Digital Applications & Data Management

WS25/26

Modul 7

Dr. Jens Kohl

Roadmap Vorlesung

1. Grundlagen Künstlicher Intelligenz, Machine Learning und Daten
2. Grundlagen Data Science
3. Angewandte Data Science (Teil 1)
4. Angewandte Data Science (Teil 2)
5. Data Science Use Case
6. Grundlagen unüberwachtes Lernen
7. Grundlagen überwachtes Lernen (tabellarische Daten)
8. Machine Learning Use Case
9. Grundlagen überwachtes Lernen (Bilddaten)
10. Transfer Learning Bilddaten und Fallbeispiel
11. Grundlagen Generative AI
12. Prompt Engineering, Agenten
13. Ausblick, Wiederholung, Fragestunde
14. Fragestunde

7. KÜNSTLICHE INTELLIGENZ:

ÜBERWACHTES (SUPERVISED) LERNEN

(TEIL 1: TABELLARISCHE DATEN)

Was machen wir heute?

Motivation

Trainieren Machine Learning Model
(vereinfacht)

```
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

random_forest = RandomForestClassifier(n_estimators=200)
random_forest.fit(X_train, y_train)

Y_prediction = random_forest.predict(X_test)
```

Messen Modellgüte per Metriken

Einsatz Confusion Matrix als Metrik für Klassifikation:

		Predicted	
		Ja	**Nein**
Tatsächlich	**Ja**	True Positive	False Negative
	Nein	False Positiv	True Negative

Accuracy = $\left(\frac{\text{korrekt vorhergesagt}}{\text{Gesamtzahl}}\right) = 97.49\%$

Anwenden Modell

```
[ ]   # notwendigen Features sind: PassengerClass, Sex (0 Frau, 1 Mann), Age,
      # SiblingSpousesPresent, ParentsChildrenPresent, fare,  AgeGroup
      Kate = [[1, 0, 17., 0, 1, 150., 2]]
      Leo = [[3, 1, 20., 0, 0, 15., 3]]
      Billy = [[1, 1, 30., 0, 0, 150., 3]]

      # make a prediction
      print("Prädiktion Überlebenschance Kate:", logmodel.predict(Kate))
      print("Prädiktion Überlebenschance Leo:", logmodel.predict(Leo))
      print("Prädiktion Überlebenschance Billy:", logmodel.predict(Billy))

    Prädiktion Überlebenschance Kate: [1]
    Prädiktion Überlebenschance Leo: [0]
    Prädiktion Überlebenschance Billy: [1]
```

Supervised Learning ermöglicht, automatisiert endliche/ abzählbare Werte vorherzusagen, bspw. Aktienkurse.

Kategorien des Machine Learning

Unsupervised Learning	Supervised Learning	Reinforcement Learning
Algorithmus lernt ohne vorher definierte Zielwerte oder Belohnung	Algorithmus lernt eine Funktion, die Eingabe-größen auf vorher definierte Outputs mappt („wahr“, „falsch“, „Hund“, Katze“).	Algorithmus lernt selbständig mit dem Ziel, eine „Belohnung“ zu maximieren.

