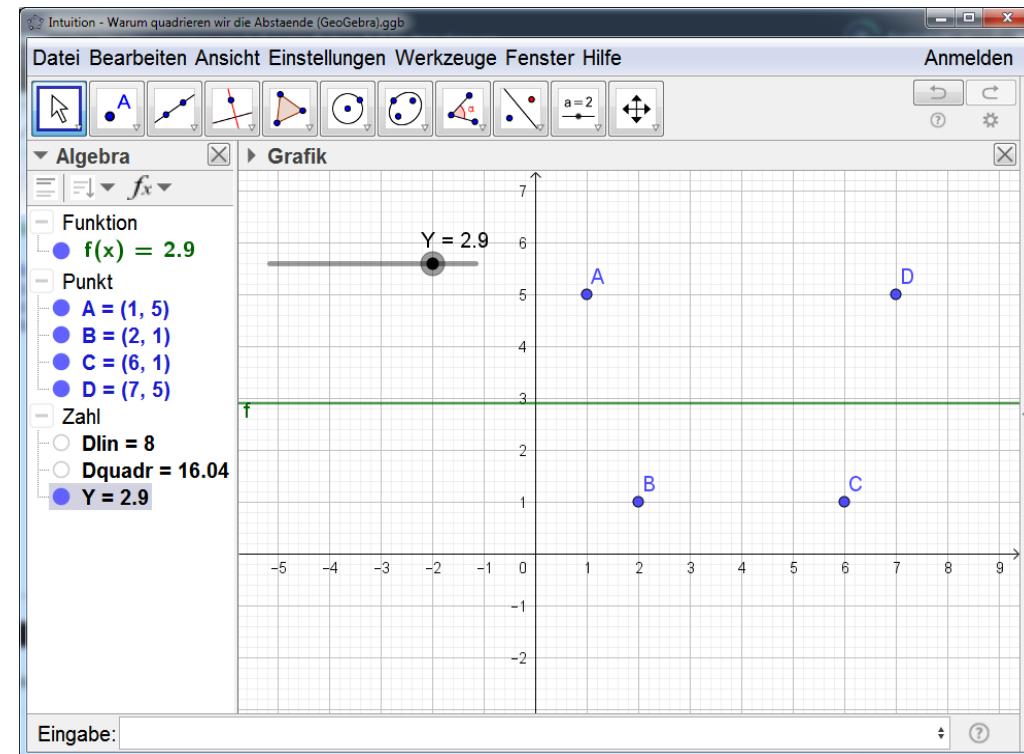
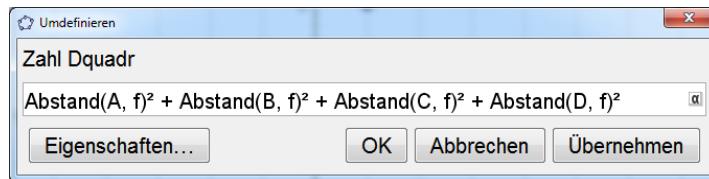
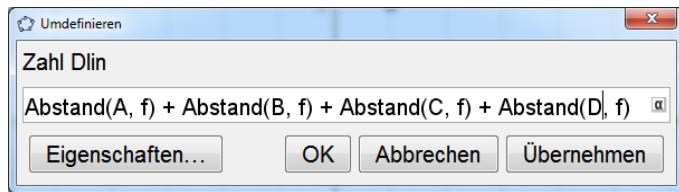
2.4 Warum quadriert man Fehler?

- **Fehlersumme S:**
$$S = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$
- Ziel: $\min(S)$
- Die Ableitung von S muss dabei 0 sein: $S' = 0$
- Die Ableitung enthält dann keine Quadrate mehr und diese kann dann mit einem Quadratischen Gleichungssystem gelöst werden.

2.4 Warum quadriert man Fehler?

– Intuitive Veranschaulichung

- GeoGebra-Datei: **QuadrierenVonAbstaenden.ggb**
- Y: **Grüne Messlinie**
- Dlin: zwischen den Punkten immer gleich!
- Dquadr: Minimum in der Mitte!



2.5 Praxisprojekt

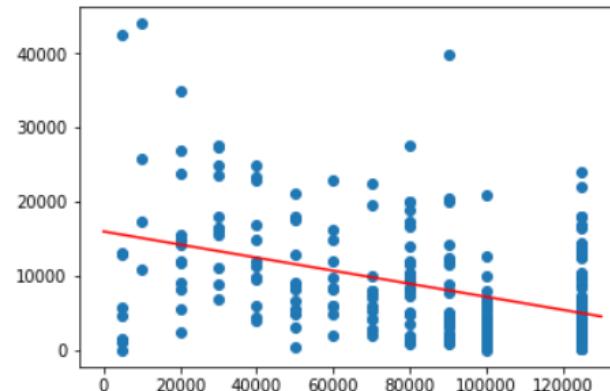
- **Öffentliche Datensätze:** <https://www.kaggle.com/datasets>
- **Beispiel:** Gebrauchtwagenpreise aus eBay-Kleinanzeigen
- **File:** autos_prepared.csv
- **Aufgabe:** Wie hängen Laufleistung und Preis zusammen?
 - **Python-Demo:**

PraxisprojektLineareRegression.ipynb

	price	yearOfRegistration	powerPS	kilometer	model	fuelType	name
0	1450	1997	75	90000	andere	benzin	Toyota_Toyota_Starlet_1_Hand__TÜV_neu
1	13100	2005	280	5000	golf	benzin	R32_tauschen_oder_kaufuen
2	4500	2008	87	90000	yaris	benzin	Toyota_Yaris_1.3_VVT_i
3	6000	2009	177	125000	3er	diesel	320_Alpinweiss_Kohlenstoff
4	3990	1999	118	90000	3er	benzin	BMW_318i_E46_+++_1_Hand_++_Liehaberfahrzeug



?

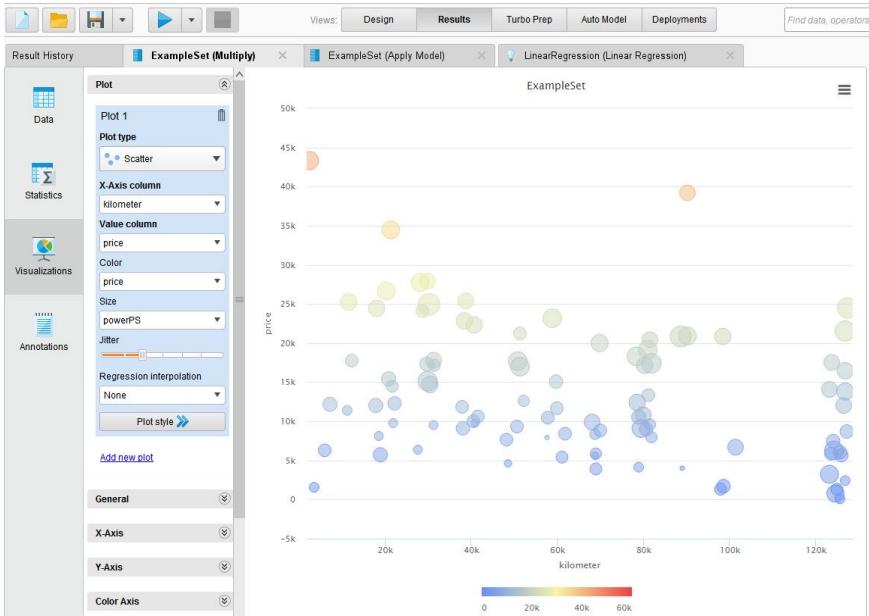


The screenshot shows the Kaggle homepage with sections for Datasets, Trending Datasets, Trending Tasks, Popular Datasets, and a sidebar for Data Scientists.

- Datasets:** Shows a search bar and filters for Datasets, Tasks, Computer Science, Education, Classification, Computer Vision, NLP, and Data Visualization. It lists trending datasets like Netflix Movies and TV Shows, COVID-19 World Vaccination Progress, HR Analytics: Job Change of Data Scientists, and All Trump's Twitter insults (2015-2021).
- Trending Datasets:** Displays cards for Netflix Movies and TV Shows, COVID-19 World Vaccination Progress, HR Analytics: Job Change of Data Scientists, and All Trump's Twitter insults (2015-2021).
- Trending Tasks:** Displays cards for Are the tweets are really effective?, Better prediction for diabetes, Topic Modelling of ML Papers using LDA (Latent Dirichlet..., and All NeurIPS (NIPS) Papers.
- Popular Datasets:** Displays cards for Reddit WallStreetBets Posts, Rain in Australia, Melbourne Housing Snapshot, and Gufhtugu Publications Dataset Challenge.
- Data Scientists:** Shows a sidebar with sections for What are Tasks?, Popular Datasets, and a list of pending work and challenges.

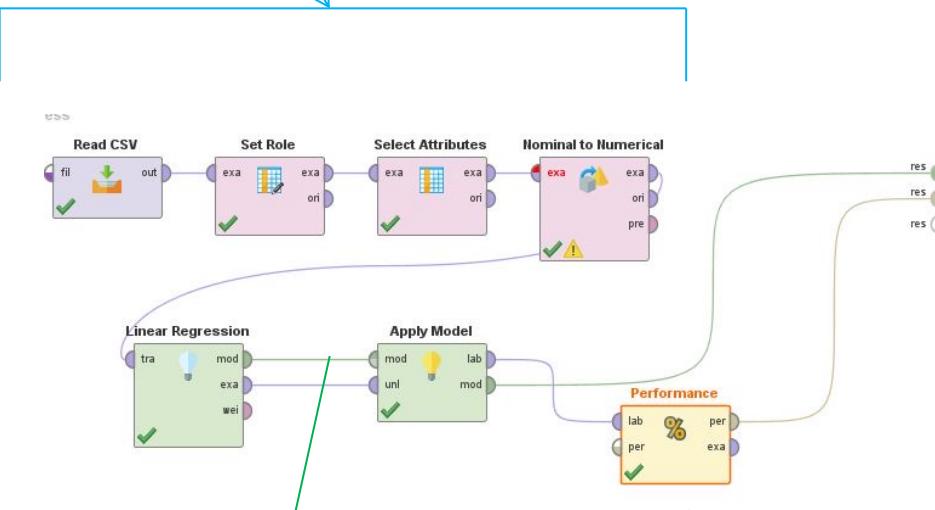
2.5 Praxisprojekt

– Anderes Tool: RapidMiner



Plots interaktiv erstellen

Vorverarbeitung



Modellbildung

LinearRegression

- 0.088 * kilometer
- + 15988.727

Performance-
Messung

root_mean_squared_error

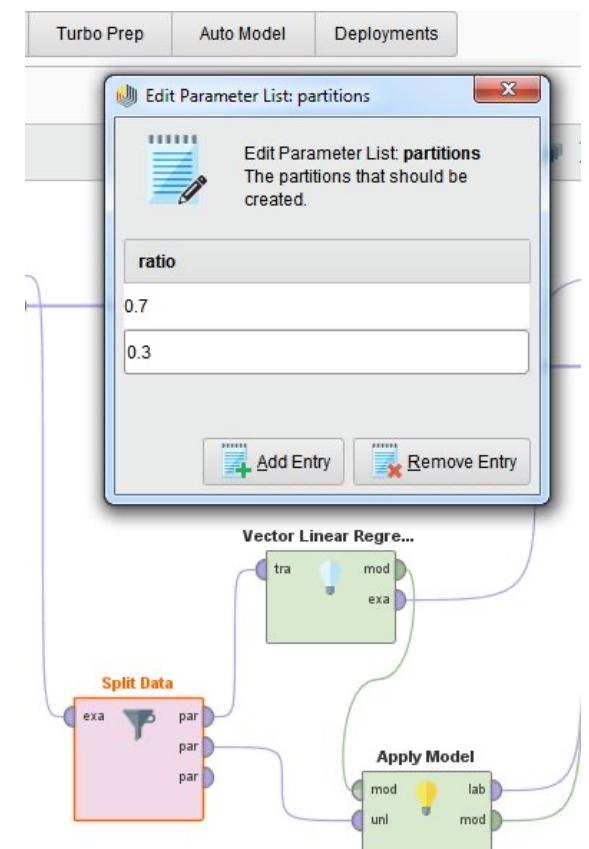
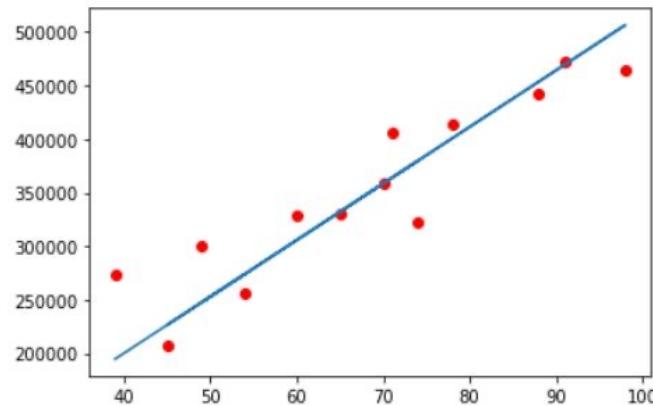
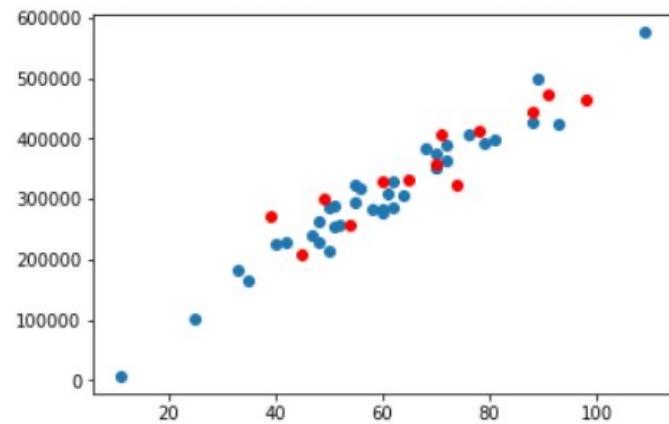
root_mean_squared_error: 7057.530 +/- 0.000

2.6 Training and Testing (Train Test)

- **Idee:** Aufteilen der Daten
- **Beispiel:**
 - $\frac{3}{4}$ der Daten zum **Trainieren**
 - $\frac{1}{4}$ der Daten zum **Testen**
- **Daten müssen gleichmäßig aufgeteilt werden**
 - **Beispiel:** 10, 11, 12, 80, 81
 - **Schlecht:**
 - **Train:** 10, 12, 11
 - **Test:** 80, 81
 - **Zufällig ist hier in der Regel am Besten!**

2.6 Training and Testing (Train Test)

- Umsetzung in Rapid Miner mit «Split Data»:
- Schematische Veranschaulichung:
 - Blau: Training, Rot: Testing
 - Werden die Testdaten gut durch das Modell (blaue Linie) dargestellt?

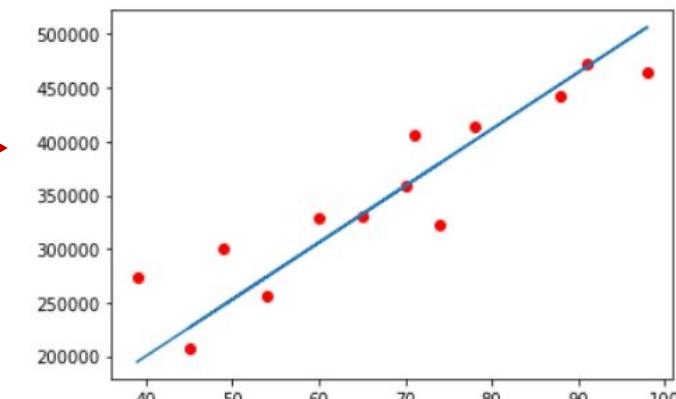
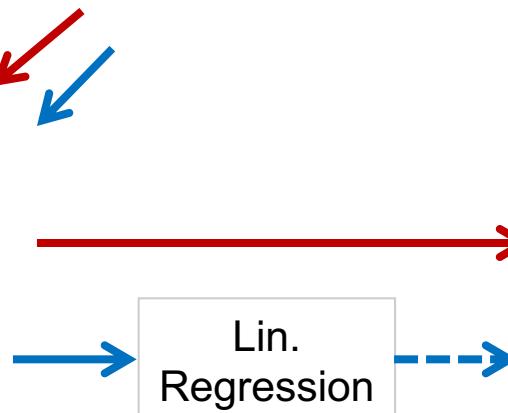
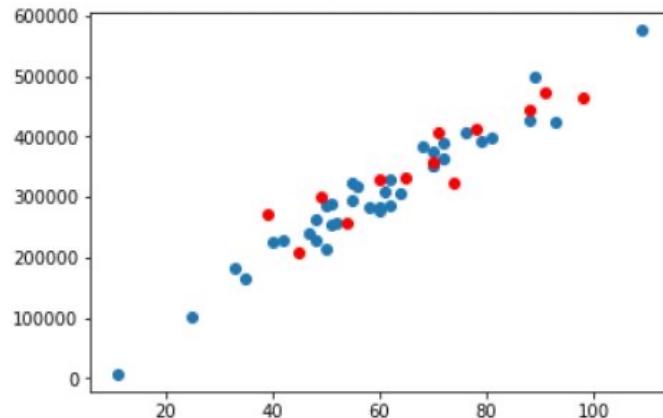


2.6 Training and Testing (Train Test)

- **Beispiel:** Immobilienkaufpreise
- **File:** [wohnungspreise.csv](#)
- **Aufgabe:** Wie hängen Quadratmeter und Preis zusammen?
 - **Python-Demo:** [Train-Test.ipynb](#)

	Quadratmeter	Verkaufspreis
0	70	351000
1	72	390000
2	91	473000
3	58	282000
4	49	300000

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state = 0, test_size = 0.25)
```

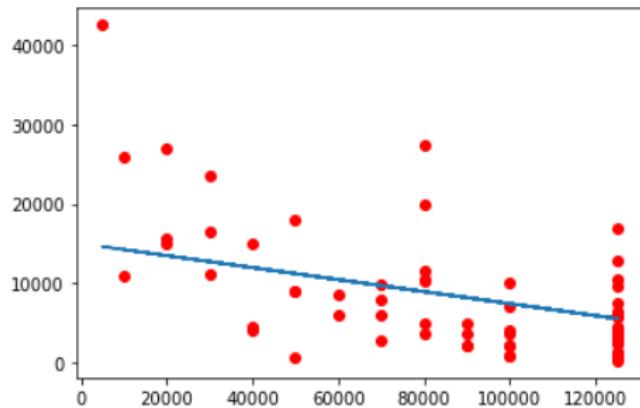
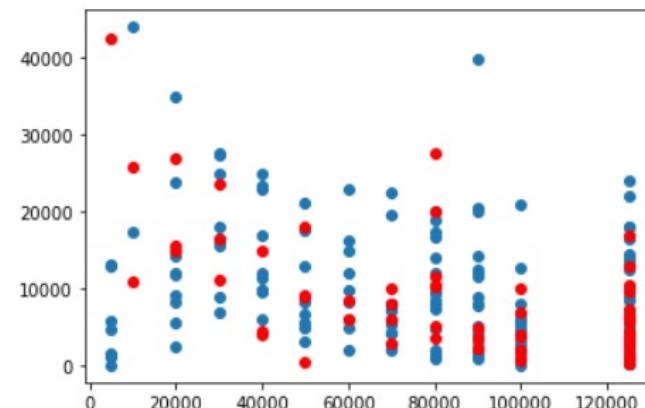


2.6 Training and Testing (Train Test)

- **Beispiel:** Gebrauchtwagenpreise
- **File:** [autos_prepared.csv](#)
- **Aufgabe:** Machen Sie ein Train Test Split und erstellen Sie ein Modell auf Basis der Train-Daten. Machen Sie einen Plot, der das Modell in den Testdaten zeigt.
 - **Python-Musterlösung:**

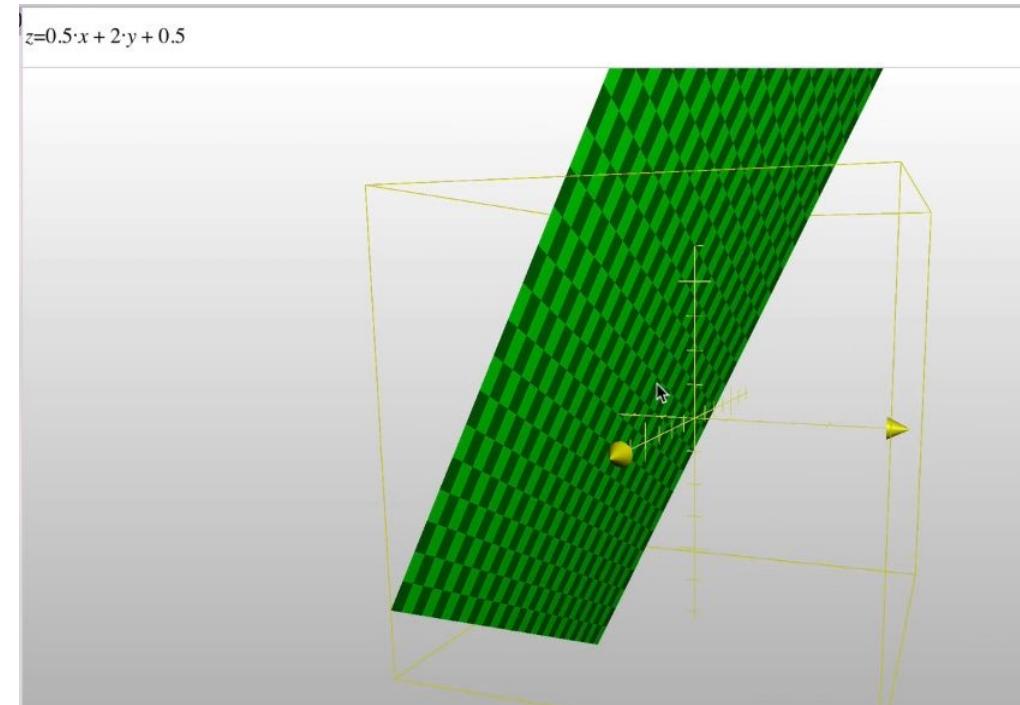
[Train-Test-Aufgabe-Autopreise.ipynb](#)

	price	yearOfRegistration	powerPS	kilometer	model	fuelType	name
0	1450	1997	75	90000	andere	benzin	Toyota_Toyota_Starlet_1_Hand__TÜV_neu
1	13100	2005	280	5000	golf	benzin	R32_tauschen_oder_kaufen
2	4500	2008	87	90000	yaris	benzin	Toyota_Yaris_1.3_VVT_i
3	6000	2009	177	125000	3er	diesel	320_Alpinweiss_Kohlenstoff
4	3990	1999	118	90000	3er	benzin	BMW_318i_E46_+++_1_Hand_+++_Liebhaberfahrzeug



2.7 Regression mit mehreren Variablen

- **Beispiel:** Verkaufspreise von Hotels schätzen
- Dazu nutzen wir die Attribute:
 - Anzahl Zimmer
 - Gewinn im letzten Jahr
 - (Stadt)
- **Veranschaulichung:**
 - Dies ist wie der Übergang von der zweidimensionalen Ebene in den dreidimensionalen Raum.
 - Wir suchen nicht nach einer Geraden, sondern z.B. nach einer Fläche.



2.7 Regression mit mehreren Variablen

- Entwurf einer **Ebenengleichung**:

- $Z = AX + BY + C$
- [Verkaufspreis] =
 $A * [\text{Zimmer}] + B * [\text{Gewinn}] + C$

- **Verallgemeinerung** mit vielen Dimensionen:

$$y_t = x_{t1}\beta_1 + x_{t2}\beta_2 + \cdots + x_{tK}\beta_K + \varepsilon_t$$

- **Formale Voraussetzungen:**

- Daten sind unkorreliert (voneinander unabhängig)
- Satz von Gauß-(Markov)

Der Kleinsten-Quadrate-Schätzer ist die beste
lineare erwartungstreue Schätzfunktion, wenn die
zufälligen Störgrößen einen Erwartungswert von
null haben.

2.7 Regression mit mehreren Variablen

- Ungutes Beispiel:

	Anzahl Zimmer	Anzahl Badezimmer	Preis
Hotel 1	100	100	100.000
Hotel 2	150	150	150.000
Hotel 3	300	300	300.000
Hotel 4	175	175	175.000



Übereinstimmung ist die schlimmste Form der Korrelation!

2.7 Regression mit mehreren Variablen

- Immer noch Redundanzen:

	Normale Zimmer	Suites	Zimmer Gesamt	Preis
Hotel I	75	25	100	100.000
Hotel 2	150	0	150	150.000
Hotel 3	200	100	300	300.000
Hotel 4	125	50	175	175.000

$$\begin{matrix} \swarrow & \nearrow \\ + & \longrightarrow & = \end{matrix}$$

2.7 Regression mit mehreren Variablen

- **Beispiel:** Hotelpreise aus mehreren Variablen ermitteln
- **File:** `hotels.csv`
- **Aufgabe:** Wie hängen Gewinn, Quadratmeter mit dem Preis zusammen?
 - **Python-Demo:** `LineareRegressionMitMehrerenVariablen.ipynb`

	Gewinn	Preis in Mio	Quadratmeter	Stadt
0	119000.0	21.88	3938.0	Berlin
1	250000.0	27.95	3986.0	München
2	250000.0	16.09	2574.0	Köln
3	145000.0	27.58	4155.0	München
4	110000.0	23.76	3795.0	Berlin



$$\begin{aligned} [\text{Preis in Mio}] &= 6.48370247 + \\ &[\text{Gewinn}] * 6.39855984 \times 10^{-6} + \\ &[\text{Quadratmeter}] * 3.89642288 \times 10^{-3} \end{aligned}$$

(Grafisch nicht mehr so einfach darstellbar)

2.8 Das Bestimmtheitsmaß R^2

R^2

- **Problem:**
 - Wir benötigen etwas, um die Qualität eines Modells messen zu können
 - Einfach eine Grafik zeichnen, um das «anzuschauen» geht bei mehreren Variablen nicht mehr.
- **Lösung:** R^2

2.8 Das Bestimmtheitsmaß R^2

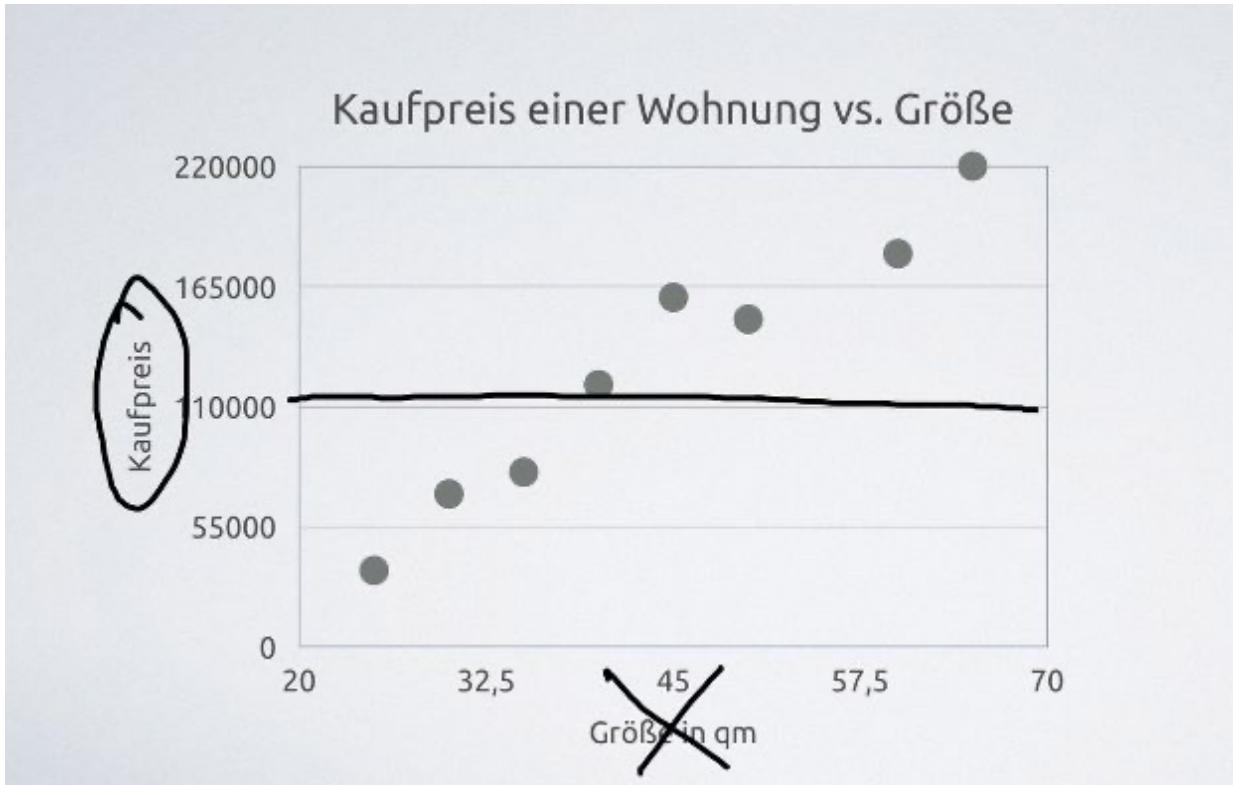
R^2

- Idee:
 - Wir berechnen den Fehler für ein «richtig schlechtes Modell»
 - R^2 : Wie gut ist «unser Modell» besser im Vergleich zum «richtig schlechten Modell»?
 - «Richtig schlechtes Modell»
 - Ein Modell, das unsere Eingabedaten gar nicht betrachtet, sondern:
 - Wir schauen nur auf den Durchschnitt von den Daten, die wir gesehen haben

2.8 Das Bestimmtheitsmaß R^2

R^2

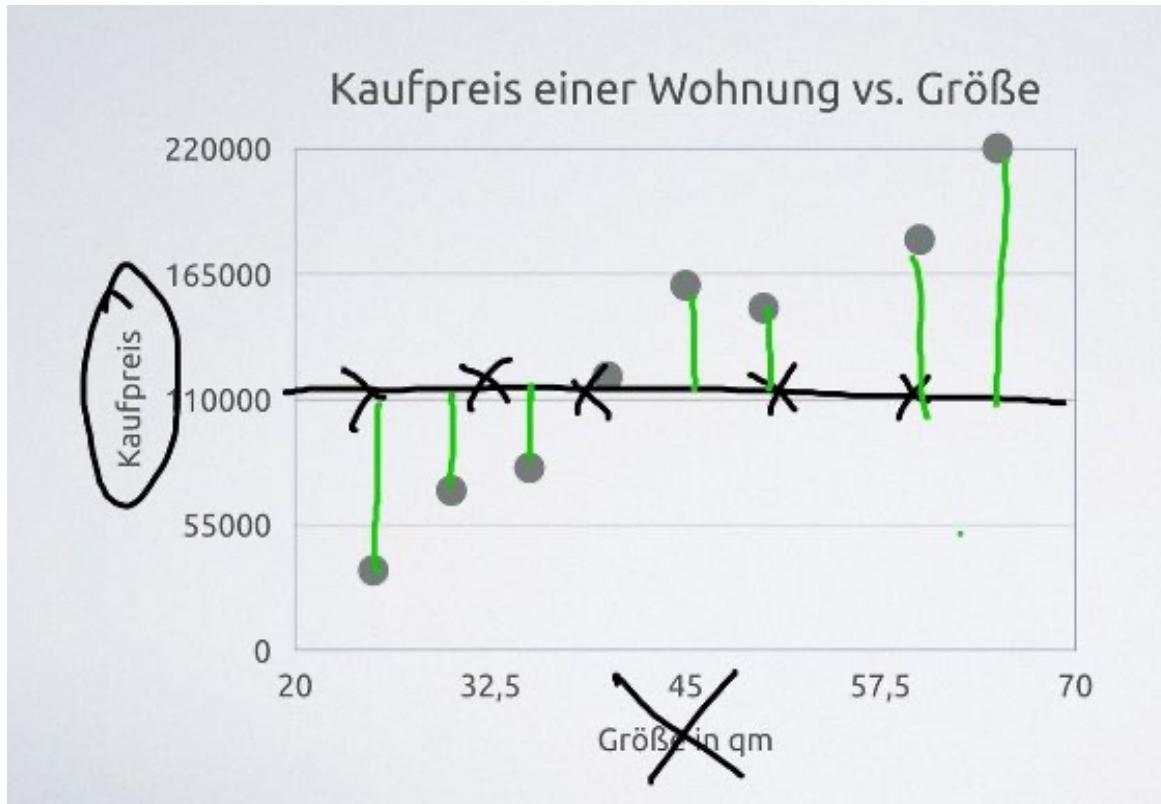
- Der Durchschnitt als «richtig schlechtes Modell»:



2.8 Das Bestimmtheitsmaß R^2

R^2

- Die Fehlerquadrate geben an, wie schlecht das «richtig schlechte Modell» konkret ist:



2.8 Das Bestimmtheitsmaß R^2

- Die **Formel**

R^2

$$R^2 := 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

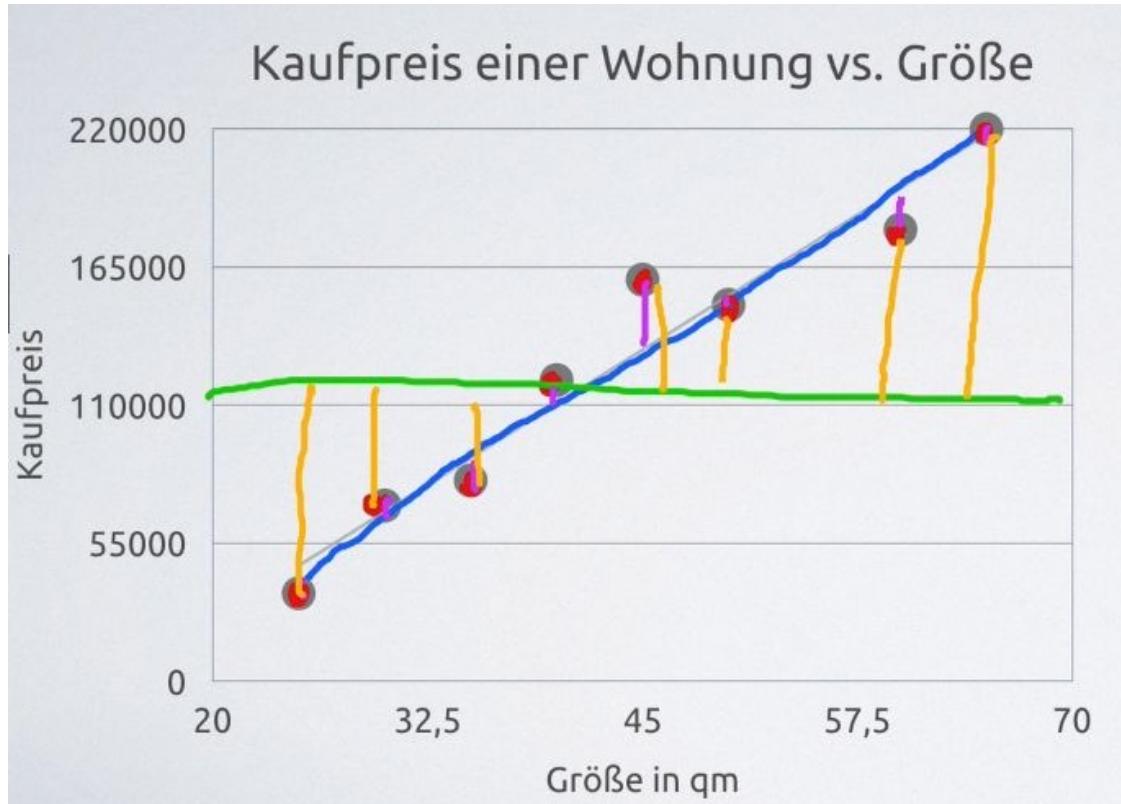
i
 y_i
 \hat{y}_i
 \bar{y}

Anz. Datenpunkte
Wert eines Punkts
Schätzwert per Modell
Durchschnittswert

2.8 Das Bestimmtheitsmaß R^2

R^2

- **Erklärung der Formel:**



$$R^2 := 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

$R = 0.85$ ist gut
 $R = 0.95$ sehr gut

2.8 Das Bestimmtheitsmaß R^2

R^2

- Demonstration: R^2 mit Python:
 - File: R2mitSpielzahlen.ipynb

```
: #r2 = r2_score(<testdata yi>, <prediction y^>)
# Siehe Tafelanschrieb
r2 = r2_score([1,1,3,5], [1,2,3,4])
print(r2)
```

0.8181818181818181

-> Tafelanschrieb hierzu

2.8 Das Bestimmtheitsmaß R^2

R^2

- **Beispiel:** Hotelpreise aus mehreren Variablen ermitteln – Evaluation des Modells
- **File:** `hotels.csv`
- **Vorgehen:** Wir nehmen die Datei `Train-Test.ipynb` und bestimmen R^2 aus den Testdaten, und dem gefundenen Modell.
 - **Python-Demo:** `R2-ScoreAusrechnen.ipynb`
 - Es gibt verschiedene Arten, wie man den R^2 -Score ermittelt:

Gewinn oder Quadratmeter mehrmals laufen lassen, ungefähren Wert ermitteln

Quadratmeter bei 0.88, Gewinn bei 0.3 --> Quadratmeter deutlich besser

beides zusammen 0.81

```
y_test_pred = model.predict(X_test)
```

```
from sklearn.metrics import r2_score  
  
r2 = r2_score(y_test, y_test_pred)  
print(r2)
```

```
0.8783249527580934
```

```
from sklearn.linear_model import LinearRegression  
  
model = LinearRegression()  
model.fit(X_train, y_train)  
# model kann auch R2 ausrechnen ....  
print(model.score(X_test, y_test))  
#(sklearn.metrics import r2_score ist also nicht nötig)
```

```
0.8783249527580934
```

2.9 Welche Spalten machen Sinn? – R² nutzbringend einsetzen

- **Übliches Verfahren :**
 1. Mehrere verschiedene Splits und Modellbildung nacheinander durchführen (For-Schleife)
 2. Als Güteabschätzung wird der Durchschnitt aller R2-Werte genommen.
- Mit diesem verbesserten **Güteabschätzverfahren** kann man zeigen, welche Spalten oder Spaltenkombination einen guten Beitrag leisten, bzw. welche unnötig oder sogar eher kontraproduktiv sind.

2.9 Welche Spalten machen Sinn? – R² nutzbringend einsetzen

- **Beispiel:** Hotelverkaufspreise – Auswahl verschiedener Spalten
- **File:** `hotels.csv`
- **Vorgehen:** Wir nehmen die Datei `Train-Test-Aufgabe-Autopreise.ipynb` als Grundlage und evaluieren mit R² unterschiedliche Spalten bzw. Spaltenkombinationen.
 - **Python-Demo:** `WelcheSpaltenMachenSinn.ipynb`
 - Der Python Zufallsgenerator ist entweder **reproduzierbar zufällig**
 - `x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 0, test_size = 0.25)`
 - Oder er ist tatsächlich zufällig, also auch **nicht reproduzierbar zufällig**
 - `x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25)`
 - Lässt man R² 1000 mal ermitteln, wird R² **realistischer**:

```
for i in range(0, 1000):  
    ...  
    scores.append(model.score(X_test, y_test))  
print(sum(scores) / len(scores))
```

→ Vergleich
... →

2.9 Welche Spalten machen Sinn? – R² nutzbringend einsetzen

- **Aufgabe:** Gebrauchtwagenpreise – Auswahl verschiedener Spalten
- **File:** [autos_prepared.csv](#)
- **Frage:** Dass Kilometer den Preis vorhersagen, haben wir ja schon gesehen. 1) Wie gut ist dieses Modell 2) Gibt es im Vergleich dazu bessere Kombinationen, bei denen Kilometer und noch ein weiteres Attribut den Preis besser als 1. vorhersagen können.
 - **Ausgangsbasis:** [PraxisprojektR2-ScoreBerechnenAusgangsbasis.ipynb](#)
 - **Lösung:** 1) sollte ca. 0,29 ergeben, bei 2) sollte man auf 0,62 kommen.
 - **Fazit:** Es lohnt sich also mit mehr als einer Spalte zu trainieren, aber 0,62 ist immer noch ein imgrunde schlechtes Modell.
- Beispiellösung: [PraxisprojektR2-ScoreBerechnenBeispiellösung.ipynb](#)

3. Datenarten

– Metrisch



– Ordinal



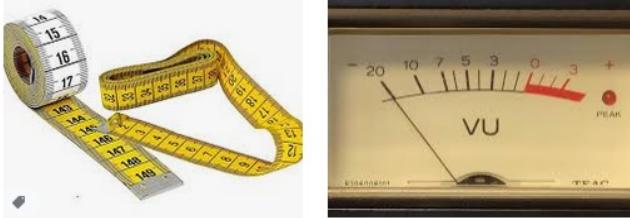
– Nominal



3.1 Grundlegende Einteilung der Datenarten

- Grundlegende Datenarten:

- Metrisch



- Ordinal



- Nominal



3.1 Grundlegende Einteilung der Datenarten

- **Metrisch:**
 - **Zahlen**
 - Die Abstände haben eine «direkte» Bedeutung
 - Meist stetig (es gibt Kommawerte dazwischen)
 - **Beispiel: Distanz**
 - $2\text{km} + 1\text{km} = 3\text{km}$
 - $4\text{km} + 1\text{km} = 5\text{km}$
 - **Beispiel: Geld**
 - Die Hälfte von 100\$ ist 50\$

3.1 Grundlegende Einteilung der Datenarten

- **Ordinal:**
 - Es gibt eine «**Rangordnung**»
 - Die **Abstände** zwischen den einzelnen Schritten dieser Ordnung haben eine unbestimmte / **unterschiedliche** Größe
 - **Beispiel: Alterseinteilung von Menschen**
 - Ist Säugling < ist Kind < ist jugendlich < ist erwachsen < ist alt
 - **Beispiel: Sportergebnis**
 - 1. Platz < 2. Platz < 3. Platz



Die Altersabstände sind ungleich!

Wie viel mehr Wert hat der 1. Platz als der 2. Platz oder der 3. Platz???

3.1 Grundlegende Einteilung der Datenarten

- **Nominal:**
 - Es gibt «**keine Rangordnung**»
 - Wird verwendet, um nicht numerische Eigenschaften zu beschreiben, die zutreffen oder nicht zutreffen.
- **Beispiele:**
 - Welches Betriebssystem benutzt Du?
 - Geschlecht
 - Studienfach

3.2 Verwendung von Datenarten im Machine-Learning

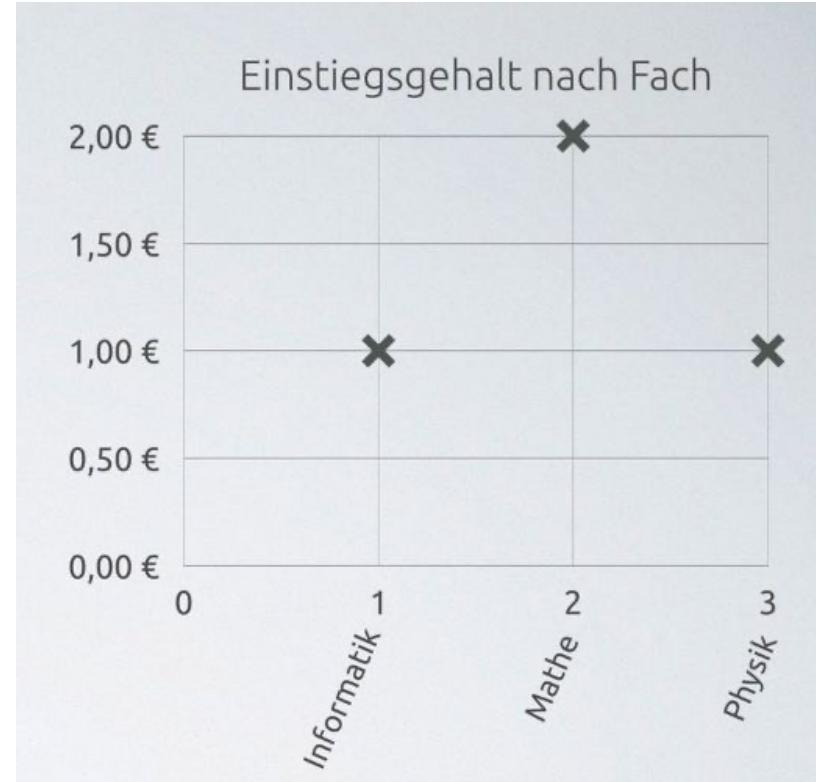
- **Metrische Daten**
 - Diese Daten können direkt in der **linearen Regression** verwendet werden

3.2 Verwendung von Datenarten im Machine-Learning

- **Nominale Daten**
 - Diese Daten müssen vorab aufbereitet werden
 - **Beispiel:** Welches Fach studierst Du?
 - Informatik, Mathematik, Physik
 - Eine mögliche Umsetzung könnte sein:
 - Informatik = 1, Mathematik = 2, Physik = 3

3.2 Verwendung von Datenarten im Machine-Learning

- Wenn wir eine Codierung Nominaler Daten in Zahlen zusammen mit dem Gehalt (einer Metrischen Größe) in ein Koordinatensystem eintragen:
- Dann erkennt man:
Eine Regression ist hier nicht möglich!
Sie würde kein sinnvolles Ergebnis liefern.



3.2 Verwendung von Datenarten im Machine-Learning

- **Regression** benötigt numerische Daten, damit eine **Formel** nach dem Schema:

$$y_t = x_{t1}\beta_1 + x_{t2}\beta_2 + \cdots + x_{tK}\beta_K + \varepsilon_t$$

möglich wird.

- Im Rahmen der Vorverarbeitung ist eine **Umwandlung ins Numerische** notwendig.
- Gängiges Verfahren: **One-Hot-Encoding**

3.2 Verwendung von Datenarten im Machine-Learning

- Das One-Hot-Encoding
 - Idee: Es werden «**Numerische Dummies**» aus Nominalen Daten erzeugt.

Fach	Informatik?	Mathematik?	Physik?
Informatik	1	0	0
Mathematik	0	1	0
Physik	0	0	1

- Nominale Daten werden in Metrische Daten überführt und können dann für die Regression verwendet werden.
- Durch die Umwandlung entstehen allerdings viele neue Spalten

3.2 Verwendung von Datenarten im Machine-Learning

- **Ordinale Daten**
 - Diese Daten haben eine **natürliche Ordnung**
 - Die **Abstände sind nicht gleich groß** bzw. sind **manchmal auch nicht definiert.**
 - Beispiele zeitlicher Beschreibungen:
 - **Morgens, Mittags, Abends**
 - **Baby, Kleinkind, Jugendlicher, Erwachsener**
 - Weiteres gutes Beispiel:
 - **Kleinunternehmen, Mittestand, Aktiengesellschaft**
 - Weniger gutes Beispiel
 - **Aachen, Berlin, Zürich** → Hier ist keine natürliche Ordnung vorhanden.

3.2 Verwendung von Datenarten im Machine-Learning

- Umwandlung Ordinaler Daten
- Option 1 – Umwandlung in Metrische Daten
 - *Baby* = 1, *Kleinkind* = 2,
Jugendlicher = 3, *Erwachsener* = 4
 - *Sehr-gut* = 1, *Gut* = 2, *Befriedigend* = 3,
Ausreichend = 4, *nicht-bestanden* = 5
 - Erwachsene sind nicht viermal so alt wie Babies. Der Unterschied zwischen *Ausreichend* und *nicht-bestanden* kann viel größer sein, als der zwischen *sehr-gut* und *gut*.
 - Unser Modell lernt daher mit einer falschen Struktur!

3.2 Verwendung von Datenarten im Machine-Learning

- Option 2 – **Umwandlung in Metrische Daten**
 - Wir behandeln Ordinale Daten wie Nominale Daten und realisieren **One-Hot-Encoding**
 - Dies ergibt sehr viel mehr Spalten
 - Die Folge ist: Mehr Dimensionen, mehr Parameter, die trainiert werden müssen

3.3 Problem der Redundanzen beim One-Hot-Encoding

- Beispiel mit Nominalen Daten in Form der **One-Hot-Encoding** dargestellt:

Fach	Informatik?	Mathematik?	Physik?	Intercept	Ergebnis
Informatik	1	0	0	1	1
Mathematik	0	1	0	1	2
Physik	0	0	1	1	1

$$Y = a \cdot [\text{Informatik?}] + b \cdot [\text{Mathe?}] + c \cdot [\text{Physik?}] + d \cdot [\text{Intercept}]$$

- Führt zu dem Gleichungssystem:

$$1 = a \cdot 1 + b \cdot 0 + c \cdot 0 + d \cdot 1$$

$$2 = a \cdot 0 + b \cdot 1 + c \cdot 0 + d \cdot 1$$

$$1 = a \cdot 0 + b \cdot 0 + c \cdot 1 + d \cdot 1$$

3.3 Problem der Redundanzen beim One-Hot-Encoding

- Das Gleichungssystem

$$1 = a \cdot 1 + b \cdot 0 + c \cdot 0 + d \cdot 1$$

$$2 = a \cdot 0 + b \cdot 1 + c \cdot 0 + d \cdot 1$$

$$1 = a \cdot 0 + b \cdot 0 + c \cdot 1 + d \cdot 1$$

hat aber **zwei Lösungen**, weil die Tabelle **Redundanzen/Abhängigkeiten** hat:

$$a = 0$$

$$b = 1$$

$$c = 0$$

$$d = 1$$

$$a = 1$$

$$b = 2$$

$$c = 1$$

$$d = 0$$

3.4 Lineare Abhangigkeiten beim One-Hot-Encoding beseitigen

- Warum sind Redundanzen oder Abhangigkeiten bose?
 - Beispiel: X1 und X2 sind identisch, dadurch wird die Formelbeschreibung des Modells uneindeutig:

X1	X2	Y	
1	1	1	$Y = 1 \cdot X1 + 0 \cdot X2$
2	2	2	$Y = 0 \cdot X1 + 1 \cdot X2$
3	3	3	$Y = 0.5 \cdot X1 + 0.5 \cdot X2$
4	4	4	$Y = 0.75 \cdot X1 + 0.25 \cdot X2$

von Auge sieht man es, aber das Modell errechnet beliebig viele Modelle, da X1 und X2 redundant sind

.... Oder anders herum gesagt: Man bekommt plotzlich beliebig viele Modelle

3.4 Lineare Abhangigkeiten beim One-Hot-Encoding beseitigen

.... Oder anders herum gesagt: Man bekommt plotzlich beliebig viele Modelle

X1	X2	Y	
1	2	1	$Y = 1 \cdot X1 + 0 \cdot X2$
2	4	2	$Y = 0 \cdot X1 + 0.5 \cdot X2$
3	6	3	$Y = 0.5 \cdot X1 + 0.25 \cdot X2$
4	8	4	$Y = 0.75 \cdot X1 + 0.125 \cdot X2$

3.4 Lineare Abhangigkeiten beim One-Hot-Encoding beseitigen

- **Linear abhangige Spalten:**

- Es gibt mindestens mehr als eine Losung
- Es gibt schlechtesten Falls unendlich viele «beste» Losungen.
- Ein Machine-Learning-Modell wurde unter diesen Umstanden allermeistens nicht die tatsachlich objektiv beste Losung erzeugen.
- Die objektiv beste Losung zeigt sich nur bei Daten ohne direkte Abhangigkeiten.
- Fazit: **Redundanzen** mussen in einem weiteren Schritt extra beseitigt werden

X1	X2	Y
1	2	1
2	4	2
3	6	3
4	8	4

$$Y = 1 \cdot X1 + 0 \cdot X2$$
$$Y = 0 \cdot X1 + 0.5 \cdot X2$$
$$Y = 0.5 \cdot X1 + 0.25 \cdot X2$$
$$Y = 0.75 \cdot X1 + 0.125 \cdot X2$$

⋮
⋮

3.4 Lineare Abhangigkeiten beim One-Hot-Encoding beseitigen

- Ubliche Vorgehensweise in Machine-Learning-Umgebungen wie z.B. in R:
Streichen einer Spalte, um Redundanz zu vermeiden:

Fach	Informatik?	Mathematik?	Physik?	Intercept	Ergebnis
Informatik	1	0	0	1	1
Mathematik	0	1	0	1	2
Physik	0	0	1	1	1

$$Y = a \cdot [\text{Informatik?}] + b \cdot [\text{Mathe?}] + \cancel{c \cdot [\text{Physik?}]} + d \cdot [\text{Intercept}]$$

$$1 = a \cdot 1 + b \cdot 0 + \cancel{c \cdot 0} + d \cdot 1$$

$$2 = a \cdot 0 + b \cdot 1 + \cancel{c \cdot 0} + d \cdot 1$$

$$1 = a \cdot 0 + b \cdot 0 + \cancel{c \cdot 1} + d \cdot 1$$

3.4 Lineare Abhangigkeiten beim One-Hot-Encoding beseitigen

- Bei **Python** braucht man bei **One-Hot-Encoding** Redundanzen nicht explizit zu berucksichtigen!
- Automatisches Vorgehen bei **Python**:
Beim **One-Hot-Encoding**:
 - Spalten werden belassen
 - Es wird nach der besten Losung mit der besten Annerung an die Trainingsdaten gesucht.Kriterium:

$$\min \sqrt{a^2 + b^2 + c^2 + d^2}$$

$$a = 1$$

$$b = 2$$

$$c = 1$$

$$d = 0$$

$$\sqrt{6}$$

$$a = 0$$
 ✓

$$b = 1$$

$$c = 0$$

$$d = 1$$

$$\sqrt{2}$$

3.5 Python-Beispiel zum One-Hot-Encoding

- **Beispiel:** Hotelverkaufspreise – One-Hot-Encoding der Städte, damit eine lin. Regression möglich wird.
 - **File:** `hotels.csv`
 - **Python-Demo:** `NominaleDatenUndLineareRegression.ipynb`
- Mit der Funktion `get_dummies(...)` kann eine Nominale Spalte per **One-Hot-Encoding ins Metrische** überführt werden.

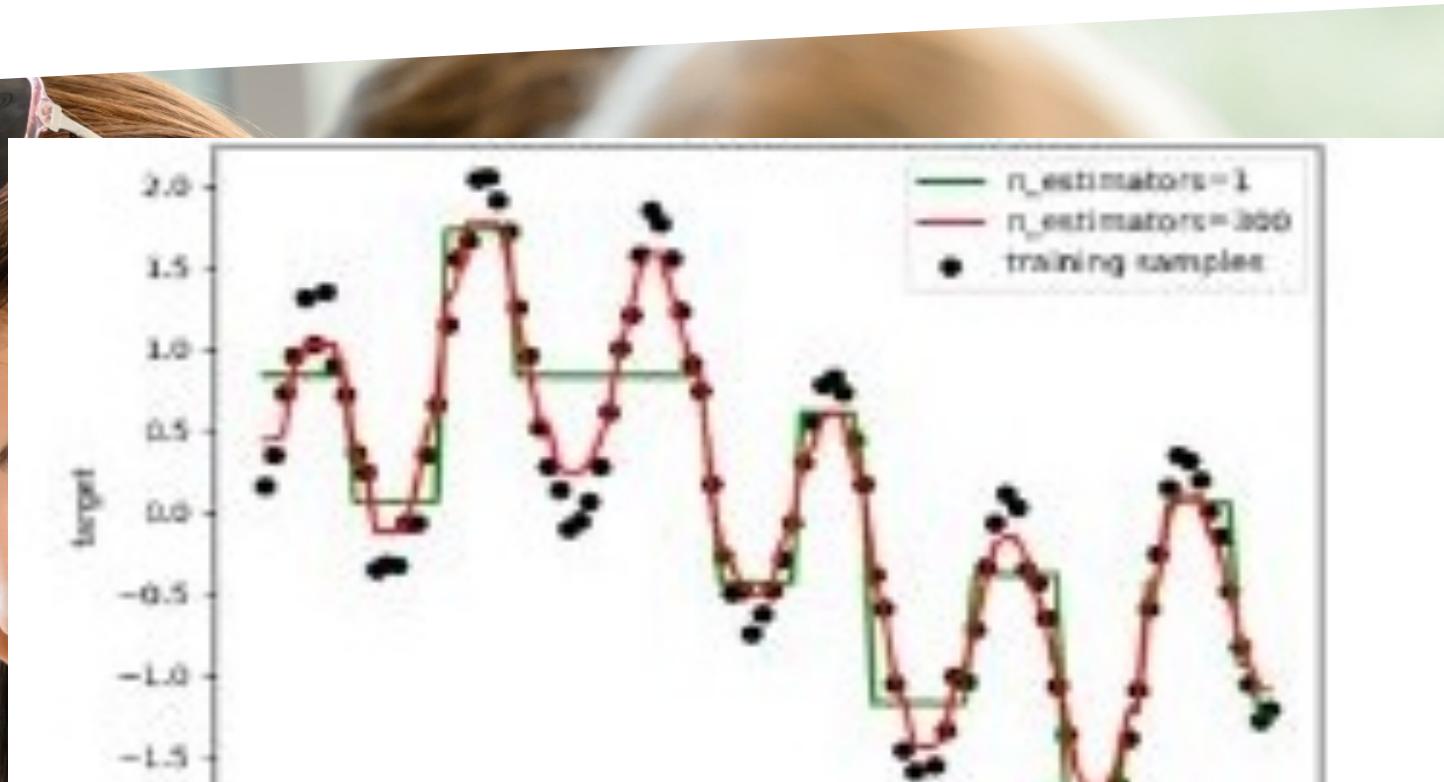
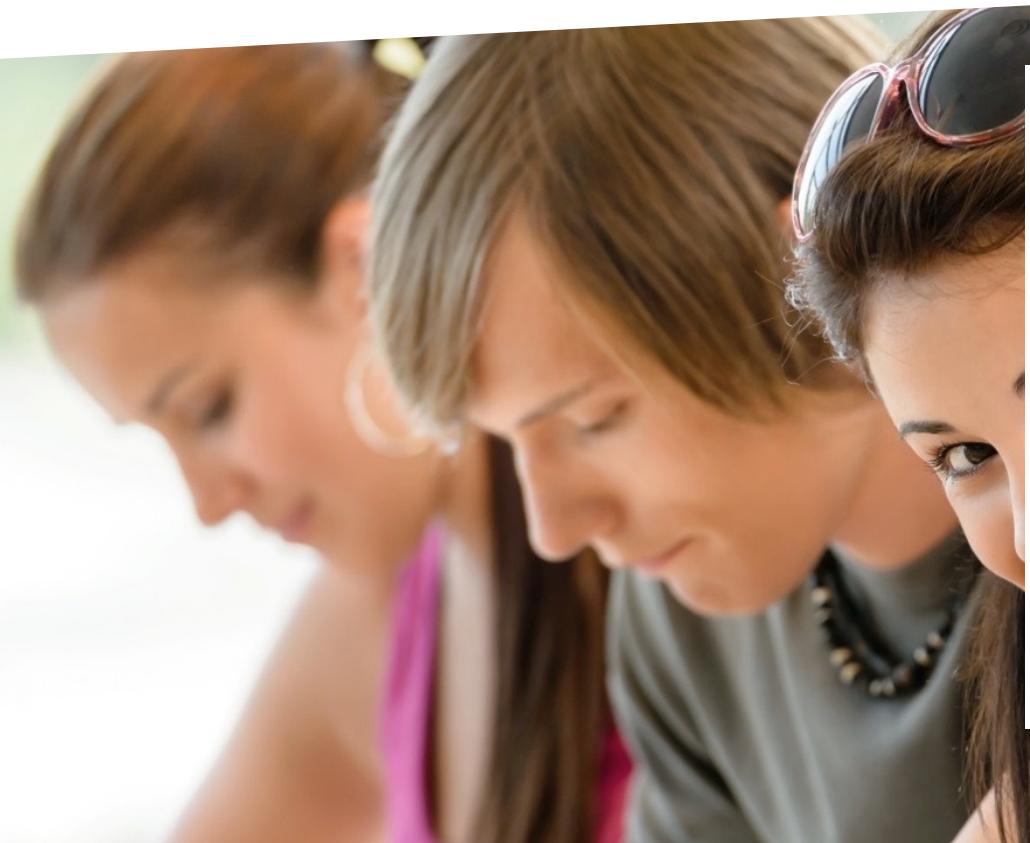
	Gewinn	Preis in Mio	Quadratmeter	Stadt
0	119000.0	21.88	3938.0	Berlin
1	250000.0	27.95	3986.0	München
2	250000.0	16.09	2574.0	Köln
3	145000.0	27.58	4155.0	München
4	110000.0	23.76	3795.0	Berlin



```
.get_dummies(df0, columns = ["Stadt"]) →
```

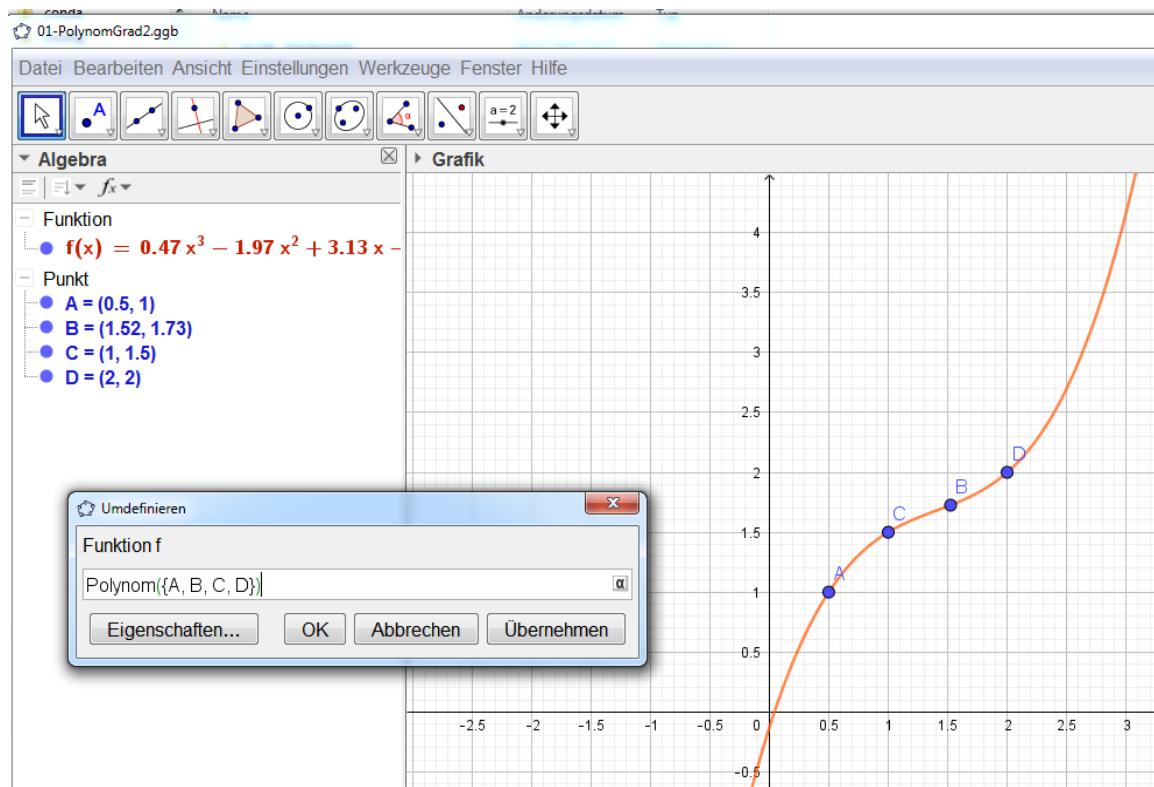
	Gewinn	Preis in Mio	Quadratmeter	Stadt_Berlin	Stadt_Köln	Stadt_München
0	119000.0	21.88	3938.0	1	0	0
1	250000.0	27.95	3986.0	0	0	1
2	250000.0	16.09	2574.0	0	1	0
3	145000.0	27.58	4155.0	0	0	1
4	110000.0	23.76	3795.0	1	0	0

4. Polynomiale Regression



4.1 Idee der Polynomialen Regression

- Zunächst: Veranschaulichung einer Gleichung 2. Grades:



Mit der Eingabe
Polynom(...) kann
eine Menge von
Punkten
angegeben werden.
Geogebra findet
dann automatisch
ein Polynom, dass
durch die Punkte
geht

Software: GeoGebra – Files auf Moodle

- File: [PolynomGrad2.ggb](#)

4.1 Idee der Polynomialen Regression

- **Idee:**
 - Das **Modell** hat **polynomiale Terme**
 - Die **polynomialen** Anteile werden separat **ergänzt** und dann in eine **lineare Regression** gegeben:



4.2 Überführung der Polynomialen Regression in einer Lineare Regression

- Beispiel: Polynom 2. Grades:

$$f(x) = ax^2 + bx + c$$

- Typischerweise liegen gegebenen Daten liegen jeweils Werte für x und das Ergebnis der Funktion $f(x)$ in Form von **Wertepaaren** vor.
- Die **Polynomiale Regression** soll jetzt a , b und c «lernen», um später aus einem gegebenen x möglichst gut das Ergebnis $f(x)$ vorherzusagen.
- Hierzu müssen die Daten, welche nur x enthalten mit einer zusätzlichen Spalte $x*x$ ergänzt werden, damit vor allem das « a » gelernt wird. Dies führt auf eine **Lineare Regression**.



x	$x * x$
1	1
3	9

4.2 Überführung der Polynomialen Regression in einer Lineare Regression

- Beispiel: Polynom 2. Grades mit mehr als zwei Dimensionen (mehrere x-Werte)

$$f(x) = ax_1 + bx_2 + cx_1^2 + dx_1x_2 + ex_2^2 + g$$

- Typischerweise liegen gegebenen Daten liegen jeweils Werte für x_1, \dots, x_n und das Ergebnis der Funktion $f(x)$ in Form von **Tupeln** vor.
- Die **Polynomiale Regression** soll jetzt **a, b, c, d, e usw.** «lernen», um später aus einer Kombination gegebener x_1, \dots, x_n möglichst gut das Ergebnis $f(x)$ vorherzusagen.
- Die Ergänzung zusätzlicher Spalten zur Überführung in eine **Lineare Regression** müssen nicht nur die x_i mit **sich selber**, sondern auch **untereinander multipliziert** werden:

x_1	x_2
1	2
3	4



x_1	x_2	$x_1 * x_1$	$x_1 * x_2$	$x_2 * x_2$	Intercept
1	2	1	2	4	1
3	4	9	12	16	1

4.3 Von Underfitting bis Overfitting – Adäquatheit des Modells

- Modelle müssen angemessen sein:



- **Underfitting**: Das Modell ist **zu grob** und bildet **zu wenig Eigenschaften** der Daten ab.
- **Overfitting**: Das Modell ist **zu fein** und bildet Zufälligkeiten (stat. Rauschen) des Examplesets ab, die für vergleichbare spätere Fälle schlechter funktionieren werden.
- Modell kann nur durch Kenntnis des physikalischen / mathematischen „Hintergrunds“ richtig ausgewählt werden. Falls dieser Hintergrund unbekannt ist, kann aber nur intuitiv entschieden werden!

4.4 Python-Beispiel zur Polynomialen Regression

- **Beispiel:** Profit von Nutzflächen, die in Länge und Breite gegeben sind.

- **File:** [fields.csv](#)

- **Python-Demo:** [PolynomialeRegressionEinführung.ipynb](#)

Ablauf anschauen

- Lineare Regression: **R² = 0,915**

- Mit

```
pf = PolynomialFeatures(degree = 2, include_bias = False)
X_train_transformed = pf.transform(X_train)
```

werden die **zusätzlichen Spalten** erzeugt: x_1 , x_2 , $x_1 \cdot x_1$, $x_1 \cdot x_2$, $x_2 \cdot x_2$

- Beispiel einer Spatenauswahl:

```
X_train_transformed = pf.transform(X_train)[:, [0, 1, 2, 3, 4]]
```

- Dies wird in eine Lineare Regression gegeben:

```
model = LinearRegression()
model.fit(X_train_transformed, y_train)
```

- Ergebnis Polynomiale Regression: **R² = 0,988**

4.4 Python-Beispiel zur Polynomialen Regression

- **Beispiel:** Daten von 50000 Diamanten: Preise, Karat, Qualität und Abmessungen.
 - Das Gewicht eines Diamanten wird oft in Karat angegeben
 - Angenommen, wir haben keine Waage. Können wir den Wert auch über Länge, Breite und Höhe eines Diamanten schätzen?
 - Vielleicht ist die Schätzung sogar besser als wenn wir das Gewicht messen?
- **File:** diamonds.csv
- **Quelle:** <https://www.kaggle.com/shivam2503/diamonds>



4.4 Python-Beispiel zur Polynomialen Regression

- **Aufgabe:** Vergleich folgender Modelle
 - Lineare Regression: Preisabschätzung auf Basis des Gewichts (Carat)
 - Lineare Regression: Preisabschätzung auf Basis der carat Länge, Breite und Höhe (Spalten x, y, z)
 - Polynomiale Regression aus Länge, Breite und Höhe (Spalten x, y, z) (mit Grad 2)
- **File:** diamonds.csv
- **Python-Beispiel-Lösung:** [BeispielLösungModellvergleichDiamanten.ipynb](#)
- R²-Ergebnisse:
 - Lineare Regression mit carat: 0,85
 - Lineare Regression mit carat, x, y, z: 0,78
 - Polynomiale Regression carat, x, y, z,: 0,86

4.4 Nützliches für Machine-Learning – Numpy Where

- Häufiges Problem: Bestimmte Inhalte aus Arrays ziehen

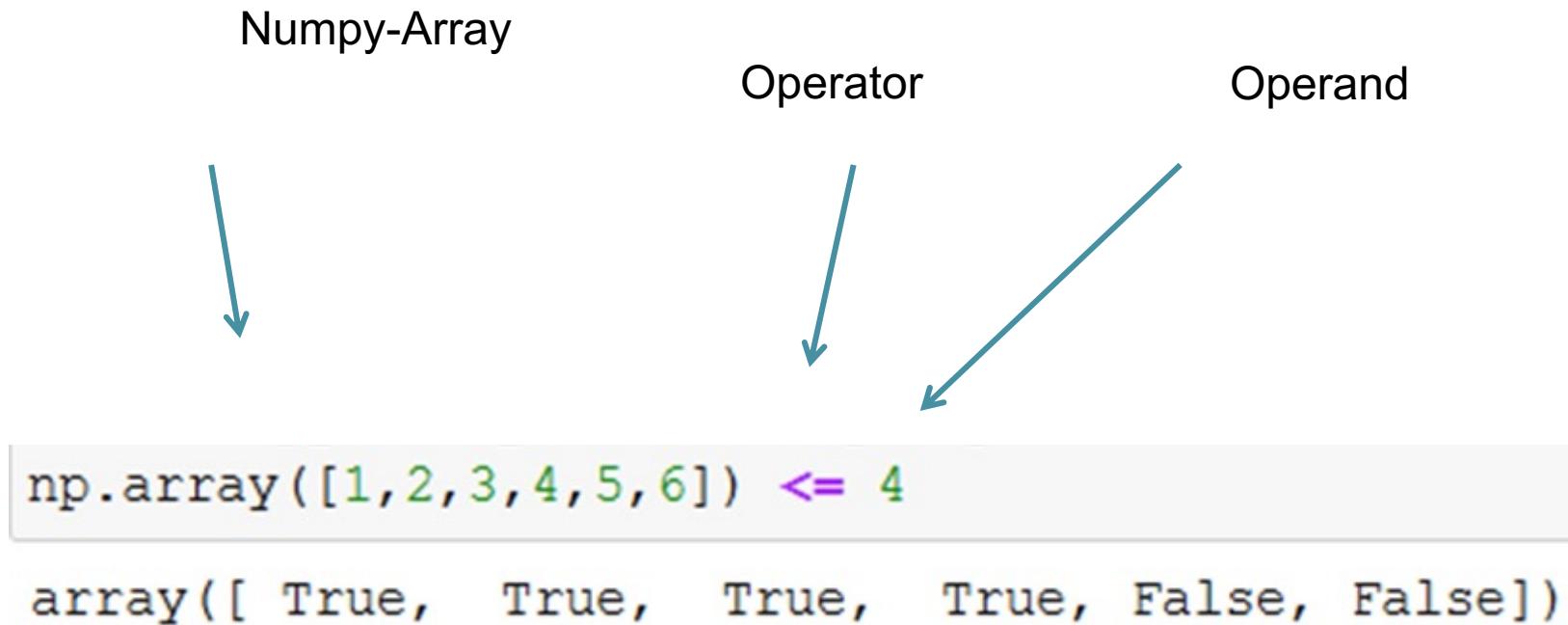
```
np.where([True, False], [1, 2], [3, 4])  
array([1, 4])
```

The diagram illustrates the use of the `np.where` function. It shows three arrays: a query array [True, False], an array to extract values from if the query is True [1, 2], and an array to extract values from if the query is False [3, 4]. Arrows point from the labels to the corresponding elements in the arrays.

- Query: Points to the first element of the query array [True, False].
- Entnahme bei `True`: Points to the first element of the second array [1, 2].
- Entnahme bei `False`: Points to the first element of the third array [3, 4].

4.4 Nützliches für Machine-Learning – Numpy Where

- **Häufiges Problem:** Operationen auf ein Array anwenden



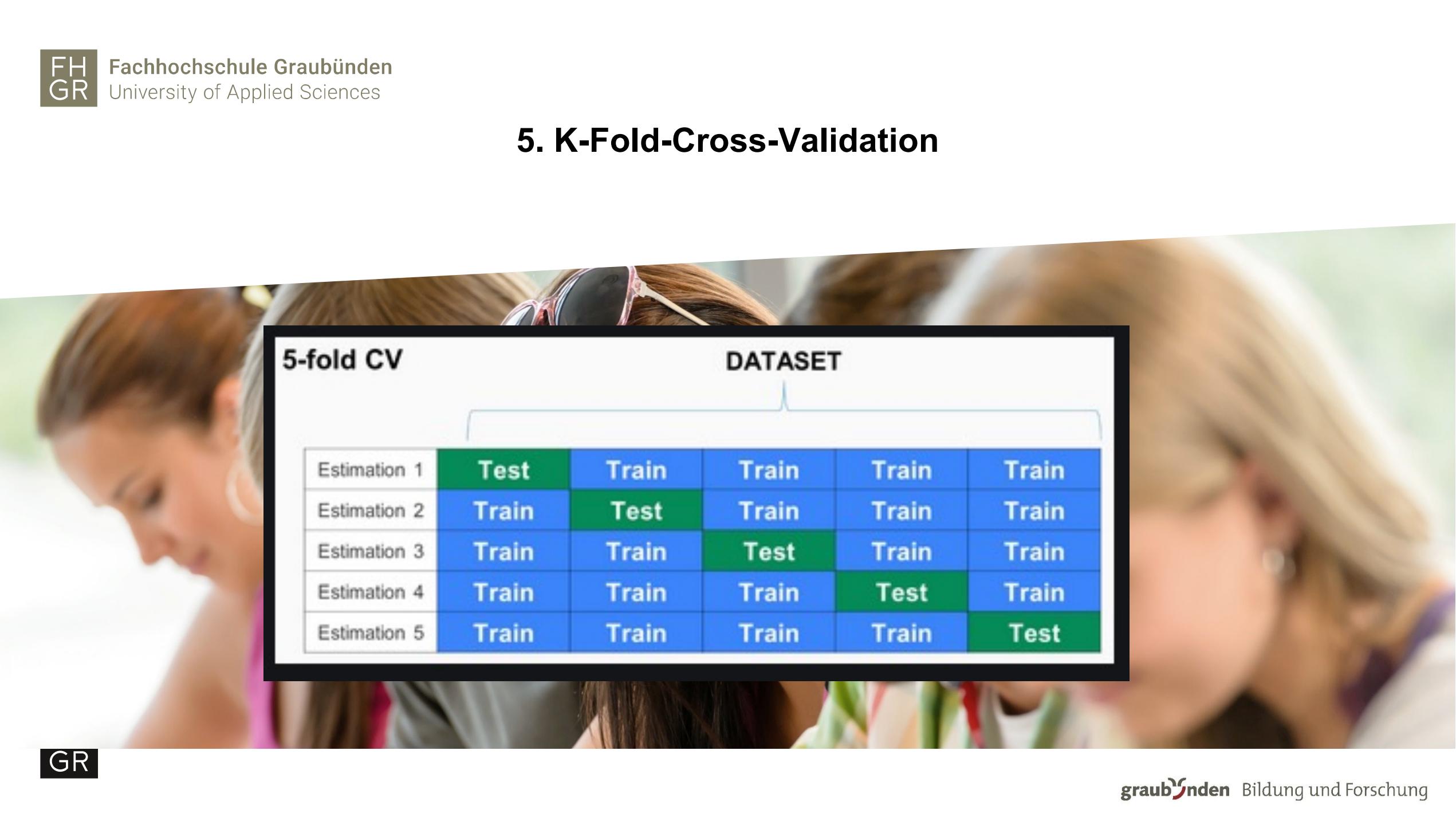
4.4 Nützliches für Machine-Learning – Numpy Where

- **Kombination:** Operationen auf ein Array innerhalb eines Numpy Where anwenden

```
np.where(np.array([1,2,3,4,5,6]) <= 4, ["bestanden"], ["nicht bestanden"])
array(['bestanden', 'bestanden', 'bestanden', 'bestanden',
       'nicht bestanden', 'nicht bestanden'], dtype='<U15')
```

- **Wichtig in der Datenvorverarbeitung:** Solche Operationen können Metrische Größen in nominale Daten umwandeln oder umgekehrt.
 - **Konkrete Anwendung:**
Farbe von Diamanten in eine Ordinale Größe verwandeln oder Daten mit einer bestimmten Farbe selektieren.
 - **Python-Beispiel:** [NützlichesInPythonFürMachineLearning-2.ipynb](#)

5. K-Fold-Cross-Validation

A blurred background image shows three people from behind, looking towards a screen. The person on the left has short brown hair and is wearing a white shirt. The person in the middle has long brown hair and is wearing a pink top. The person on the right has long blonde hair and is wearing a dark top.

5-fold CV

DATASET

	Test	Train	Train	Train	Train
Estimation 1	Test	Train	Train	Train	Train
Estimation 2	Train	Test	Train	Train	Train
Estimation 3	Train	Train	Test	Train	Train
Estimation 4	Train	Train	Train	Test	Train
Estimation 5	Train	Train	Train	Train	Test

5.1 Idee der K-Fold-Cross-Validation

- **Ziel:** Der R²-Wert soll möglichst wenig Zufälligkeiten enthalten.
 - Leider ist der R²-Wert mal besser, mal schlechter, je nachdem, welche zufälligen Trainings- und Testdaten man hat.

Ausgangsbasis:



5.1 Idee der K-Fold-Cross-Validation

- **Idee:**
 - Daten werden in k Blöcke aufgeteilt
 - Es werden mehrere Modell auf den gleichen Daten trainiert
 - K-1 Blöcke werden dann jeweils die Trainings-Daten, der restliche Block enthält dann die Testdaten
 - Wir bekommen mehrere Modelle und mehrere Genauigkeits-/ R^2 -Werte
 - Aus der Mittelung der Genauigkeits-/ R^2 -Werte werden Schwankungen ausgeglichen
 - Genaue und reproduzierbare Genauigkeits-/ R^2 -Werte.

5.1 Idee der K-Fold-Cross-Validation

– Beispiel:

- Die R²-Werte sind die unterste Zeile. **Training: Orange, Testing: blau**
- Erst durch die Mittelung der R²-Werte wird klar, dass die Lineare Regression besser abschneidet:

Lineare Regression: 0.85		
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
0.8	0.9	0.85

Polynomiale Regression: 0.75		
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
0.5	0.75	1

Spitzenwert hier

5.2 Die K-Fold-Cross-Validation in Python

- Prinzip:

```
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression

kf = KFold(n_splits = 10, shuffle = True)
score = 0
for train_index, test_index in kf.split(X):
```

...Modellbildung...

```
    model.fit(X_train, y_train)
    score += model.score(X_test, y_test)
print(score / kf.get_n_splits(X))
```

10x Score ermitteln

Durchschnitt der Scores

- **Python-Demo: K-Fold-Cross-Validation-Teil1.ipynb**

- **Python-Demo: K-Fold-Cross-Validation-Teil2.ipynb**

5.2 Die K-Fold-Cross-Validation in Python

- **Kurzschreibweise:** Alles in einer Zeile....

```
scores = cross_val_score(LinearRegression(), X, Y, cv = KFold(n_splits = 10))
```

- **Python-Demo:** [K-Fold-Cross-Validation-Kurzschreibweise.ipynb](#)
- **Die Repeated-K-Fold-Cross-Validation**

- Idee ist die Ersetzung der For-Schleife.:

```
scores = cross_val_score(LinearRegression(), X, Y, cv = RepeatedKFold())
```

- Default sind 10 Wiederholungen. Höhere Werte sind selten sinnvoll
 - **Python-Demo:** [Repeated-K-Fold-Cross-Validation.ipynb](#)

5.3 Modellvergleich mit der K-Fold-Cross-Validation

- **Aufgabe:** Vergleich folgender Modelle für Diamantendaten
 - **Modell 1:** Lineare Regression: Preisabschätzung auf Basis des Gewichts (Carat)
 - **Modell 2:** Lineare Regression: Preisabschätzung auf Basis der Länge, Breite und Höhe (Spalten x, y, z)
- **File:** diamonds.csv
- **Ausgangsbasis:** BeispielLösungModellvergleichDiamanten.ipynb (aus Kapitel 4)
- **Beispiel-Lösung:** Lösung-K-Fold-Crossvalidation.ipynb

(Achtung Rechenzeit!)

- Modell 1: [0.85073998 0.85434192 0.84308919 ... 0.8503476 0.85604449 0.85196604]
Durchschnitt der Scores: 0.8492823956742708
- Modell 2: [0.55119189 0.78372608 0.78037373 ... 0.7905612 0.78000833 0.78642454]
Durchschnitt der Scores: 0.7723086644890922

6. Statistische Grundlagen

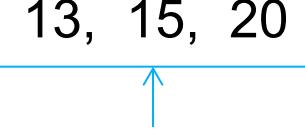


6.2 Mittelwert und Median

– Der Mittelwert

- Vorgang:
 - Wir summieren alle Werte auf
 - Anschließend: Teilen durch die Anzahl der Elemente
- Formel:
$$\bar{x}_{\text{arithm}} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$$
- Beispiel:
 - Temperaturwerte seien gegeben: **15° 13° 20°**
 - Berechnung: $(15 + 13 + 20) / 3 = 48 / 3 = 16$
 - Mittelwert also: **16°**

6.1 Mittelwert und Median

- Der Median
 - Vorgang:
 - Wir sortieren alle Daten
 - Der Median ist das Element in der Mitte
 - Beispiel:
 - Temperaturwerte seien gegeben: 15° 13° 20°
 - Daten sortieren: $13, 15, 20$
 - Wert in der **Mitte** ablesen:

 - Median also : 15°

6.1 Mittelwert und Median

- Python: Mittelwert und Median berechnen
 - Datei: **MittelwertUndMedian.ipynb**

```
import numpy as np  
  
incomes = [10000, 30000, 5000000]
```

```
np.median(incomes)
```

```
np.mean(incomes)
```

```
import pandas as pd  
  
df = pd.read_csv("./diamonds.csv")  
  
df.head()
```

```
df["carat"].mean()
```

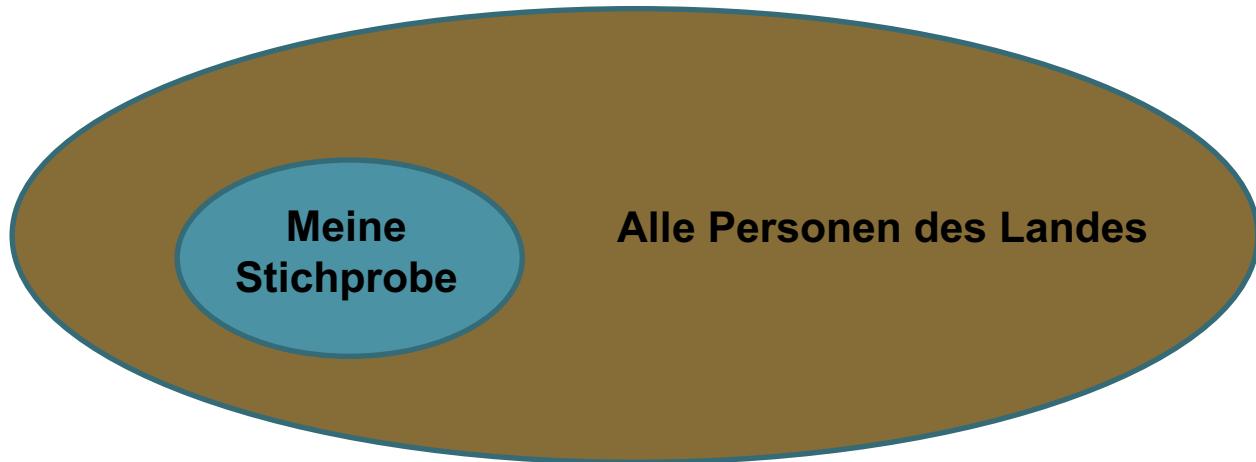
```
df["carat"].median()
```

```
df.mean()
```

```
df.median()
```

6.2 Varianz

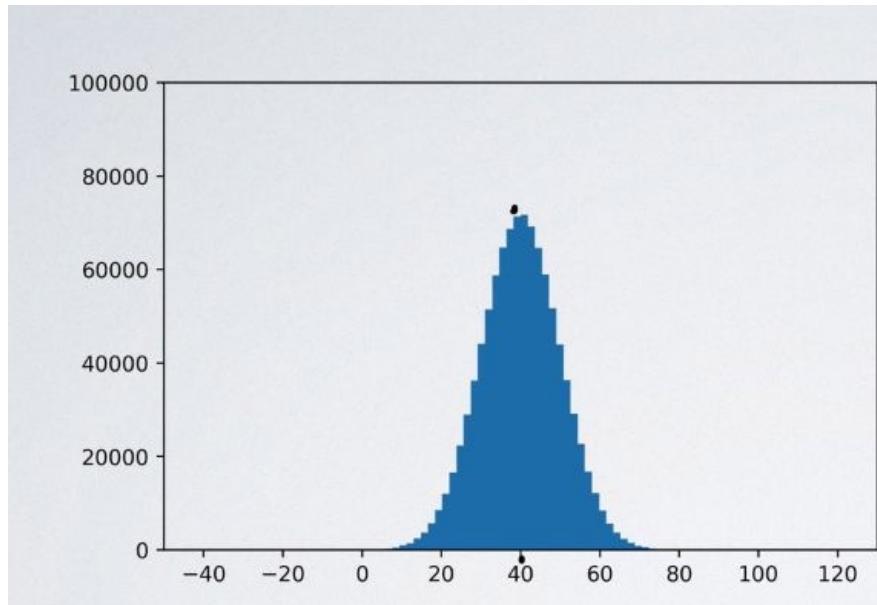
- **Stichproben als Ausgangsbasis:**
 - Beispiel:
 - Ich möchte das Durchschnittseinkommen eines Landes bestimmen
 - Ich rufe 10.000 Leute an, und frage Sie nach ihrem Einkommen.
 - Dies ist eine Stichprobe



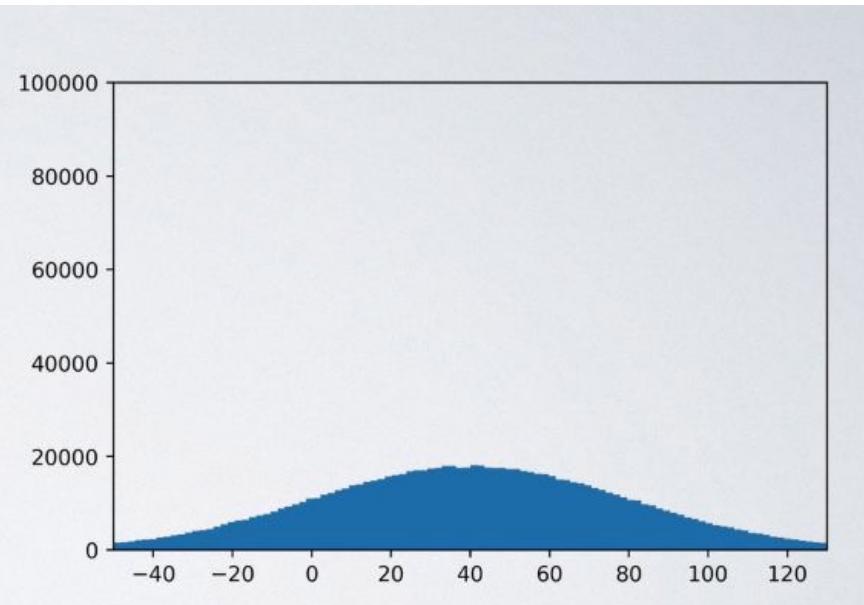
6.2 Varianz

– Motivation:

- Wir würden gerne abschätzen können, ob die Daten nah beieinander oder weit voneinander weg liegen.



weniger Streuung



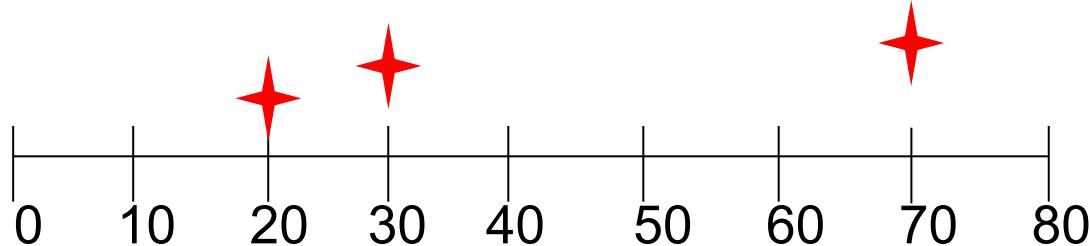
mehr Streuung

6.2 Varianz

- Formel zur Berechnung der Varianz:

$$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

- Beispielszenario: Gegeben seien die Werte 20, 30, 70

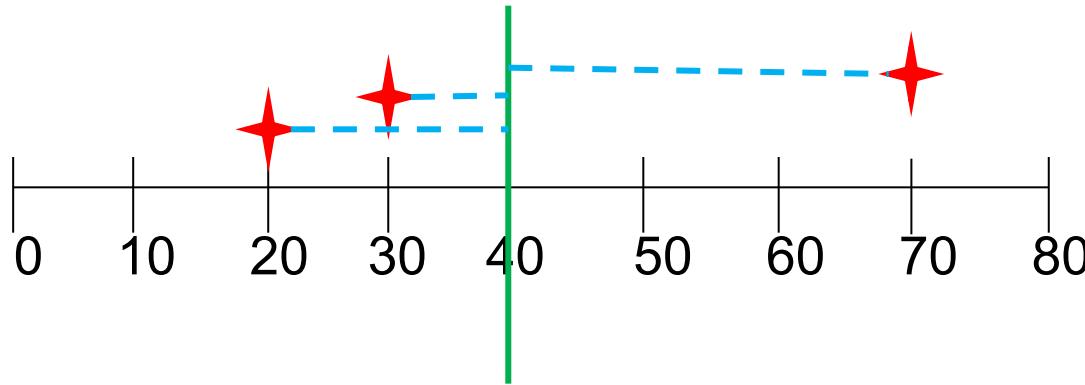


6.2 Varianz

- Formel zur Berechnung der Varianz:

$$\frac{1}{n} \sum_{i=1}^n (\underline{x_i} - \underline{\bar{x}})^2$$

- Beispielszenario: Gegeben seien die Werte 20, 30, 70
- Durchschnitt: $(20 + 30 + 70) / 3 = 40$
- Varianz: $1/3 * (30^2 + 10^2 + 20^2) = 1/3 * 1400 = \underline{466,6666}$



6.2 Varianz

- **Korrigierte Stichprobenvarianz:** Da durch die Schätzung des Stichprobenmittels ein Freiheitsgrad verbraucht wird, ist es üblich die empirische Varianz mit dem Faktor $\frac{1}{n-1}$ zu „korrigieren“.
 - **Formel der Korrigierten Stichprobenvarianz:**
$$v^*(x) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$
- Die **Korrigierte Stichprobenvarianz hat für das Machine-Learning wenig Bedeutung**, da hier meist mit großen Datenmengen gearbeitet wird und sich die Werte von Varianz und Korrigierter Stichprobenvarianz kaum unterscheiden.

6.3 Standardabweichung

- Formel zur Berechnung der Standardabweichung:

$$s = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

- **Idee:** Wir berechnen die Wurzel der Varianz
- **Vereinfachte / intuitive Interpretation:**
Die durchschnittliche Entfernung zum Mittelwert ist **s**

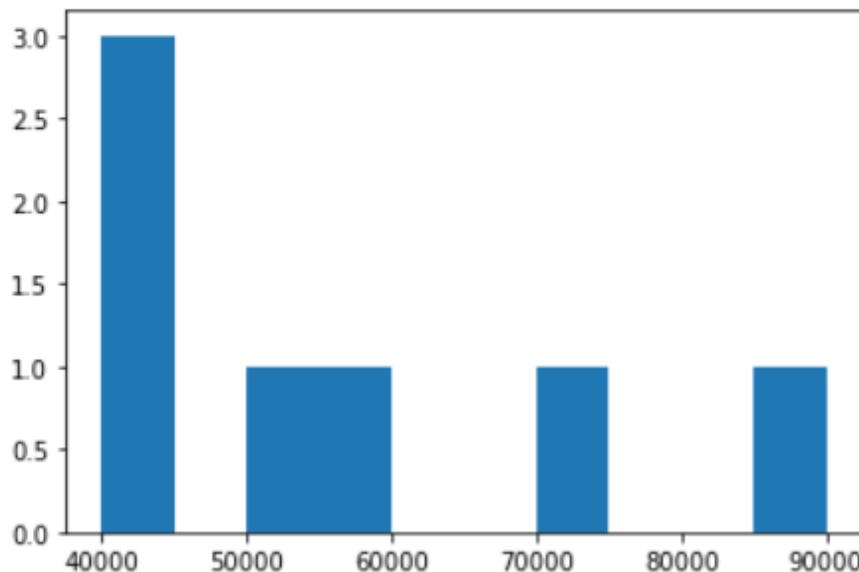
6.4 Histogramme: Verteilung Visualisieren

- Python-Demo: HistogrammeZeichnen.ipynb

Direktive nur
für Jupyter
(kein Python-Code)

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist([40000, 40000, 41000, 50000, 55000, 70000, 90000])
plt.show()
```

Für Jupyter
eigentlich nicht
nötig, aber für
normales Python

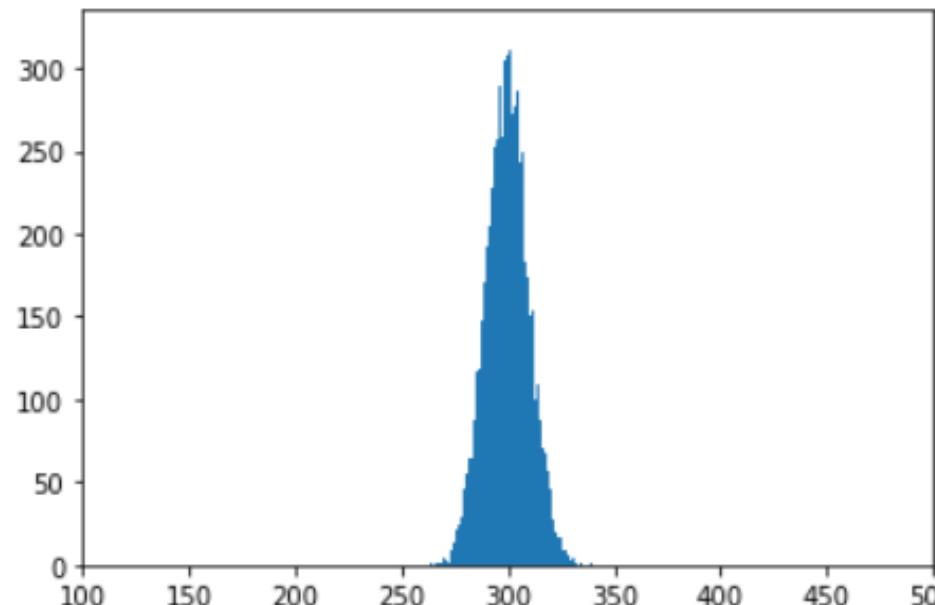


6.4 Histogramme: Verteilung Visualisieren

- Python-Demo: *VerteilungenVisualisieren.ipynb*

```
import numpy as np  
levels = np.random.normal(300, 10, 10000)  
#Siehe: https://numpy.org/doc/stable/reference/random/generator.html  
  
%matplotlib inline  
  
import matplotlib.pyplot as plt  
  
plt.hist(levels, bins = 100)  
plt.xlim(100, 500) #Begrenzung auf X-Ebene  
plt.show()
```

Mittelwert Standardabweichung
Größe der (Zufalls-) Stichprobe



6.5 Kovarianz

$$s_{xy} := \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

- **Idee:**
 - Prüfen, ob zwei Spalten A und B in einem Daten sich **ähnlich zueinander verhalten**
- **Beispiel:**
 - Wenn A hohe Werte hat, dann hat B auch hohe Werte **-> Positive Kovarianz**
 - Wenn A hohe Werte hat und B in diesen Fällen niedrige Werte **-> Negative Kovarianz**
 - Unabhängigkeit von A und B (mal so, mal so) **-> Kovarianz um die 0**
- **Praktisches Problem:** Die Kovarianz kann Werte eines beliebigen Zahlenbereichs annehmen
 - *Schwer zu interpretieren.*

6.6 Korrelationskoeffizient

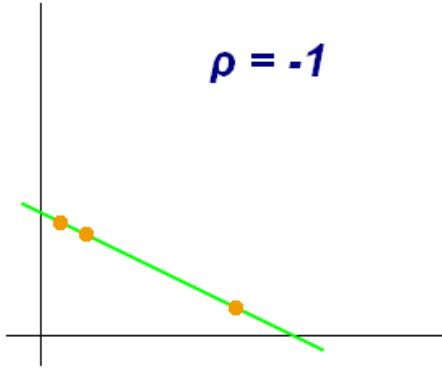
$$r_{xy} := \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- **Idee:**
 - Wie die Kovarianz, aber es gibt Ergebnisse im Zahlenbereich zwischen [-1 1]
- **Beispiel:**
 - Wenn A hohe Werte hat, dann hat B auch hohe Werte: $> 0 \leq 1$)
 - Wenn A hohe Werte hat und B in diesen Fällen niedrige Werte: $\geq -1 > 0$
 - Unabhängigkeit von A und B (mal so, mal so): 0
- **Praktisches Problem:** Die Kovarianz kann Werte eines beliebigen Zahlenbereichs annehmen
 - *Schwer zu interpretieren.*

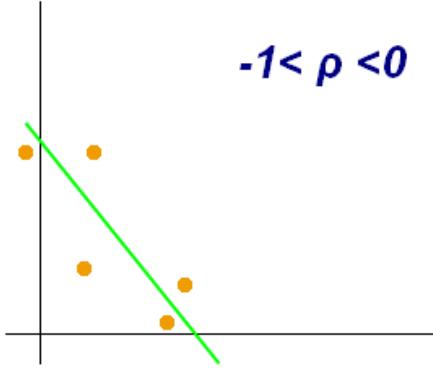
6.6 Korrelationskoeffizient

$$r_{xy} := \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$

$\rho = -1$

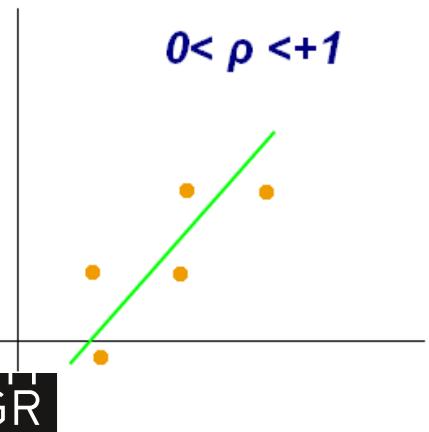


$-1 < \rho < 0$

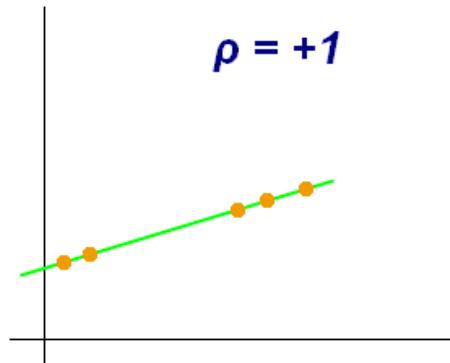


Python-Demo:
[Korrelationsanalyse.ipynb](#)

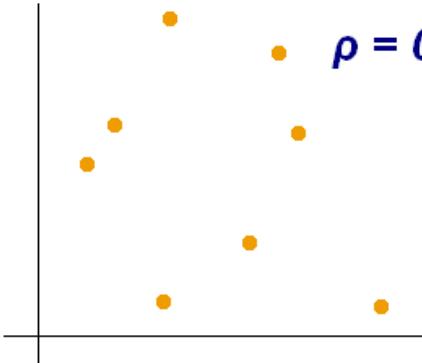
$0 < \rho < +1$



$\rho = +1$



$\rho = 0$



6.6 Korrelationskoeffizient

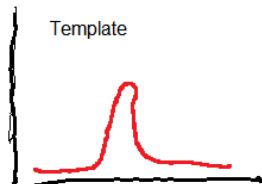
– Eindimensional:

- Z.B.: Signale über der Zeit. Das Template ist die Signalform.
Ausgabe: Ein Korrelationskoeffizient.

```
#zunächst: Eindimensional...
match_template(np.array([[0,0,0,1,2,6,2,1,0,0,0,0,0]]), np.array([[0,0,0,1,3,5,2,2,0,0,0,0,0]]))
<   array([[0.9577338]]) >
```

– Kreuzkorrelation

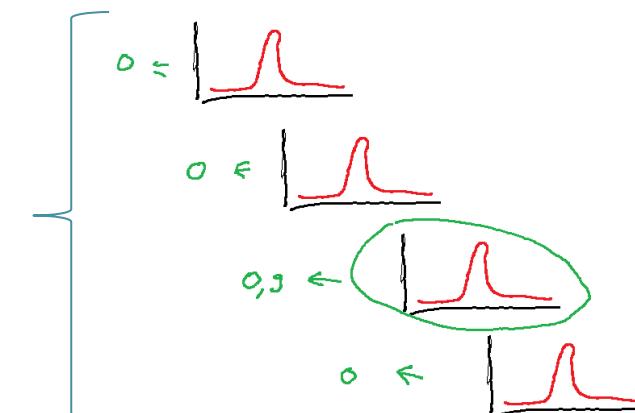
- Ein eindimensionales Datenobjekt (Signal) wird mit sich selber über die X-Achse permultiert. So kann ein Signal-Template über einen längeren Zeitraum in den Daten gesucht werden. Es ergibt sich dann der Auftrittszeitpunkt des Signals.