Überwachtes vs. Unüberwachtes Lernen

Vergleich Algorithmen

	Überwachtes (supervised) Lernen	Unüberwachtes (unsupervised) Lernen
Eingabe	Gekennzeichnete Daten	Ungekennzeichnete Daten
	Von Experten vorgegebener Input/ korrekte Antworten (Zahlen, Eigenschaften)	Input wird nicht von Experten vorgegeben, sondern wird (meist aus lokalen Informationen) eigenständig erarbeitet
Eignung	Vorhersagen/ Einteilung zukünftiger Beobachtungen	Analysen/ Verstehen von Datenmengen
Vorgehen	Regressionen, Klassifikationen	Clustering, Dimensionsreduktion
Ausgabe	Reale Werte oder Kategorien	Homogene Klassen und Regeln zur Beschreibung von versteckten Mustern und Strukturen in den Inputdaten
Datenmenge	Auch mit relativ geringer Datenmenge möglich	Meist nur mit sehr großer Datenmenge möglich
Trainingsumfeld	Meist offline	Meist real time
Haupt-Anwendungen	Vorhersagen von Kundenverhalten, Gruppenzugehörigkeiten	Identifizierung von Ähnlichkeiten (von Gruppen) oder Anomalien (in Daten), Produktempfehlungen
Vorteile	- Kontrolle über die Modellerstellung - Einfachere Fehlerentdeckung und -behebung - Ergebnisse sind nachvollziehbar - Wenig(er) Daten nötig	- Kein aufwändiger Experteninput (Kosten, Zeit) nötig - Arbeit mit unstrukturierten Inputdaten möglich - Minimierung des menschlichen Fehlers - Erlangung von neuem Wissen über Daten (für Menschen nicht ersichtliche Zusammenhänge)
Nachteile	- Zeit- und kostenintensive Trainingsdaten-Generierung - Keine Möglichkeit der Erkennung und Darstellung von neuen Kategorien, wenn diese in den Trainingsdaten nicht vorhanden sind	- Neu entstandenen Klassen sind oft nicht relevant - Wenig Kontrolle über die Modellerstellung - Geringe Nachvollziehbarkeit der Ergebnisse - Meist viele Inputdaten und hohe Rechnerleistung nötig

Quelle: Gröner & Heinecke, „Kollege KI“

Supervised Learning

Übersicht

- **Ziel**: Modell um für beliebige **Eingaben $X_{i...n}$** eine **Zielgröße Y** möglichst genau zu bestimmen oder anzunähern.

- **Ermöglicht**:
  - **Regression**: Vorhersage eines kontinuierlichen Wertes Y (bspw. natürliche oder reelle Zahl wie Aktienkurs).
  - **Klassifikation**: Vorhersage eines diskreten Wertes Y (bspw. wahr/ falsch, Tierart, Personenentdeckung, ….).

- **Was schauen wir uns?**
  - Gesamter Workflow: von Datenaufbereitung, Trainieren, Testen bis hin zur Anwendung.
  - Ausgewählte „simple“ Algorithmen für Regression und Klassifikation (weitere Algorithmen im Backup).
  - Kombinierte Algorithmen (Ensemble Learning).
  - Metriken: wie gut ist das Modell?.

Supervised Learning

Regression

- **Ziel**: Vorhersage eines Wertes aus einer großen Wertemenge (kontinuierliche Variable).

- **Ermöglicht**: Prädiktion eines Wertes

- **Aber**:
  - Hoher Aufwand für Bestimmen der Zielwerte (= Label). Dies ist beim unüberwachtem Lernen nicht vorhanden.
  - Ungenauer als Klassifikation (wir versuchen einen genauen Wert aus einer großen Wertemenge vorherzusagen).
  - Diese zusätzliche "Genauigkeit" wird oft nicht benötigt; z.B. reicht im Aktienhandel oft Prädiktion, ob Kursziel überschritten.

- **Anwendungsbeispiele**:
  - Aktienkursvorhersage: Prädiktion eines genauen Kurswertes.
  - Biomonitoring: Prädiktion Cholesterin-/ Insulinspiegel.
  - ...

Supervised Learning

Klassifikation

- **Ziel**: Vorhersage eines Wertes aus einer kleineren, abzählbaren Menge (diskrete Variable).

- **Ermöglicht**: Klassifizieren, d.h. Einordnen eines Wertes in eine endliche Gruppe bzw. Prädiktion endlichen Wertes.

- **Aber**:
  - Hoher Aufwand für Bestimmen der Zielwerte (Labeln). Dies ist beim unüberwachtem Lernen nicht vorhanden.
  - Genauigkeit hängt von der Größe der Gruppe ab → für sehr große Gruppen haben wir ja quasi eine Regression.

- **Anwendungsbeispiele**:
  - Online-Werbung: klickt der User die Werbung an oder nicht an?
  - Bilderkennung: Erkennen eines bestimmten Tieres, Schriftzeichen, Bildinhalte bewerten, …
  - Übersetzung: Deutsch zu Spanisch, …
  - …

Supervised Learning

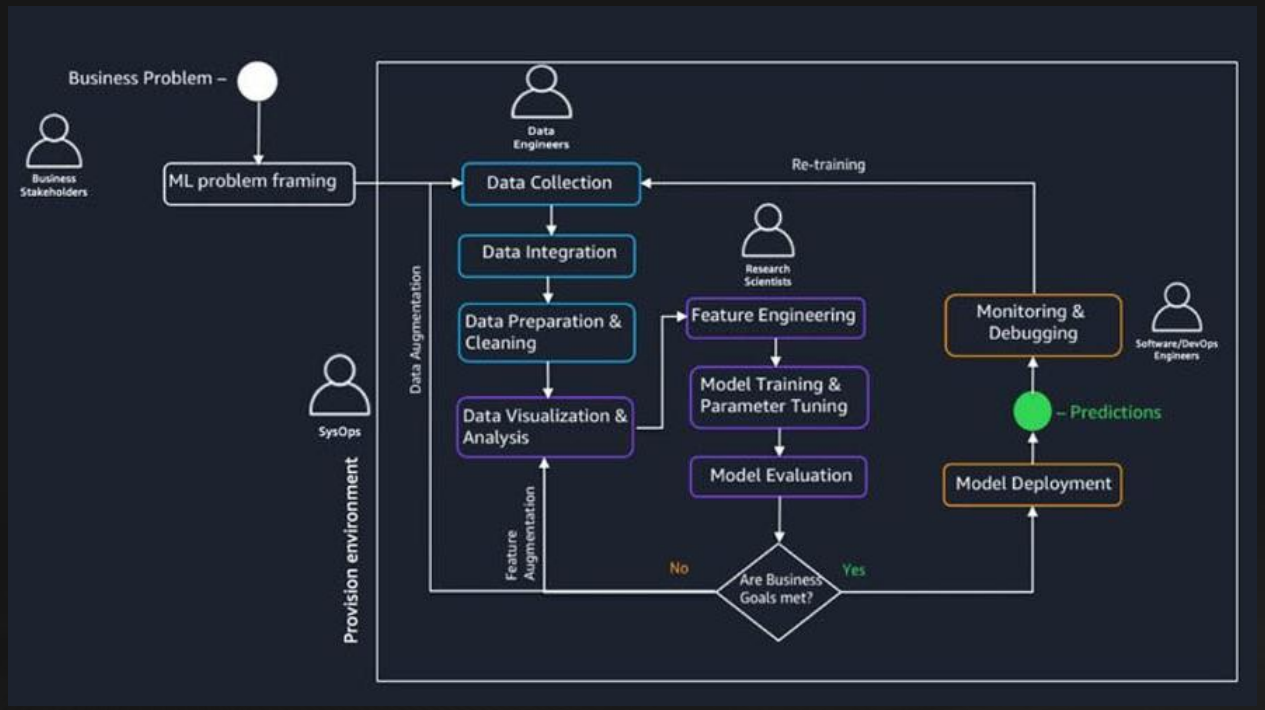
Übersicht

- Training mit beschrifteten Daten: Algorithmus wird anhand eines Trainingsdatensatz (ca. 70% Daten) trainiert, bei dem die Ergebnisvariable (Label) und somit Eingabe-Ausgabe-Paare bekannt sind.
  ➔ Algorithmus lernt selbstständig, einer Eingabe die gewünschten Ausgaben zuzuordnen.
- Auswahl Algorithmus: Auswahl des richtigen Algorithmus ist entscheidend für die Leistung.
  Die für uns relevanten Algorithmen des überwachten Lernens sind
  - Lineare Regression (für Regressionsaufgaben),
  - Logistische RegRession (für Klassifizierungsaufgaben),
  - Support Vector Machines (sowohl Klassifizierung als auch Regression),
  - Entscheidungsbäume/ RANDOM FOREST
  - Neuronale Netze für sehr große Datensätze (nächste woche).
- Parameter-Tuning und Optimierung: Hyperparameter, die zur Leistungsoptimierung angepasst werden können (nicht Inhalt dieses Kurses)
- Modellvorhersage: Sobald Modell trainiert ist, wird es verwendet, um Ergebnisvariable vorherzusagen. Anschließend wird Vorhersage mit realem Wert verglichen um zu sehen, wie gut die Genauigkeit des Modells ist (Testen Modell auf Trainingsdaten).
- TESTING: Nach dem Training wird die Leistung des Modells anhand eines separaten Testdatensatzes (ca. 30 % Daten) ausgewertet und validiert.