Kreuzkorrelation

Python-Demo:

Korrelationsanalyse.ipynb

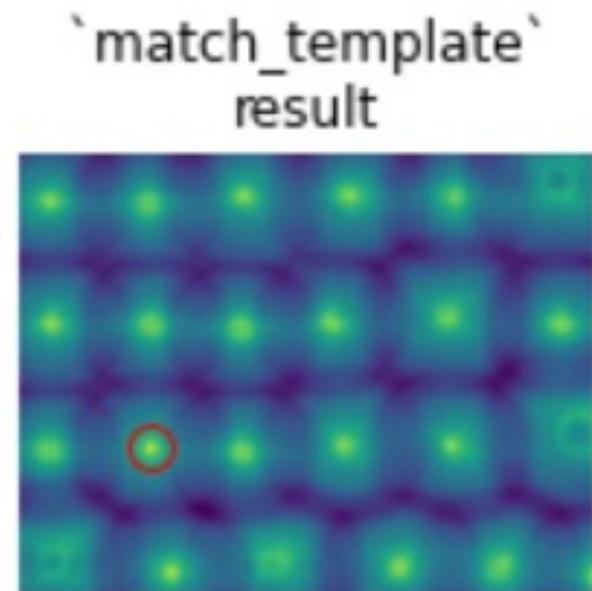
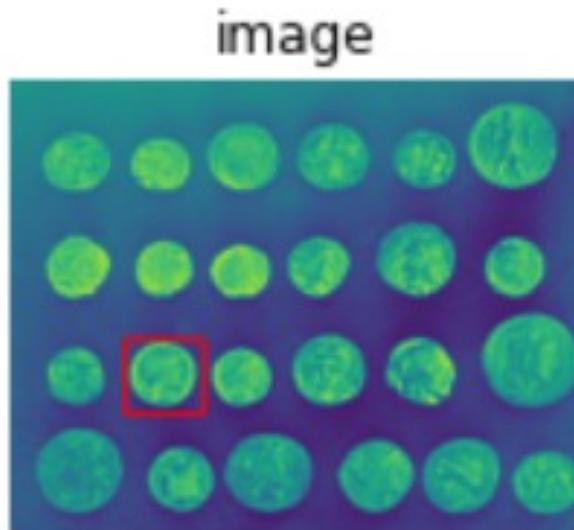
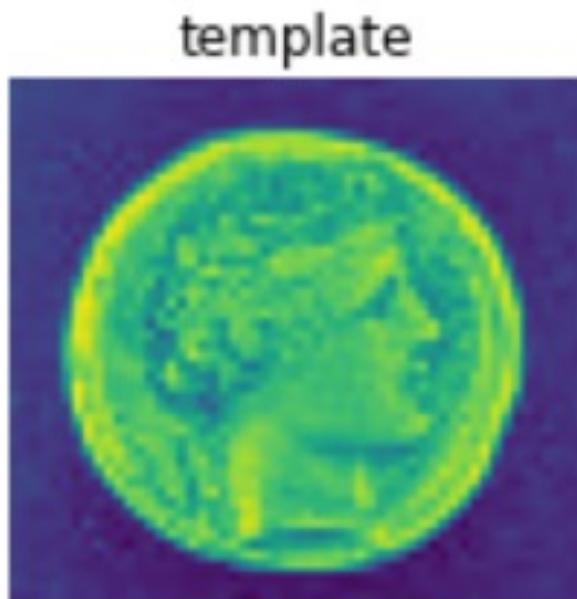


6.6 Korrelationskoeffizient

- **Zweidimensional**

- Z.B.: Bilddateien. Template ist ein kleines Bild.
Frage: Wo kommt dieses Template im großen Bild vor?
- Ausgabe: Bild mit einem «Hotspot», das die Position der höchsten Übereinstimmung zeigt.

Python-Demo:
[Korrelationsanalyse.ipynb](#)



6.7 Praxisprojekt

- Beispiel: Gehaltsliste aller Bediensteten der Stadt San Francisco
 - File: [sf_salaries.csv](#)
 - Python-Demo: [ProjektStatistischeGrundlagen.ipynb](#)
 - Gegen Fehler bei Datentypwechsel: `df = pd.read_csv("./sf_salaries.csv", low_memory=False)`
 - Filtern: `df[df["TotalPay"] > 50000] #filtert die ganze Tabelle nach Eigenschaften in einer Spalte`

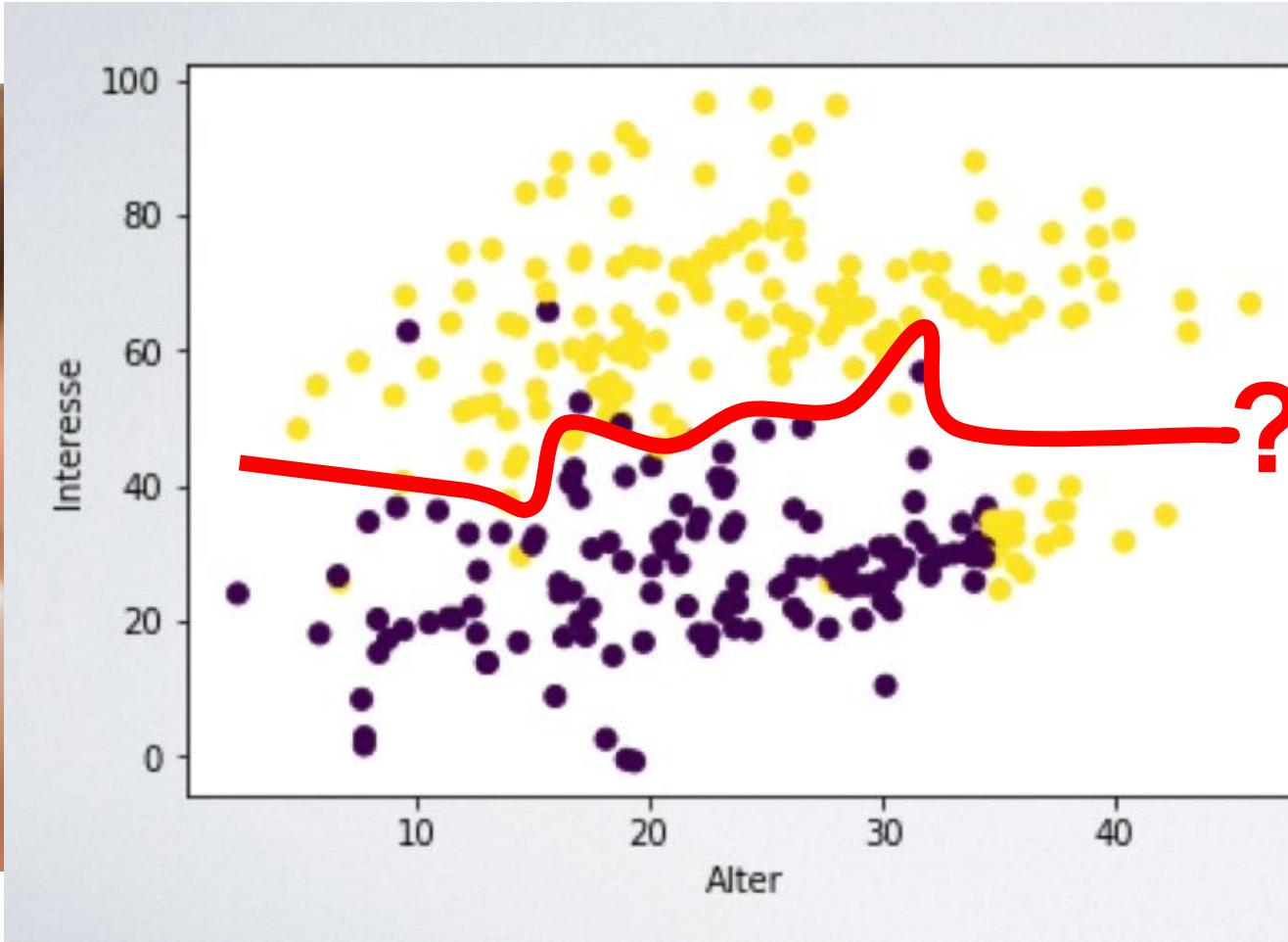
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.0	400184.25	NaN	567595.43	567595.43	2011	NaN	
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	

6.7 Praxisprojekt

Übung machen

- **Aufgaben**
 - File: [sf_salaries.csv](#)
 - **Python-Ausgangsbasis:** [ProjektStatistischeGrundlagen.ipynb](#)
 - Lösung: [ProjektStatistischeGrundlagenLösung.ipynb](#)
- **Aufgabe 1: Bezogen auf die Spalte «Total Pay»: Welche Aussagen sind richtig?**
 - Im Jahr 2014 war das Durchschnittseinkommen geringer als im Jahr 2013
 - Im Jahr 2013 war das Durchschnittseinkommen geringer als im Jahr 2012
 - Im Jahr 2012 war das Durchschnittseinkommen geringer als im Jahr 2011
 - Keine der oben genannten Aussagen ist richtig
- **Aufgabe 2: Bezogen auf die Spalte «Total Pay»: Wie verhalten sich Median und Durchschnitt zueinander?**
 - Der Median ist größer als der Durchschnitt
 - Der Durchschnitt ist größer als der Median
 - Beide sind identisch

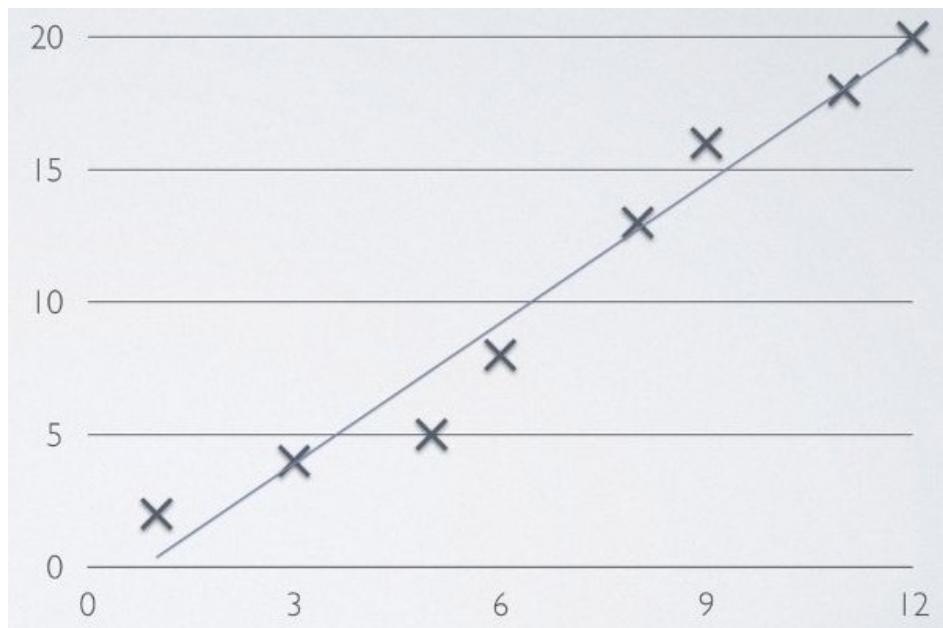
7. Klassifizierung Grundlagen



7.1 Allgemeines zur Klassifizierung

- Rückblick auf die Regression:

Wir sagen Zahlenwerte vorher....



7.1 Allgemeines zur Klassifizierung

- **Beispiele für (binäre) Klassifizierung:**
 - Ist ein Pilz essbar oder nicht?
 - Wird ein Bewerber erfolgreich sein?
 - Ist ein Tumor gutartig oder bösartig?
 - Hat eine Person eine Krankheit oder nicht?

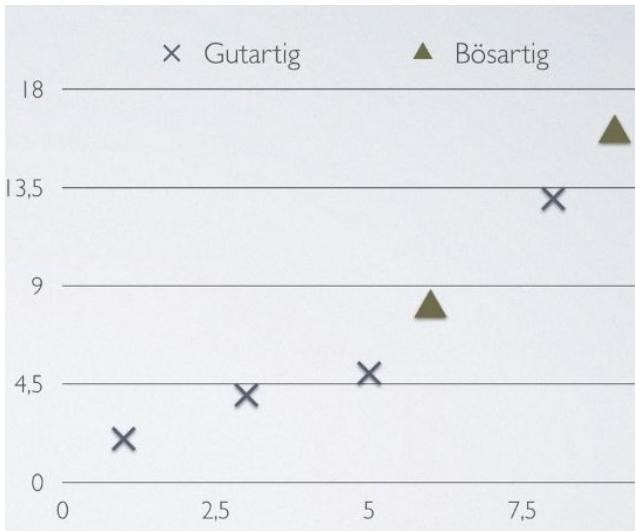
7.1 Allgemeines zur Klassifizierung

- **Beispiele für Klassifizierung mit mehreren Kategorien (nicht binär):**
 - Was sieht man auf dem Bild: Ein Schuh, ein T-Shirt oder eine Hose?
 - Was für eine Ziffer wurde geschrieben: 0, 1, ..., 9 ?
 - Ist ein Tumor gutartig oder bösartig?
 - Hat eine Person eine Krankheit oder nicht?

7.1 Allgemeines zur Klassifizierung

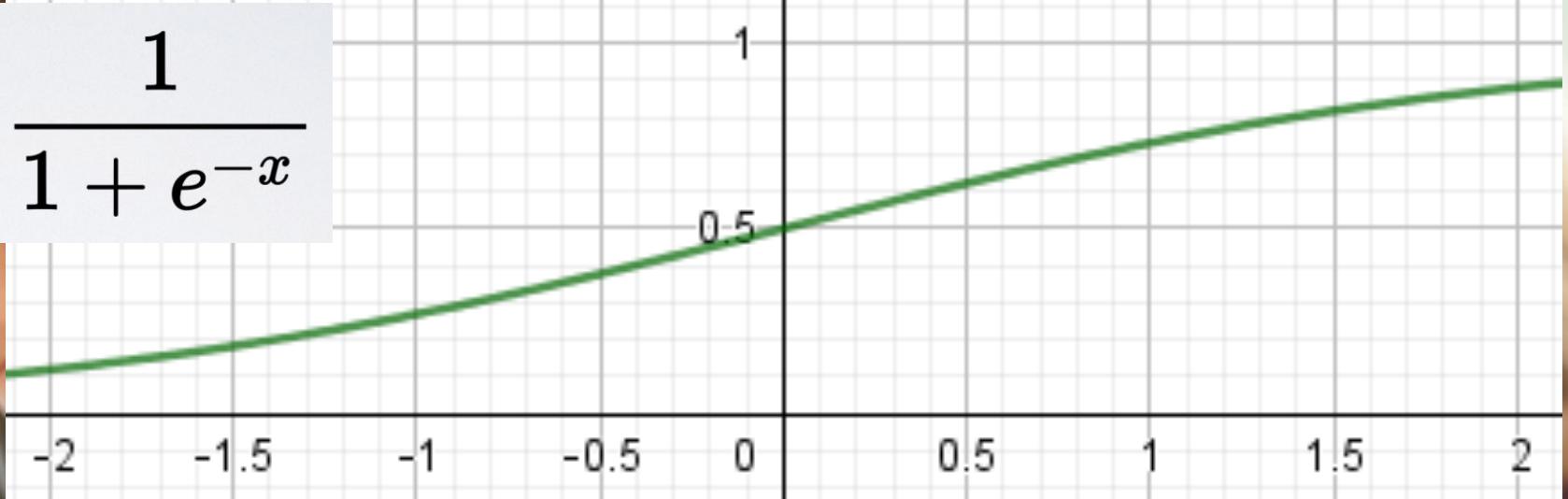
- **Was wollen wir erreichen?**

- Wir trainieren ein Modell mit bekannten Daten.
Man nennt dieses Prinzip auch
Supervised Learning
- Anschließend kann unser Modell unbekannte
Daten klassifizieren



8. Klassifizierung mit Logistischer Regression

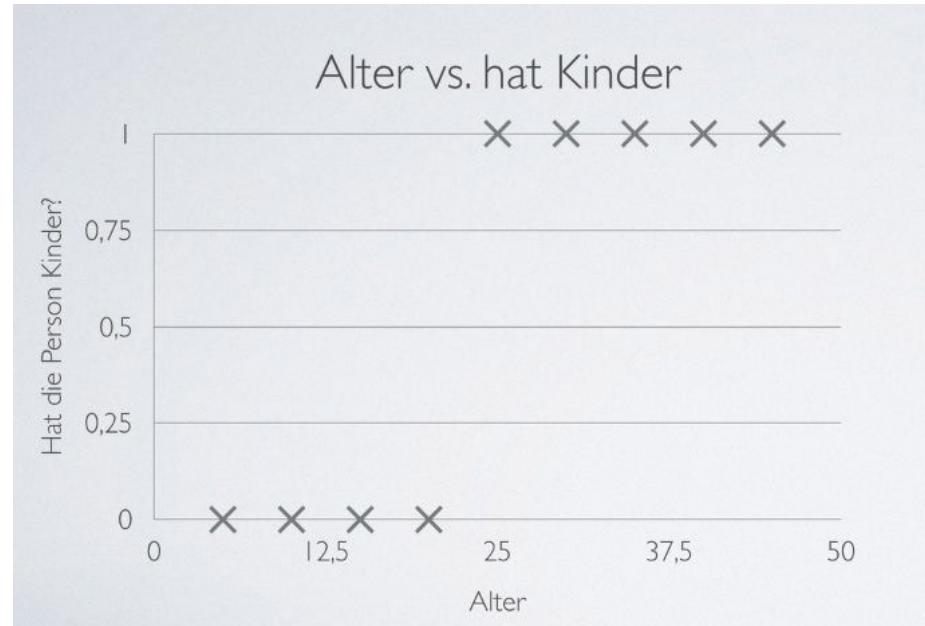
$$f(x) = \frac{1}{1 + e^{-x}}$$



8.1 Allgemeines zur Logistischen Regression

- Die Logistische Regression ist ein Verfahren zur **binären Klassifizierung**
- **Beispielfrage:** Haben Menschen in Abhängigkeit vom Alter selber Kinder?
 - Es ist eine Frage, die binär entscheidbar ist.
 - **Die Logistische Regression verlangt numerische Daten.** Wir definieren:
0: nein 1: ja

- **Visualisierung:**



8.1 Allgemeines zur Logistischen Regression

?

- **Gedankenexperiment:** Man könnte die Trennung mit einer Geraden $f(x) = a * x + b$ realisieren.....



-90 jährige hätten dann eine 200%-ige Wahrscheinlichkeit Kinder zu haben, was die Realität nicht sinnvoll abbildet.

8.1 Allgemeines zur Logistischen Regression

- **Verbesserungsversuch:**



-Dieser Ansatz ist aber unpraktisch, da das Modell davon ausgeht, ein 5-jähriger mit gleicher Wahrscheinlichkeit Vater wird wie ein 18-jähriger.

8.1 Allgemeines zur Logistischen Regression

- Verbesserungsversuch II:



-Dieser Ansatz ist aber unpraktisch, da man mehrere Geradengleichungen und Intervalldefinitionen bräuchte.

8.1 Allgemeines zur Logistischen Regression

- **Ultimative Verbesserung:** Eine Funktion mit etwa diesem Aussehen:

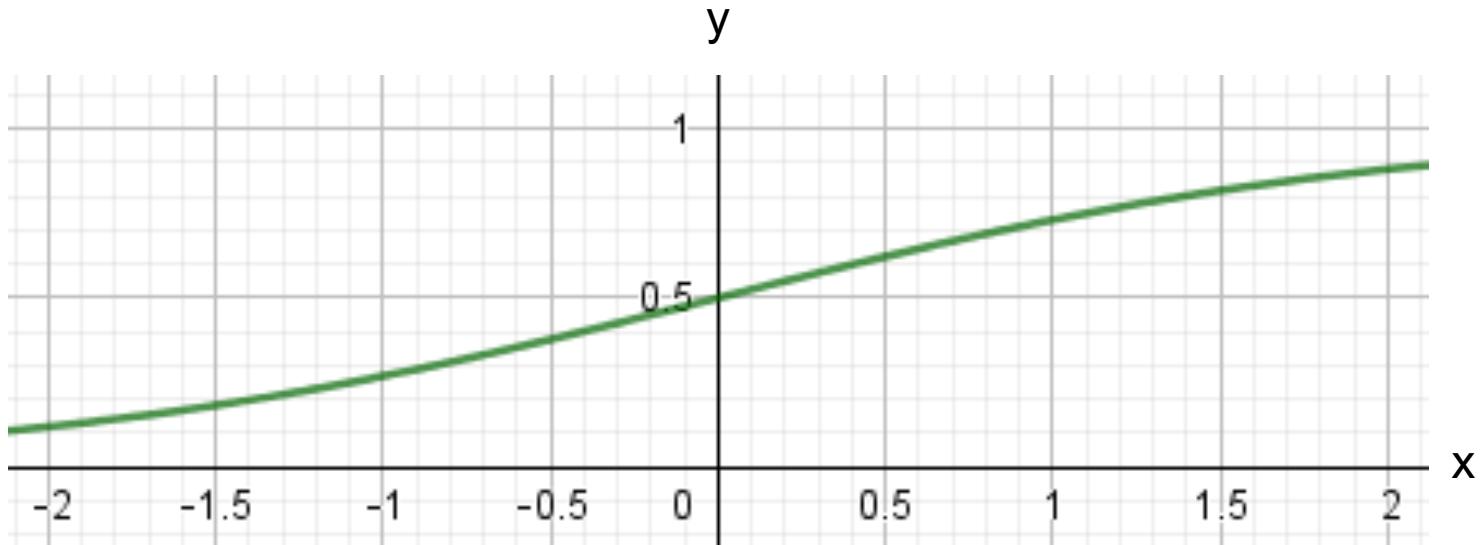


-die **Sigmoid-Funktion** ist solch eine Funktion!

8.1 Allgemeines zur Logistischen Regression

- Die **Sigmoid-Funktion**

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



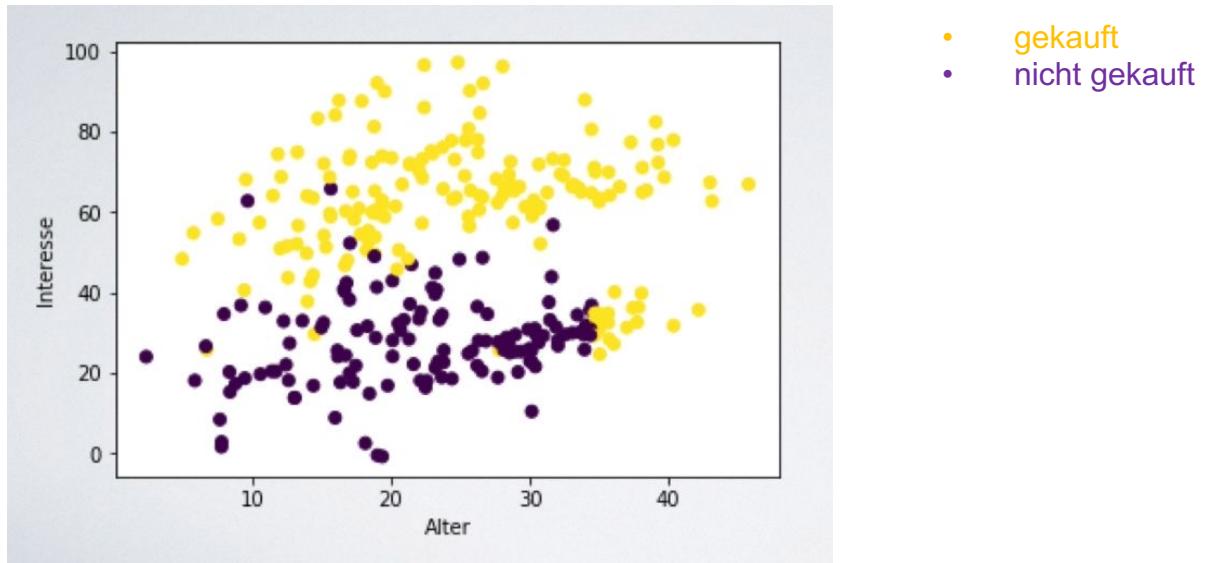
- **GeoGebra-Datei:** [SigmoidDemo.ggb](#)
- **Parametrierung:** Ersetzt man x mit $x' = a x + b$ kann über **a** und **b** der Verlauf **flacher / steiler** bzw. **nach vorne / nach hinten** geschoben werden

8.2 Beispiel zur Logistischen Regression

- **Beispiel: Wir starten eine Kickstarter-Kampagne**
 - Wir haben eine Sportkamera entwickelt, die 3D-Videos aufnimmt
 - Wir haben Daten über ersten 891 Kundenregistrierungen auf unserer Web-Plattform
 - Datei: **classification.csv**
Python-Demo: [BeispieldatenAnschauen.ipynb](#)
 - **Python-Demo:** [LogistischeRegressionDurchführen.ipynb](#)
 - **Über die Kunden wissen wir:**
 - **Alter**
 - **Interesse** an Sport (mit der Registrierung werden Fragen gestellt)
 - Ein Teil der Kunden hat dann tatsächlich **gekauft**, der andere Teil hat **nicht gekauft**.
 - Frage: **Wer kauft, wer kauft nicht?**

8.2 Beispiel zur Logistischen Regression

- Visualisierung der Daten:



- Scaling:
 - Sigmoid-Funktion braucht kleine Werte um 0
 - Scaling formal: Standardabw.=1 Durchschn.=0
 - Effekt: Werte zwischen ca. -3/+3

```
scaler = StandardScaler()  
scaler.fit(X_train)  
X_train = scaler.transform(X_train)  
X_test = scaler.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression  
model = LogisticRegression()  
model.fit(X_train, y_train)
```

- Logistische Regression
 - Durchführen

8.3 Der Standard-Scaler

– Beispiel zum Scaler:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
arr = [[0.1, 500, 2, 30, 0],
       [0.02, -500, 2, -30, 0],
       [-0.1, -100, 10, 0, 0],
       [-0.02, 100, 2, 0, 0],
       ]
scaler.fit(arr)
scaler.transform(arr)
```

Standartscaler berücksichtigt Streuung = 1, sodass breite Streuungen ebenfalls herunterskaliert gut ersichtlich sind

Scaling:

- Durchschnitt = 0
- Standardabweichung = 1

```
array([[ 1.38675049,  1.38675049, -0.57735027,  1.41421356,  0.        ],
       [ 0.2773501 , -1.38675049, -0.57735027, -1.41421356,  0.        ],
       [-1.38675049, -0.2773501 ,  1.73205081,  0.        ,  0.        ],
       [-0.2773501 ,  0.2773501 , -0.57735027,  0.        ,  0.        ]])
```

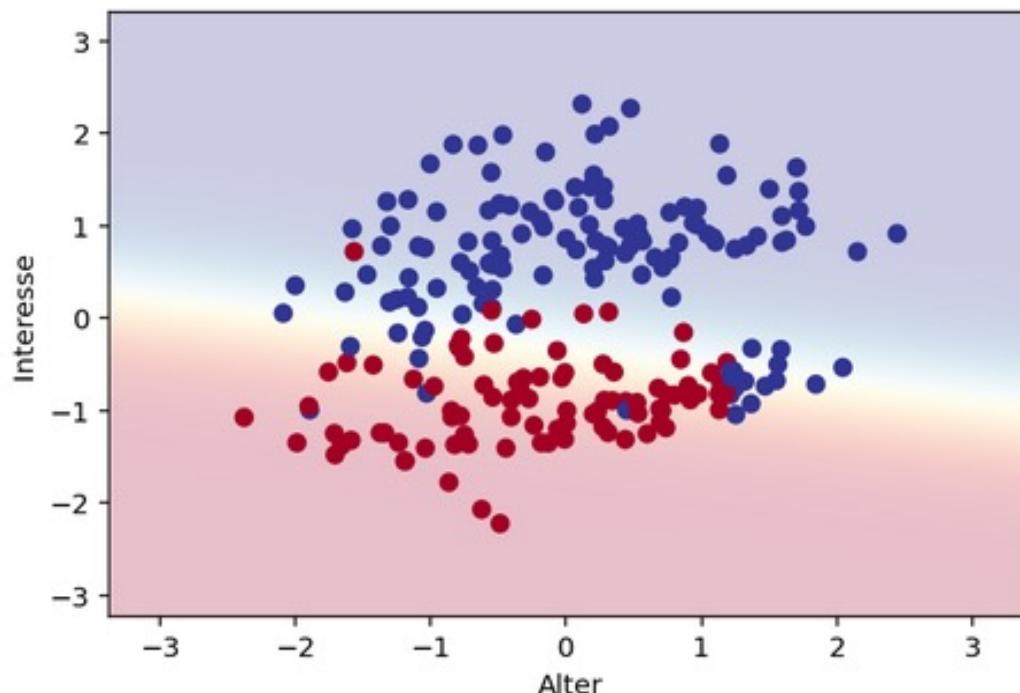
Scaling

4. Spalte:	Ergebnis:	Standardabw:
30	1.41421356	$ 1.41421356 ^2$
-30	-1.41421356	$ -1.41421356 ^2$
0	0	$ 0 ^2$
0	0	$ 0 ^2$
	0	
		4 : 4 = 1

Skalierung um 0 (0 Symmetrisch)

8.4 Entscheidungsgrenze in der Logistischen Regression

- **Entscheidungsgrenze:** Den Verlauf der Sigmoid-Funktion visualisieren.
 - Datei: [classification.csv](#)
 - Datei: [helper.py](#) – Macht die eigentliche Visualisierung
 - **Python-Demo:** [LogistischeRegressionEntscheidungsgrenze.ipynb](#)



8.5 Aufgabe zur Logistischen Regression

- **Beispiel: Diagnose von Brustkrebs**
 - Quelle: <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>
 - **Datei:** `cancer.csv`
Ausgangsbasis: `AufgabeBrustkrebs.ipynb` / `KlassifizierungsTemplate.ipynb`
 - **Muss vorhanden sein:** `helper.py`
 - **Viele Spalten:** `df.columns` zeigt, dass der Datensatz 32 Spalten hat:
 - Hinweise:
 - Spalten löschen geht so:

```
df = df.drop("id", axis = 1) # axis = 1 bedeutet "Spalte", axis = 0 wäre "Zeile"
```
 - `X` hat alle Spalten außer „`id`“ und „`diagnosis`“
 - `y` hat nur die Spalte „`diagnosis`“
 - Musterlösung: `BrustkrebsMusterlösung.ipynp`

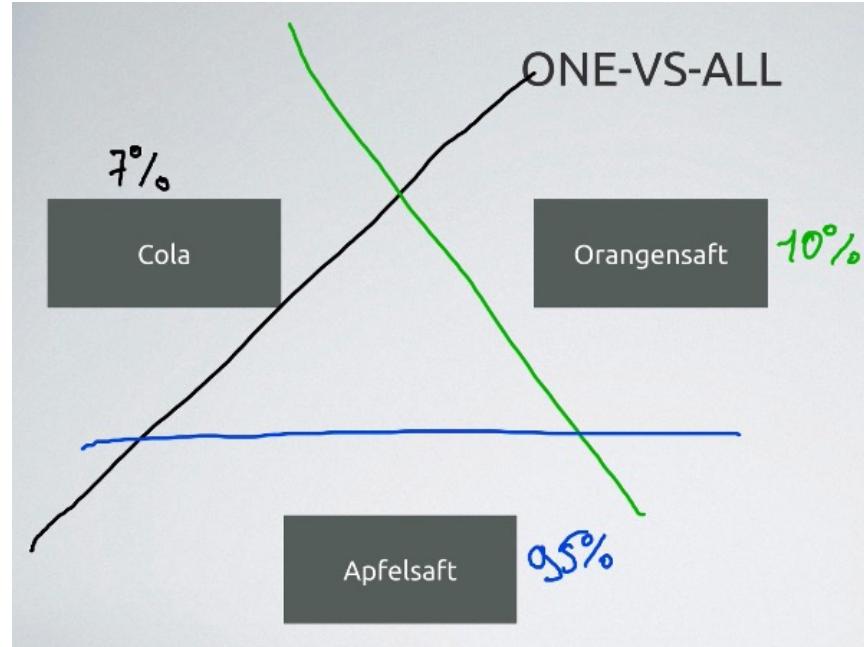
8.6 Logistische Regression mit mehreren Klassen

- **Beispiel:** Ein Patient kommt in eine Arztpraxis
 - Erkältung?
 - Grippe?
 - Bronchitis?
- **Problem:** Die Logistische Regression kann eigentlich nur eine binäre Unterscheidung treffen
 - Möglichkeit 1:
 - Z.B.: Erkältung vs. Keine Erkältung
 - Man braucht nur 3 Modelle, aber das Kann aber zu Unklarheiten führen: Erkältung: ja, Grippe: ja Bronchitis: ja (unklar was genau, oder alles gleichzeitig?)
 -Dieser Ansatz ist aber unpraktisch, da man mehrere Geradengleichungen und Intervalldefinitionen bräuchte.



8.6 Logistische Regression mit mehreren Klassen

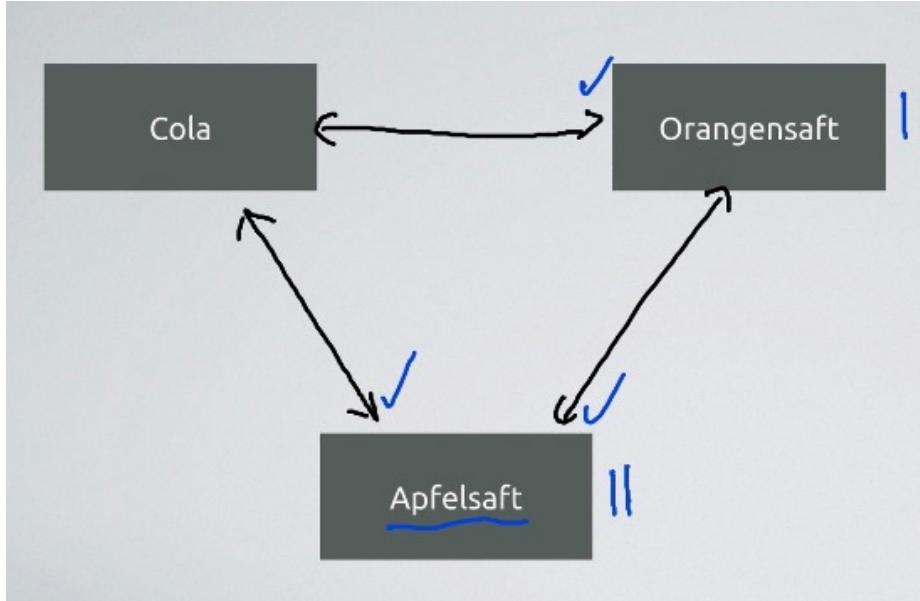
- Möglichkeit 2: One vs All



- Es gewinnt die Klasse mit dem höchsten Wert: *Apfelsaft* 95%
- Die Linien verkörpern Modelle Man braucht so viele Modelle wie es Klassen gibt
- Man muss immer mit allen vorhandenen Daten trainieren

8.6 Logistische Regression mit mehreren Klassen

- Möglichkeit 3: One vs One



- Es gewinnt die Klasse mit den meisten Ergebnissen > 50%
Apfelsaft (2 Stimmen) gewinnt! - Orangensaft (1 Stimme) und Cola (0 Stimmen)
- Man braucht mehr Modelle, muss diese aber mit weniger Daten füttern.
Das Modell Apfelsaft vs. Orangensaft braucht z.B. keine Cola-Beispiele.

8.7 Vergleich One vs. One mit One vs. All

- Gegeben seien **K Klassen**
 - **One vs All:**
 - benötigt **K Modelle**
 - Weniger Modelle notwendig, wenn viele Klassen gegeben sind.
 - **One vs One:**
 - benötigt **$K*(K-1) / 2$ Modelle**
 - Problematisch bei vielen Klassen (**$K*K!!!$**)
 - Geeignet für große Datenmengen weil die Modelle jeweils nur Anteile der Daten benötigen.

8.7 Vergleich One vs. One mit One vs. All

- Beispiel zu One vs All / One vs. One
 - Quelle: Open Food Facts
 - Datei: [foods.csv](#)
 - Python-Demo: [OneVsAll-OneVsOne.ipynb](#)

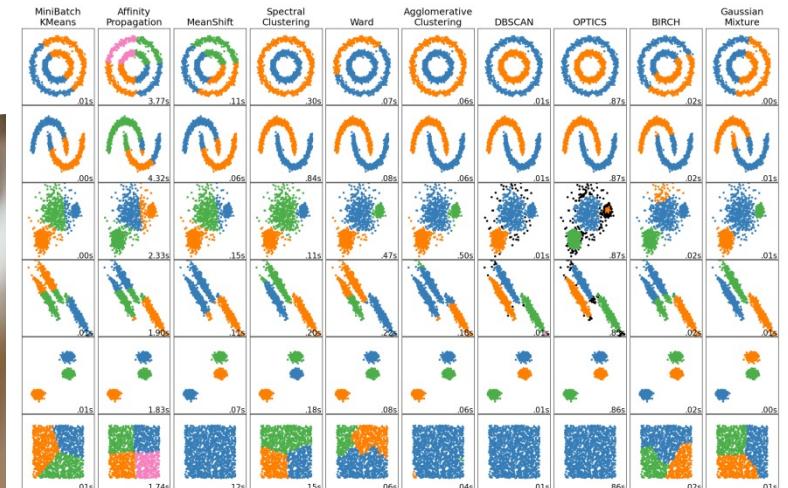
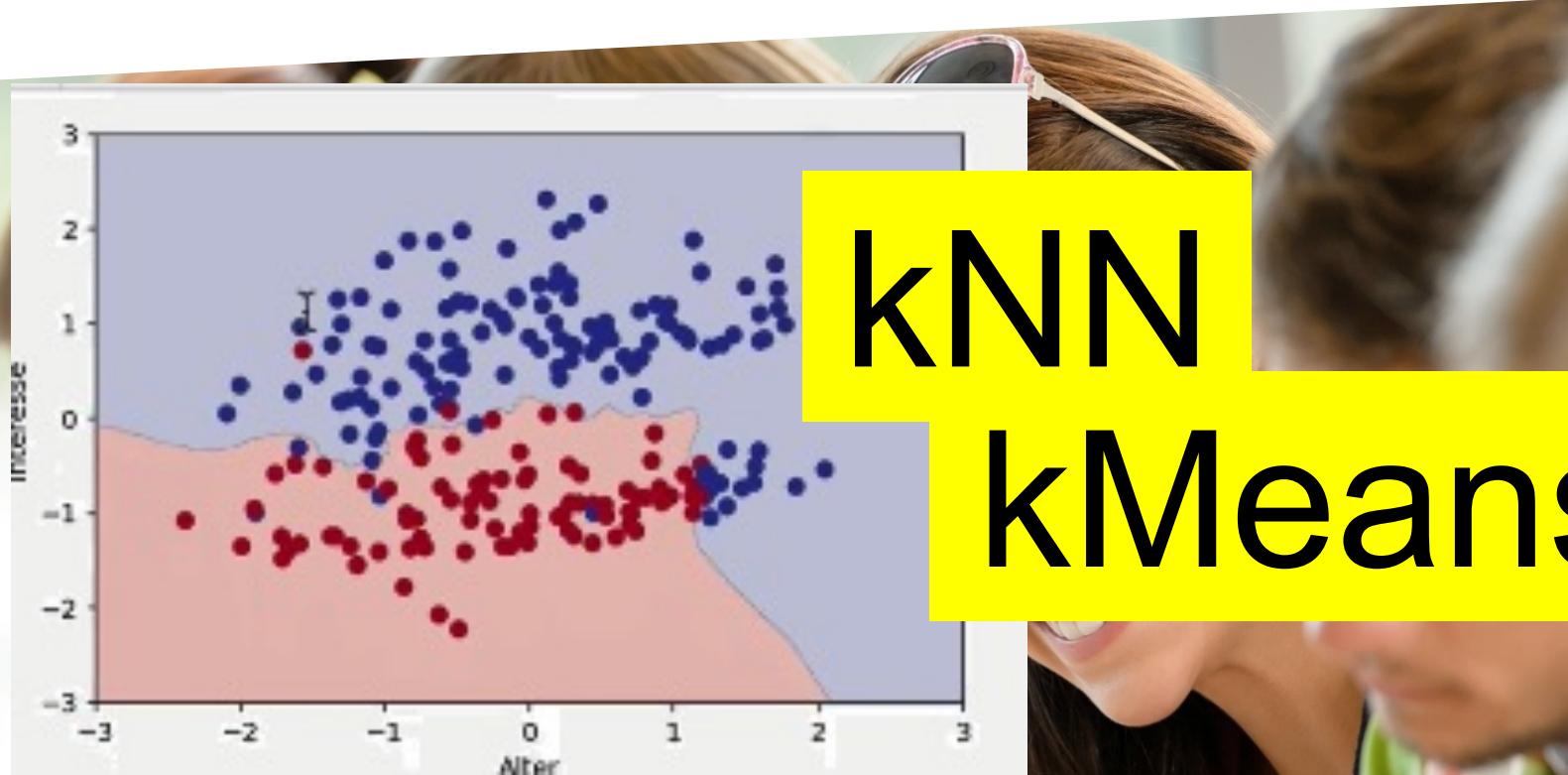
- One vs All:

```
model = LogisticRegression(multi_class="ovr")
model.fit(X_train, y_train)
```

- One vs One:

```
from sklearn.multiclass import OneVsOneClassifier
model = OneVsOneClassifier(LogisticRegression())
model.fit(X_train, y_train)
```

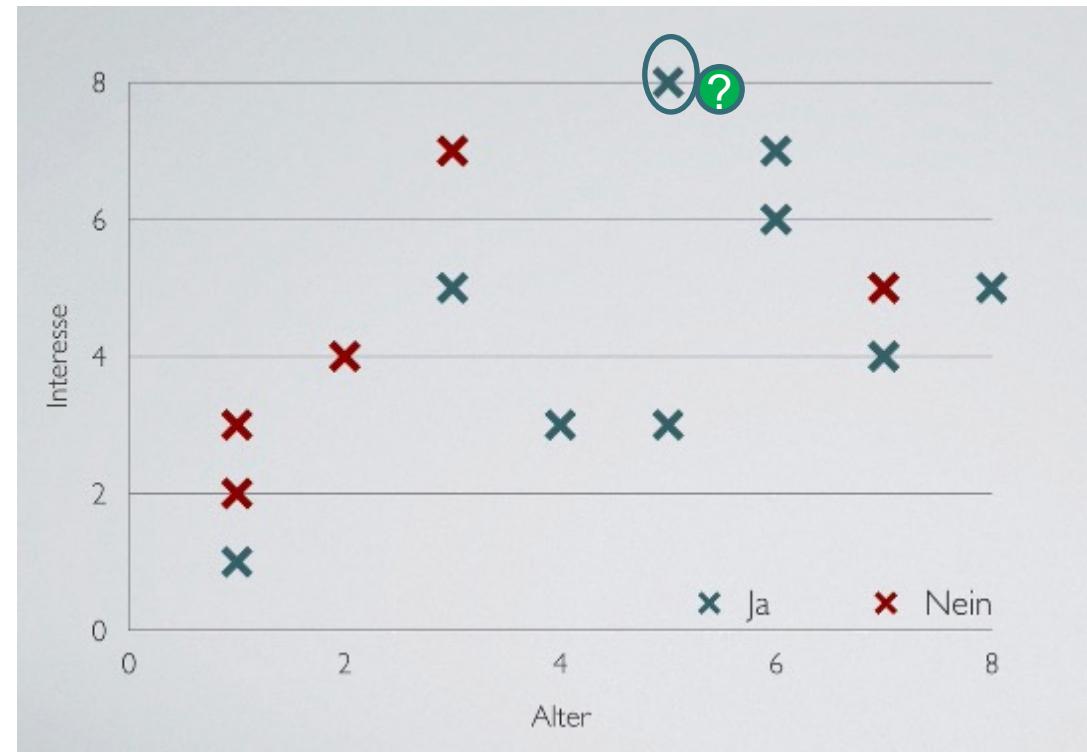
9. Clustering



$$d_E(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

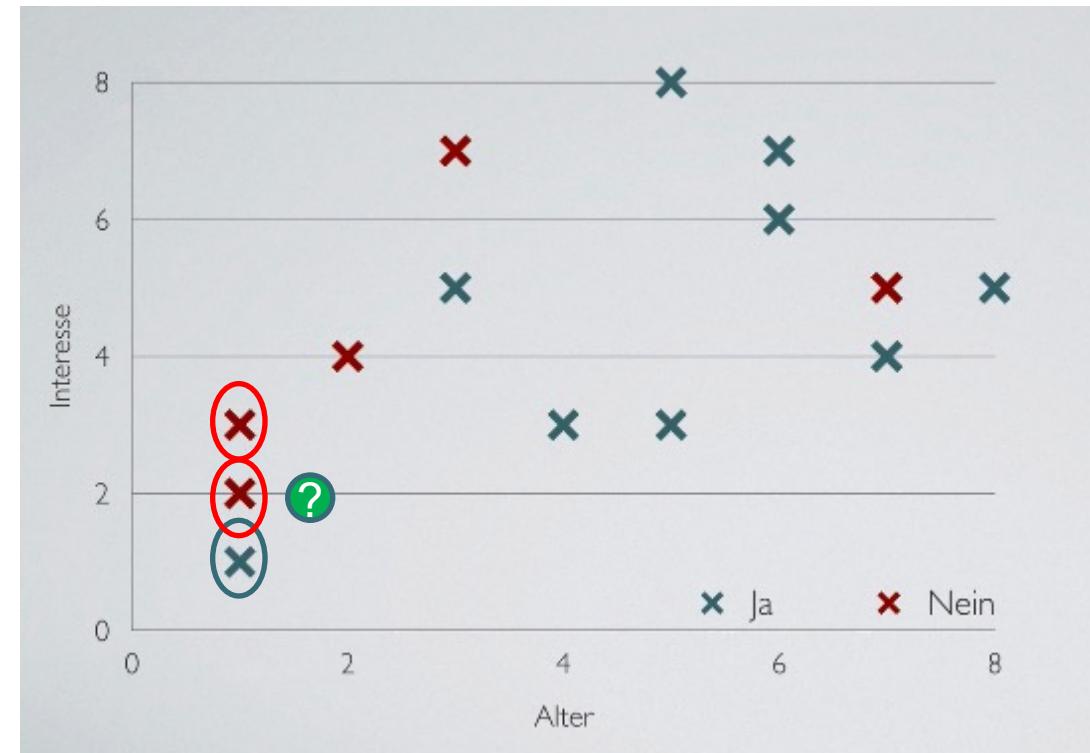
9.1 Clustering und Klassifizierung mit Nächste Nachbarn (kNN)

- **Idee:** Wir wollen Objekte einer Klasse zuordnen und orientieren uns an Objekten in der Nähe, die bereits einer Klasse zugeordnet sind.
- **Beispiel k = 1:**
 - Wir müssen nur einen, also «den nächsten Nachbarn» bestimmen und dessen Klasse (Label) übernehmen.
 - Der nächste Nachbar ist «blau», daher ist die gesuchte Klasse auch «blau»



9.1 Clustering und Klassifizierung mit Nächste Nachbarn (kNN)

- **Idee:** Wir wollen Objekte einer Klasse zuordnen und orientieren uns an Objekten in der Nähe, die bereits einer Klasse zugeordnet sind.
- **Beispiel k = 3:**
 - Wir müssen nur einen, also «3 nächste Nachbarn» bestimmen und dessen Klasse (Label) übernehmen.
 - Der nächsten Nachbarn sind «rot», «rot», «blau», also **mehrheitlich «rot»**, daher weisen wir der gesuchten Klasse «rot» zu.

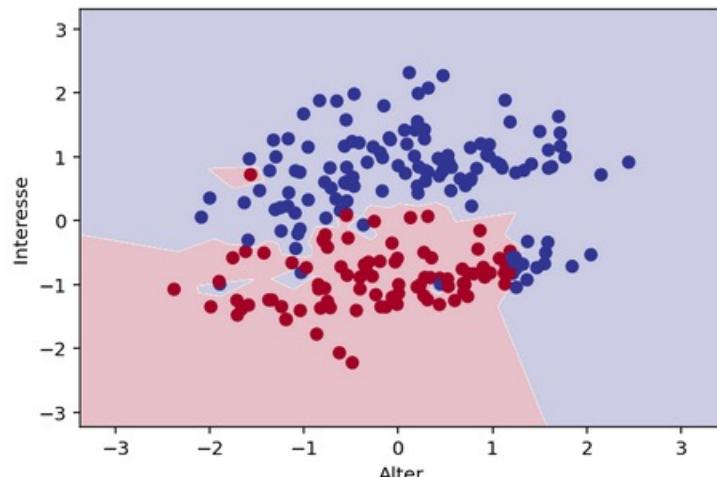


9.2 k-NN: Zur Wahl von k

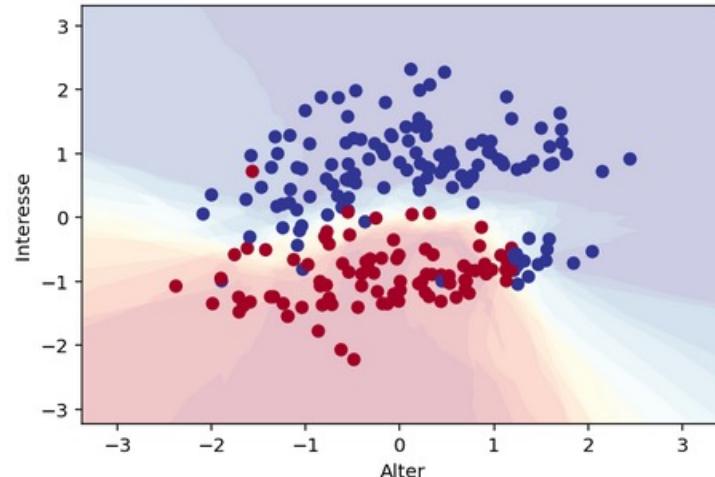
- Praktische Überlegungen:
 - **k sollte immer ungerade sein:** Damit bekommt man bei binären Klassierungen immer ein eindeutiges Ergebnis
 - **Wie große sollte k sein?**
 - **k soll so groß wie möglich** sein, damit der Fehler geringer wird
 - **K sollte klein sein,** damit wir uns nur auf nahe liegende Punkte beschränken, damit der Fehler geringer wird.
 - **Fazit:** Man **probiert** am besten dann verschiedene k-Werte aus...

9.3 Beispiel zur Klassifizierung mit k-NN mit unterschiedlichen k-Werten

- **Beispiel: Wir starten eine Kickstarter-Kampagne**
 - Wir haben eine Sportkamera entwickelt, die 3D-Videos aufnimmt
 - Wir haben Daten über ersten 891 Kundenregistrierungen auf unserer Web-Plattform
 - Wir untersuchen die Auswirkung von k . Wo liegt das Optimum?
 - Datei: [classification.csv](#)
- Python-Demo:** [K-NearestNeighbourAuswirkungVonK.ipynb](#)



$k=1$



$k=15$

9.4 Der Iris-Datensatz von 1936

- Wichtiges Standardbeispiel zum Umgang mit unscharfen Klassierungen
- Quelle: *Fisher R.A.:The Use of Multiple Measurements in Taxonomic Problems, 1936*
- Gegeben sind Daten von **150 Schwertlilien**, je 50 von **3 Arten**:



Iris setosa



Iris versicolor

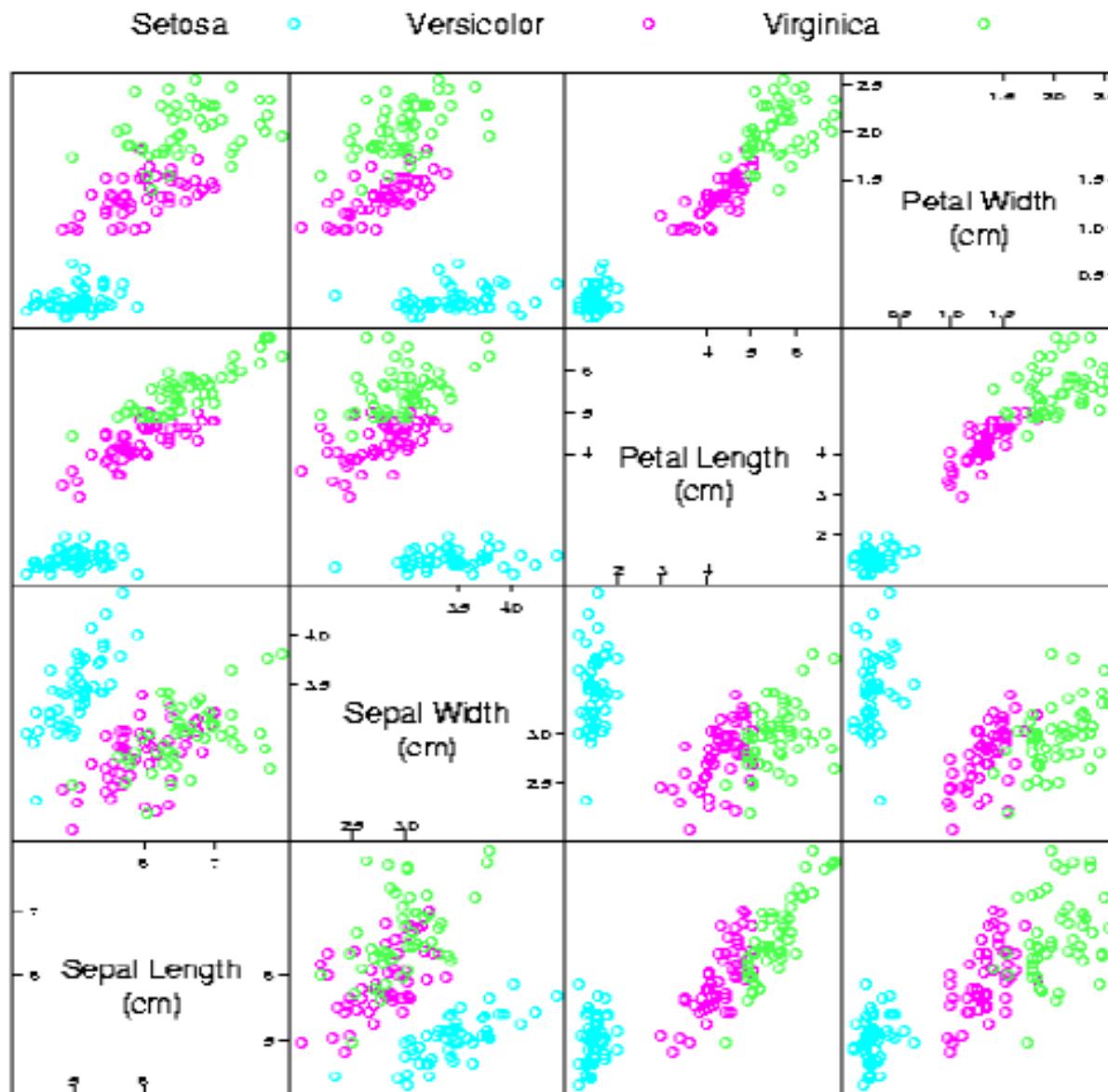


Iris virginica

- Spalten: Enthalten die Längen und Breiten von Sepal- und Petal-Blättern

9.4 Der Iris-Datensatz von 1936

- Scatterplot-Matrix zur Verdeutlichung:



9.5 Praxisprojekt: Vergleich kNN / Logistische Regression mit dem Iris-Datensatz

- **Aufgabe:**

Vorhersage der Spalte Species und Qualitätsvergleich verschiedener Algorithmen

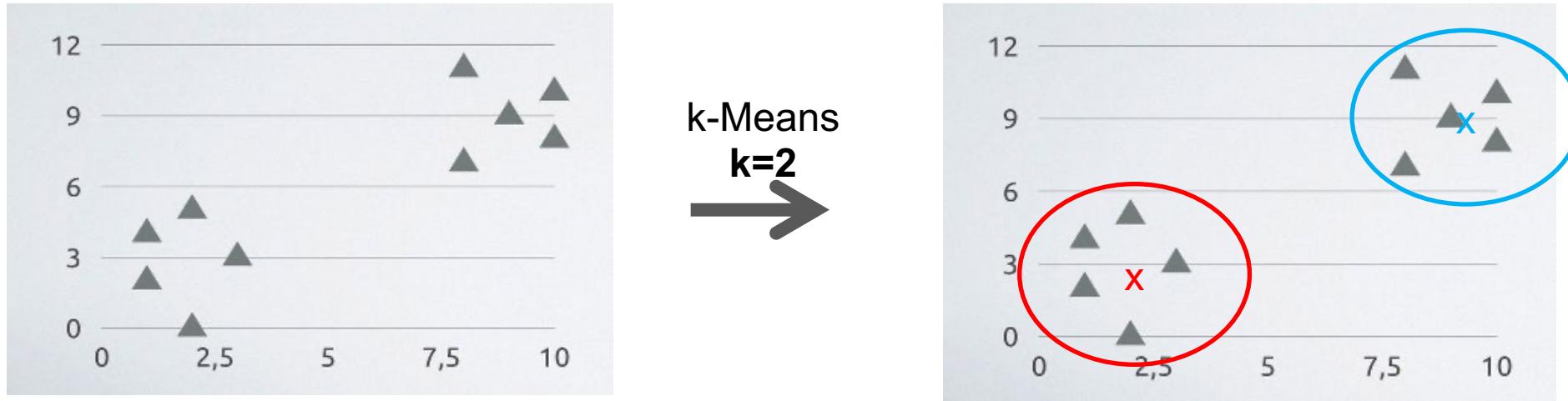
- Datei: [iris.csv](#)
Python-Datei: [IrisAufgabe.ipynb](#)
- Python-Datei: [KlassifizierungsTemplate.ipynb](#) (aus Kapitel 8)

- **R² - Ergebnisse:**
 - Logistische Regression ca. 97%
 - k Nächste Nachbarn: ca. 97%
 - Lineare Regression: ca. 88%

- Musterlösung: [IrisMusterlösung.ipynb](#)

9.6 K-Means-Clustering – Klassifizierung ohne Trainingsdaten

- **Problem:** Manchmal hat man beim Klassifizieren gar keine Trainingsdaten. Wir möchten wissen, ob es Cluster gibt, und wenn ja, wie viele und welche Struktur dort ist. Dies nennt dies auch ***Unsupervised-Learning*** oder auch ***Lazy-Learning***
- **Mögliche Lösung:** Der **k-Means** Clustering-Algorithmus
- **Idee von k-Means:**
 - Man muss zuvor festlegen, wie viele Cluster man finden möchte. **Beispiel:**



Wir Menschen sehen intuitiv, dass es sich um zwei Gruppen handelt. Also: **k=2**
Es ergibt sich das **K-Means-Clustermodell**: Cluster 1: (2,3|2,6) Cluster 2: (9,0|9,0)

9.6 K-Means-Clustering – Klassifizierung ohne Trainingsdaten

- Zwischen einzelnen Datenpunkten eines Clusters gibt es eine ~~xe~~ Mitte. Diese kann z.B. das Arithmetische Mittel der Koordinaten sein.
Man spricht hier auch von **Zentroiden** oder **Means**.

bei hinzufügen von neuen Datenpunkten "wandert" der Zentroid

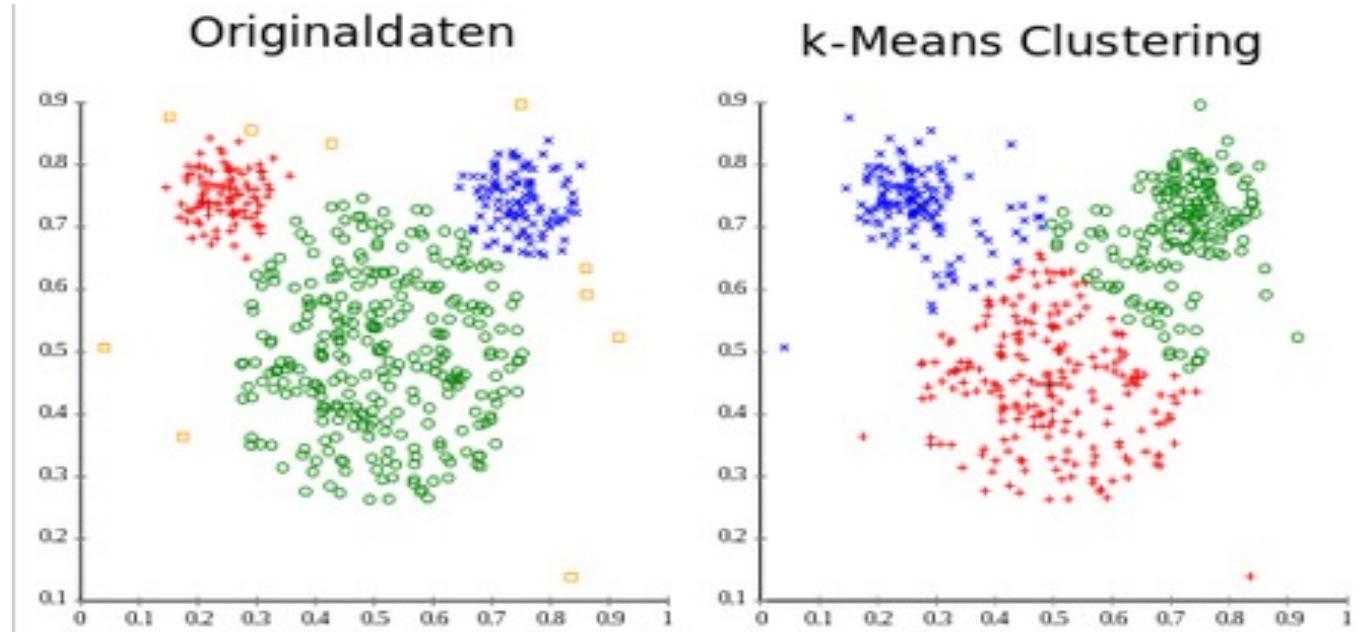
- **Neue Datenpunkte:** Die Zugehörigkeit eines neuen Datenpunktes \vec{p} zu Cluster 1 oder Cluster 2 wird so ermittelt: Bestimme die Entfernung $d_E(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$ zu den Mittelpunkten (Means) der Cluster.
Entfernung z.B. mit der **Euklid'schen Distanz**:

- Ordne x dem nächstgelegenen Cluster i zu und berechne den Mittelpunkt für i neu. Usw.
- Beispiel mit Animation zur Veranschaulichung:

<https://towardsdatascience.com/log-book-guide-to-distance-measuring-approaches-for-k-means-clustering-f137807e8e21>

9.6 K-Means-Clustering – Klassifizierung ohne Trainingsdaten

- **Problem des k-Means:** K-Means funktioniert nur gut, wenn die Cluster räumlich gut getrennt sind und ähnlich gestreut sind und die Streuung kugelförmig ist.
- Hierzu ein Beispiel, was passiert, wenn dies nicht der Fall ist:



Wie man sieht, kommt es im Grenzbereich der Cluster zu Falschzuweisungen aufgrund der Entfernungseinschätzung.

- **Verbesserungspotenzial:**
 - Es gibt viele **verschiedene Entfernungsmße** mit ihren Vor- und Nachteilen.
 - Es gibt sehr viele **verschiedene Clustering-Algorithmen** (s. übernächste Folie)

9.6 K-Means-Clustering – Klassifizierung ohne Trainingsdaten

- Entfernungsmaße für K-Means:

- Euklid'sche-Distanz:

$$d_E(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Ziemlich gängig. Einfach und intuitiv.

Nachteile: Ausreißer haben starken Einfluss.
Keine Berücksichtigung der Streuung

- Minkowski-Distanz
auch bez. als
Average Euclidean:

$$d(\vec{p}, \vec{q}) = \left(\sum_{i=1}^n (p_i - q_i)^r \right)^{1/r}$$

Vorteil: Besser als die Euklid'sche Distanz, die im Falle von $r=2$ sogar enthalten ist. Ausreißer können hier untergewichtet werden.

Nachteile: Redundante Daten ziehen den Mittelpunkt.

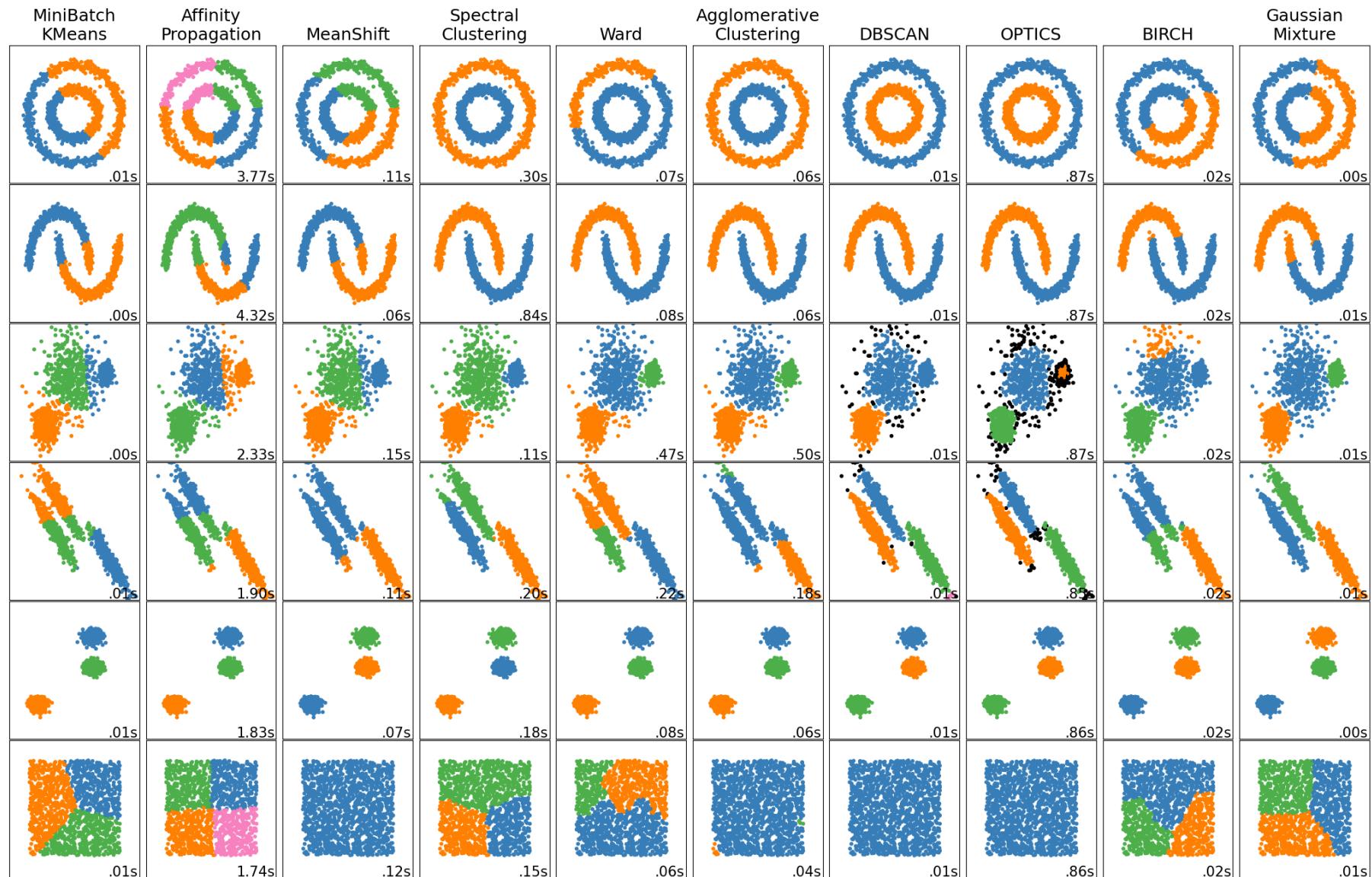
- Manhattan-Distanz:

$$d_M(\vec{p}, \vec{q}) = \sum_{i=1}^n |p_i - q_i|$$

Vorteil: Sehr einfach und performant und etwas robuster gegen Ausreißer.

Nachteile: Ausreißer haben immer noch Einfluss.

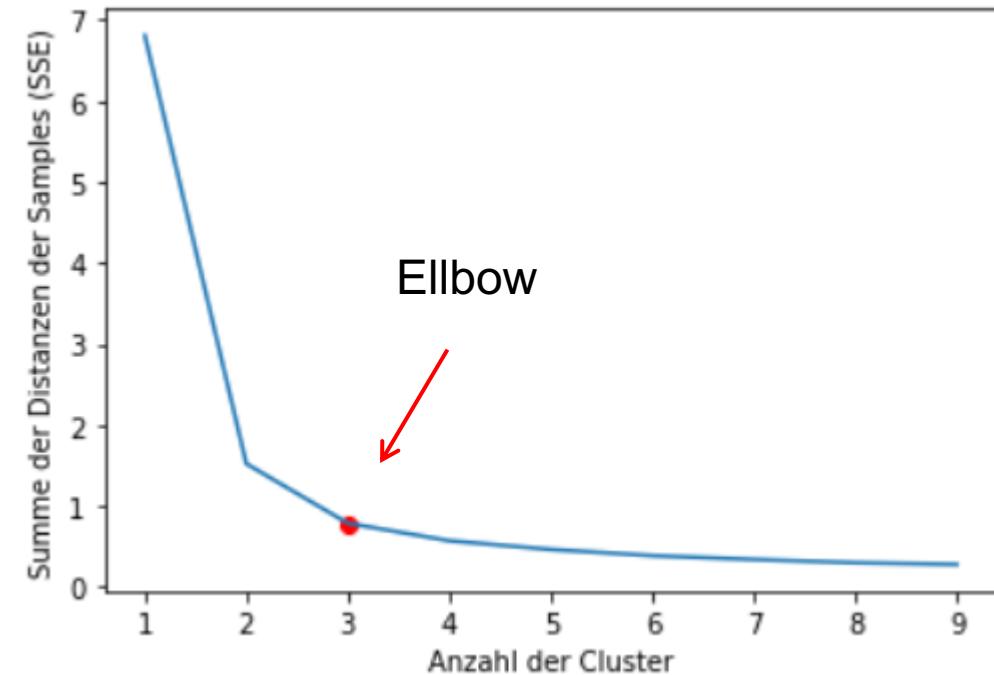
9.7 Verschiedene-Clustering-Algorithmen mit ihren Stärken und Schwächen



Quelle: https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

9.8 Unsupervised-Learning: Wie viele Cluster gibt es? - Gibt es eine Struktur?

- Das **Ellbow-Kriterium** zur Bestimmung der passenden Clusterzahl: Für uns Menschen reicht ein Blick und wir «sehen» auf einem wieviele Cluster vorhanden sind. In solchen Fällen bietet sich auch eine Automatisierung an:
 - Datei: [iris.csv](#)
 - Python-Datei:
[Clusterzahl-Ellbow-Kriterium.ipynb](#)
 - Idee: Man probiert mehrere Cluster-Anzahlen aus und schaut wie sich die Summe der Entfernungen zu den Zentroiden entwickelt. Dort wo die Kurve nur noch linear sinkt, ist der **Ellenbogen** und die passende Zahl
 - Das Beispiel zeigt die Iris-Daten, die mit k-Means mit $k=[1\dots 10]$ geclustert wurden. -> Offensichtlich gibt es drei Cluster.

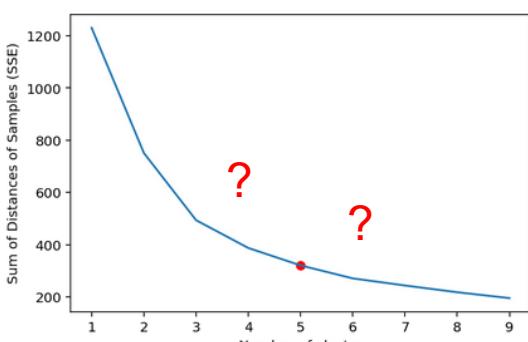
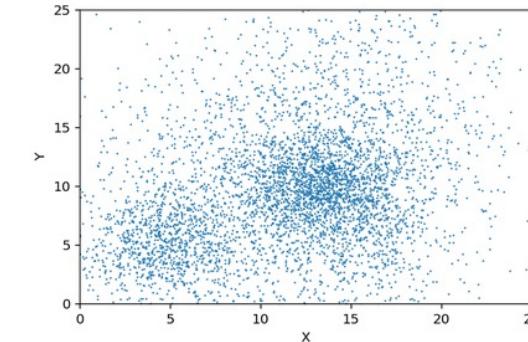
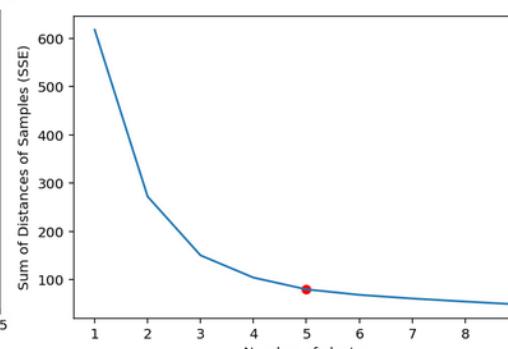
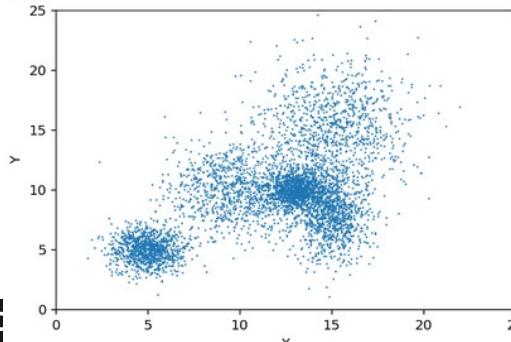
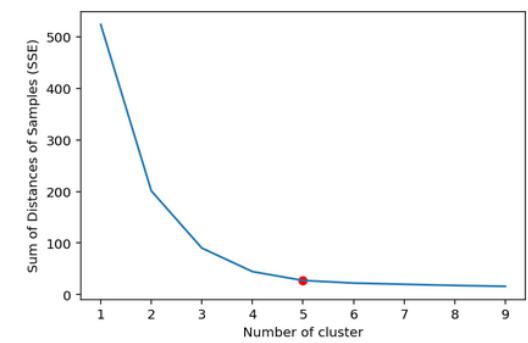
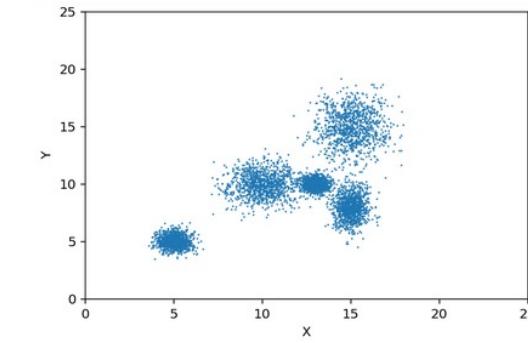
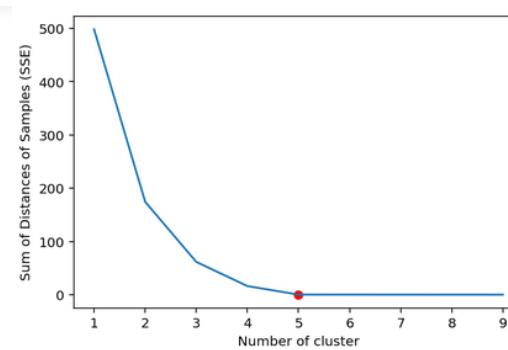
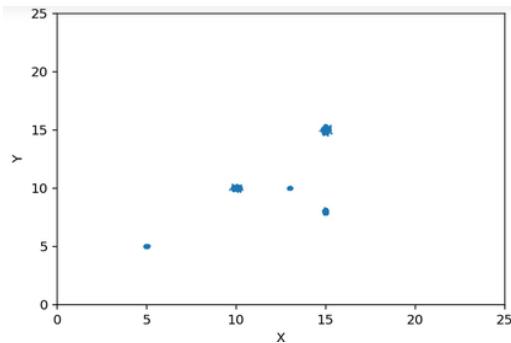


9.8 Unsupervised-Learning: Wie viele Cluster gibt es? - Gibt es eine Struktur?

- Das Ellbow-Kriterium in Abhängigkeit von der Verteilung:

- Python-Datei:

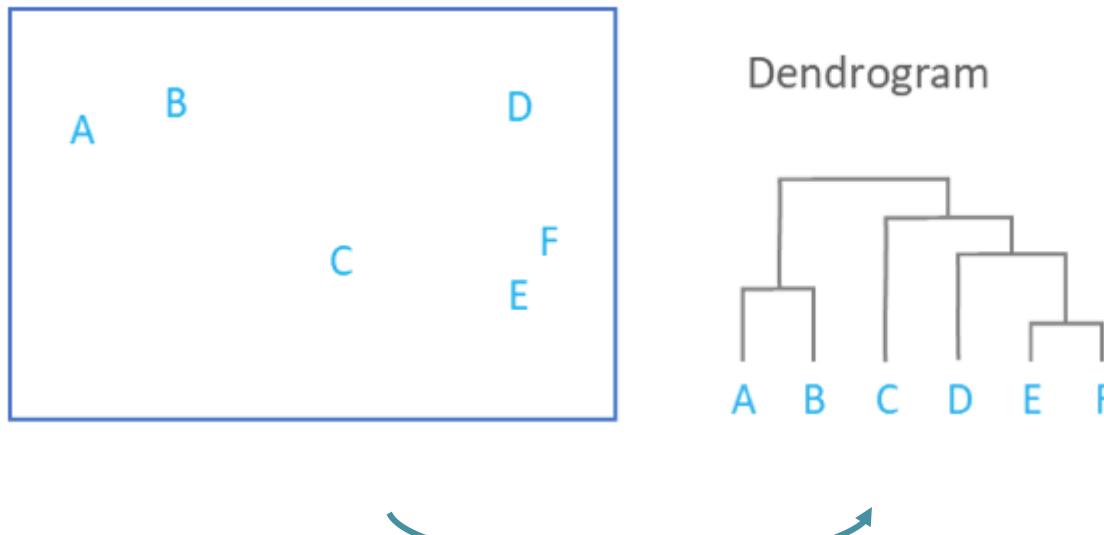
[Ellbow-Kriterium-Verteilungen.ipynb](#)



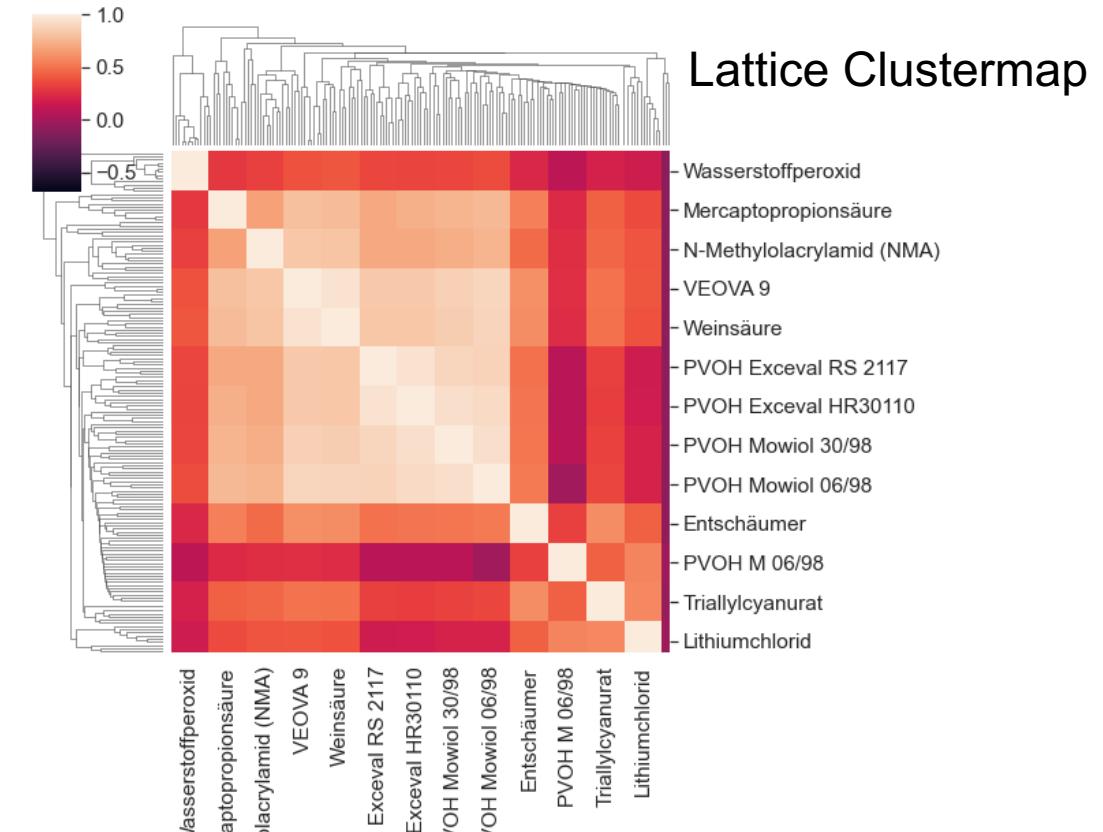
9.8 Unsupervised-Learning: Wie viele Cluster gibt es? - Gibt es eine Struktur?

- **Agglomeratives Clustering:** Die Frage nach der absoluten Anzahl von Clustern ist oft nicht die Lösung. Oft verbirgt sich eine Hierarchie hinter der Struktur der Cluster. Hier braucht es einen Algorithmus für das **Hierarchische Clustering**

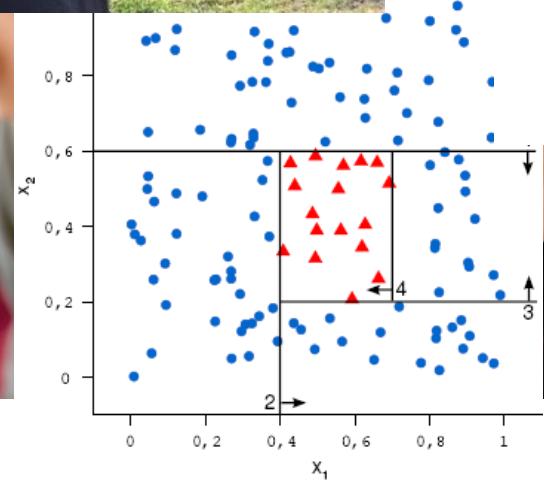
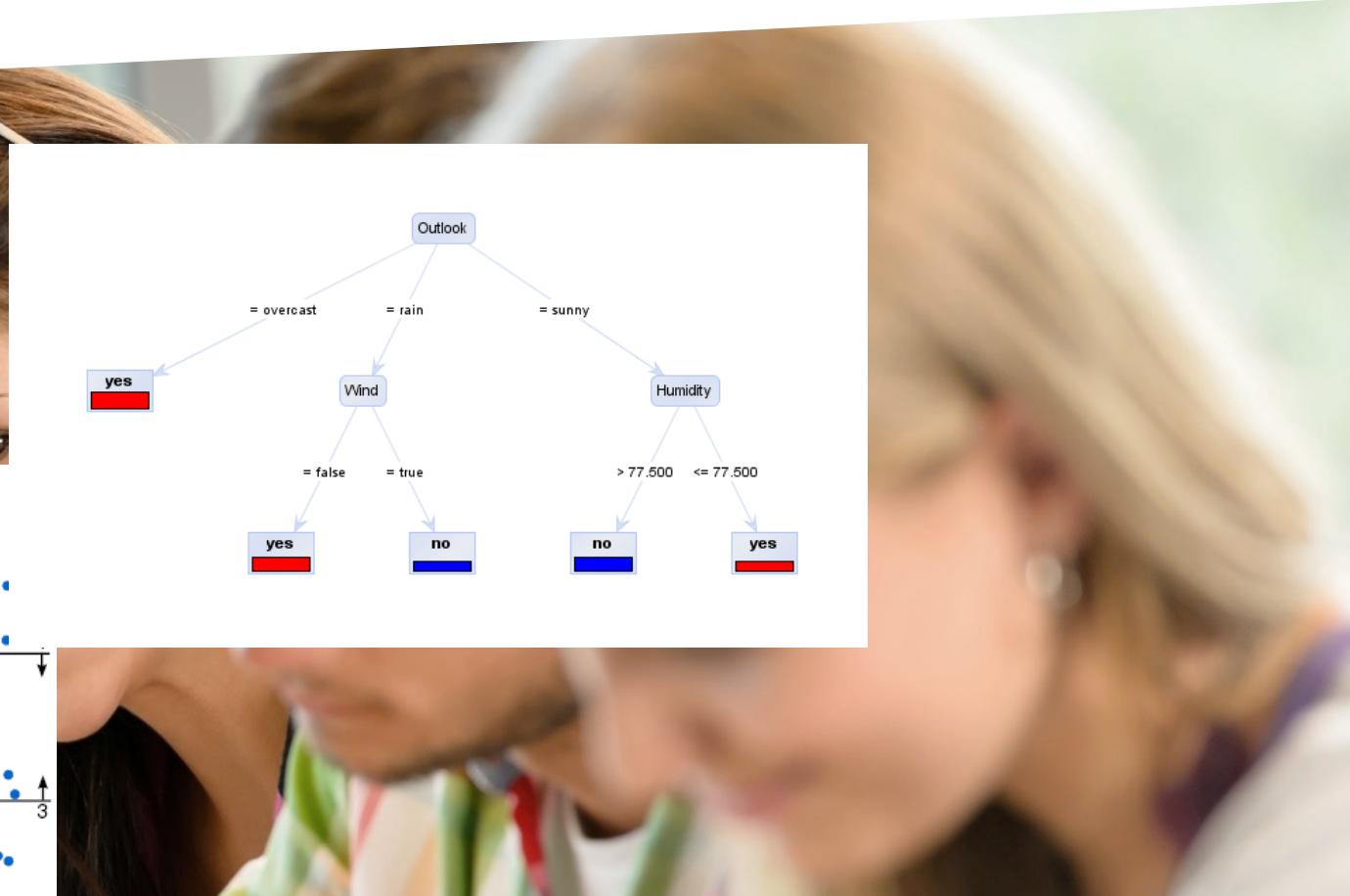
- Ein Dendrogramm ist das Ergebnis und zeigt die Struktur:



```
from sklearn.cluster import AgglomerativeClustering  
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')  
cluster.fit_predict(X)
```



10. Klassifizierung mit Entscheidungsbäumen



10.1 Zum Begriff der Entropie

- **Entropie in der Physik / Thermodynamik:** Ein System aus einem kalten und einem heißen Körper in einem praktisch abgeschlossenen System wird durch Wärmetransport den Temperaturunterschied ausgleichen. Die Entropie ist ansteigend und das Ganze ist irreversibel. Niemals wird in einem geschlossenen System sich der kältere Körper abkühlen, und den wärmeren Körper erhitzen.
- **Entropie In der Informatik / Informationswissenschaft:**
 - Wie viel Information ist enthalten?
 - Anschaulich: Entropie ist das Gegenteil von Ordnung

10.1 Zum Begriff der Entropie



Delivery: Lasagne al Forno - Bild vo...
tripadvisor.de

- **Bspiel:**
 - Lieferdienst ruft 1 x pro Woche an: „Welches Essen möchtest Du haben?“
 - Antwort: „Lasagne“
 - Wenn das 1 Jahr so geht: **Lernt der Lieferdienst mit jedem Anruf etwas dazu?**
 - **Nein:**
 - Der Lieferdienst weiß ja vorher schon, dass Du eh „Lasagne“ bestellst!
 - **Entropie / Informationsgehalt: 0**

10.1 Zum Begriff der Entropie



- **Bsp 2:**
 - Lieferdienst ruft 1 x pro Woche an: „Welches Essen möchtest Du haben?“
 - Antwort: [...immer was anderes...]
 - Wenn das 1 Jahr so geht: **Lernt der Lieferdienst mit jedem Anruf etwas dazu?**
 - **Ja:**
 - Jeder Anruf ist eine „wichtige“ Information und alle 365 sind maximal ungeordnet.
 - **Entropie / Informationsgehalt: 1**

10.1 Zum Begriff der Entropie



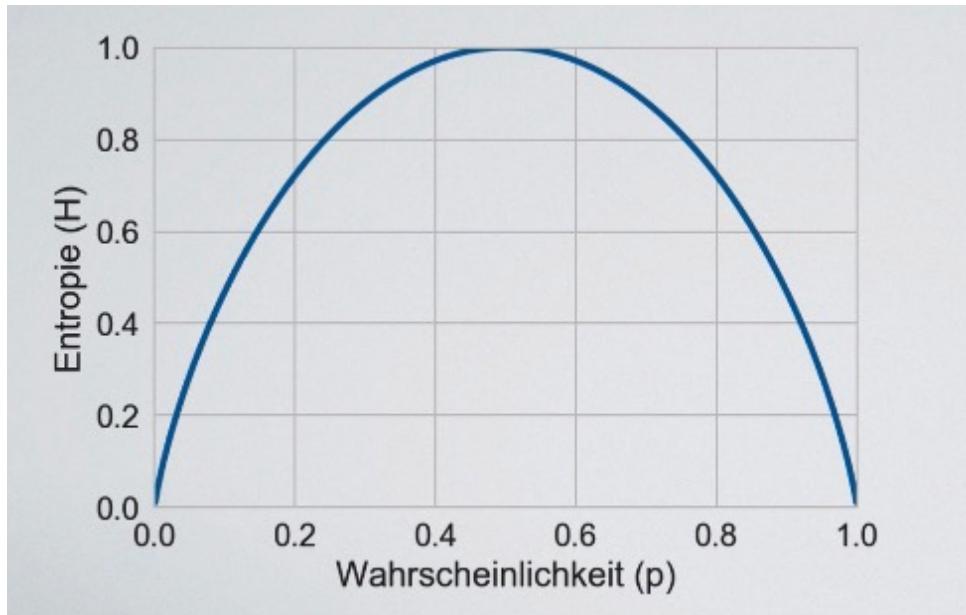
- **Bespiel3: Münzwurf**
 - **Münzwurf** Kopf oder Zahl?
 - Es kann jetzt sein, die Münze ist „gezinkt“ und das Ergebnis ist immer „Kopf“
 - Informationsgehalt / Entropie = 0
 - Es kann jetzt sein, die Münze ist „nicht gezinkt“ und das Ergebnis ist 50% so oder so
 - Informationsgehalt / Entropie = 1

10.1 Zum Begriff der Entropie

- Formel der Entropie bei einem Zufallsexperiment:

$$H = -p \cdot \log_2 p - (1 - p) \cdot \log_2 (1 - p)$$

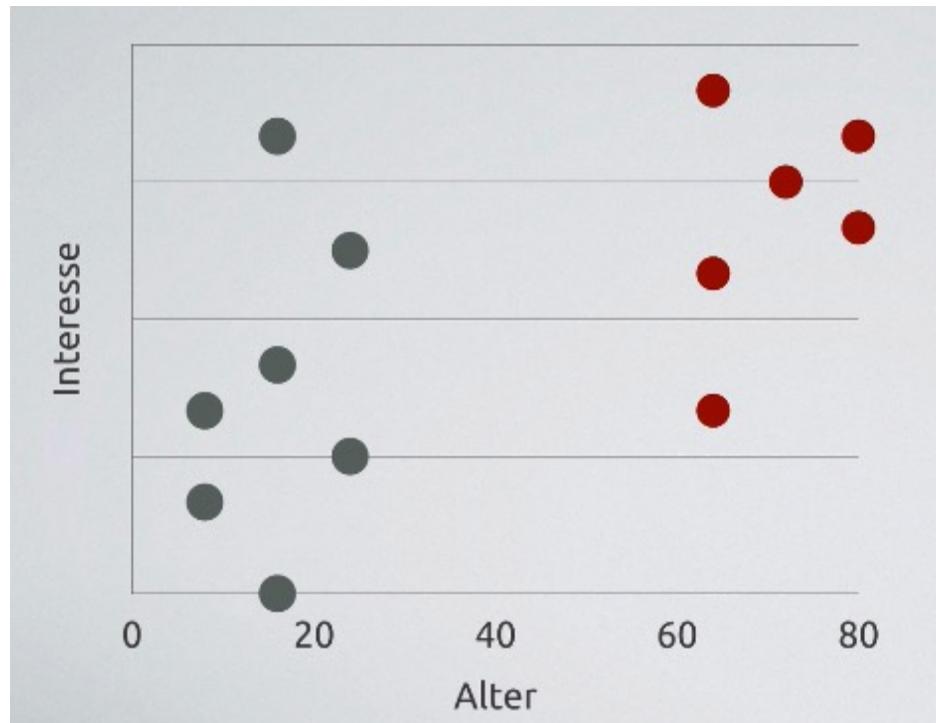
- Verlauf der Entropie in Abhängigkeit zu Wahrscheinlichkeit:



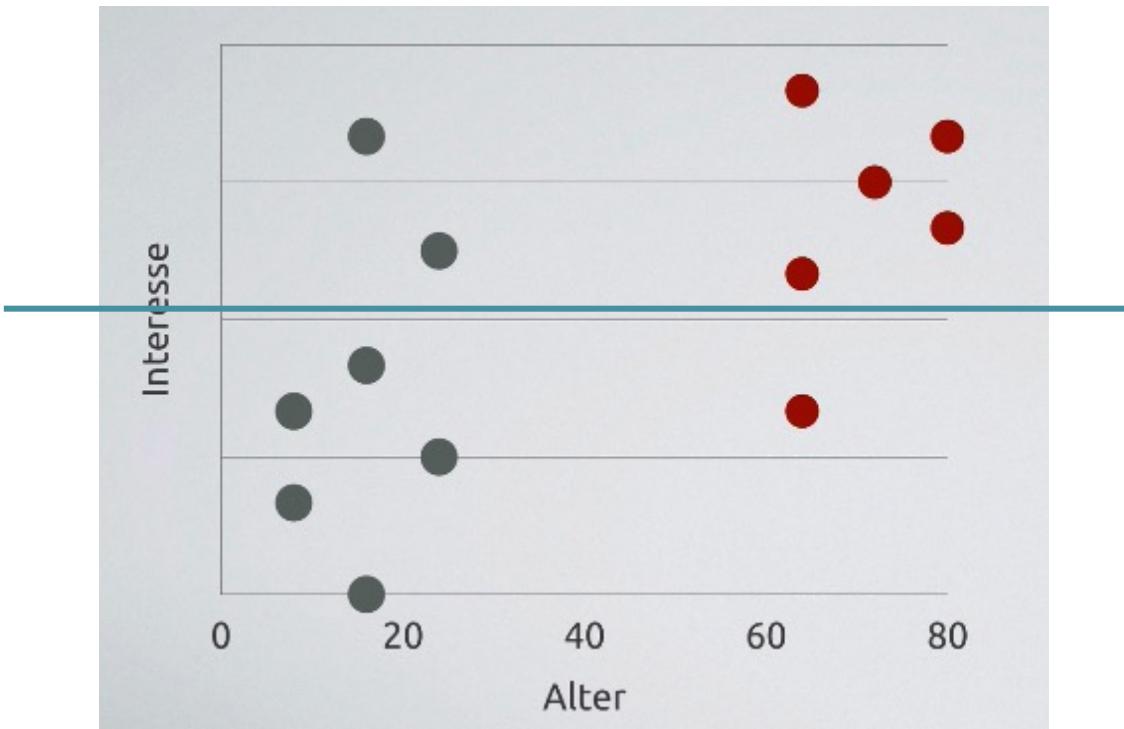
Bei einem Würfelexperiment wäre die Wahrscheinlichkeit eine „6“ zu werfen 1/6. Die Entropie eines Wurfes wäre dann ca. 0,5.

10.2 Idee der Entscheidungsbäume

- Gegeben seien folgende Daten mit hoher Entropie, die nahe der 1 ist.
 - Der Entscheidungsbaumgenerator sucht nach **waagerechten- und senkrechten Trennlinien**, welche die **Entropie senken**.

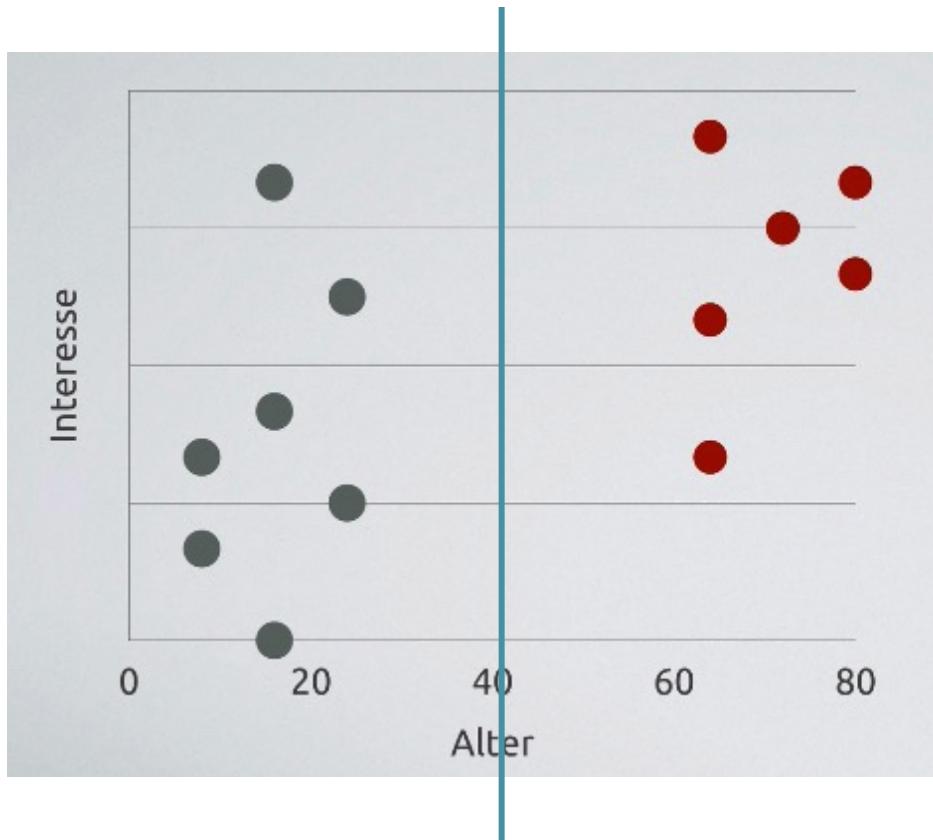


10.2 Idee der Entscheidungsbäume



- Die **Entropie** wäre in den beiden Bereichen **oben** und **unten** geringer...
- Die Suche geht aber weiter.... Gesucht wird eine Unterteilung mit möglichst viel Ordnung also möglichst geringer Entropie.

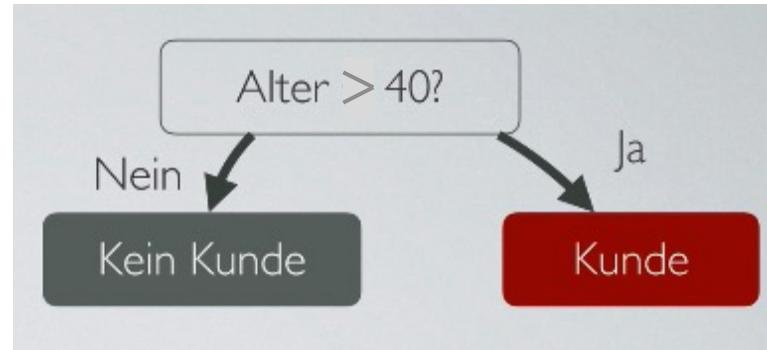
10.2 Idee der Entscheidungsbäume



- Die **Entropie** wäre in den beiden Bereichen **links** und **rechts sogar 0**
- Damit wär eine **ideale Aufteilung gefunden**.

10.2 Idee der Entscheidungsbäume

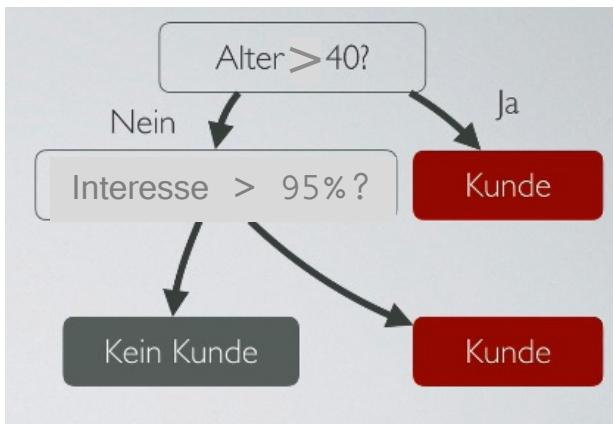
- Das führt zu folgendem **Entscheidungsbaum**:



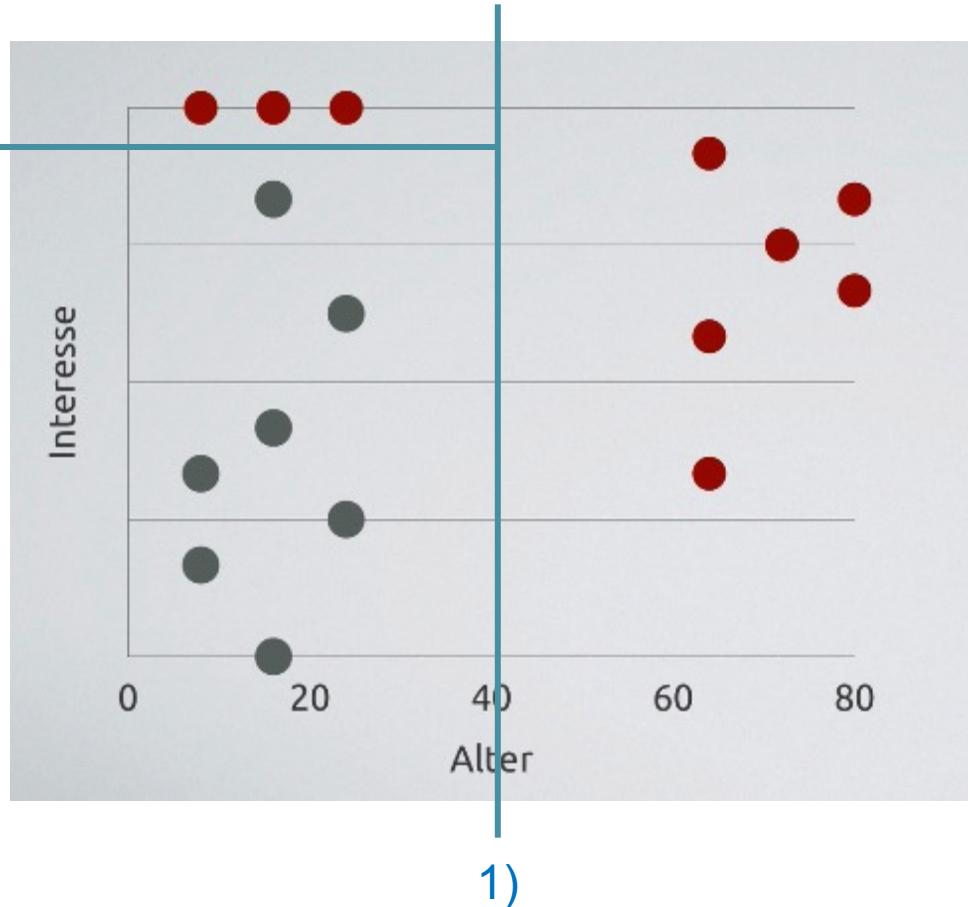
- Leider sehen reale Daten anders aus und die Bäume sind meist komplizierter....

10.2 Idee der Entscheidungsbäume

- Komplizierteres Beispiel:
 - Schritt 1: Alter < 40
 - Schritt 2: Interesse > 95%
- Entscheidungsbaum:



2)



1)

10.3 Eigenschaften von Entscheidungsbäumen

- **Vorteile von Entscheidungsbäumen**
 - **Einfaches Modell:** Entscheidungen sind sehr gut nachvollziehbar
 - **Schnelles Modell:** Programmtechnisch sind das ja nur geschachtelte if-then-else-Strukturen, die wenig Rechenzeit verbraten
 - Daten müssen vorab kein Scaling durchlaufen
 - Aller Erfahrung können viele Klassifikationsprobleme damit recht gut gelöst werden.
- **Nachteile von Entscheidungsbäumen**
 - Passt sich teils stark den Daten an, also anfällig für **Overfitting** / Überanpassung
 - Das Modell antwortet nur mit ja / nein oder der Klassenzugehörigkeit. Keine Prozentangaben / Wahrscheinlichkeitswerte, um abzuschätzen wie sicher eine Schätzung des Modells ist.

10.4 Beispiel zu Entscheidungsbäumen

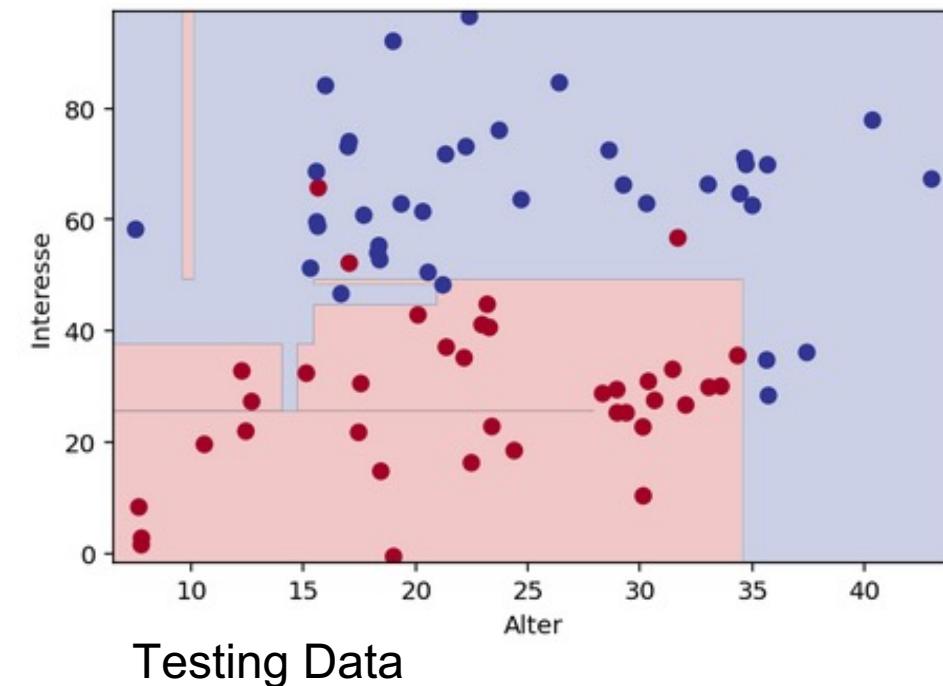
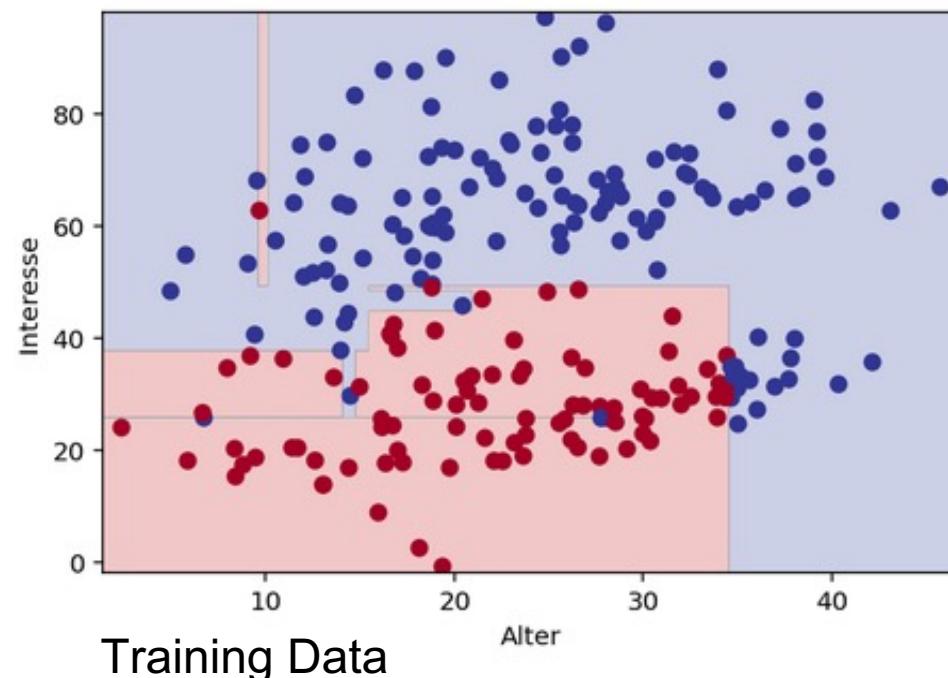
- **Beispiel: Wir starten eine Kickstarter-Kampagne**
 - Wir haben eine Sportkamera entwickelt, die 3D-Videos aufnimmt
 - Wir haben Daten über ersten 891 Kundenregistrierungen auf unserer Web-Plattform
 - Wir erzeugen einen **Entscheidungsbaum**, der auf das **Minimieren der Entropie** strebt:
 - Datei: **classification.csv**
 - **Python-Demo: Entscheidungsbaum.ipynb**

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = "entropy")
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

0.9466666666666667

10.4 Beispiel zu Entscheidungsbäumen

- Man sieht die Einteilung mit waagerechten und senkrechten Linien
- Einzelne Ausreißer werden extra bedient. Das sieht nach Überanpassung / Overfitting aus!
- Auch die Testdaten passen häufig nicht in die filigranen Strukturen, was ebenfalls ein Indiz für Überanpassung ist.



10.5 Entscheidungsbäume visualisieren

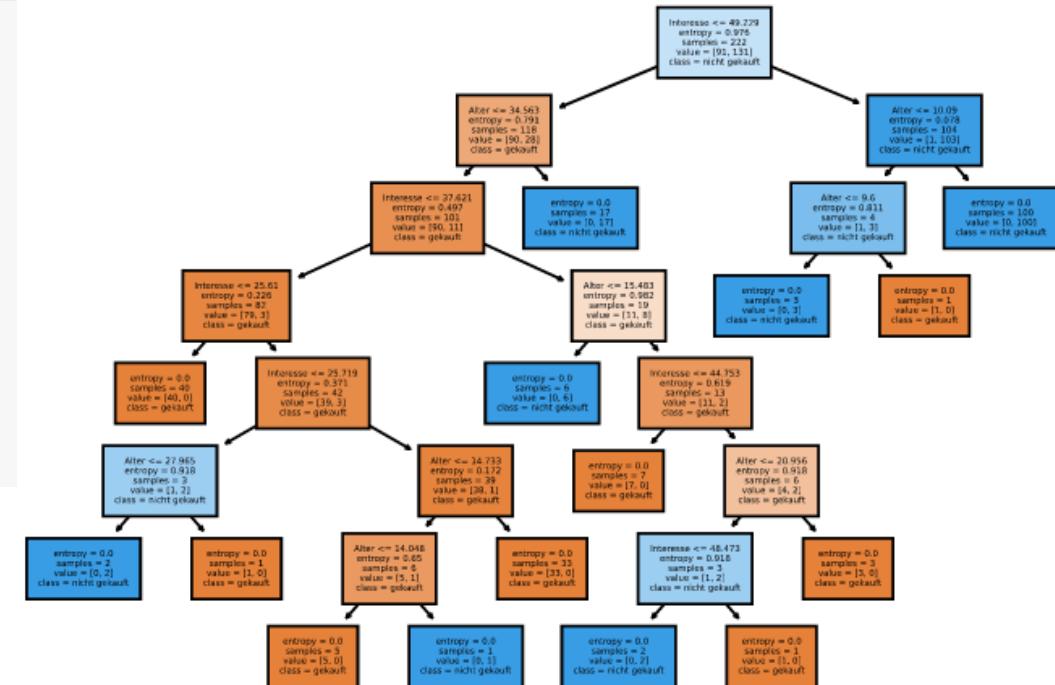
– Beispiel: Entscheidungsbaum Visualisieren

- Datei: **classification.csv**

Python-Demo: EntscheidungsbaumVisualisieren.ipynb

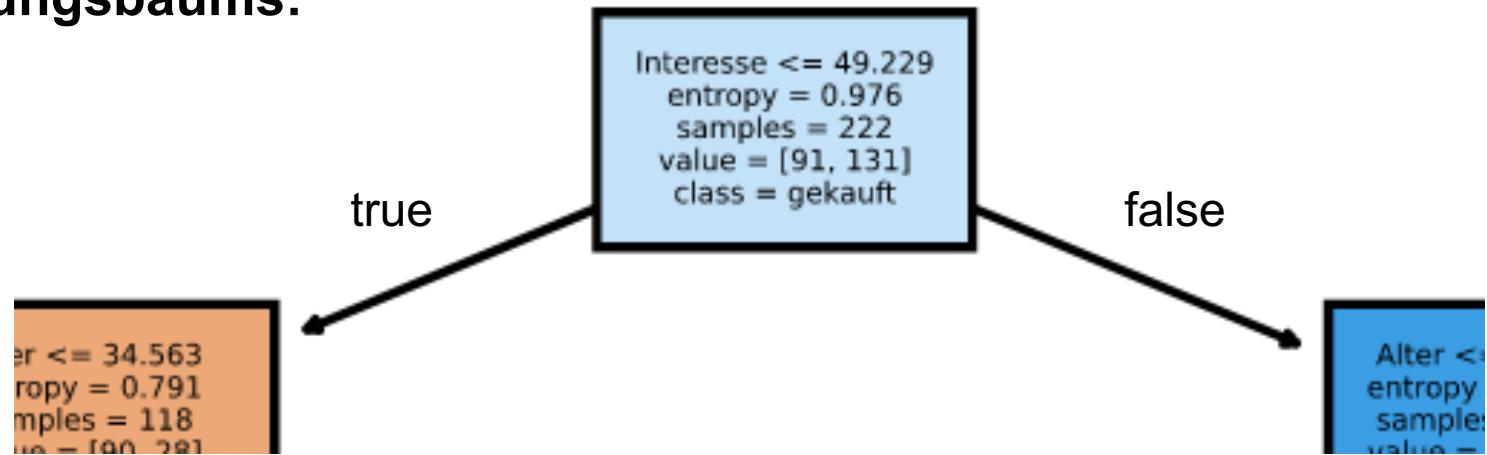
- `plot_tree()` hat umfangreiche Parameter. Dokumentation mit **plot_tree?**

```
%matplotlib inline
%config InlineBackend.figure_formats = set(["svg"])
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(dpi=80)
plot_tree(model,
          fontsize=3,
          feature_names = ["Alter", "Interesse"],
          class_names = ["nicht gekauft", "gekauft"],
          filled = True
         )
plt.show()
```



10.5 Entscheidungsbäume visualisieren

- Lesen eines Entscheidungsbaums:



- Ist das **Interesse <= 49.229** ?
 - Falls **ja**, gehe nach **links**
 - Falls **nein**, gehe nach **rechts**
- Man sieht die Anzahl der Samples und die Entropie. Bei Verzweigungen nur wegen einigen wenigen Samples haben wir wahrscheinlich **Overfitting**
- Je kleiner die **Entropie**, desto höher die Ordnung. Je mehr Ordnung, desto stärker die **Farbe**.

10.6 Entscheidungsbäume beschränken

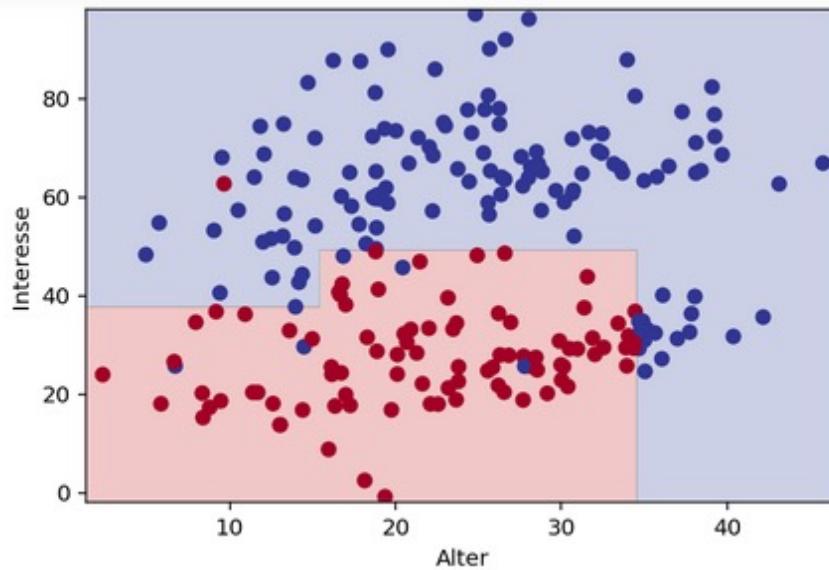
- Beispiel: Entscheidungsbaum Beschränken
 - Datei: [classification.csv](#)
 - Python-Demo: [EntscheidungsbaumBeschränken.ipynb](#)
 - Vermeidung von Overfitting durch
 - **max_depth:** Begrenzung der Tiefe oder einer
 - **min_samples_leaf:** Mindestanzahl von Samples zur Errichtung eines Knotens.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = "entropy",
                                max_depth = 4,
                                min_samples_leaf = 3)
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

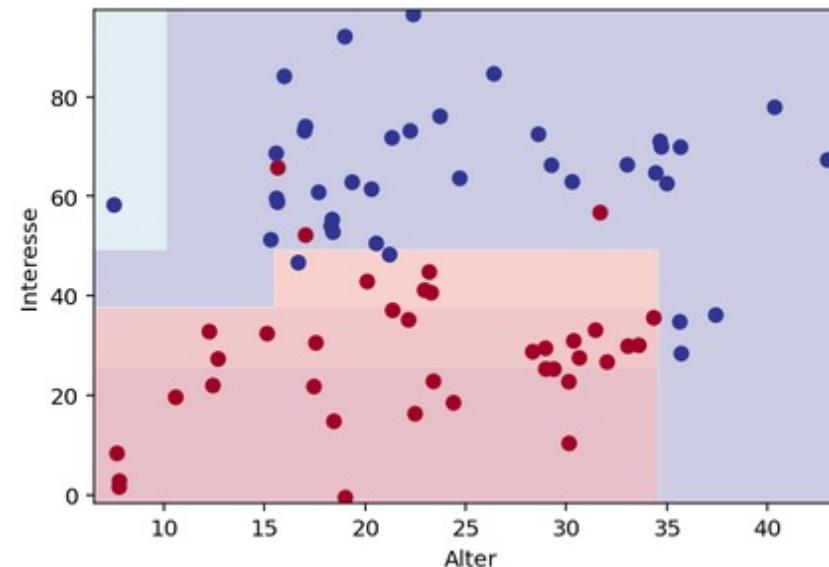
0.9333333333333333

10.6 Entscheidungsbäume beschränken

- Einzelne Ausreißer werden nicht mehr extra bedient.
- R^2 -Wert so gut wie gleich.
- **Heuristik:** Baum solange beschneiden, bis R^2 sich beginnt zu verschlechtern.
- Mit `proba = True` kann man für Bereiche auch **Wahrscheinlichkeiten visualisieren**



Training Data



Testing Data / `proba=True`

10.6 Entscheidungsbäume beschränken

- Beispiel: Entscheidungsbaum Beschränken
 - Datei: **classification.csv**
 - Python-Demo: **EntscheidungsbaumBeschränken.ipynb**
 - Vermeidung von Overfitting durch
 - **max_depth**: Begrenzung der Tiefe oder einer
 - **min_samples_leaf**: Mindestanzahl von Samples zur Errichtung eines Knotens.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = "entropy",
                                max_depth = 4,
                                min_samples_leaf = 3)
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

0.9333333333333333

10.7 Praxisbeispiel zur Giftigkeit von Pilzen

- **Aufgabe: Gegeben ist ein Datensatz von Pilzen**
 - Quelle: <https://www.kaggle.com/uciml/mushroom-classification>
 - Es gibt giftige und ungiftige Arten
 - Wir haben 8124 Daten mit 23 Spalten zu Form Farbe bzw. Zu Beschaffenheit von Stil und Hut usw.
 - Wir wollen zur Vorhersage der Giftigkeit einen Entscheidungsbaum erzeugen.
 - Datei: **mushroom.csv**
 - **Python-Aufgabe: PilzeAufgabe.ipynb**
 - **Hinweis:** One-Hot-Encoding aller Spalten machen. Aus «class» werden zwei Spalten: Eine von beiden genügt hier, da diese 1 oder 0 enthält.
 - Musterlösung: **PilzeMusterlösung.ipynb**



10.8 Random Forest

- **Idee: Aus einzelnen Bäumen macht man einen Wald...**
 - Statt nur einen Baum zu erzeugen, werden gleich mehrere Bäume erzeugt.
 - Bei jedem Schritt im Aufbau schließt man ein paar zufällig gewählte Spalten aus.
 - Dadurch wird jeder Baum aus dem Wald ein bisschen anders
- **Vorteile**
 - Daten müssen nicht aufwändig aufbereitet werden
 - Algorithmus kommt mit vielen Spalten klar
 - Prozentwert kann angegeben werden
- **Nachteile**
 - Modell nicht so anschaulich wie beim einzelnen Baum
 - Hoher Rechenaufwand



10.8 Random Forest

- Beispiel: Random Forest in der Praxis
 - Datei: **classification.csv**
 - Python-Demo: **RandomForest.ipynb**
 - Wichtigster Parameter
 - **N_estimators: Anzahl der Bäume** im Wald (default: 100)

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(criterion = "entropy", n_estimators=15)
model.fit(X_train, y_train)

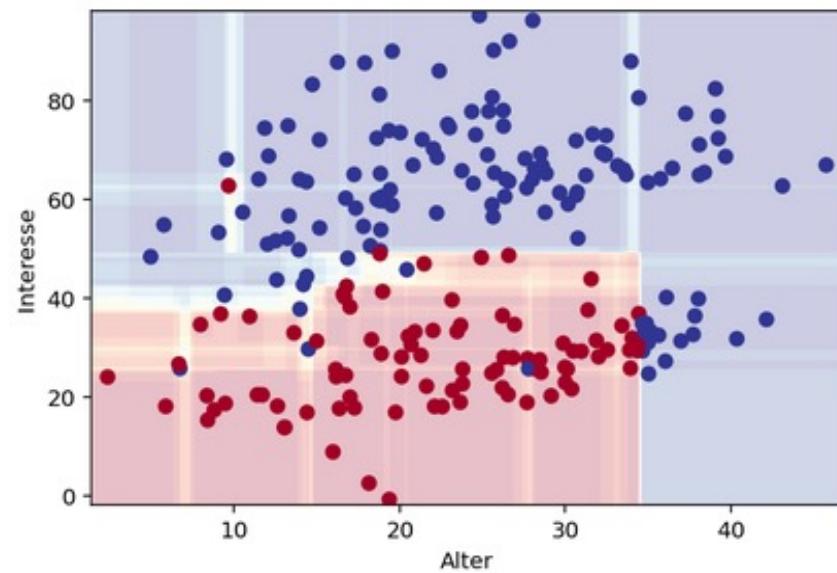
print(model.score(X_test, y_test))
```

0.9466666666666667

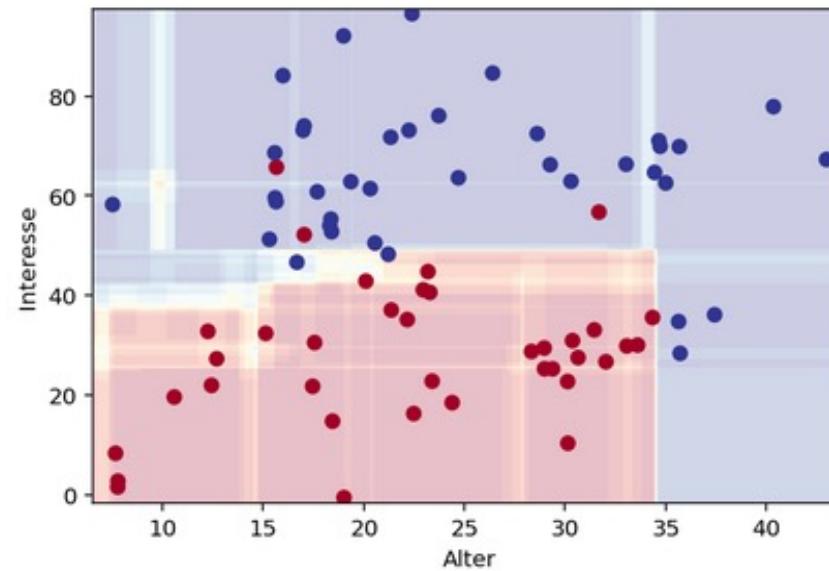
- Leicht verbessertes Ergebnis gegenüber dem singulären Entscheidungsbaum

10.8 Random Forest

- Die Bewertungen der einzelnen Bäume werden gewichtet, was mit `proba = True` kann gut sichtbar wird. Die Struktur erinnert an vielen Stellen an gleitende Übergänge



Training Data



Testing Data

11. Klassifizierung mit Naivem Bayes

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$



Thomas Bayes
1702 - 1761

11.1 Einführung

- **Der Naive Bayes basiert auf Wahrscheinlichkeitsrechnung**
 - Die **naive** Grundannahme ist dabei, dass **jedes Attribut nur vom Klassenattribut abhängt**, obwohl dies in der Realität selten zutrifft. Trotzdem funktioniert es meist gut.
 - Der Naive Bayes kann auch als **Probabilistisches Machines Lernen** bezeichnet werden
 - Eine Anwendung des Naiven Bayes im **Information Retrieval** wird dort als **Probabilistic Retrieval** bezeichnet und begründet dort ein eigenes Retrievalmodell.
- **Bedingte Wahrscheinlichkeiten**
 - **P(A)** die A-priori-Wahrscheinlichkeit für ein Ereignis A und
 - **P(B | A)** die Wahrscheinlichkeit für ein Ereignis B unter der Bedingung, dass A eingetreten ist und
 - **P(B)** die A-priori-Wahrscheinlichkeit für ein Ereignis B
 - Für zwei Ereignisse A und B mit $P(B) > 0$ lautet die **Bedingte Wahrscheinlichkeit** dann also: **P(A|B)**.

11.1 Einführung

- **Die Auftrittswahrscheinlichkeiten von Daten beschreiben**
 - Wir wollen Trainingsdaten der **Tabelle X** mit bedingten Wahrscheinlichkeiten ausdrücken
 - Wir nehmen hierbei an, es gäbe eine Klasse: **Relevant R**
 - Ein Objekt des Datensatzes **X** wird als $\vec{x} = (x_1, \dots, x_n)$ bezeichnet
 - Die Wahrscheinlichkeit, dass ein Objekt \vec{x} zu **R** gehört ist dann: $P(R|\vec{x})$
 - Man kann nur über $P(x)$, $P(R)$, oder $P(\vec{x}|R)$ Aussagen machen, also muss umgeformt werden.
 - **Der Satz von Bayes dient zur Umformung.** Der Satz lautet:

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

11.2 Herleitung des Probabilistischen Machine-Learning

- Ansatz zum Naiven Bayes:

$$P(R | \vec{x}) = P(\vec{x} | R) \cdot \frac{P(R)}{P(\vec{x})}$$

...mit Bayes...

Erklärung: z.B. $P(x | R)$ ist die Wahrscheinlichkeit, dass ein Datenobjekt, beschrieben durch x für zur Klasse R gehört.

- $P(\vec{x} | R)$ kann auf unterschiedliche Art berechnet werden:

- **Multinomial Naiver Bayes:** Wenn der Vektor \vec{x} nur Werte wie beim One-Hot-Encoding enthält, also diskrete Häufigkeiten angibt.
- **Gauß'scher Naiver Bayes:** Geeignet, wenn der Vektor \vec{x} kontinuierliche Werte enthält

11.3 Der Multinomiale Naive Bayes

- **Multinomialer Naiver Bayes:** Wenn der Vektor x nur Werte wie beim One-Hot-Encoding enthält, also diskrete Häufigkeiten angibt

- **Chancenverhältnisse (Odds)**

- Ist neben P eine andere Möglichkeit Wahrscheinlichkeiten anzugeben.
 - Für $P(X) = 0,5$ kann man z.B. auch sagen:
 - Es steht 50:50, dass X eintritt, oder die Wahrscheinlichkeit, dass X eintritt, steht 1:1 $O(x) = 1$

$$O(X) = \frac{P(X)}{1 - P(X)}$$

$$O(X) = \frac{P(X)}{P(\bar{X})}$$

x1	x2	Class
0	1	0
1	0	1
2	1	1
3	1	0
4	0	1

11.3 Der Multinomiale Naive Bayes

- Ansatz zum Multinomialen Naive Bayes mit Chancenverhältnissen:

...mit Bayes...

$$O(R | \vec{x}) = \frac{P(R | \vec{x})}{P(\bar{R} | \vec{x})} = \frac{P(\vec{x} | R) \cdot \frac{P(R)}{P(\vec{x})}}{P(\vec{x} | \bar{R}) \cdot \frac{P(\bar{R})}{P(\vec{x})}}$$

kürzt sich

Erklärung: z.B. $P(\vec{x} | R)$ ist die Wahrscheinlichkeit, dass ein Objekt, beschrieben durch \vec{x} für zur Klasse R gehört.

11.3 Der Multinomiale Naive Bayes

- Weitere Umformungen:

$$O(R | \vec{x}) = \frac{P(\vec{x} | R) \cdot P(R)}{P(\vec{x} | \bar{R}) \cdot P(\bar{R})} = \frac{P(\vec{x} | R)}{P(\vec{x} | \bar{R})} \cdot O(R)$$

11.3 Der Multinomiale Naive Bayes

- Berechnung für Datenobjekt Nr. 2: $\vec{x} = (1, 1)$ Class: nR

- x_1 und x_2 sind bedingt unabhängig,
dann rechnet man:

$$P((1, 1) | R) = \underbrace{P(x_1=1 | R)}_{\text{Wahrscheinlichkeit, dass der Wert } X_1 = 1 \text{ in einem Datenobjekt der Klasse R steht.}} \cdot \underbrace{P(x_2=1 | R)}_{\text{Wahrscheinlichkeit, dass der Wert } X_2 = 1 \text{ in einem Datenobjekt der Klasse R steht.}}$$

Wahrscheinlichkeit, dass der Wert $X_1 = 1$ in einem Datenobjekt der Klasse R steht.

$X_1 = 1$
kommt 1 mal bei Klasse R vor
Die Klasse R hat 3 Objekte

$$P(x_1=1 | R) = 1 / 3 = 0,33333$$

Wahrscheinlichkeit, dass der Wert $X_2 = 1$ in einem Datenobjekt der Klasse R steht.

$X_2 = 1$
kommt 3 mal bei Klasse R vor
Die Klasse R hat 3 Objekte

$$P(x_2=1 | R) = 3 / 3 = 1,0000$$

x1	x2	Class	
0	1	0	nR
1	0	1	R
2	1	1	nR
3	1	1	R
4	0	1	R

11.3 Der Multinomiale Naive Bayes

x1	x2	Class	
0	1	0	nR
1	0	1	R
2	1	1	nR
3	1	1	R
4	0	1	R

$$P((1, 1) | \bar{R}) = P(x_1=1 | \bar{R}) \cdot P(x_2=1 | \bar{R})$$

Wahrscheinlichkeit, dass der Wert $X_1 = 1$ in einem Datenobjekt der Klasse R steht.

$X_1 = 1$
kommt 2 mal bei Klasse nR vor
Die Klasse R hat 2 Objekte

$$P(x_1=1 | \bar{R}) = 2 / 2 = 1,0000$$

Wahrscheinlichkeit, dass der Wert $X_2 = 1$ in einem Datenobjekt der Klasse R steht.

$X_2 = 1$
kommt 1 mal bei Klasse nR vor
Die Klasse R hat 2 Objekte

$$P(x_2=1 | \bar{R}) = 1 / 2 = 0,5000$$

11.3 Der Multinomiale Naive Bayes

- Chancenverhältnis $O(R)$:

$$O(R) = \frac{P(R)}{P(\bar{R})}$$

Es gibt 3 Objekte der Klasse R
Es gibt 2 Objekte der Klasse nR
 $3 / 2 = 1,5$

Es gibt 3 Objekte der Klasse R
5 Objekte insgesamt
 $3 / 5 = 0,6$

Es gibt 2 Objekte der Klasse nR
5 Objekte insgesamt
 $2 / 5 = 0,4$

x1	x2	Class
0	1	0
1	0	1
2	1	1
3	1	1
4	0	1

11.3 Der Multinomiale Naive Bayes

- Konkrete Berechnung:

$$O(R | (1,1)) = \frac{P(x_1=1 | R) \cdot P(x_2=1 | R)}{P(x_1=1 | \bar{R}) \cdot P(x_2=1 | \bar{R})} \cdot O(R)$$

$$= \frac{\frac{1}{3} \cdot \frac{1}{2}}{\frac{1}{3} \cdot \frac{1}{2}} \cdot \frac{3}{2}$$

$$= \frac{1}{1} = \underline{\underline{1.0}}$$

x1	x2	Class
0	1	nR
1	0	R
2	1	nR
3	1	R
4	0	R

- Das Chancenverhältnis, dass die Kombination $x=(1,1)$ zur Klasse R gehört, ist 1. Das heißt: Die Wahrscheinlichkeit ist gleich groß, dass $x=(1,1)$ zu nR gehört.

11.4 Ein E-Mail Spam Filter mit Multinomial Naive Bayes

- Bespieldaten: „SMS Spam Collection Dataset“
Quelle: <https://www.kaggle.com/uciml/sms-spam-collection-dataset>
- 5574 SMS-Nachrichten, die als „spam“ oder „ham“ (Nicht-Spam) gekennzeichnet wurden:

A v1	A v2	A	A
class	spam	5169 unique values	[null] bt not his girlfrnd... Other (47)
ham	87% 13%	99% 0% 1%	[null] MK17 92H. 450Ppw ... Other (10)
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got a...		
ham	Ok lar... Joking wif u oni...		
spam	Free entry in 2 a wkly comp to win FA Cup final tkt 21st May 2005. Text FA to 87121 to receive entr...		
ham	U dun say so early hor... U c already seen em...		

11.4 Ein E-Mail Spam Filter mit Multinomial Naive Bayes

- **Exkurs: Prinzip der Vorverarbeitung von Texten im Information-Retrieval**
 1. **Tokenizing:** Aus dem jeweiligen Text werden Wörter extrahiert.
 2. **Stoppwortelimination:** Worte wie „der“, „die“, „ein“ „bald“, „nichts“, „mehr“, ... sind nicht sinntragend und werden gefiltert
 3. **Stammformenreduktion:** Aus Wörtern wie „gemacht“, „getan“, „Häuser“ wird: „machen“, „tun“, „Haus“
 4. **Lemmatisierung:** Orte, Personen, Zeitbegriffe, Marken usw. werden erkannt und semantisch verortet.
- In unserem Beispiel wollen ist aber nur 1) explizit enthalten, um eine **Wort-Dokument-Matrix** aufzubauen

11.4 Ein E-Mail Spam Filter mit Multinomial Naive Bayes

- Nach der Vorverarbeitung der Texte erhalten wir eine Datentabelle der Art:

Nachricht	Jetzt	kaufen	Schnäppchen	Angebot
Jetzt kaufen	1	1	0	0
Schnäppchen kaufen	0	1	1	0
Jetzt Schnäppchen kaufen	1	1	1	0
Kaufen, Angebot!	0	1	0	1
Kaufen, kaufen, kaufen!	0	0	0	0

Ein Spalte, bei der ein Wort nur ein einziges Mal im Korpus vorkommt, ist wahrscheinlich vernachlässigbar.

Es gibt unterschiedliche Verfahren:
Hier könnte man 1 oder 3 eintragen

- In Wirklichkeit gibt es bei Texten aber **sehr sehr viele Spalten**.
Anzahl Spalten = Anzahl Wörter im gesamten Textkorpus!!!

11.4 Ein E-Mail Spam Filter mit Multinomial Naive Bayes

- Python-Beispiel: **NaiveBayesMultinomialSpamfilter.ipynb**
 - Daten: **spam.csv**

```
X = df["message"]
y = df["type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

```
cv = CountVectorizer()
cv.fit(X_train)
X_train = cv.transform(X_train)
X_test = cv.transform(X_test)
```

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

0.9863603732950467

	type	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Parameter zur Einschränkung:

- *min_df*: Mindesthäufigkeit – wegen sehr seltener Wörter
- *max_df*: Maximalhäufigkeit – wegen sehr häufiger Wörter
- *max_features*: Anzahl Wörter
(Einschränkung nur zu Testzwecken wg. Performance sinnvoll!!!)

Sparse-Matrix
(Wort-Dokument-Matrix)

11.5 Gauß'scher Naiver Bayes

- Wir erinnern uns:

- Ansatz zum Naiven Bayes:

Erklärung: z.B. $P(x | R)$ ist die Wahrscheinlichkeit, dass ein Datenobjekt, beschrieben durch x für zur Klasse R gehört.

$$P(R | \vec{x}) = P(\vec{x} | R) \cdot \frac{P(R)}{P(\vec{x})}$$

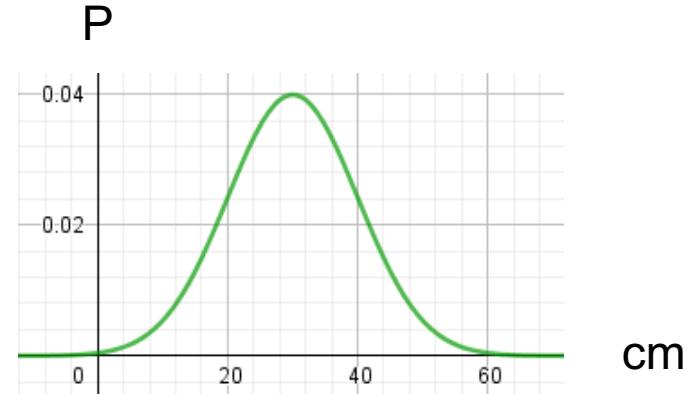
...mit Bayes...

- $P(\vec{x} | R)$ kann auf unterschiedliche Art berechnet werden:

- **Multinomial Naiver Bayes:** Wenn der Vektor \vec{x} nur Werte wie beim One-Hot-Encoding enthält, also diskrete Häufigkeiten angibt.
 - **Gauß'scher Naiver Bayes:** Geeignet, wenn der Vektor \vec{x} kontinuierliche Werte enthält

11.5 Gauß'scher Naiver Bayes

- **Normalverteilung (Gaußverteilung) am Beispiel:**
 - Wir sägen 100 Bretter auf 1m zu
 - Qualitätssicherung: Wir messen nach!
 - 1) 3.01m 2) 2.99m 3) 3.02m ... 100) 2.99m
 - Beobachtung: Werte nah bei 3.00m kommen oft vor, je weiter entfernt, desto seltener kommt ein Wert vor. Diese Verteilung und vieles Andere in der Natur entspricht der Normalverteilung.
- **Idee:** Die Daten einer Spalte sind auch normalverteilt. Wir können dann, um $\vec{P}(\vec{x} | \mathcal{R})$ zu berechnen, über die Formel der Normalverteilung, die Anteile jeder Spalte bestimmen und verrechnen. Für jede Spalte müssen Durschnitt μ und Varianz σ aus den Trainingsdaten ermittelt werden.



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

11.5 Gauß'scher Naiver Bayes

– Anschauliches Beispiel:

- Wir wollen Katzen und Hunde anhand Gewicht und Größe mit Gauß'schem NB klassifizieren.

- Ein unbekanntes Tier **B** habe folgende Daten:

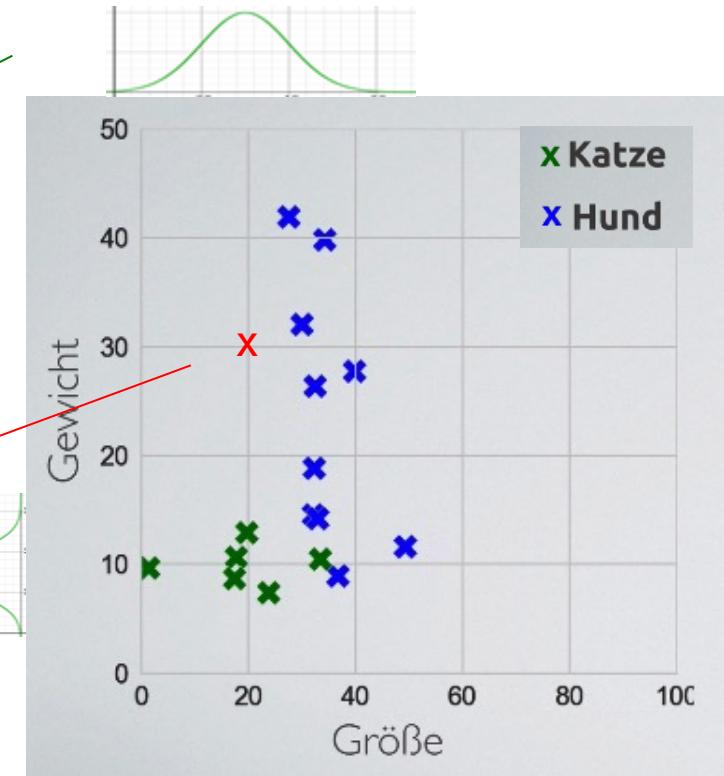
$$B = \{\text{Gew} = 30, \text{Gr} = 20\}$$

– Mit Satz von Bayes

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

Größen-
Normalverteilung
bei Katzen

Gewichts-
Normalverteilung
bei Katzen



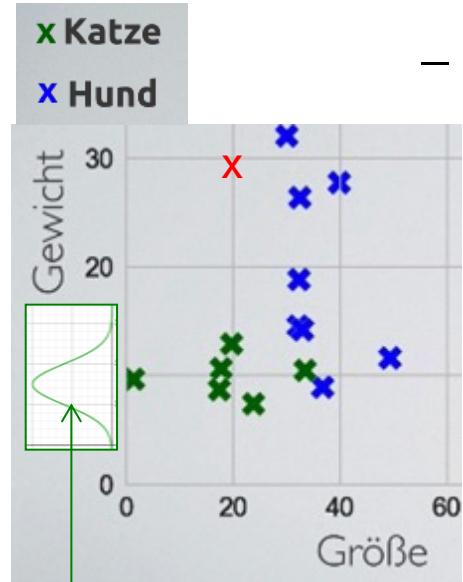
...ergibt sich dann:

$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$

11.5 Gauß'scher Naiver Bayes

- Wie berechnen wir z.B. jetzt $P(\text{Gew}=30, \text{Gr}=20 | \text{Katze})$?
 - **Annahme:** die beiden Größen seien bedingt unabhängig (obwohl das ja in Wirklichkeit nicht so ist, weswegen das Verfahren auch als **Naiver Bayes** bezeichnet wird).



– Konkrete Berechnung:

$$P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze}) = P(\text{Gew} = 30 | \text{Katze}) \cdot P(\text{Gr} = 20 | \text{Katze})$$

- $P(\text{Gew}=30 | \text{Katze})$ wird so ergibt sich aus der Normalverteilung der Gewichtswerte von Katzen aus unseren Trainingsdaten:
 1. Durchschnitt μ und Varianz σ^2 ermitteln. Beispiel: [NormalverteilungAusPunkten.ggb](#)
 2. $P(\text{Gew}=30 | \text{Katze}) = f(30)$ mit der Formel:
 3. Ergebnis: ca. 0

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

11.5 Gauß'scher Naiver Bayes

$$B = \{\text{Gew} = 30, \text{Gr} = 20\}$$

- Da wir beide Wahrscheinlichkeiten der bedingten Klassenzugehörigkeit nur vergleichen wollen, fällt der Nenner jeweils weg:

$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$



$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})$$

$$P(\text{Hund}) = [\text{Anzahl Hunde}] / [\text{Anzahl Tiere insgesamt}]$$

11.5 Gauß'scher Naiver Bayes

$$B = \{Gew = 30, Gr = 20\}$$

- Da wir beide Wahrscheinlichkeiten der bedingten Klassenzugehörigkeit nur vergleichen wollen, fällt der Nenner jeweils weg:

$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) = \frac{P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})}{P(\text{Gew} = 30, \text{Gr} = 20)}$$



$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})$$



$$P(\text{Hund}) = [\text{Anzahl Hunde}] / [\text{Anzahl Tiere insgesamt}]$$

11.5 Gauß'scher Naiver Bayes

- Wenn alle Ergebnisse vorliegen, z.B.:

0,03

$$P(\text{Katze} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Katze}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Katze})$$

$$P(\text{Hund} | \text{Gew} = 30, \text{Gr} = 20) \sim P(\text{Hund}) \cdot P(\text{Gew} = 30, \text{Gr} = 20 | \text{Hund})$$

0,55

...dann entscheidet man sich für die Klasse mit dem größeren Ergebnis.

- **Endergebnis:** $B = \{\text{Gew} = 30, \text{Gr} = 20\}$ ist ein **Hund**

11.5 Gauß'scher Naiver Bayes

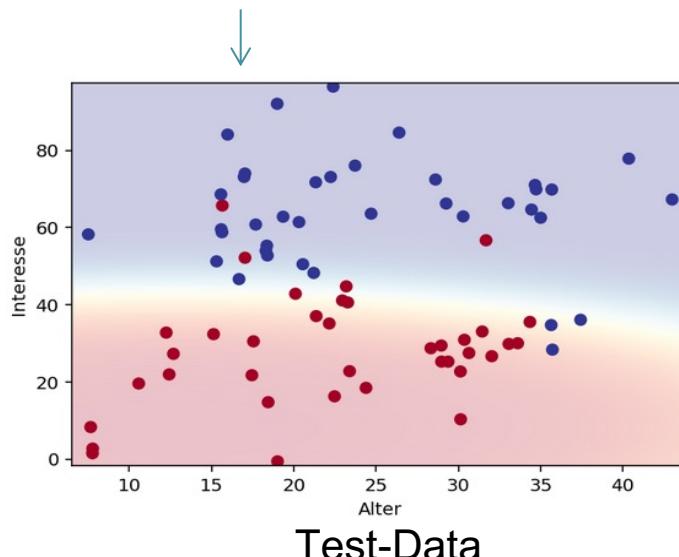
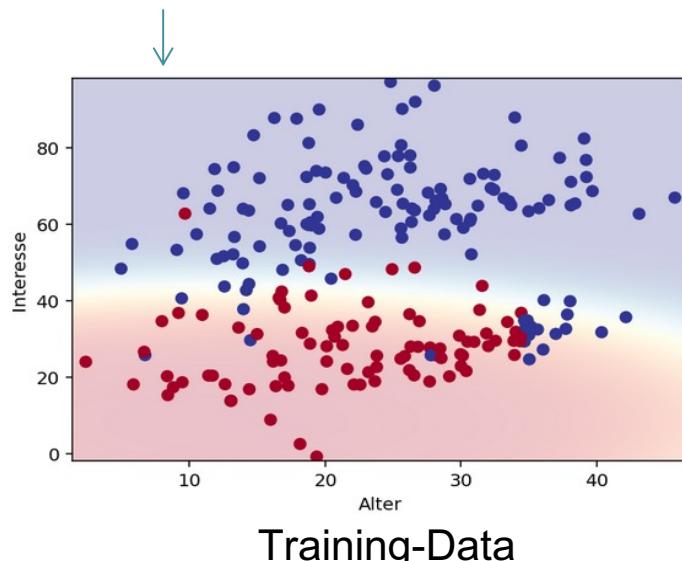
- Praxisbeispiel Gauß'scher Naiver Bayes

- Python: [NaiveBayesGauss.ipynb](#)
- Daten: [classification.csv](#)

	age	interest	success
0	23.657801	18.859917	0.0
1	22.573729	17.969223	0.0
2	32.553424	29.463651	0.0
3	6.718035	25.704665	1.0
4	14.401919	16.770856	0.0