Generischer Workflow

Generischer Workflow Supervised Learning

Übersicht

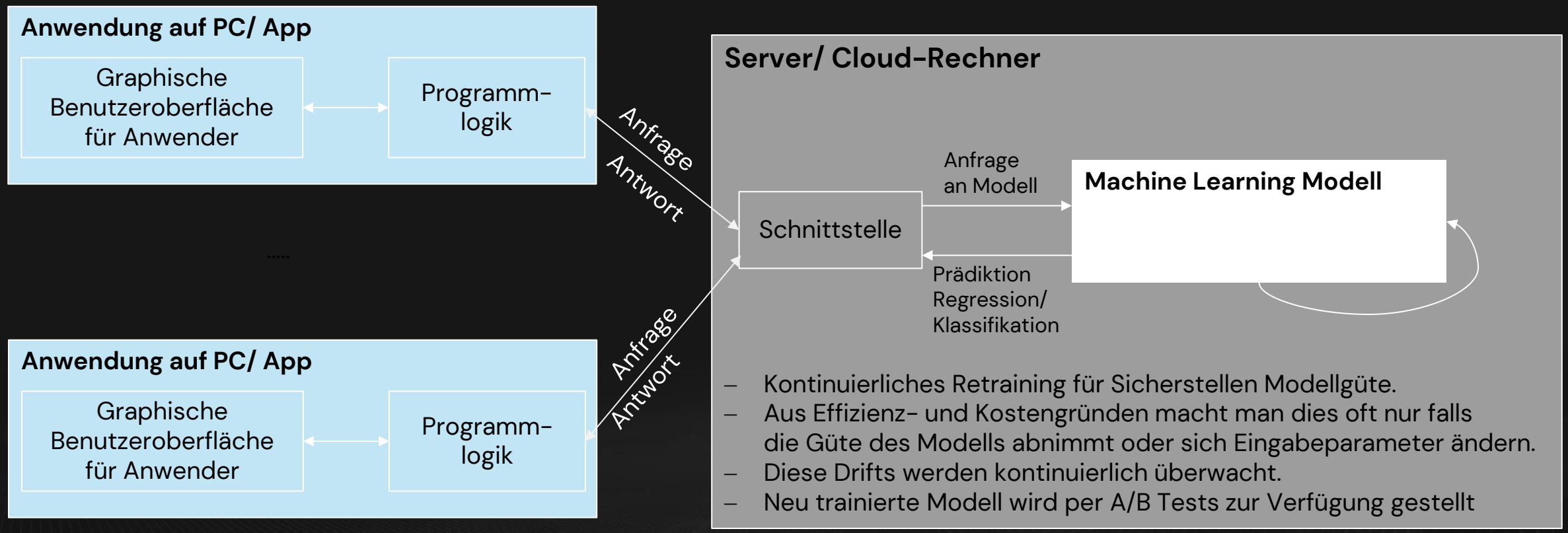


1. Customer/ Business Problem und Metriken definieren
2. Daten organisieren, aufbereiten, säubern, ggf. neue Features bauen
3. Machine Learning Vorbereiten:
   - Aufteilen Daten aufteilen in Trainings- und Testmenge (und ggf. Validierungsmenge).
   - Algorithmus und Kosten-/Optimierungsfunktion wählen und Modellparameter initialisieren.
4. Training: schrittweise Optimierung Modellparameter bis Modell möglichst gute Performance für Trainingsmenge hat.
5. Modell(-güte) validieren mit Testmenge, dessen Elemente Modell nicht kannte. Falls Modellgüte i.O., weiter zu Schritt 6. Sonst zu Schritt 5.
6. Deployment: Modell „live“ vor Kunde schalten.
7. Monitoring: kontinuierliches Überprüfen Modellgüte [„Modelldrift“] und Modellinput [„Parameterdrift“]