```
from sklearn.naive_bayes import GaussianNB  
model = GaussianNB()  
model.fit(X_train, y_train)  
print(model.score(X_test, y_test))
```

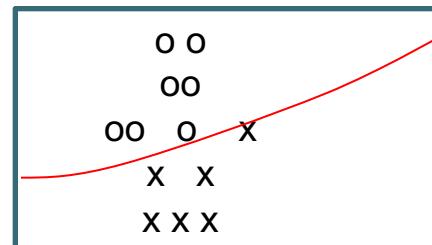
0.8933333333333333



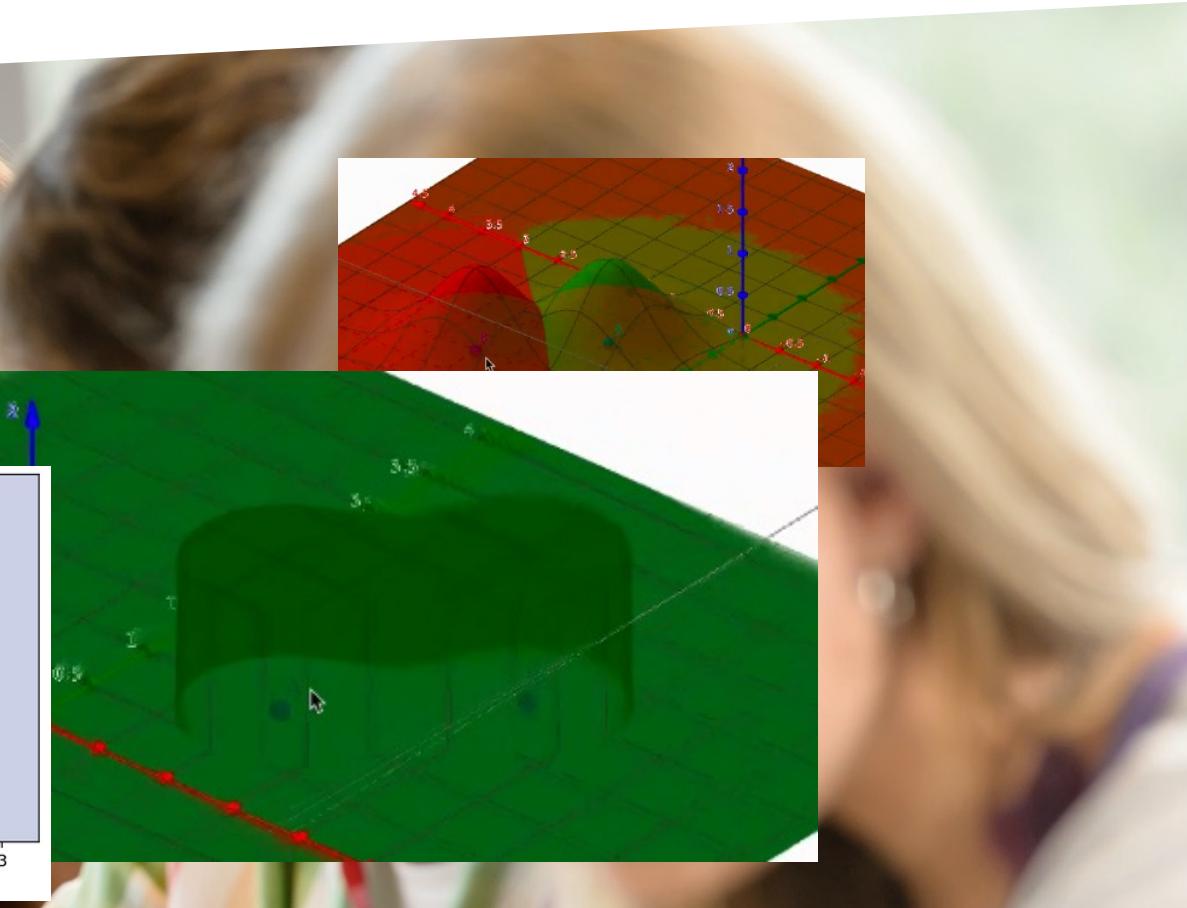
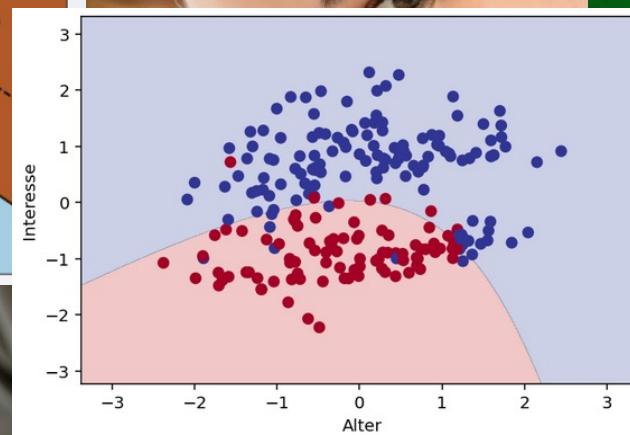
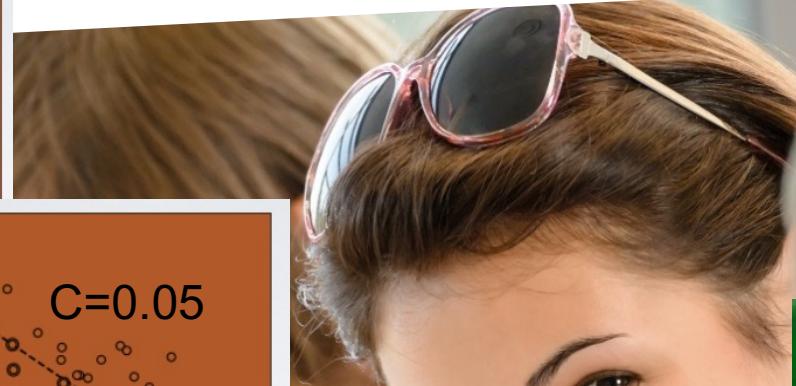
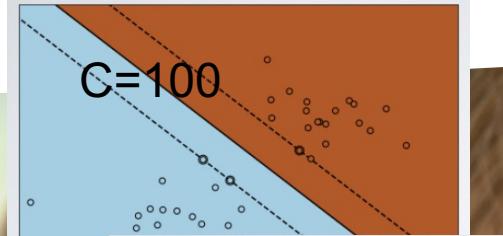
Bemerkung:
Trennbereiche sind
gebogen

11.6 Eigenschaften des Naiven Bayes

- **Vorteile:**
 - Einfaches Modell
 - Kommt gut mit sehr vielen Spalten klar
 - Sehr performant (schnell berechenbar)
 - Man kann Zwischenergebnisse speichern, z.B. Spam-Wahrscheinlichkeiten für Wörter in der Sparse-Matrix
 - Ergibt Zugehörigkeitswerte für jede Klasse: Damit kann man die Verlässlichkeit der Aussage erkennen (Klassenzugehörigkeit eindeutiger oder weniger eindeutig)
- **Nachteile:**
 - Grundannahmen sind oft nicht gegeben. Reale Probleme sind oft nicht naiv.
 - Gauß'scher Naive Bayes ergibt nur einfach gebogene Trennbereiche zwischen den Klassen



12. Klassifizierung mit Support Vector Machines

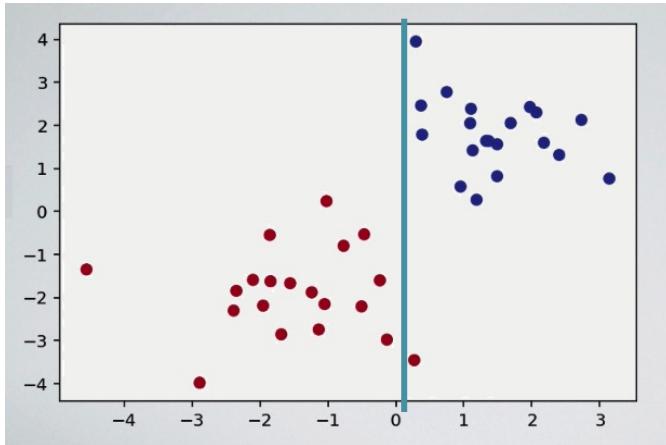


12.1 Einführung

- **Support Vectors basieren auf dem Prinzip**
 - Der Abstand zwischen den Datenpunkten und der Entscheidungsgrenze zu maximieren. Support-Vector-Machines sind also sog. Large Margin Classifiers.
 - Im Gegensatz zu Entscheidungsbäumen, wo die Trennlinien entweder nur senkrecht oder waagrecht sind, sind es bei Support-Vector-Machines meist **schräge Linien**.
 - In **zweidimensionalen** Szenarien liefern SVMs tatsächlich **Linien**
 - In **dreidimensionalen** Szenarien liefern SVMs **Ebenen** (Flächen)
 - In **höherdimensionalen** Szenarien liefern SVMs etwas, das nicht mehr anschaulich ist und deshalb als **Hyperplane** bezeichnet wird.

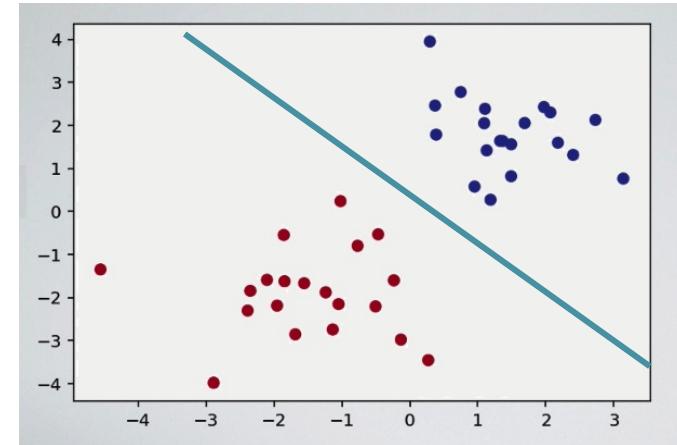
12.1 Einführung

- Anschauungsbeispiel: Vergleich Entscheidungsbaum vs. Support-Vector



Entscheidungsbaum

Trennung oft knapp/fehlerhaft

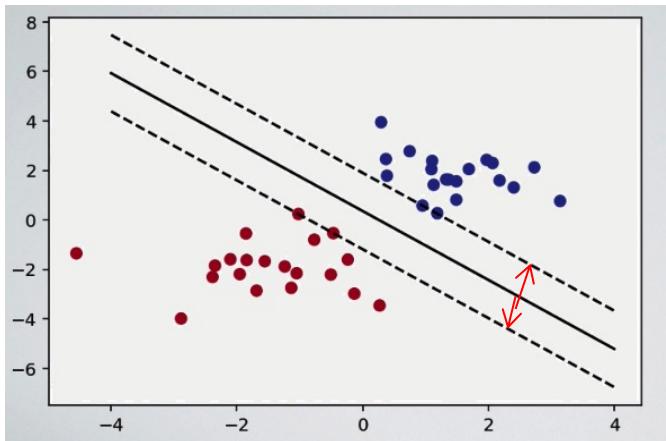


Support-Vector

Trennung mittig, mit max. Abstand, häufig besser

12.1 Einführung

- Anschauungsbeispiel: Vergleich Support-Vector vs. Logistische Regression



Support-Vector

Optimierung: Winkel und Position
der Trennlinie zu **maximalem Abstand**



Logistische Regression

Optimierung: S-Kurve soll möglichst
wenig Fehler von den Daten abweichen.

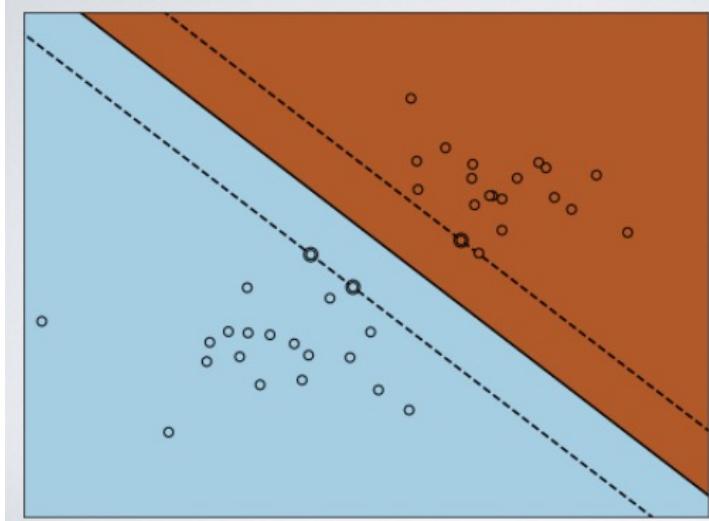
- Ergibt de facto oft ähnliche Ergebnisse, aber in manchen Situationen ist das Eine oder das Andere besser

12.2 Das Optimierungsverfahren in Support-Vector-Machines

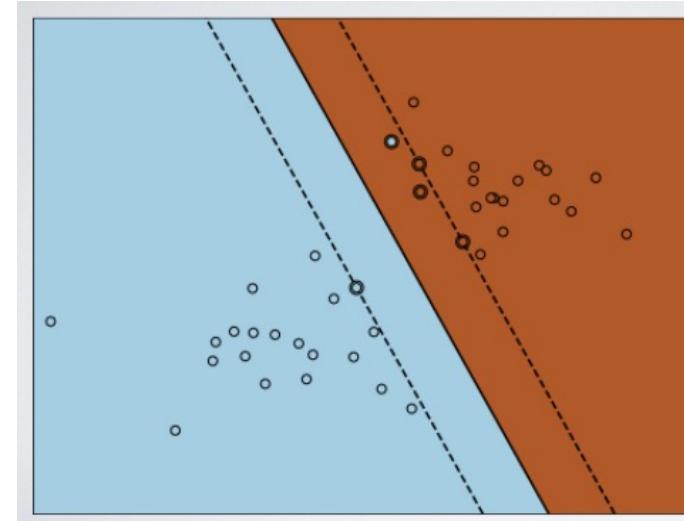
- **Support-Vector:** „Stützvektoren“ kommen vom Ursprung und zeigen auf nächstgelegene Punkte entlang der Grenze zwischen den Klassen.
- **Optimierung:** Position und Winkel der Trennlinie wird auf maximalen Abstand zu den Enden der Support-Vektors optimiert. Anders gesagt: Die **Breite der Straße (Margin), die hindurchführt, wird maximiert.**
- Der **Parameter C:**
 - **Grosses C:** Geringe Fehlertoleranz. Dies führt zu schmäleren Trennbereichen. Einzelne Punkte sind wichtiger. Neigt zum **Overfitting!**
 - **Kleines C:** Grosse Fehlertoleranz. Es werden mehrere Punkte beider Klassen miteinbezogen. Ausreisser fallen dann nicht so ins Gewicht. Dies führt zu breiteren Trennbereichen und SVM folgt eher der großen Tendenz der gesamten Daten. Neigt zum **Underfitting!**
 - **Die Wahl von C:** Es gibt **keine Faustregel.**
Pragmatisch: Unterschiedliche Werte probieren, um R^2 zu maximieren

12.2 Das Optimierungsverfahren in Support-Vector-Machines

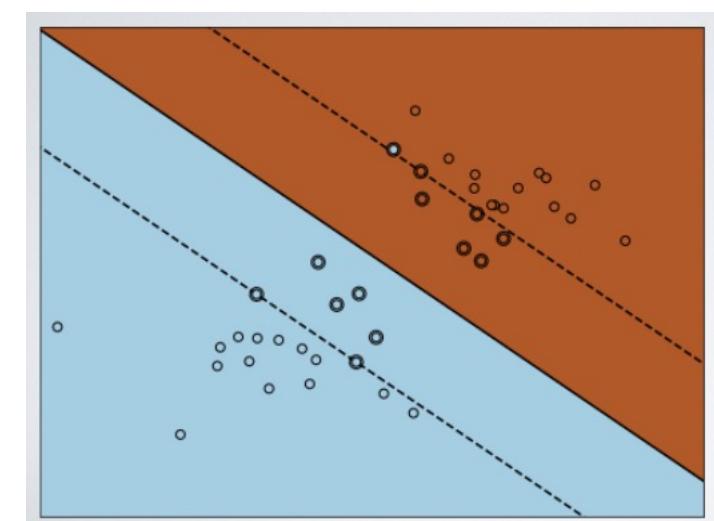
- Veranschaulichung der Auswirkungen des Parameters „C“:



C=100



C=1



C=0.05

- Man sieht, wie der Margin (Trennbereich) immer breiter wird und einzelne falsche Punkte dann auch toleriert werden. **Kleines C also nur bei tatsächlichen Ausreisern!**

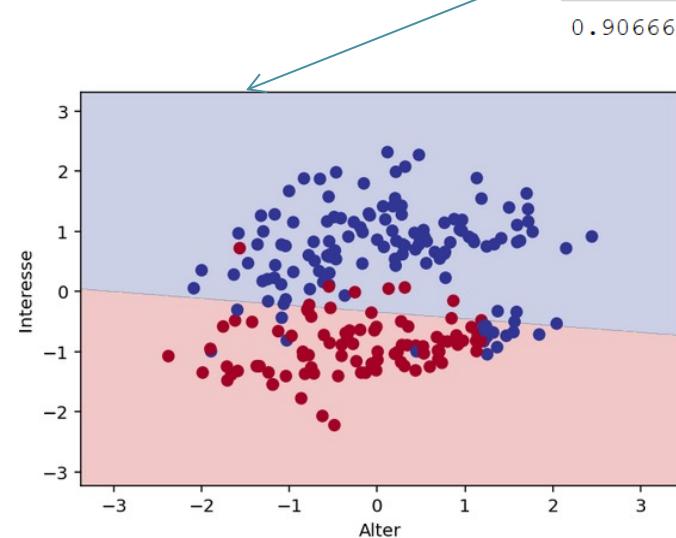
12.3 Praxisprojekt zu Support Vektor Machines mit linearem Kernel

- Python: **SupportVectorMachine.ipynb**
- Daten: **classification.csv**

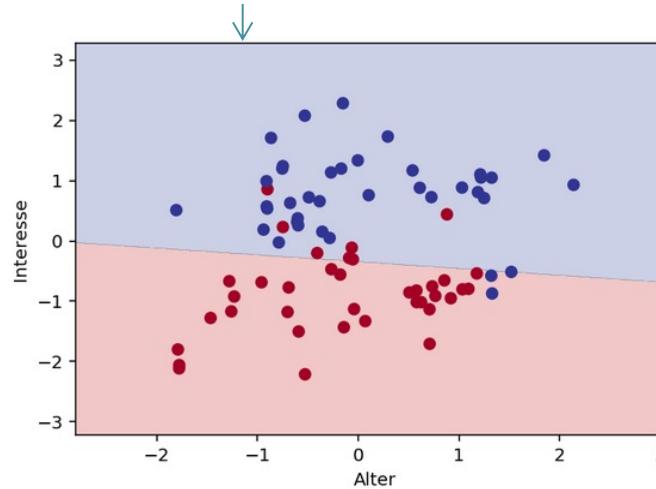
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.svm import SVC
model = SVC(kernel = "linear", C = 0.03)
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

0.9066666666666666



Training-Data



Test-Data

	age	interest	success
0	23.657801	18.859917	0.0
1	22.573729	17.969223	0.0
2	32.553424	29.463651	0.0
3	6.718035	25.704665	1.0
4	14.401919	16.770856	0.0

Parameter C

Bemerkungen:

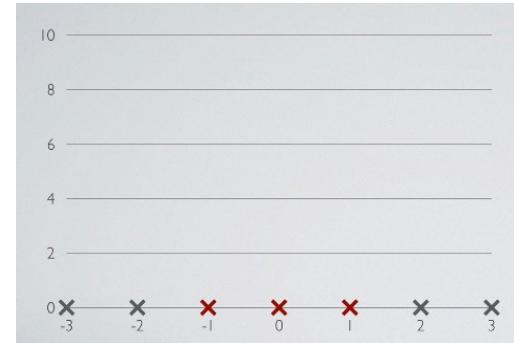
- Trennbereiche linear, und ggf. schräg.
- Ergebnis 0.906 ist nicht so gut.
- „Kurvig“ wäre besser! Siehe folgende Seite...

12.4 Polynomiale Kernel für Support-Vector-Machines

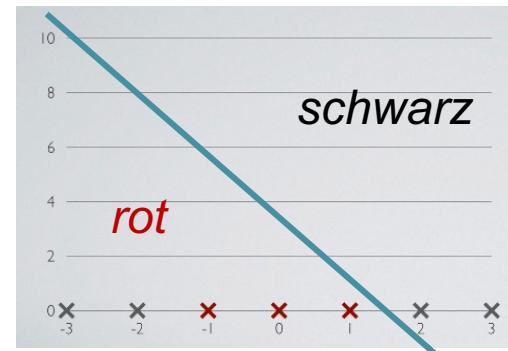
- Bisher haben wir gesehen: Support-Vektor-Machines liefern Trennlinien zwischen Daten unterschiedlicher Klassen
 - Eine Support-Vector-Machine mit **linearem Kernel**: Daten werden mit einer **Gerade**, einer **Ebene** oder einer **Hyperplane** getrennt.
 - Dieser Ansatz liefert nur dann gute Ergebnisse, wenn die Daten „linear trennbar“ sind. Dies ist z.B. der Fall, wenn die Daten verschiedener Klassen sich stärker unterscheiden.
 - Eine Support-Vector-Machine kann einen **polynomialen Kernel** haben: Daten werden mit einer **gekrümmten-** bzw. **gekurvten Linie**, einer **gekrümmten-** oder **gekurvten Fläche** oder einer ‚**Polynomial Hyperplane**‘ getrennt.
 - Dieser Ansatz ist eine wichtige Option, falls lineare Kernel keine guten Ergebnisse liefern, wenngleich diese Variante rechenaufwändiger ist

12.4 Polynomiale Kernel für Support-Vector-Machines

- **Anschaungseispiel:**
 - Angenommen es sollen diese Punkte rot und schwarz linear getrennt werden:
 - Mit Geraden gibt es leider nur Varianten mit erheblichen Fehlern, also konkret:



oder



12.4 Polynomiale Kernel für Support-Vector-Machines

- Polynomiale Erweiterung: Wir **transponieren** die eindimensionalen Punkte in eine zweite Dimension y und mit der Abbildung: $y = x^2$



- Durch die **Polynomiale Transformation** der Daten können die Punkte fehlerfrei **linear getrennt** werden

12.4 Polynomiale Kernel für Support-Vector-Machines

- Python: **SupportVectorMachinePolynomialerKernel.ipynb**
- Daten: **classification.csv**

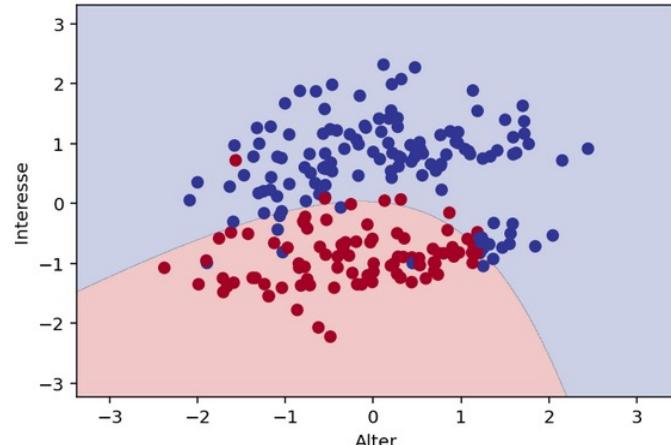
	age	interest	success
0	23.657801	18.859917	0.0
1	22.573729	17.969223	0.0
2	32.553424	29.463651	0.0
3	6.718035	25.704665	1.0
4	14.401919	16.770856	0.0

```
from sklearn.svm import SVC
import numpy as np

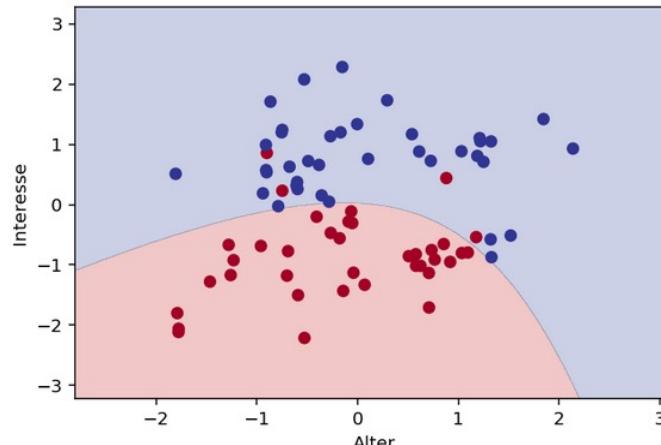
model = SVC(kernel = "poly", degree = 2, coef0 = 1)
model.fit(X_train, y_train)

print(model.score(X_test, y_test))
```

0.9466666666666667



Training-Data



Test-Data

$$(x * v + \text{coef}0)^{\text{degree}}$$

Bemerkungen:

- v sind die Gewichte, die optimiert werden
- $\text{coef}0$ gibt der Quadraturierung mehr Gewicht.
- Trennlinie nun gebogen!
- $R^2 = 0,946$ ist besser als $R^2 = 0,90$ (lineares SVM)

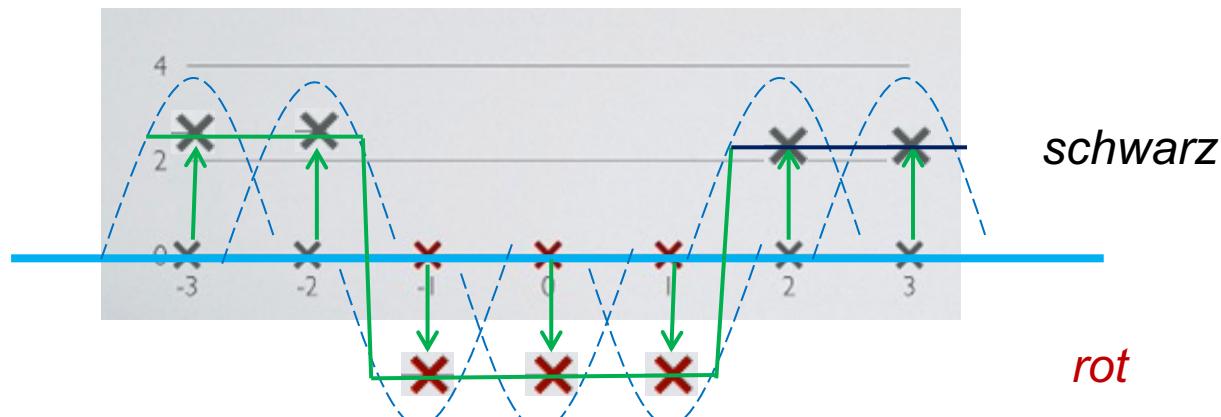
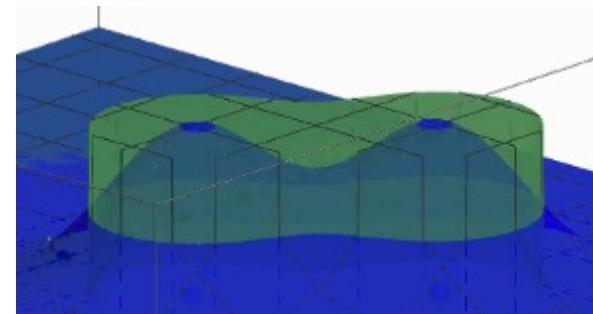
12.5 Radial Basis Function Kernel (RBF) für Support-Vector-Machines

- **Radial Basis Function Kernel (RBF)** wird am häufigsten bei SVMs verwendet.
 - Der RBF-Kernel kann ganz neue Zusammenhänge modellieren
 - Der RBF-Kernel **modelliert sehr verschieden** im Vergleich zu anderen Modellen. RBF ist also das Mittel der Wahl, wenn andere Algorithmen nicht so gut funktionieren.
 - Die Kernelfunktion ist:
 - x steht für eine Koordinate, (Landmark) die im Zweidimensionalen ein Punkt (x,y) sein kann.
 - Das σ definiert die Steigung des Bergs um die „Landmark“

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

12.5 Radial Basis Function Kernel (RBF) für Support-Vector-Machines

- **Idee:** Es wird um die Datenpunkte mithilfe der **Kernelfunktion** ein Radius gelegt. Die Radien von nahen Datenpunkten überlappen sich und so entsteht ein größerer Bereich innerhalb dem die Daten **transponiert** werden.
- **Veranschaulichung** (GeoGebra): **RBF-Kernel.ggb**
- Hier eine Veranschaulichung der **Separierung per Transponierung RBF-Kernel**:



12.5 Radial Basis Function Kernel (RBF) für Support-Vector-Machines

- Der Parameter Gamma

- Bisher kennen wir nur Sigma σ

Im Machine-Learning ist aber Gamma γ eher gebräuchlicher. **Es gilt:**

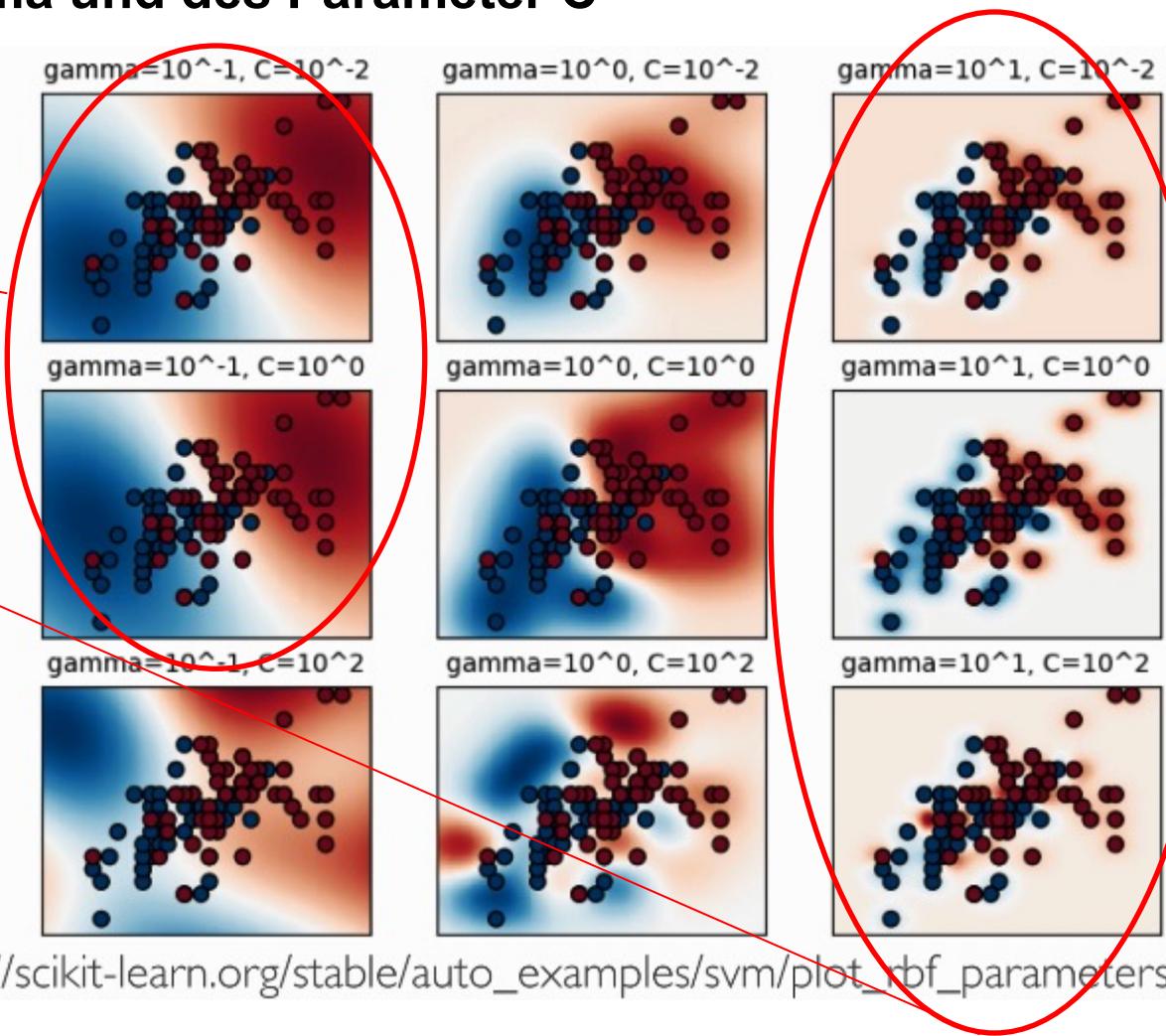
$$\gamma = \frac{1}{2\sigma^2}$$

- **Hohes γ :** Hügel der Kernelfunktion ist steiler, einzelne Punkte sind wichtig, allgemeiner Trend ist weniger wichtig. Neigt zu ***Overfitting!***
 - **Niedriges γ :** Hügel der Kernelfunktion ist flacher, einzelne Punkte sind weniger wichtig, allgemeiner Trend ist wichtiger. Neigt zu ***Underfitting!***
 - Die Auswirkungen von Parameter γ und Parameter C sind sehr ähnlich, daher werden diese beiden Parameter gemeinsam verändert.

12.5 Radial Basis Function Kernel (RBF) für Support-Vector-Machines

- Auswirkungen des Parameter Gamma und des Parameter C

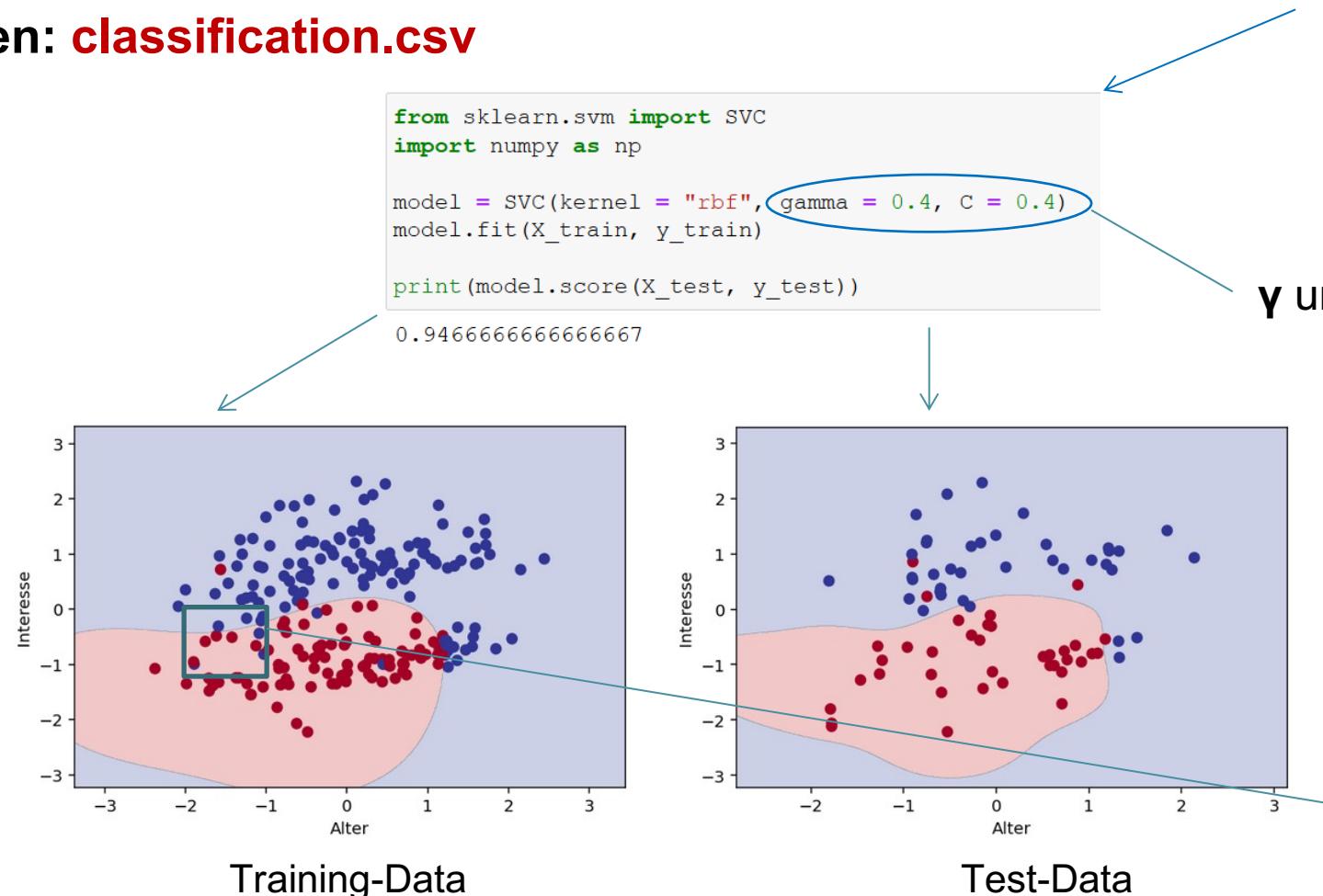
- Zu niedriges γ , zu niedriges C:
Underfitting!
- zu hohes γ , zu hohes C:
Overfitting!



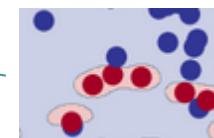
http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html 220

12.5 Radial Basis Function Kernel (RBF) für Support-Vector-Machines

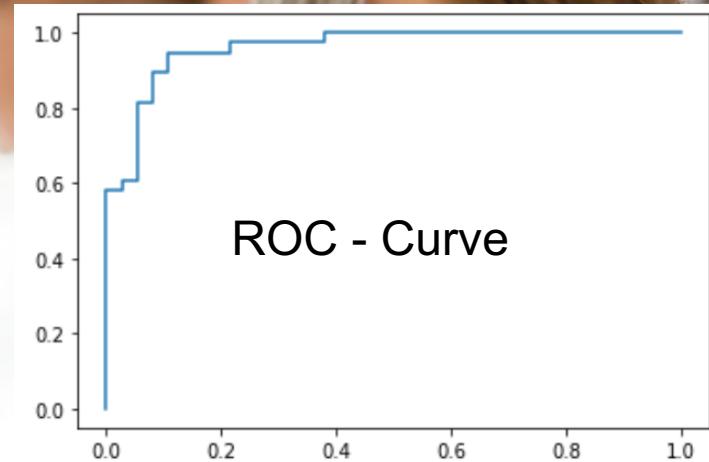
- Python: **SupportVectorMachineRBFFKernel.ipynb**
- Daten: **classification.csv**



- Bemerkungen:**
- $R^2 = 0,946$ ist besser als $R^2 = 0,90$ (lineares SVM)
 - Bei ganz hohem γ und C gibt es nur noch Inseln um Punkte (Overfitting!)



13. Evaluation und Optimierung von Modellen



	Person ist krank (r_p+f_n)	Person ist gesund (f_p+r_n)
Confusion Matrix		
Test positiv (r_p+f_p)	richtig positiv (r_p)	falsch positiv (f_p)
Test negativ (f_n+r_n)	falsch negativ (f_n)	richtig negativ (r_n)

```
pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier(n_neighbors = 3))
])
```

13.1 Die Genauigkeit von Modellen

- Probleme der Modellbewertung

- Beispiel

- Ich habe einen Medizinischen Test entwickelt.
Dieser Test liegt zu 99% richtig, also $R^2 = 0,99$
 - Ist dieser Test gut oder schlecht?
 - Angenommen: 99% der Personen sind tatsächlich gesund und nur 1% ist krank.
 - Der Test könnte jetzt ein „Fake“ sein und bei allen immer sagen: Du bist gesund und es gäbe nie den Fall, dass jemand als „krank“ markiert wird.
 - Dann wäre der Test bei 99% der Personen richtig, obgleich das Modell des Tests ja Betrug wäre und eigentlich die 1% der Kranken gar nicht bemerkt würden. Er wäre also de facto sehr schlecht, obgleich $R^2 = 0,99$
 - $R^2 = 0,99$ kann bei mengenmäßig unausgeglichenen Klassen irreführend sein.

13.1 Die Genauigkeit von Modellen

- **R²** ist nur vertretbar, bei ziemlich **ausgeglichenen Klassen**, also z.B. bei Klassen, die annähernd gleich groß sind.
 - 50% gut / 50% schlecht
 - 33,3% Iris setosa / 33,3% Iris versicolor / 33,3% Iris virginica
 - usw.
 - Meist wird die Klassenausgeglichenheit künstlich erzeugt:
50% „Nicht-Betrugsfälle“ / 50% „Betrugsfälle“ (obwohl diese real selten sind)
- Das Beispiel mit dem **Medizinischen Test** zeigt:
 - **Wir brauchen ein neues Maß, um solche unausgeglichenen Klassen zu bewerten**

13.2 Recall und Precision

- Beispiel: Corona PCR-Test

Recall und Prescision aus Data Management anschauen

- Bei 100 an Covid-19 erkrankten Personen, werden 98% gefunden.
- Der PCR-Test hat also eine **Sensitivität** von 98%
- Der **Recall** ist damit bei 0,98
- **True-Positive-Rate** (TPR) bei 0,98
- Bei 100 mit dem PCR-Test positiv getesteten Personen, sind nur 95% tatsächlich positiv! – Ungefähr jeder 20. Test ist also falschpositiv.
- Die **Precision** ist also bei 0,95
- Bei 100 Personen mit negativem Testergebnis, sind ca. 99% der Testergebnisse korrekt und schließen eine Infektion zu ca. 99% korrekt aus.
- Die **Spezifität** ist also bei ca. 99%
- **False-Positive-Rate** (FPR) bei 0,01

13.2 Recall und Precision

- Python: **Recall und Precision.ipynb**
- Daten: **classification.csv**

```
model = LogisticRegression()
model.fit(X_train, y_train)

print(model.score(X_test, y_test))
```

0.88

```
y_test_pred = model.predict(X_test)
```

```
from sklearn.metrics import precision_score, recall_score

print(precision_score(y_test, y_test_pred))
print(recall_score(y_test, y_test_pred))
```

0.822222222222

0.973684210526

13.3 Die «Confusion Matrix»

- Folgende Darstellung für den **Medizinischen Test**:

	Realität: gesund	Realität: nicht gesund
Arzt: gesund	Korrekt	Patient wird nicht behandelt
Arzt: nicht gesund	Gesunder wird behandelt	Korrekt

verschieden
kritische
Fehlerfälle

- **Motto:** lieber behandelt man 10 umsonst, als dass man einen Fall nicht erkennt.

13.3 Die «Confusion Matrix»

– Weiteres Beispiel: Gerichtsentscheidung

		Realität: unschuldig	Realität: schuldig
Gericht: unschuldig	Korrekt	Schuldiger wird freigesprochen	
	Unschuldiger wird verurteilt		
Gericht: schuldig		Korrekt	In dubio pro reo

13.3 Die «Confusion Matrix»

– Definition

r_p : Richtig Positiv / True Positive

f_p : Falsch Positiv / False Positive

f_n : Falsch Negativ / False Negative

r_n : Richtig negativ / True Negative

	Person ist krank (r_p+f_n)	Person ist gesund (f_p+r_n)
Test positiv (r_p+f_p)	richtig positiv (r_p)	falsch positiv (f_p)
Test negativ (f_n+r_n)	falsch negativ (f_n)	richtig negativ (r_n)

13.3 Die «Confusion Matrix»

- Python: **ConfusionMatrix.ipynb**
- Daten: **classification.csv**

```
y_test_pred = model.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix  
confusion_matrix(y_test, y_test_pred)
```

Das Format einer `confusion_matrix`:

	Modell: Nicht wahr	Modell: Wahr
Realität: Nicht wahr	Richtig negativ	Falsch positiv
Realität: Wahr	Falsch negativ	Richtig positiv