Ziel: Lernen eines möglichst genauen Modells H(Input) = Zielgröße

Generischer Workflow Supervised Learning

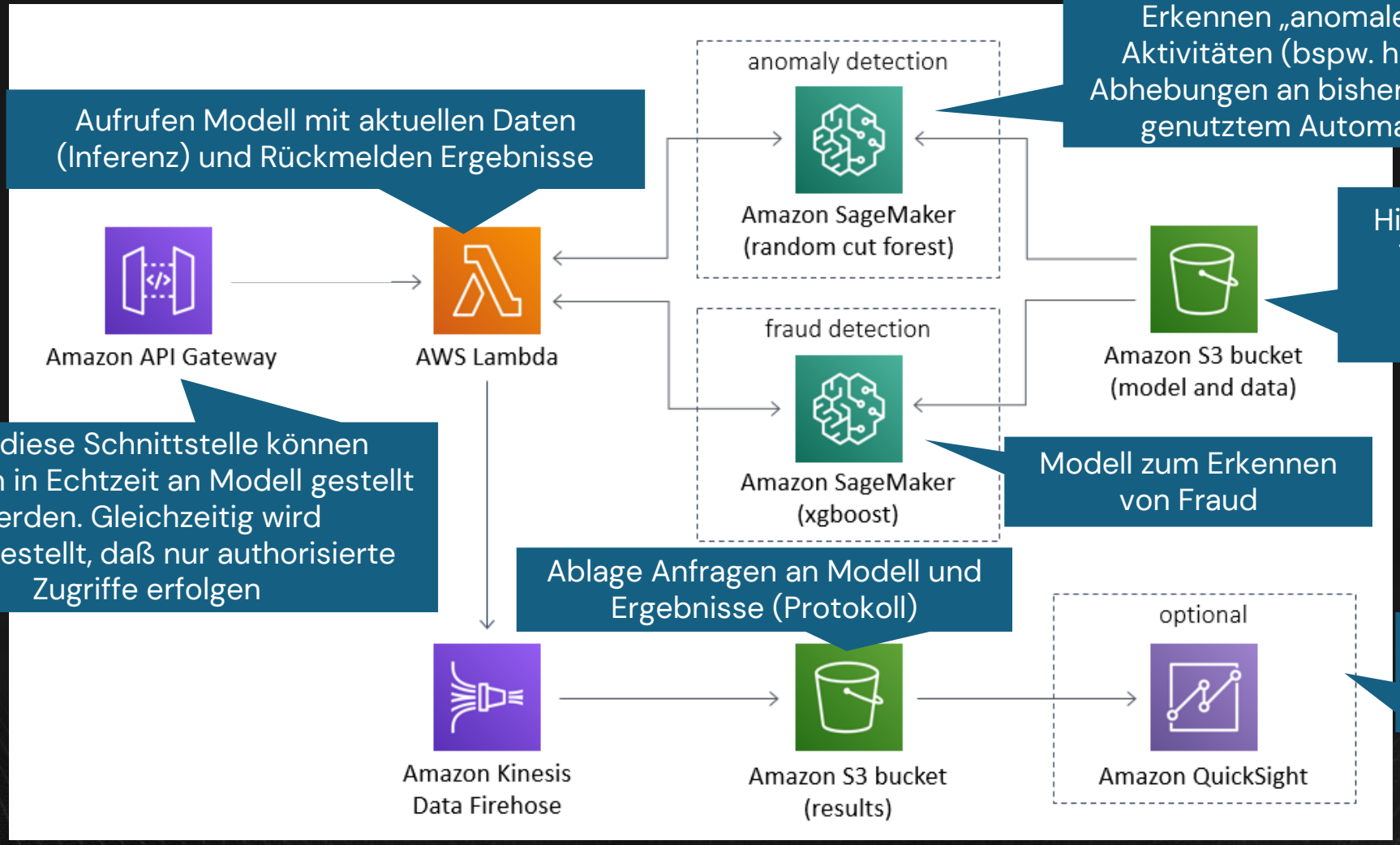
Wie wird so ein Modell im realen Leben eingesetzt (vereinfacht)?