```
array([[29,  8],  
       [ 1, 37]], dtype=int64)
```

13.3 Die ROC-Curve

- **Idee:**

Wir betrachten Wahrscheinlichkeiten, nicht die eigentlichen Ergebnisse

- **Beispiel:**

- Wir hätten einen Indikator, der sagt mit welcher Wahrscheinlichkeit ein Patient krank ist:
 - Patient 1: 90%
 - Patient 2: 52%
 - Patient 3: 10%
 - Patient 4: 1%
- Wo ist nun die Entscheidungsgrenze? – Bei 50% - Bei 10%?
- Die **ROC-Kurve** ist nun eine Methode, um Modelle zu vergleichen, wenn die Entscheidungsgrenzlinie nicht mittig, sondern z.B. irgendwo zwischen 1% und 10% gezogen würde.

13.4 Die ROC-Curve

- **True Positive Rate (Sensitivität / Recall):**
 - Anteil der tatsächlich Kranken (positiv = krank, negativ=gesund), bei denen die Krankheit vom Model „vorhergesagt“ bzw. „entdeckt“ wurde:

$$P(\text{positives Testergebnis} | \text{tatsächlich krank}) = \frac{r_p}{r_p + f_n}$$

13.4 Die ROC-Curve

- **True Positive Rate (TPR/ Sensitivitt / Recall):**
 - Anteil der tatschlich Kranken (positiv = krank, negativ=gesund), bei denen die Krankheit vom Model „vorhergesagt“ bzw. „entdeckt“ wurde:

$$P(\text{positives Testergebnis} | \text{tats\"achlich krank}) = \frac{r_p}{r_p + f_n}$$

- Beispiele:
 - Tatsächlich: 1 1 1 0 0 0
 - Modell sagt: 1 1 1 1 1 1
 - **TPR:** $3 / (3 + 0) = 1.0$
 - **TPR:** $0 / (0 + 0) = 0.0$
 - **TPR:** $0 / (0 + 0) = 0.0$

- Beispiel
 - Tatsächlich: 1 1 1 1 0 0 0 1 1 1 0 0 0
 - Modell sagt: 1 1 0 0 1 1 0 1 1 1 0 0 0
 - **TPR:** $2 / (2 + 2) = 0,5$ **TPR:** $3 / (3 + 0) = 1,0$

13.4 Die ROC-Curve

- **False Positive Rate (FPR):**
 - Anteil der Gesunden, die trotzdem ein positives Testergebnis bekommen haben (obwohl diese ja negativ sein sollten):

$$P(\text{positives Testergebnis} | \text{tatsächlich gesund}) = \frac{f_p}{r_n + f_p}$$

13.4 Die ROC-Curve

- **False Positive Rate (FPR):**

- Anteil der Gesunden, die trotzdem ein positives Testergebnis bekommen haben (obwohl diese ja negativ sein sollten):

$$P(\text{positives Testergebnis} | \text{tatsächlich gesund}) = \frac{f_p}{r_n + f_p}$$

- Beispiel:

- Tatsächlich: 1 1 1 0 0 0

- 1 1 1 1 1 1

- 0 0 0 0 0 0

- Modell sagt: 1 1 1 1 1 1

- 0 0 0 0 0 0

- 1 1 1 1 1 1

- FPR: $3 / (0 + 3) = 1.0$

- FPR: $0 / (0 + 0) = 0.0$

- FPR: $5 / (0 + 5) = 1.0$

- Beispiel

- Tatsächlich: 1 1 1 1 0 0 0

- 1 1 1 0 0 0

- Modell sagt: 1 0 0 1 1 1 0

- 1 1 1 0 0 0

- FPR: $2 / (3 + 1) = 0.5$

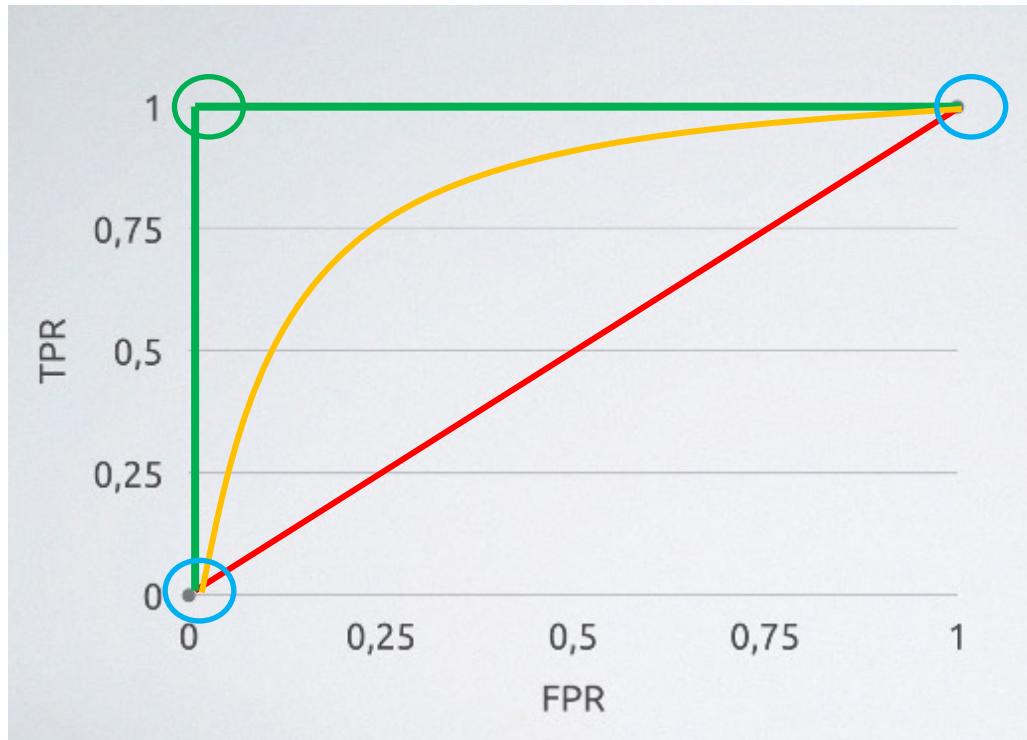
- $0 / (3 + 0) = 0.0$

13.4 Die ROC-Curve

- Beobachtung in der Realität:
 - Zur Erinnerung:
 - TPR: Anteil der tatsächlich Kranken, bei denen das Modell richtig vorhersagt
 - FPR: Anteil von Gesunden, die ein positives Ergebnis haben
 - Konkrete Extremfälle:
 - **Tatsächlich: Alle sind krank:** 1 1 1 1 1 1
 - **Modell sagt: Keine Person ist krank:** 0 0 0 0 0 0 TPR: 0 FPR: 0
 - **Tatsächlich: niemand ist krank:** 0 0 0 0 0 0
 - **Modell sagt: Alle Personen sind krank:** 1 1 1 1 1 1 TPR: 0 FPR: 1
 - **Tatsächlich und
Modell sind identisch (Optimum):** 1 1 1 0 0 0 TPR: 1 FPR: 0

13.4 Die ROC-Curve

- Beispiel von **Receiver Operating Characteristic (ROC) - Curves:**



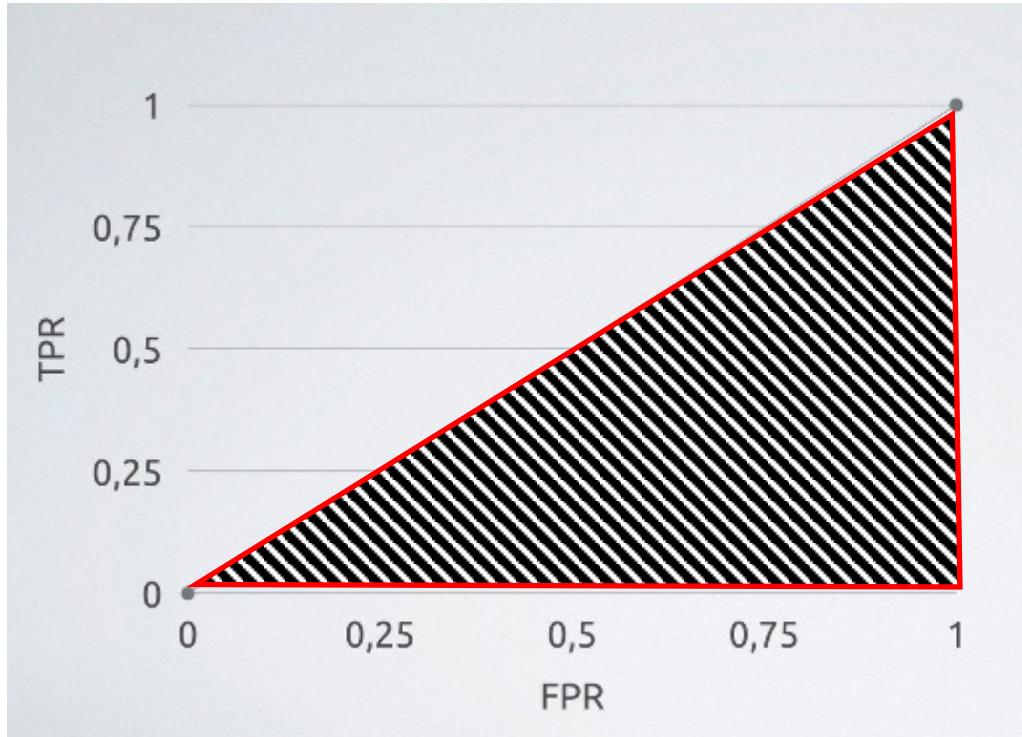
Die rote **Verbindungsline** steht für ein sehr **schlechtes** und zufälliges Modell

Die grüne **Verbindungsline** steht für das **beste** denkbare **Modell**.
FPR von 0 kombiniert mit TPR von 1

Die orangene **Verbindungsline** steht für ein Modell, mit gewissen Fehlern wie es **tatsächlich** in der Praxis vorkommen könnte.

13.4 Die ROC-Curve

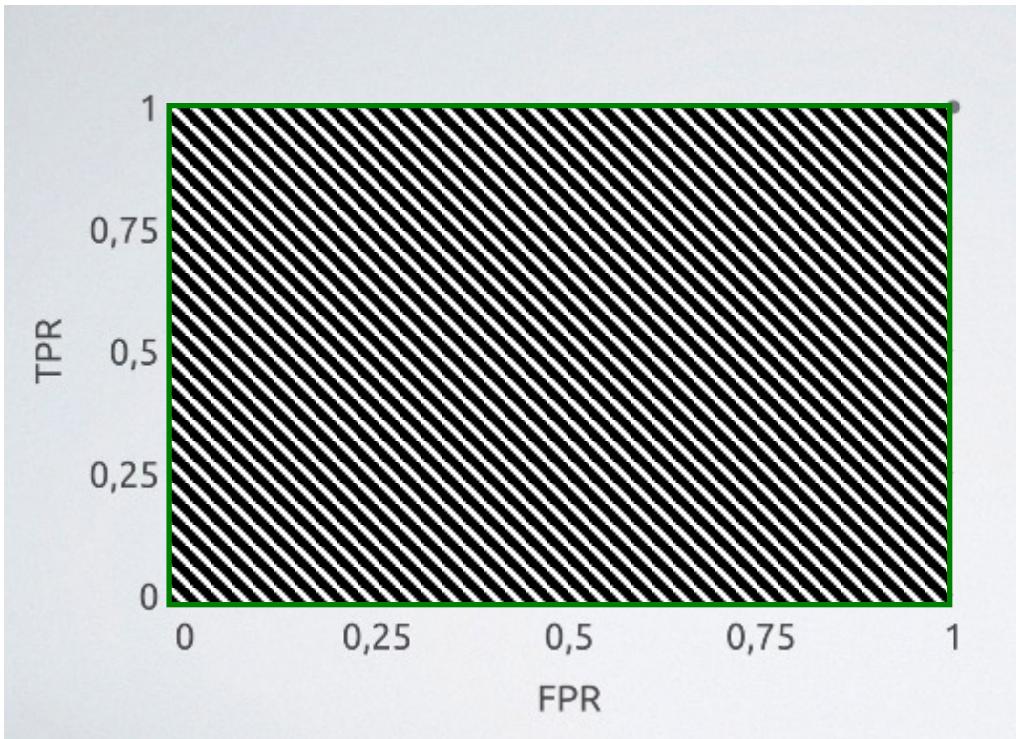
- AUC-Score aus Receiver Operating Characteristic (ROC) - Curves:



Die Fläche unter der roten
Verbindungsline
Hat einen Wert von 0,5
das ist der niedrigste hier
denkbare Wert.

13.4 Die ROC-Curve

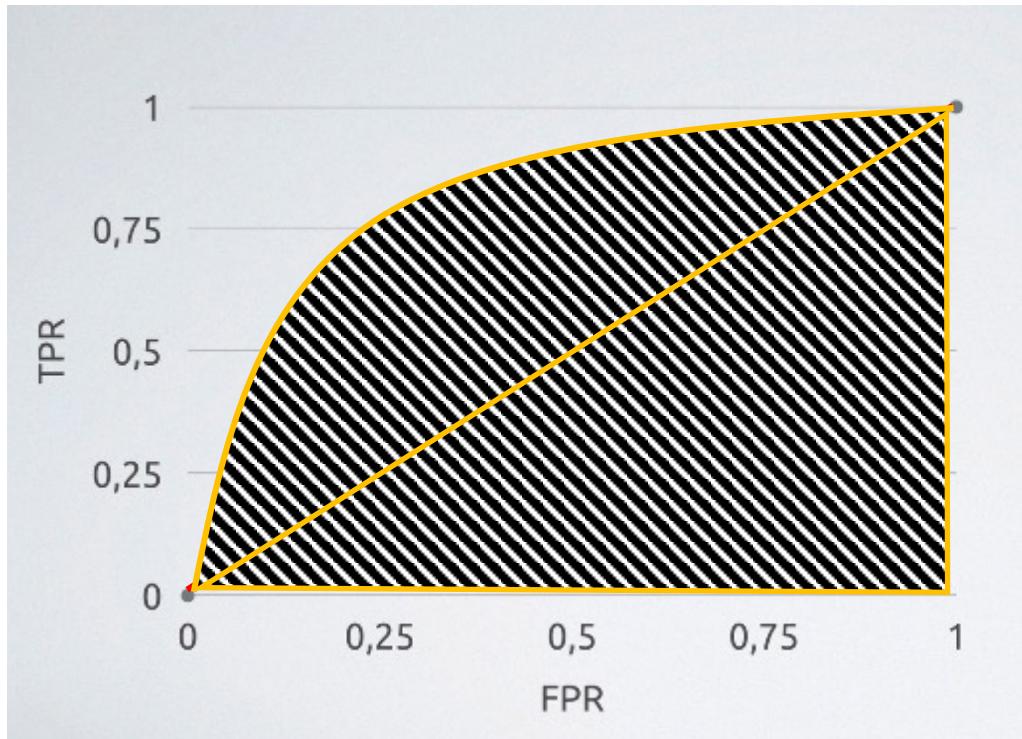
- AUC-Score aus Receiver Operating Characteristic (ROC) - Curves:



Die Fläche unter der grünen
Verbindungsline
hat einen Wert von 1
Das ist der beste hier
denkbare Wert.

13.4 Die ROC-Curve

- AUC-Score aus Receiver Operating Characteristic (ROC) - Curves:



Die Fläche unter der
orangene Verbindungsline
Hat einen Wert von z.B. 0,75

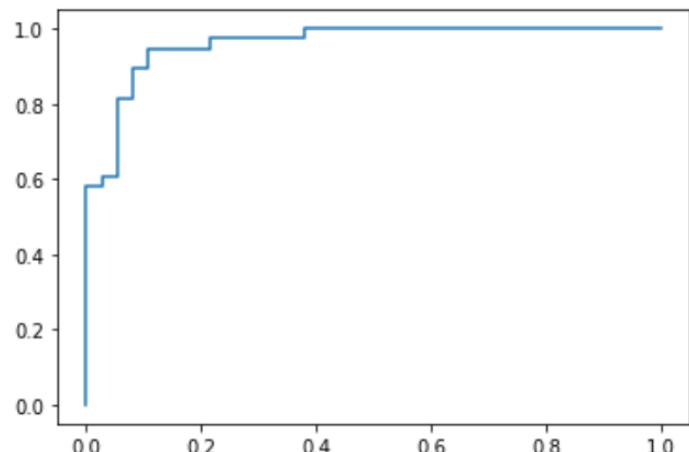
13.4 Die ROC-Curve

- **Nutzen der ROC-Curve:**
 - Das Aussehen der ROC-Kurve kann der direkten Beurteilung dienen. Je weiter links oben die Kurve verläuft, desto besser ist es
 - **Area under Curve Score (AUC-Score):**
Aus der ROC-Kurve ergibt sich ein Flächenwert, den man als Indikator für die Beurteilung von Modellen bzw. zu deren Optimierung verwenden kann.

13.4 Die ROC-Curve

– Die ROC-Curve in Python:

- Beispiel anhand der Logistischen Regression, da hier Zugehörigkeitswahrscheinlichkeiten von Klassen erzeugt werden.
- **Python: ROC-Curve.ipynp**
- **Daten: classification.csv**



Area under Curve Score:

```
roc_auc_score(y_test, y_test_proba)
```

0.9601706970128023

```
from sklearn.metrics import roc_curve, roc_auc_score  
fpr, tpr, thresholds = roc_curve(y_test, y_test_pred)
```

```
%matplotlib inline  
import matplotlib.pyplot as plt  
plt.plot(fpr, tpr)  
plt.show()
```

```
model.predict_proba(X_test)
```

```
array([[1.25569216e-02, 9.87443078e-01],  
[6.10372152e-01, 3.89627848e-01],  
[9.72537407e-01, 2.74625928e-02],
```

```
[8.01568647e-01, 9.93596550e-01],
```

```
[9.93596550e-01, 1.0]
```

```
[:, 1]
```

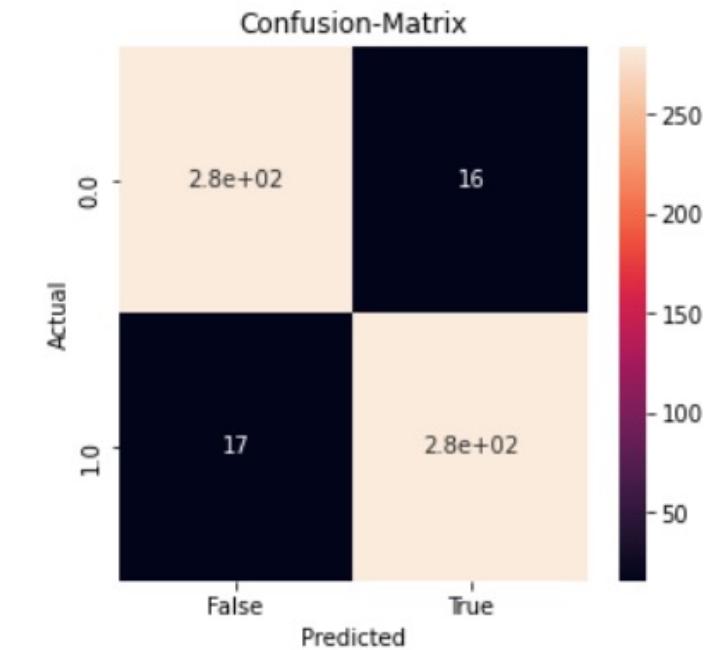
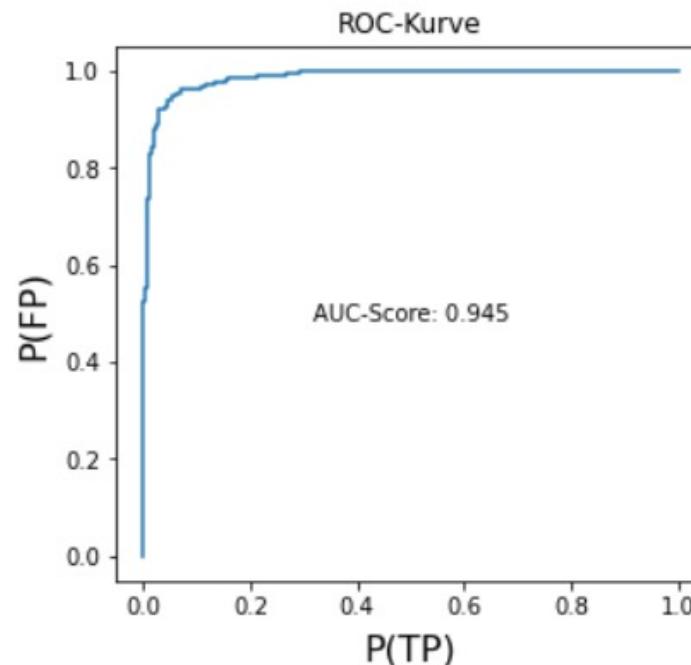
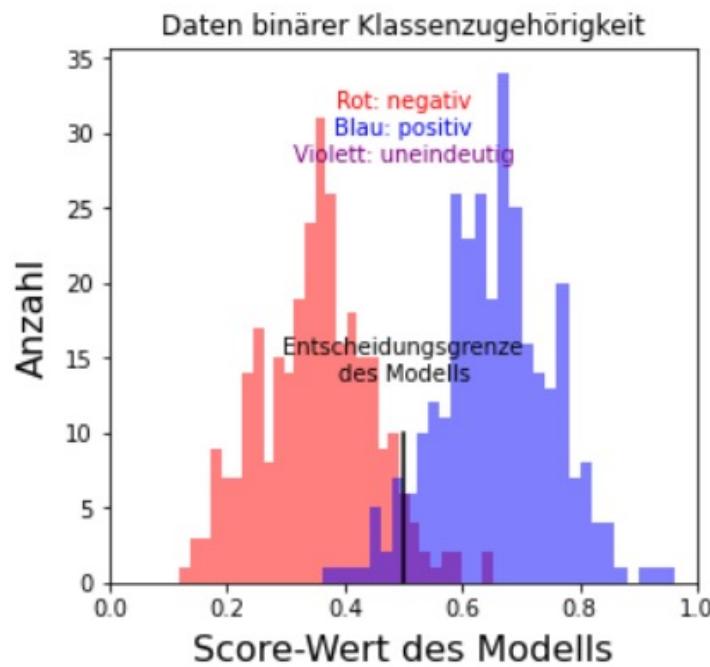
Nur die Wahrscheinlichkeit, dass X = 1 ist

```
y_test_proba
```

```
9.87443078e-01]  
3.89627848e-01]  
2.74625928e-02]  
8.01568647e-01]  
9.93596550e-01]
```

13.4 Die ROC-Curve

- Fortsetzung ROC-Curve:
 - Beispiel anhand einer simulierten binären Klassifikation.
 - Python: [ROC-Curve.ipynp](#)
 - Daten: Zwei Gauss-Verteilungen (Mediane können interaktiv verändert werden)



13.5 Hyperparameter

- **Idee: Struktur eines Modells von außen beeinflussen.** Z.B.:
 - k bei k-NN
 - Gamma und C bei einer SVM
 - Anzahl Levels eines Entscheidungsbaums
 - ...
- **Mögliche Ziele:**
 - Abwägung zwischen Overfitting und Underfitting
 - Maximierung von R^2
 - Maximierung von AUC
 -
- **Mögliches Vorgehen:**
 - Verschiedene Hyperparameter ausprobieren
 - Systematisches Optimieren von Hyperparameter durch sog. **Pipelining**

13.5 Hyperparameter

- Die Scikit-Learn-Pipeline

- Python: **Hyperparameter optimieren (Pipeline).ipynb**

- Motivation: Dieser *Teil* soll mehrfach wiederholt werden, wobei dann der Wert `n_neighbours` später immer anders sein kann.

```
1 from sklearn.preprocessing import StandardScaler
2
3 scaler = StandardScaler()
4 scaler.fit(X_train)
5
6 X_train = scaler.transform(X_train)
7 X_test = scaler.transform(X_test)
```



```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 model = KNeighborsClassifier(n_neighbors = 3)
4 model.fit(X_train, y_train)
5
6 print(model.score(X_test, y_test))
```

- Für jedes Training muss aber der Scaler immer jeweils neu auf die Trainingsdaten angewendet werden. Eine Pipeline stellt eine Umgebung zur Verfügung, die mehrfache Ausführungen unterstützt und die ein Modell z.B. mit einem Scaler kombiniert aufrufbar macht:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier(n_neighbors = 3))
])

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

pipeline.fit(X_train, y_train)
pipeline.score(X_test, y_test)

0.9733333333333338
```

13.5 Hyperparameter

- Das GridSearchCV
 - Python: **Hyperparameter optimieren (GridSearchCV).ipynb**
 - Prinzip: Man kann Parameter in einer Pipeline mit der Funktion set_params abändern: <id>__<variable> = ...
- Mehrere Werte können mit der Klasse GridSearchCV nacheinander probiert werden:

```
from sklearn.model_selection import GridSearchCV

clf = GridSearchCV(pipeline, param_grid = {
    "knn__n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
})
clf.fit(X, y)

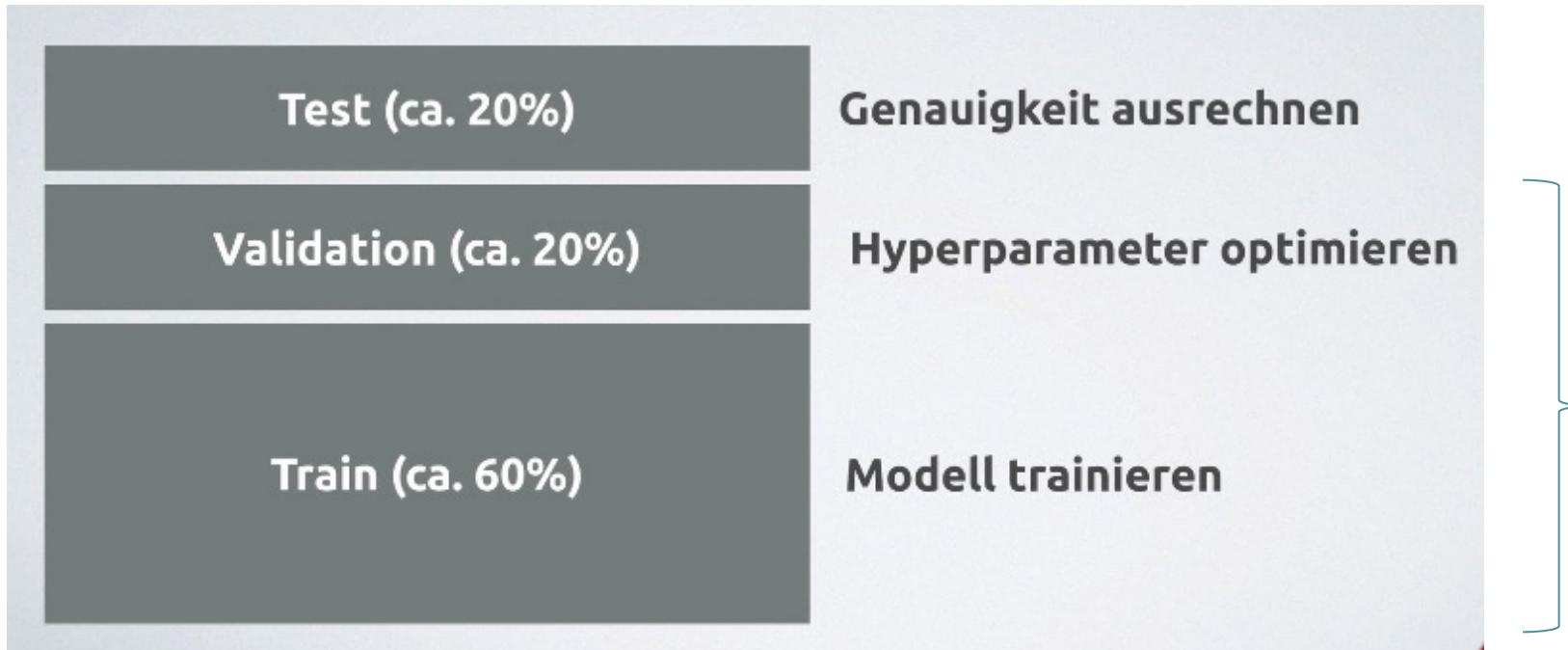
print(clf.best_params_)
{'knn__n_neighbors': 10}
```

```
pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier(n_neighbors = 3))
])
```

Bester Wert für
n_neighbours: ist also 10

13.6 Train / Test / Validation

- Prinzip:



...kann auch in Form einer
Cross-Validation
realisiert sein...

13.6 Train / Test / Validation

- Python: Hyperparameter optimieren (Train, Validation, Test).ipynb

```
from sklearn.model_selection import GridSearchCV

clf = GridSearchCV(pipeline, param_grid = {
    "knn__n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
})
clf.fit(X_train, y_train)

print(clf.best_params_)
```

{'knn__n_neighbors': 3}

```
# das sind komplett unbekannte Daten:
print(clf.score(X_test, y_test))
```

0.9333333333333333

```
#das sind Trainingsdaten
print(clf.best_score_)
```

0.9370707070707069

13.6 Train / Test / Validation

- Python: **Hyperparameter optimieren (Musterlösung).ipynb**
- **Ergebnisse:**

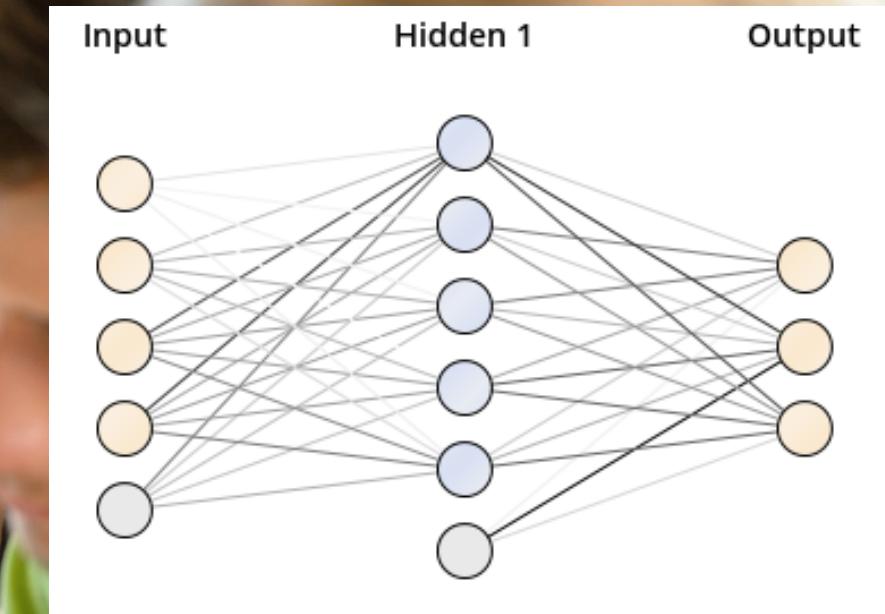
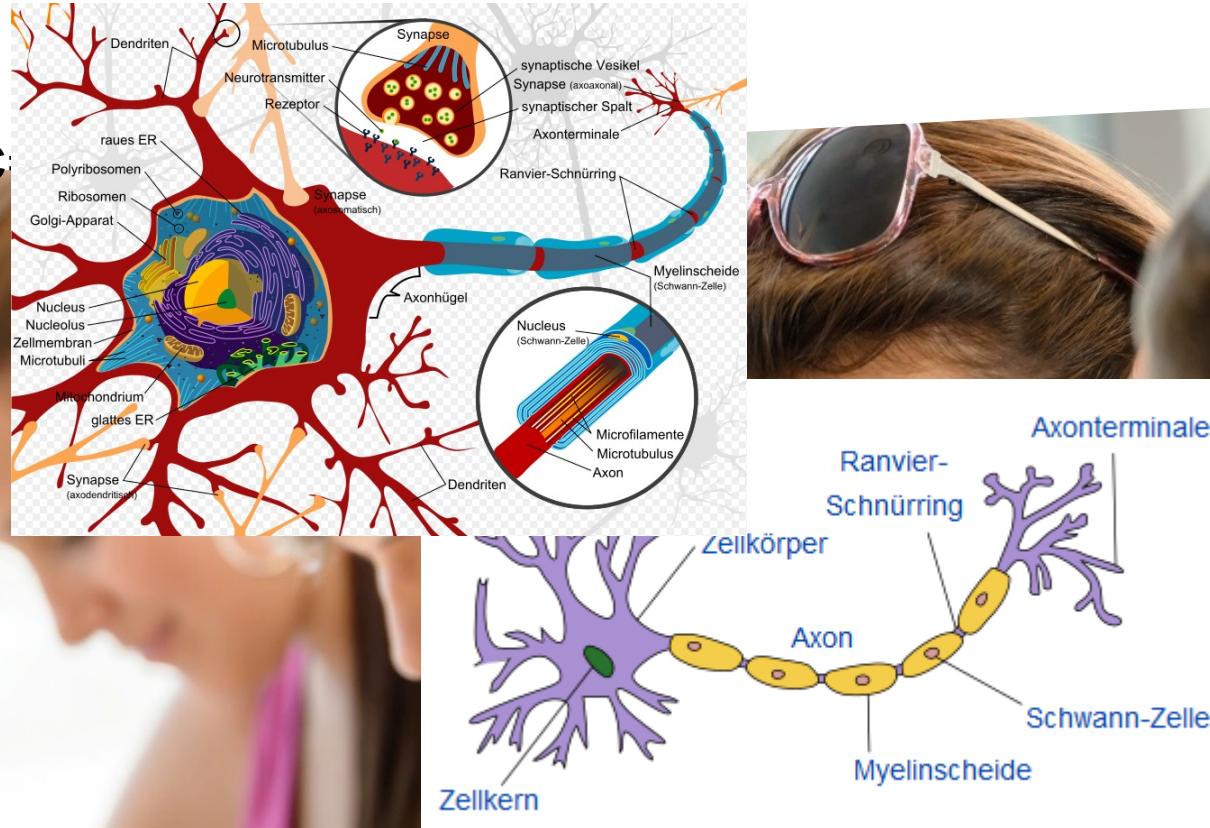
```
print(clf.best_params_)

{'svm__C': 1.5, 'svm__gamma': 1}
```

```
: print(clf.score(X_validation, y_validation))
```

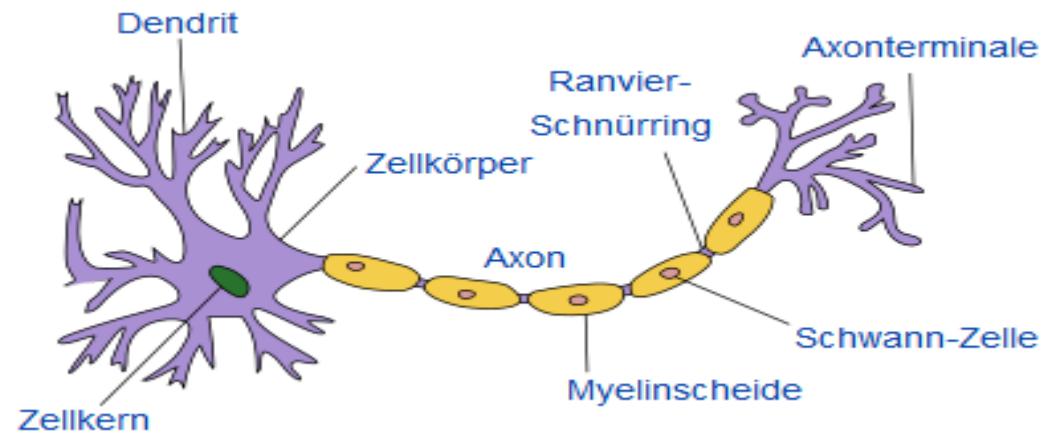
```
0.933333333333
```

14. Klassifizierung mit Neurolalen Netzen

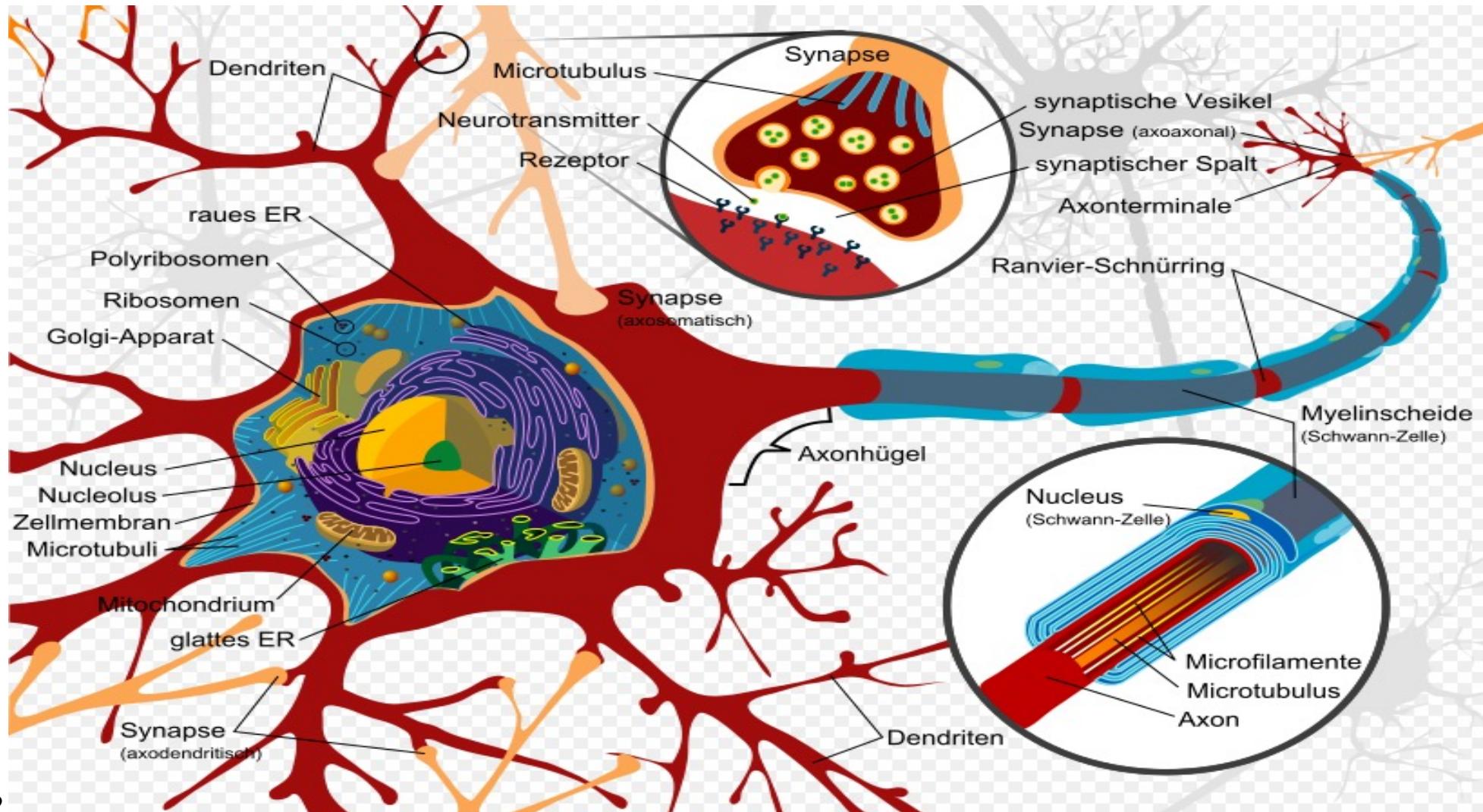


14.1 Das biologische Vorbild Neuonaler Netze

- Vorbild ist das Gehirn: Ein Dendritenbaum einer Nervenzelle kann mehrere Tausend Synapsenkontakte haben.
- Idee: Nevezellen (Neuronen) im Computer simulieren.

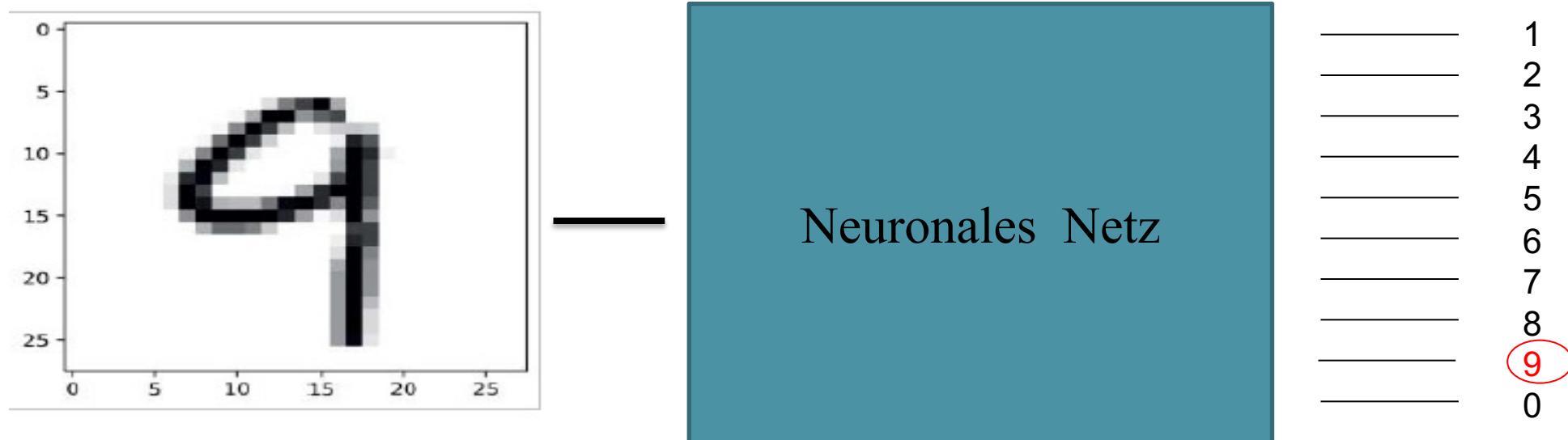


14.1 Das biologische Vorbild Neuronaler Netze



14.2 Artificial Intelligence – Das Neuronale Netz war zuerst da

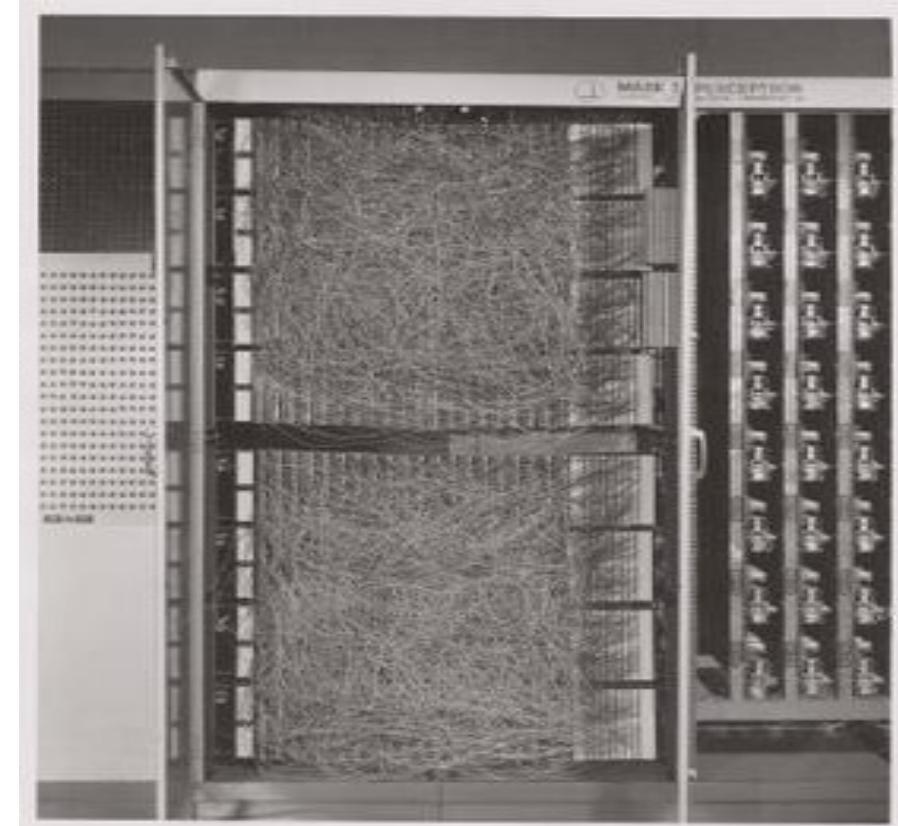
Beispiel Zeichenerkennung:



14.2 Artificial Intelligence – Das Neuronale Netz war zuerst da

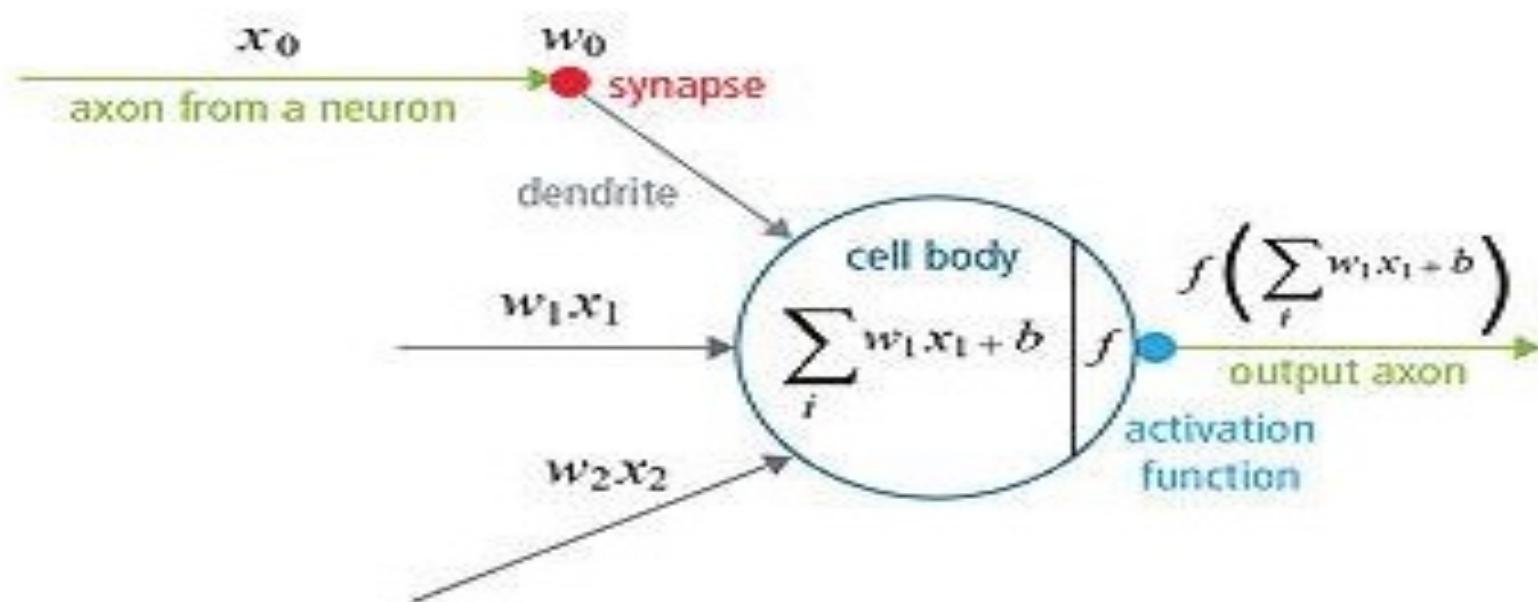
Historie

- 1954; Minski: *Neural Nets and the Brain Model Problem*, Dissertation
- 1956: Minski et al. begründen den Begriff AI
- 1958: MARK I: Erste Implementation einer Perceptron Engine.
20x20 Fotozellen
Potenziometer mit Motor zum Lernen
(rechts)



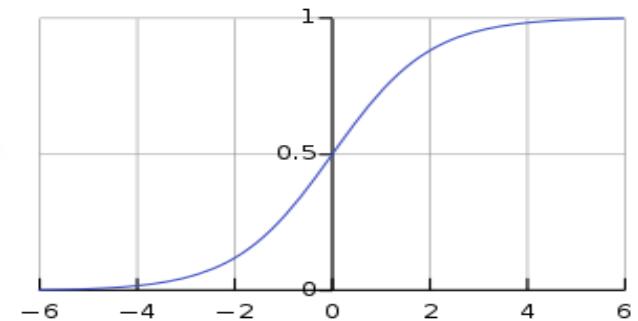
14.3 Vom Perceptron zum Neuronalen Netz

- Das Perceptron - Mathematisches Modell eines Neurons:



- x: Eingänge
- f: Aktivierungsfunktion
(oft als S oder σ geschrieben)
- w: Gewichte der Eingänge
- b: zus. „Hebewert“ (bias)

Typische Aktivierungsfunktion
 f ist die Sigmoid-Funktion,
die auch als **Logistische Funktion**
bezeichnet wird.

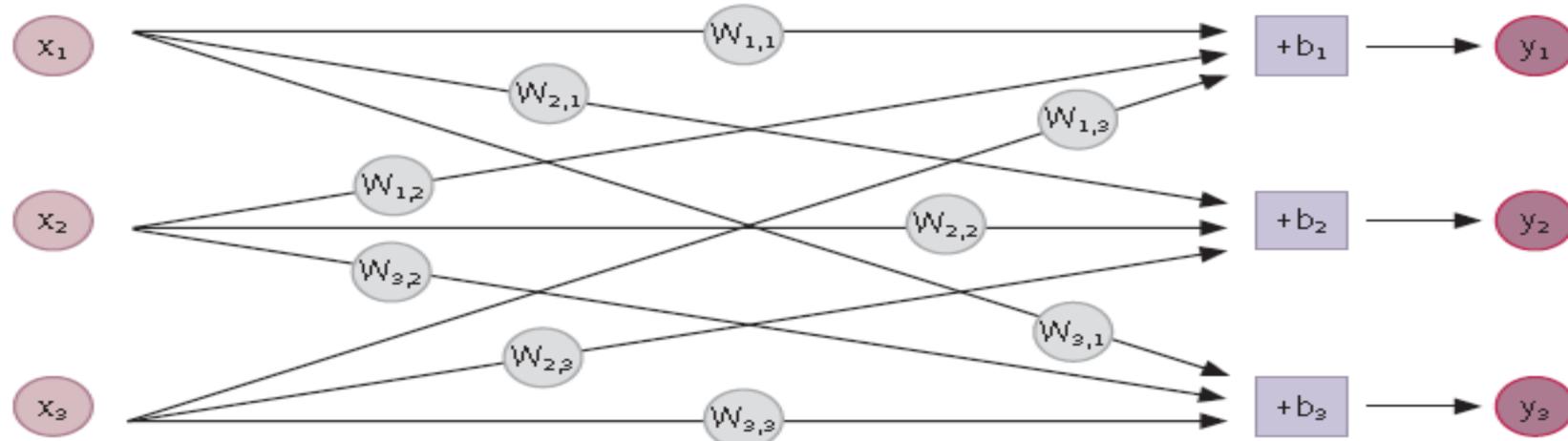


$$S(x) = \frac{1}{1 + e^{-x}}$$

Effekt:
negative Werte werden ≈ 0
große Werte ≈ 1

14.3 Vom Perceptron zum Neuronalen Netz

- Jede Eingabe X beeinflusst jede Ausgabe Y mit einem bestimmten Faktor W



... 1. Anmerkung: Gehirne sind aber komplizierter!

... 2. Anmerkung: Zwischenlayer fehlt auch noch...

14.3 Vom Perceptron zum Neuronalen Netz

- Jede Eingabe X beeinflusst jede Ausgabe Y mit einem bestimmten Faktor W

als Formel:

$$\begin{aligned}y_1 &= W_{1,1} x_1 + W_{1,2} x_2 + W_{1,3} x_3 + b_1 \\y_2 &= W_{2,1} x_1 + W_{2,2} x_2 + W_{2,3} x_3 + b_2 \\y_3 &= W_{3,1} x_1 + W_{3,2} x_2 + W_{3,3} x_3 + b_3\end{aligned}$$

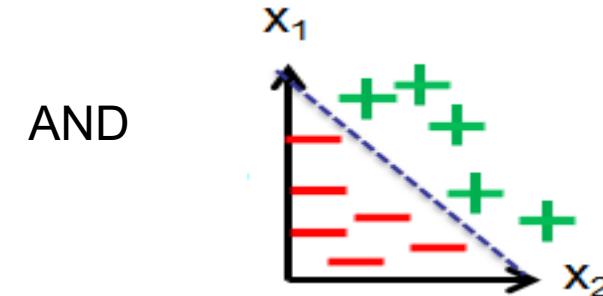
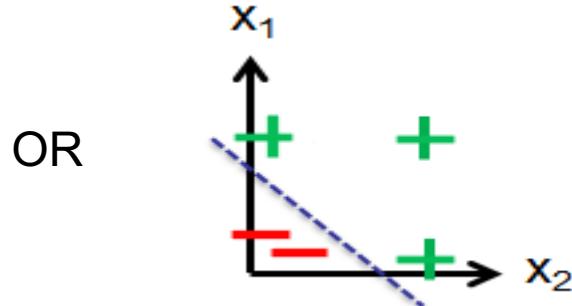
in Matrix-Schreibweise:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

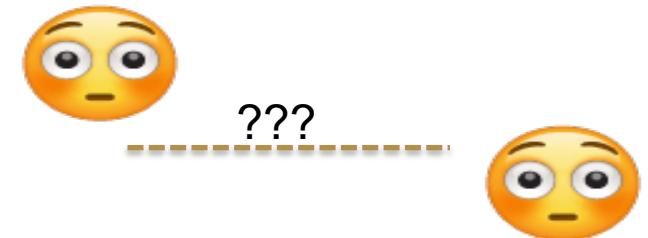
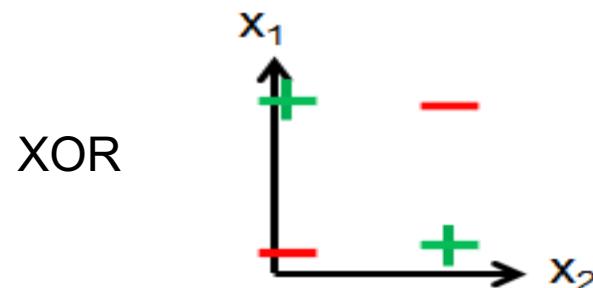
14.3 Vom Perceptron zum Neuronalen Netz

- Das XOR-Problem

- AND, OR und NOT sind mit einem Perceptron darstellbar.

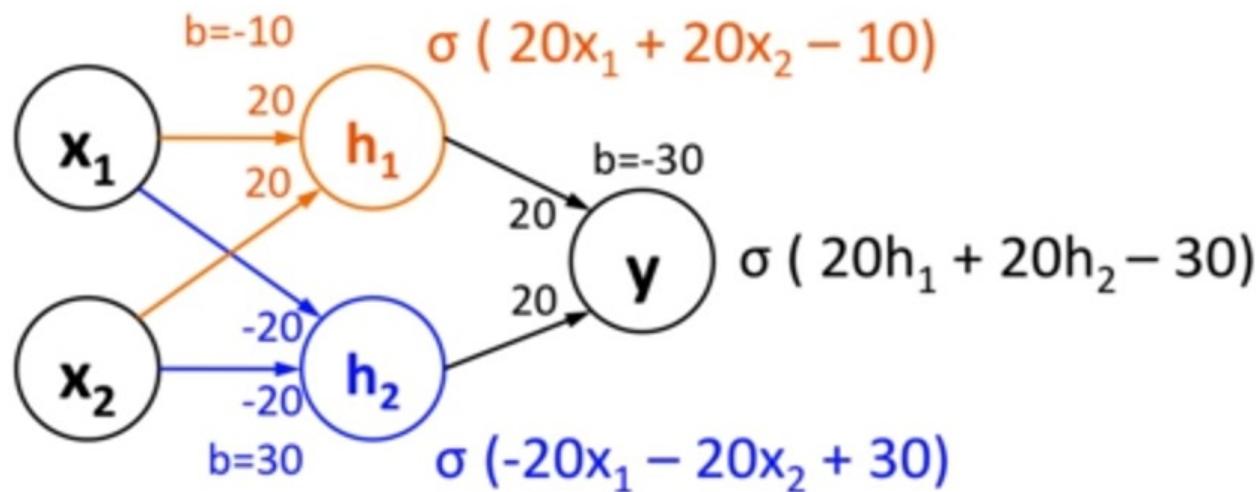
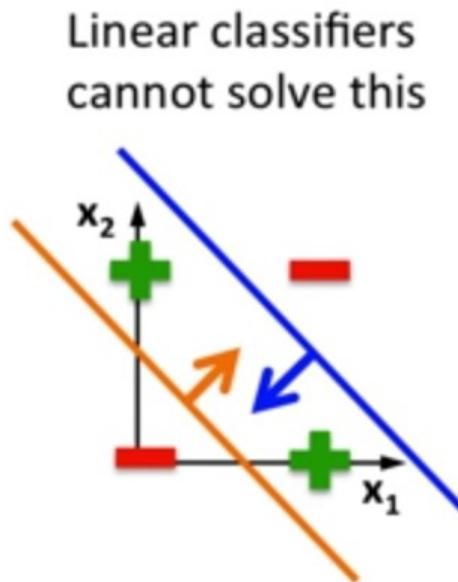


- nicht aber das **XOR** (1. Krise der KI 70er)



14.4 Das Multilayer Perceptron / Neuronale Netze mit Hidden Layer lösen XOR

- Lösung des XOR-Problems mit Multilayer-Perceptrons MLP (Hidden Layers):



$$\sigma(20 \cdot 0 + 20 \cdot 0 - 10) \approx 0$$

$$\sigma(20 \cdot 1 + 20 \cdot 1 - 10) \approx 1$$

$$\sigma(20 \cdot 0 + 20 \cdot 1 - 10) \approx 1$$

$$\sigma(20 \cdot 1 + 20 \cdot 0 - 10) \approx 1$$

$$\sigma(-20 \cdot 0 - 20 \cdot 0 + 30) \approx 1$$

$$\sigma(-20 \cdot 1 - 20 \cdot 1 + 30) \approx 0$$

$$\sigma(-20 \cdot 0 - 20 \cdot 1 + 30) \approx 1$$

$$\sigma(-20 \cdot 1 - 20 \cdot 0 + 30) \approx 1$$

$$\sigma(20 \cdot 0 + 20 \cdot 1 - 30) \approx 0$$

$$\sigma(20 \cdot 1 + 20 \cdot 0 - 30) \approx 0$$

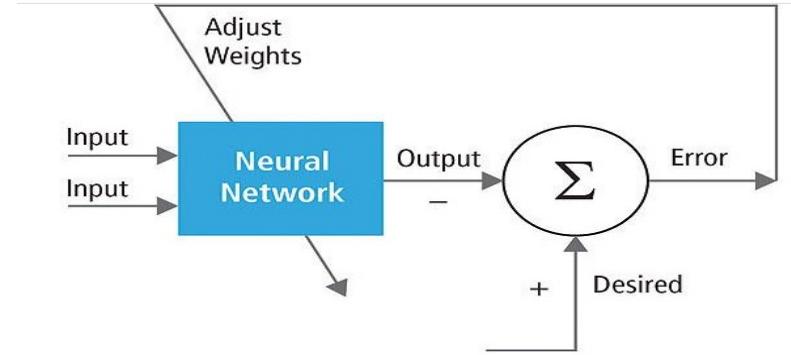
$$\sigma(20 \cdot 1 + 20 \cdot 1 - 30) \approx 1$$

$$\sigma(20 \cdot 1 + 20 \cdot 1 - 30) \approx 1$$

14.5 Backpropagation – Neuronale Netze lernen langsam aber gezielt

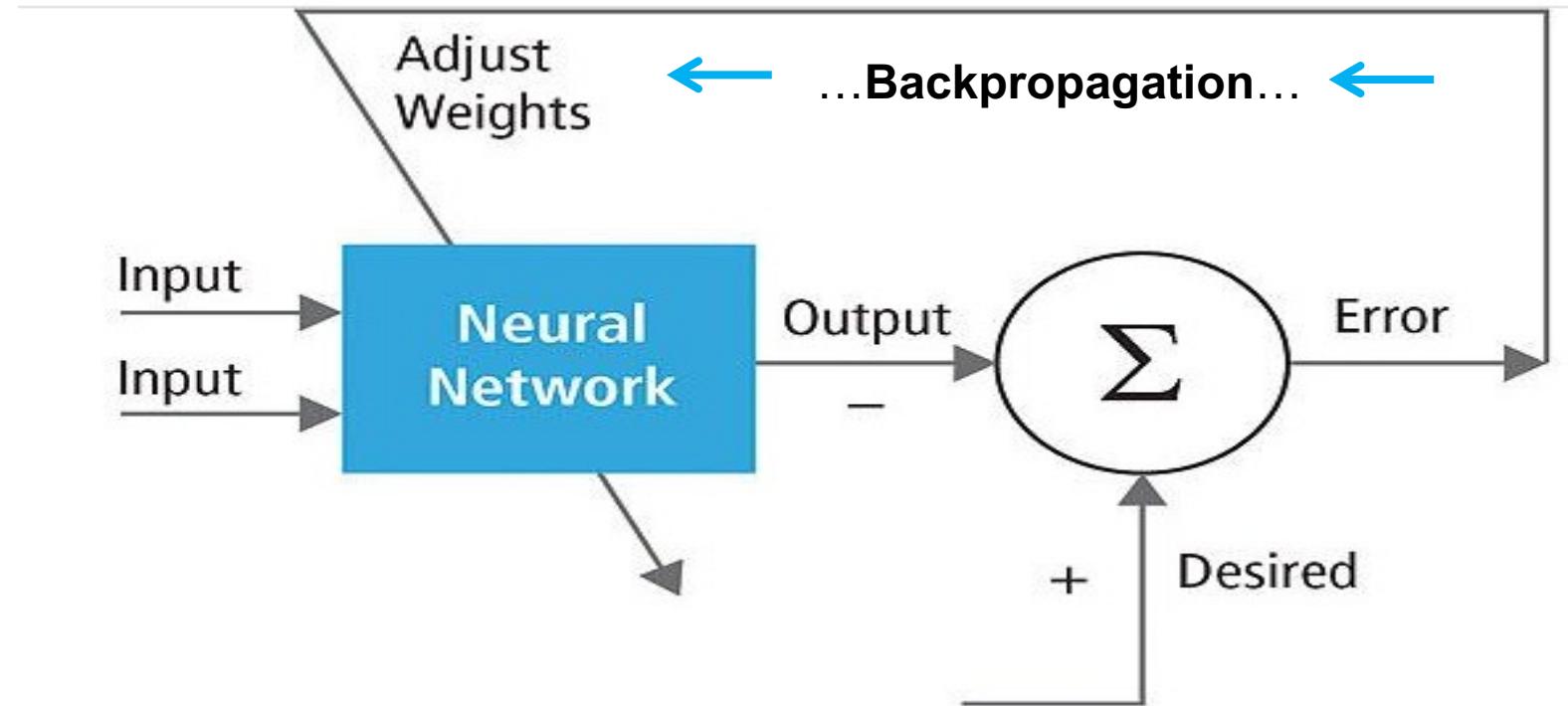
Backpropagation für Multilayer Perceptrons

- Rumelhart et al: 1986
- Jeder Ausgang j hat einen Targetwert: t_j
- Ausgabe des Neurons i : o_i
- Error: $E = \frac{1}{2} \sum (t_i - o_i)^2$
- Lernrate (Veränderung pro Iteration): η
- Gewichtsänderung pro Iteration: $\Delta w_{ij} = -\eta \delta_j o_i$
- δ_j ist enthält die Ableitung σ' der Aktivierungsfunktion σ und enthält damit auch die Ableitungen der σ -Funktionen der vorangegangenen Neuronen.
- Wiederhole: $w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta w_{ij}$ solange bis E akzeptabel



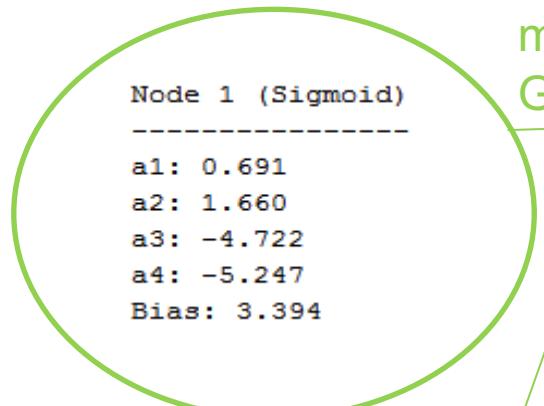
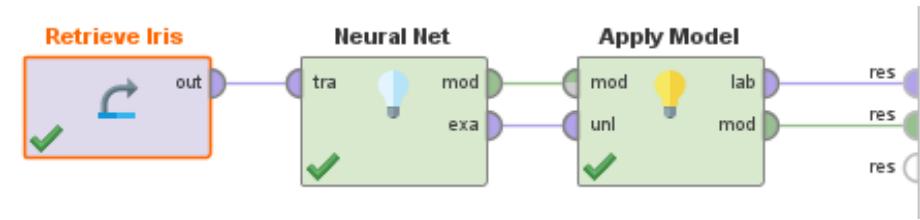
14.5 Backpropagation – Neuronale Netze lernen langsam aber gezielt

- Gewichte müssen „trainiert“ werden.
- Viele tausend mal „rotiert“ hierzu das folgende System:

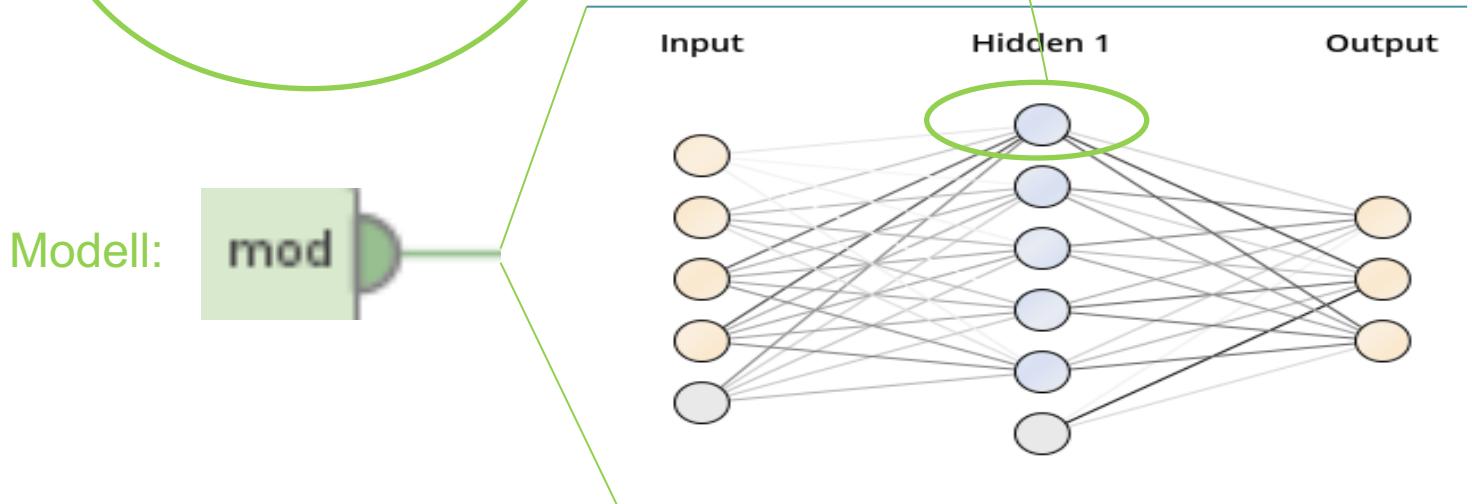


14.6 Neuronale Netze im praktischen Einsatz

– Neuronale Netze mit RapidMiner



Eingänge eines Perceptrons
mit deren jew.
Gewicht (Trainingsergebnis)



14.6 Neuronale Netze im praktischen Einsatz

Trainingsgrundlage
(Learning-Target)

Vorhersage
(Größter Wert der
Ergebnisse des
Neuronalen Netzes)

Ergebnisse der Neuronalen-Netzes



label	prediction(label)	confidence(Iris-setosa)	confidence(Iris-versicolor)	confidence(Iris-virginica)
Iris-setosa	Iris-setosa	0.956	0.044	0.000
Iris-setosa	Iris-setosa	0.873	0.127	0.000
Iris-setosa	Iris-setosa	0.953	0.047	0.000
Iris-setosa	Iris-setosa	0.939	0.061	0.000
Iris-setosa	Iris-setosa	0.953	0.047	0.000
Iris-setosa	Iris-setosa	0.940	0.060	0.000
Iris-setosa	Iris-setosa	0.960	0.040	0.000
Iris-setosa	Iris-setosa	0.952	0.048	0.000
Iris-setosa	Iris-setosa	0.959	0.041	0.000
Iris-setosa	Iris-setosa	0.954	0.046	0.000
Iris-versicolor	Iris-versicolor	0.013	0.962	0.024
Iris-versicolor	Iris-versicolor	0.012	0.953	0.034
Iris-versicolor	Iris-versicolor	0.007	0.900	0.093
Iris-versicolor	Iris-versicolor	0.010	0.953	0.037
Iris-versicolor	Iris-versicolor	0.007	0.996	0.107

14.6 Neuronale Netze im praktischen Einsatz

– Neuronale Netze mit Python

Code: **Keras-Beispiel.ipynb**

Beispieldaten:

- Input Variables (X):
 - Number of times pregnant
 - Plasma glucose concentration a 2 hours in an oral glucose tolerance test
 - Diastolic blood pressure (mm Hg)
 - Triceps skin fold thickness (mm)
 - 2-Hour serum insulin (mu U/ml)
 - Body mass index (weight in kg/(height in m)²)
 - Diabetes pedigree function
 - Age (years)
- Output Variables (y):
 - Class variable (0 or 1)

Struktur des Netzes und Typ der Aktivierungsfunktion
(Erklärung zu ReLU siehe hinten)

```
# define the keras model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Festlegung, wie die Backpropagation funktioniert.

Adam: Kombination aus

- Root Mean Square Propagation (RMSProp)
- MOMENTUM: Stochastisch erzeugte Mini-Batches, damit lokale Minima unwahrscheinlich werden

```
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
```

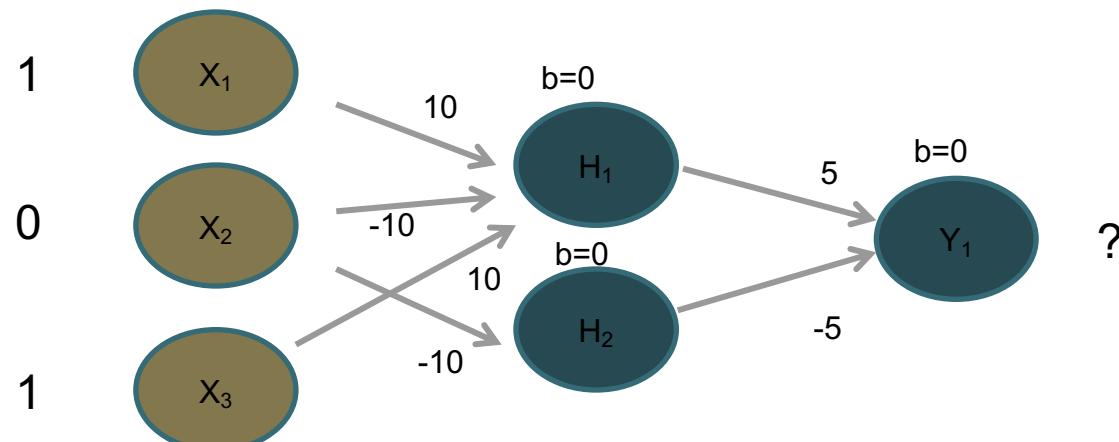
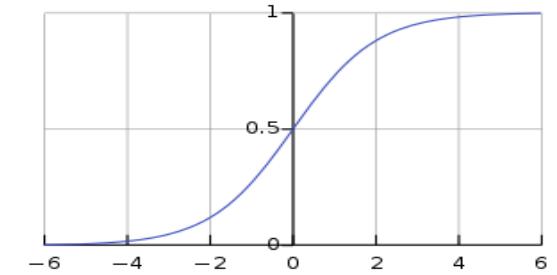
```
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

24/24 [=====] - 0s 478us/step - loss: 0.4699 - accuracy: 0.7760
Accuracy: 77.60

Übung

- Gegeben sei folgende Aktivierungsfunktion:
- Gegeben sei folgendes Neuronales Netz mit Eingängen X_{1-3} und dem Ausgang an Y_1 . (Nur an H und Y befindet sich die Aktivierungsfunktion σ)

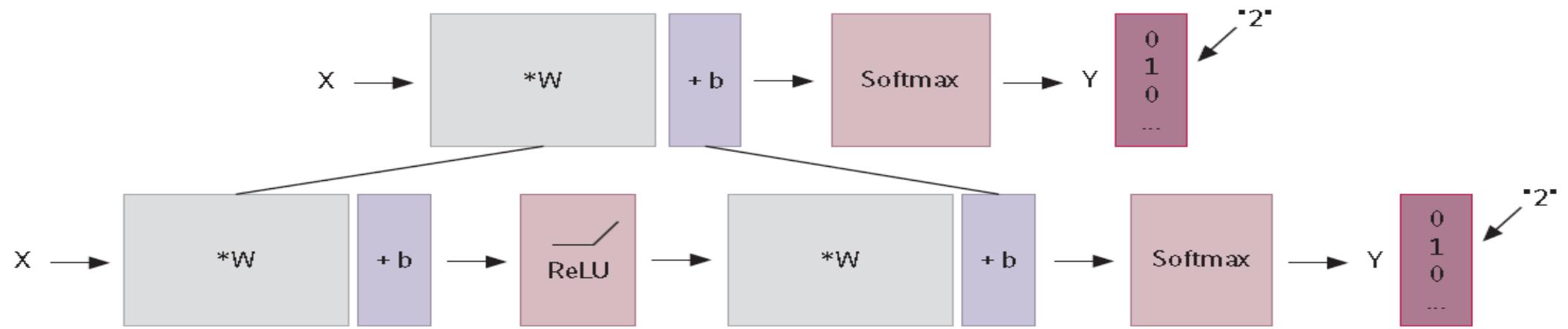
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



- Berechnen Sie Y_1 (Die Ergebnisse der σ Funktion können geschätzt werden).

14.7 Deep-Learning

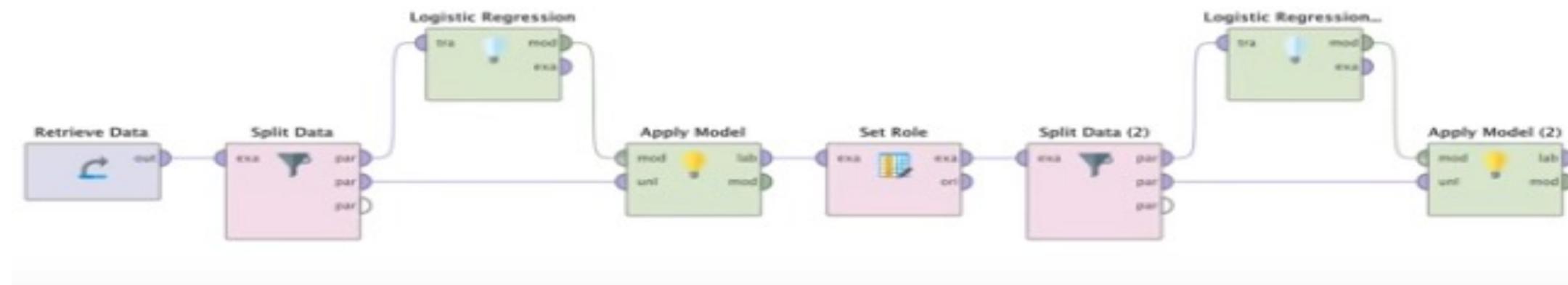
- Tiefe Neuronale Netze haben viele Zwischenlayer, die ggf. mit Schachtelungen realisiert sind:



- ReLU gibt Werte nur ab einer Schwelle weiter (s.h.)
- Softmax (ähnlich Sigmoid) normiert auf den Bereich $[0,1]$, hebt große Werte hervor und verkleinert kleine Werte (s.h.)

14.7 Deep-Learning

- Das hierarchische Stapeln geht auch in RapidMiner:

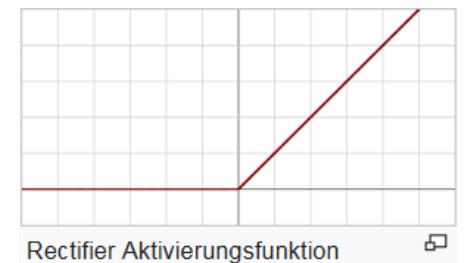


14.7 Deep-Learning

– Aktivierungsfunktionen im Deep-Learning:

- **Die Sigmoid-Funktion**: $\sigma(x) = \frac{1}{1 + e^{-x}}$ ist noch gebräuchlich, ist aber so etwas wie „Old School“.
- **ReLU: „Rectifier Linear Unit“** ist eine alternative Aktivierungsfunktion. Sie ist einfacher, performanter und besser. 2011 wurde nachgewiesen, dass ReLU im Deep-Learning bessere Ergebnisse bringt, als die Sigmoid-Funktion.
- **Softmax**: Bei Klassifizierern mit sich gegenseitig ausschließenden Klassen, bringt Softmax Vorteile: Er hebt große Werte hervor und verkleinert kleine

$$\phi(z) = \max(0, z)$$



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

$$\vec{z} \text{ sei } \begin{bmatrix} 8 \\ 5 \\ 0 \end{bmatrix}$$

$$\begin{aligned} e^{z_1} &= e^8 = 2981.0 \\ e^{z_2} &= e^5 = 148.4 \\ e^{z_3} &= e^0 = 1.0 \end{aligned}$$

$$\sum_{j=1}^K e^{z_j} = e^{z_1} + e^{z_2} + e^{z_3} = 2981.0 + 148.4 + 1.0 = 3130.4$$

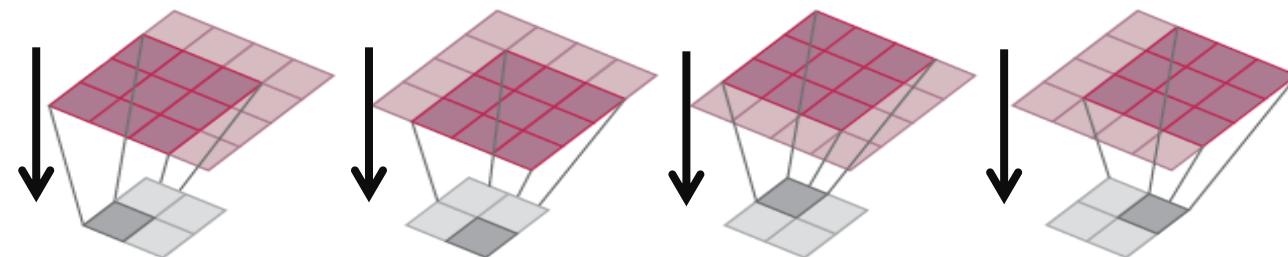
$$\begin{aligned} \sigma(\vec{z})_1 &= \frac{2981.0}{3130.4} = 0.9523 \\ \sigma(\vec{z})_2 &= \frac{148.4}{3130.4} = 0.0474 \\ \sigma(\vec{z})_3 &= \frac{1.0}{3130.4} = 0.0003 \end{aligned}$$

$$= \sigma(\vec{z})$$

14.7 Deep-Learning

- **Dense-Layers:** Diese stellen den Normalfall dar, wo alle Neuronen mit allen Neuronen der vorhergehenden Ebene verbunden sind.
- **Dropout:** Regularisierungsmethode zur Vermeidung von Overfitting. Es wird ein bestimmter Anteil an Neuronen (z.B. 30%) einfach ausgeschaltet. Das ist also das Gegenteil eines Dense-Layers.
- **Convolutional Filtering :** Ist Zusätzlich im Deep Learning üblich. Bei sehr vielen Eingabewerten (Bilder oder Klänge) muss zwischen den Ebenen eine spezielle *Faltungsverarbeitung* erfolgen. Wird vor allem in der Bildverarbeitung verwendet, um die Anzahl der Pixel zu reduzieren.

Anschaulich:



14.7 Deep-Learning

– Beispiel: Convolutional Filtering

Quelle: github.com/udacity/conv_2d_tensorflow

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3 ₀	2 ₁	1 ₂	0
0	0 ₂	1 ₂	3 ₀	1
3	1 ₀	2 ₁	2 ₂	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2 ₀	1 ₁	0 ₂
0	0	1 ₂	3 ₂	1 ₀
3	1	2 ₀	2 ₁	3 ₂
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0 ₀	0 ₁	1 ₂	3	1
3 ₂	1 ₂	2 ₀	2	3
2 ₀	0 ₁	0 ₂	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0 ₀	1 ₁	3 ₂	1
3	1 ₂	2 ₂	2 ₀	3
2	0 ₀	0 ₁	2 ₂	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3 ₀	1 ₁	2 ₂	2	3
2 ₂	0 ₂	0 ₀	2	2
2 ₀	0 ₁	0 ₂	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1 ₀	2 ₁	2 ₂	3
2	0 ₂	0 ₂	2 ₀	2
2	0 ₀	0 ₁	0 ₂	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1 ₀	3 ₁	1 ₂
3	1	2 ₂	2 ₂	3 ₀
2	0	0 ₀	2 ₁	2 ₂
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

0	1	2
2	2	0
0	1	2

mit Filter-Kernell

Übung

- Gegeben sei folgende Matrix:
- Gegeben sei der Filter-Kernel:
- Frage: Welchen Wert ergibt das Convolutional Filtering?

- Antwort:

$$0*0 + 1*1 + 3*2 + 1*2 + 2*2 + 2*0 + 0*0 + 0*1 + 2*2 =$$

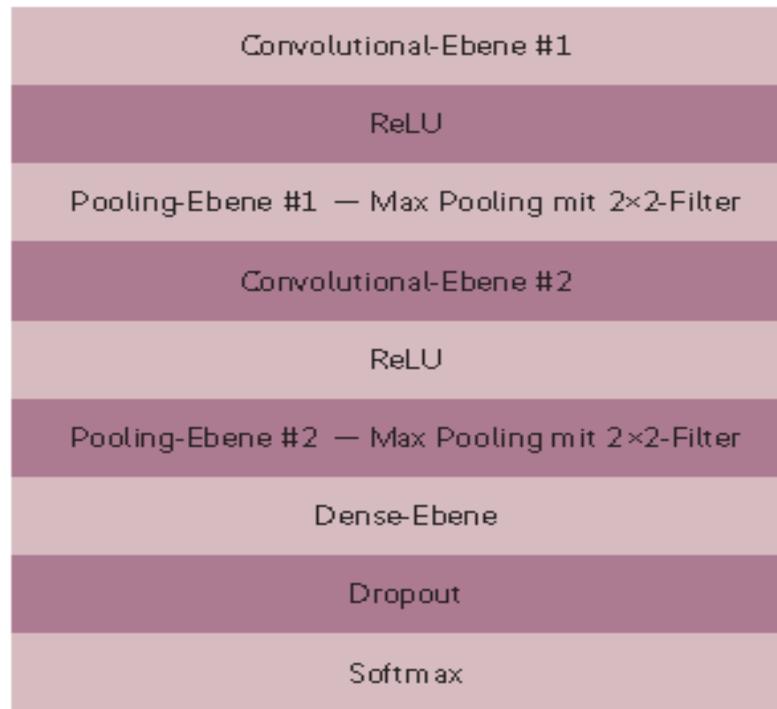
$$1 + 6 + 2 + 4 + 4 = \mathbf{17}$$

0 ₀	1 ₁	3 ₂
1 ₂	2 ₂	2 ₀
0 ₀	0 ₁	2 ₂

0	1	2
2	2	0
0	1	2

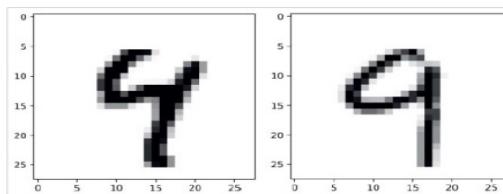
14.7 Deep-Learning

- Typische Ebenenstruktur beim Deep-Learning



14.7 Deep-Learning

- Typische Performance:
 - Handgeschriebene Zahlen können mit DeepLearning zu ca. 97% erkannt werden.
 - Bei normalen Neuronalen Netzen nur ca. 92%
- Typische Unterscheidungsprobleme, die für uns Menschen unproblematisch sind, aber die zu gelegentlichen Fehlern führen:



14.7 Deep-Learning

Deep-Learning hat Grenzen.

Die aktuelle Technologie ist hier noch überfordert...



14.7 Deep-Learning



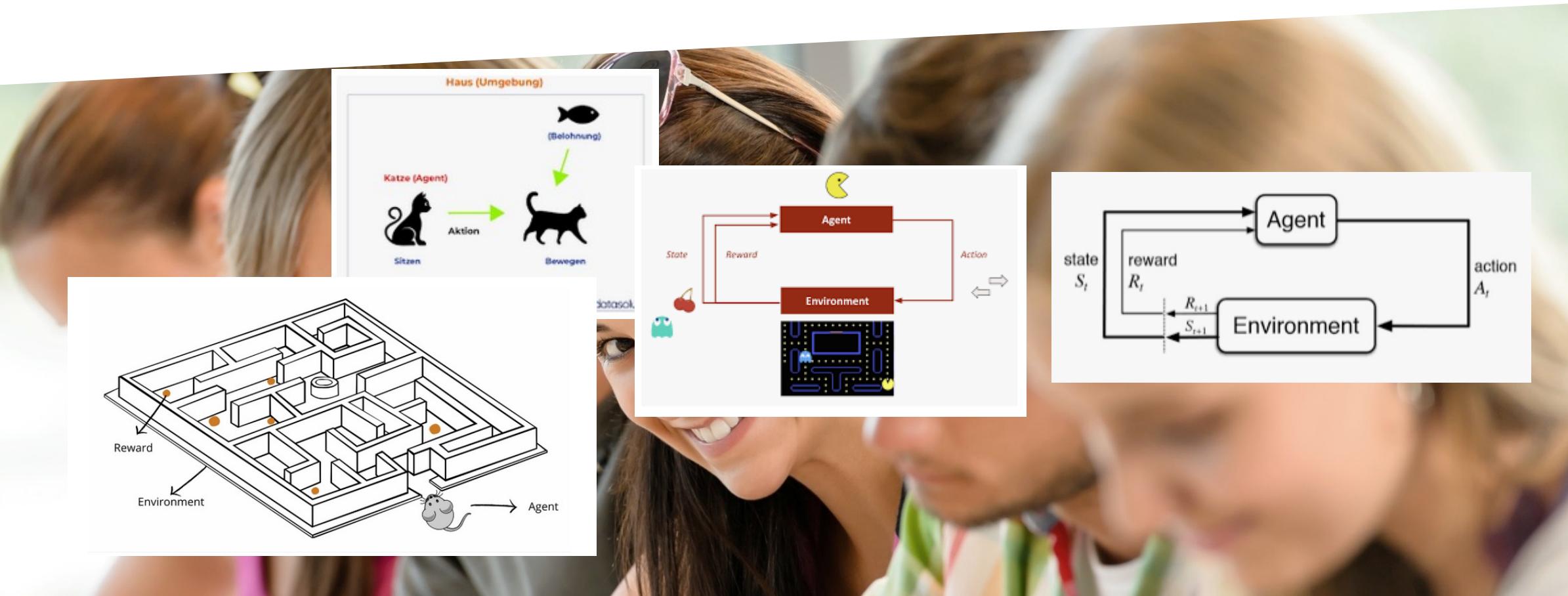
14.7 Deep-Learning



14.7 Deep-Learning



15. Reinforcement Learning – Bestärkendes Lernern



15.1 Einordnung des Reinforcement-Learning

- Wir kennen ja bereits:
 - **Supervised Learning**: Wir haben ganz viele Daten, die eventuell über einen langen Zeitraum gesammelt wurden. Wir haben zusätzlich Information zur Klassenzugehörigkeit oder Werte, für die wir ein Modell suchen.
 - **Unsupervised Learning**: Wir haben ganz viele Daten, die eventuell über einen langen Zeitraum gesammelt wurden. Wir suchen nach Mustern wie Gruppierungen von Daten-Objekten, die einander ähnlich sind. (z.B. Cluster) oder nach verborgenen Abhängigkeiten/Regeln (z.B. „b wird kleiner, wenn a größer wird“)
- Und nun das **Das Reinforcement-Learning**: Wir haben keine Daten, sondern „Dinge in der realen Welt“ oder „Dinge einer simulierten Welt“ (Environment) werden „ausprobiert“. Bei Probieren hat man häufig Misserfolge oder seltener (relative) Erfolge. Dies ergibt einen „Belohnungswert“ (Reward). Das System lernt dann durch Herumprobieren, welche Aktionen (des Agenten) gut sind und belohnt diese. Der Agent kann dadurch mit vielen Versuchen immer besser und besser werden. Möglich ist damit ***Steuern und Regeln, Robotik usw.***

15.2 Veranschaulichung des Reinforcement-Learning

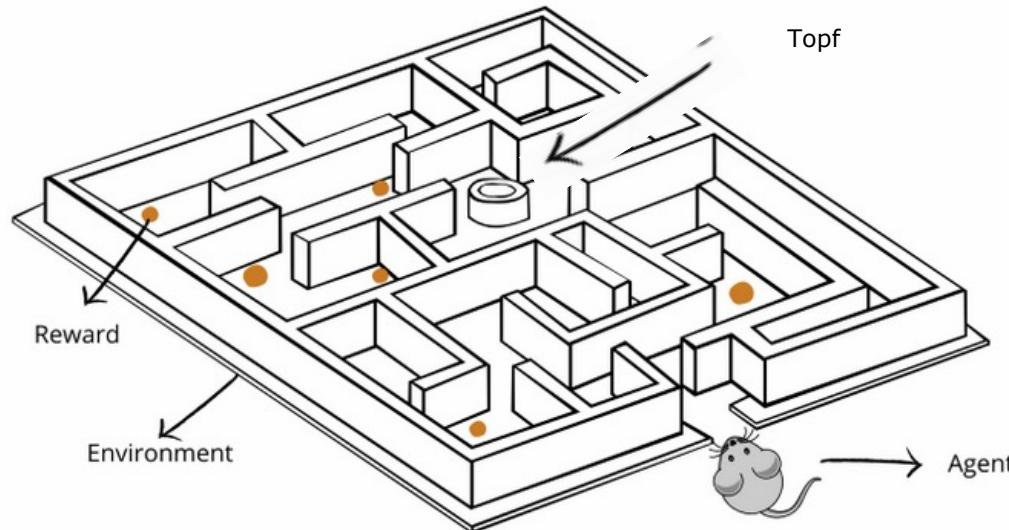
- Grundprinzip:



- Ziel ist also, dass die Katze „lernt“, wie und wohin sie sich bewegen muss, um eine Belohnung zu erhalten.

15.2 Veranschaulichung des Reinforcement-Learning

- **Grundprinzip:**

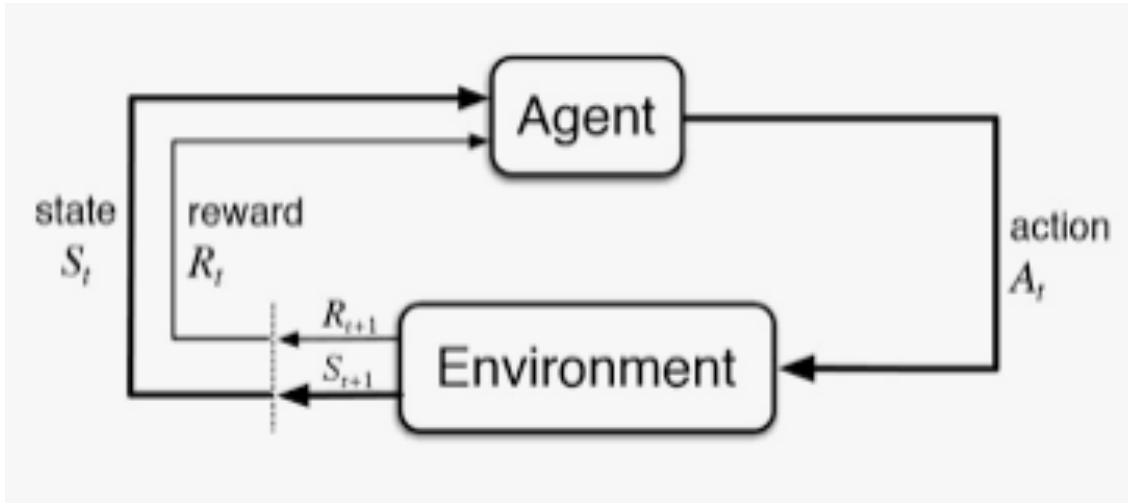


https://www.youtube.com/watch?v=UB_37encRCI

- Ziel ist also, dass die Maus „lernt“, wie und wohin sie sich bewegen muss, um eine Belohnung zu erhalten. Die Belohnung kann nun für einen pragmatischen Nutzen eingesetzt werden: **Idee:** Setze die Belohnungen so, damit die Maus irgendwann nach vielen Versuchen, auswendig den *Topf* findet.
 - **Environment:** Das Labyrinth
 - **Agent:** Die Maus, die trainiert werden soll und dabei lernt
 - **Reward:** Belohnung, damit die Maus lernt an bestimmten Stellen richtig abzubiegen und nach vielen Versuchen den Weg auswendig weiss.

15.2 Veranschaulichung des Reinforcement-Learning

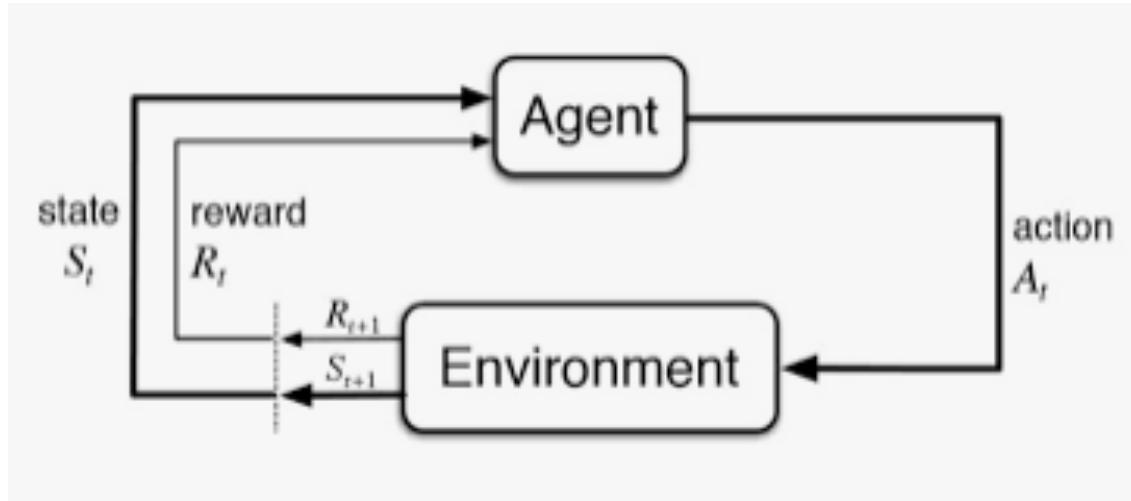
- Theoretische Sicht:



- **Lernprozess:** Dies ist eine Iteration (zyklische Wiederholung). Fortgrender Ablauf:
 - Ein **Status** S_t und eine **Belohnung** R_t wirken auf den **Agent** ein, der daraufhin eine **Aktion** A_t macht.
 - Die **Umgebung** meldet hier einen **neuen Status** S_{t+1} und eine **neue Belohnung** R_{t+1} zurück.
 - Der Agent „lernt“ mit sehr vielen Zyklen (z.B. 100000) allmählich, seine Aktionen so zu verbessern, dass die Belohnung langsam maximiert wird.

15.2 Veranschaulichung des Reinforcement-Learning

- **Theoretische Sicht:**



- **Agent:** Dies kann z.B. ein Computerprogramm sein, das mit einem neuronalem Netz gesteuert wird und das Gegenstand des Lernvorgangs ist. Hier gibt es sehr viele verschiedene Verfahren (A2C, ACER, ACKTR, DDPG, Q, DQN, GAIL, HER, PPO1, PPO2, SAC, TD3, TRPO,...) die meistens mit Neuronalen Netzen realisiert sind. Möglich sind auch Genetische Algorithmen.
- **Environment:** Das kann die reale Welt oder eine Computersimulation sein.

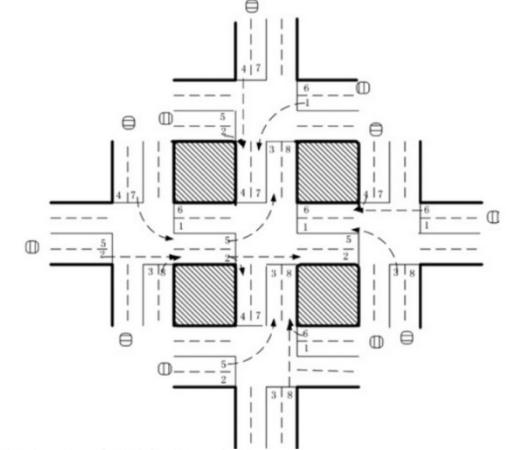
15.2 Veranschaulichung des Reinforcement-Learning

- **Beispiele für Environments:**
 - **Kraftstoffpreise:** Die Preise werden zu verschiedenen Zeitpunkten über oder unter den Durchschnittspreisen erhöht bzw. gesenkt. Dies macht ein „Agent“, der scheinbar zufällige Auf- und Abschläge realisiert. Der daraus resultierende Umsatz geht in den Reward ein: Ist der Umsatz zu einer bestimmten Uhrzeit an einem bestimmten Wochentag bei einem hohen Preis nicht viel geringer als bei sonst niedrigen Preisen wird der Reward hoch sein. So lernt der Agent an welchen Wochentagen, zu welchen Uhrzeiten Preisaufschläge eher toleriert werden.
<https://www.cirrussoft.com/de/blog/kuenstliche-intelligenz-macht-die-preise/>
 - **Angebote und Werbung verbessern:** Alibaba hat einen Agenten im entwickelt, der Angebote und Werbung aussucht und diese unterbreitet.
J. Jin, C. Song, H. Li, K. Gai, J. Wang, W. Zhang. Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising. arXiv preprint arXiv:1802.09756, 2018
 - **Robotik:** Für die Steuerung eines Roboters kann z.B. ein Video-Stream ausgewertet werden. Der Agent lernt auf dem Video zu erkennen, wie die Bewegungen des Roboters optimiert werden können.
S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end Training of Deep Visuomotor Policies. arXiv preprint arXiv:1504.00702, 2015

15.2 Veranschaulichung des Reinforcement-Learning

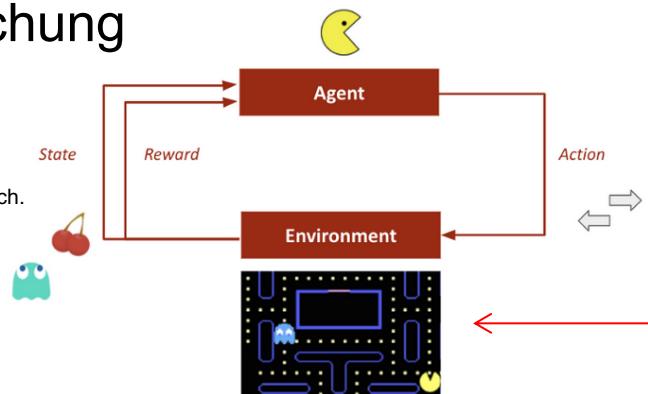
- **Optimierung einer Ampelsteuerung: Ziel ist die Maximierung des Verkehrsflusses bzw. die Stauvermeidung.**

I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," IET Intelligent Transport Systems, 2010

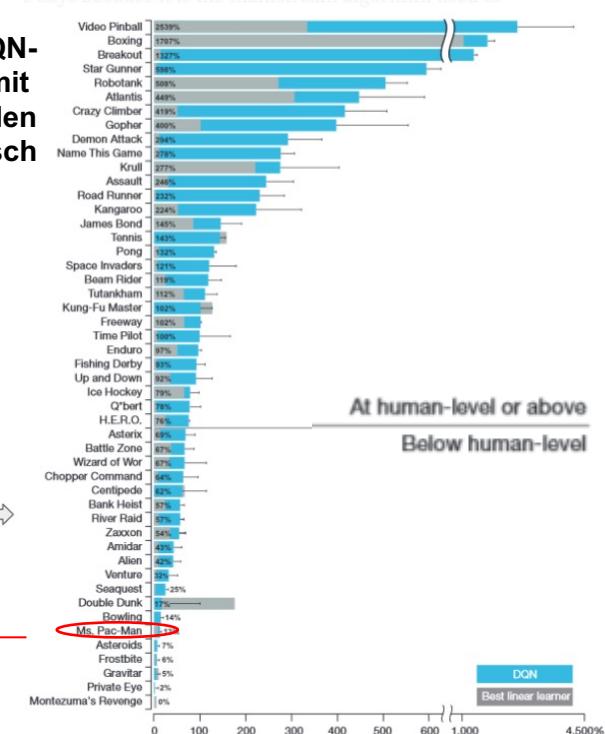


- **Computerspiele:** Das ist eher als Demonstrator oder zur Evaluation gedacht und nicht, um reale Probleme zu lösen. Als Environments werden alte C64-Nintendo-Gameboy oder Arcade-Spiele verwendet. In Rahmen der Forschung können die Eigenschaften von Algorithmen untersucht werden.

D. Silver et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587). 2016.



Vergleich des DQN-Agent mit menschlichen Spielen
100% = Mensch



15.2 Veranschaulichung des Reinforcement-Learning

– Beispiele für Environments:

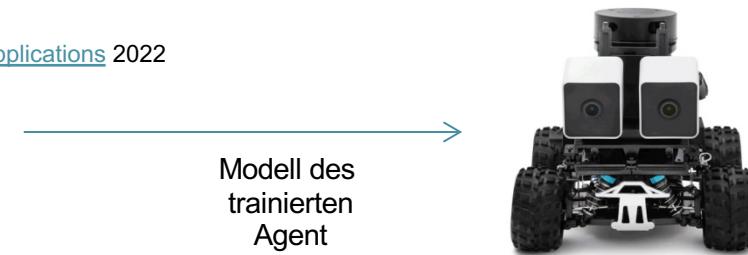
- **Robotik:** Für die Steuerung eines Roboters kann z.B. ein Video-Stream ausgewertet werden. Der Agent lernt auf dem Video zu erkennen, wie die Bewegungen des Roboters optimiert werden können.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end Training of Deep Visuomotor Policies.
arXiv preprint arXiv:1504.00702, 2015
<https://www.youtube.com/watch?v=Q4bMcUk6pcw&t=42s>



- **Autonomes Fahren:** Der Agent wird in einer Simulation trainiert. Das trainierte Modell kann dann in ein reales Fahrzeug (bzw. ein Fahrzeugmodell) geladen werden.

D. Wicki 10 Real Life Applications of Reinforcement Learning, <https://neptune.ai/blog/reinforcement-learning-applications> 2022



15.3 Beispiel: «Lunar Lander» lernt auf dem Mond zu landen

- **Lunar Lander -Beispiel:**
 - **Python: Reinforcement Learning - Beispiel Mondlandung.ipynb**
 - Trainiert das Landen auf dem Mond. 200.000 Versuche (Trial and Error) führen zum Erfolg
 - **Das Python-Package „gym“:** Stammt von OpenAI aus dem Packet Stable _baselines und dient zum Entwickeln und Vergleichen von Algorithmen für das Reinforcement-Learning.
 - **Simuliert den Lunar Lander:** Dieser hat Triebwerke...
 - **Die Klasse „Env“:** Das Environment „env“
 - simuliert die Schwerkraft auf dem Mond
 - variiert die Mondoberfläche
 - variiert die Anfangsbedingungen
 - **Ein Action-Object action verkörpert ein Beispiel eines Versuchs**
 - Mit env.render() wird die Situation grafisch dargestellt.
 - Mit env.step(action) wird eine Aktion ausgelöst, z.B. das Zünden eines der Triebwerke für eine bestimmte Zeit.