„Optimales“ Model: Modell und Hyperparameter verbessern, bis bessere Performance als Mensch und kein Overfitting!

Generischer Workflow Supervised Learning

Modell at scale



Quelle: Amazon Web Services Referenzarchitekturen: Link

Veranschaulichung Machine Learning am Fallbeispiel „Einweisung Krankenhaus“

Case Study Supervised Learning: Einweisung Krankenhaus

Vorgehen Case Study

Ask an interesting question

„Kann anhand von ausgewählten Patientendaten bestimmt werden, ob ein Patient ins Krankenhaus aufgenommen werden muß oder es reicht, ihn ambulant zu behandeln?“

Get the Data

Explore the Data

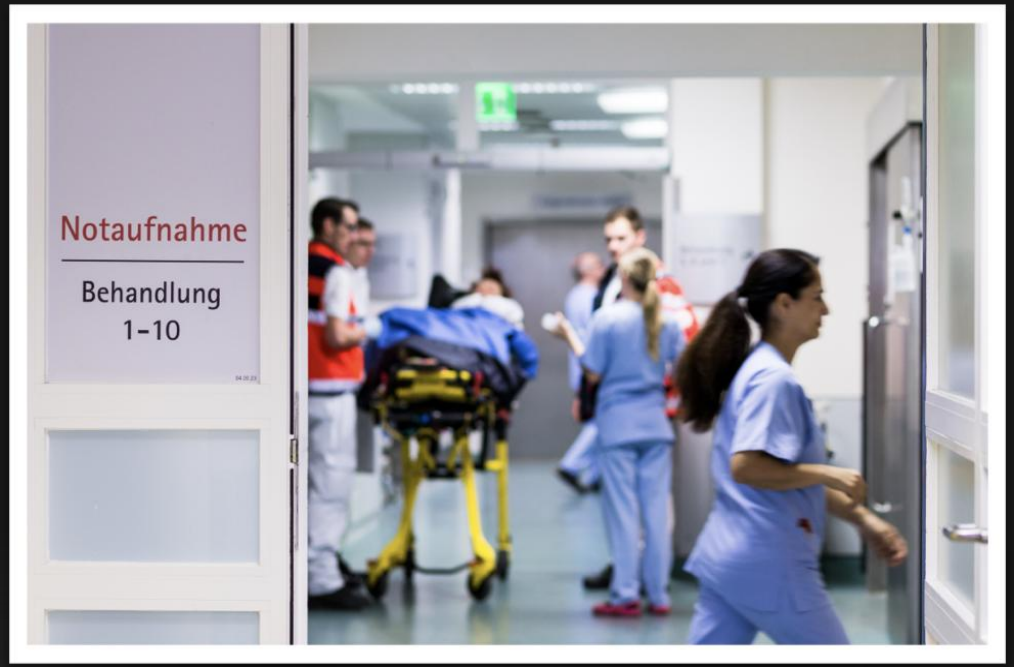
Model the Data

Communicate/Visualize the Results

Case Study Supervised Learning: Einweisung Krankenhaus

Beschreibung

- Patientenliste „KRANKENHAUS.CSV“:
- Kleiner Datensatz (4400 Zeilen à 11 Spalten)
- Deckt ganzen Workflow inkl. üblicher Probleme ab
- Fragestellung einfach verständlich und interessant…
- Vorschlag für mögliche Lösung ist auf ILIAS.



Einfaches Einführungsbeispiel, aber deckt gesamten Workflow ab

Case Study Supervised Learning: Einweisung Krankenhaus

Vorgehen Case Study

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

Auf den nächsten Folien detailliert

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 2: Get the Data

- Nachdem wir die zu untersuchende Fragestellung definiert haben, beschaffen wir uns die Daten.
- Für Laden und Auswerten Daten gibt es in Python viele Programmbibliotheken (Libraries), die uns die Detailarbeit der Programmierung abnehmen.
- Auch können wir im Notebook per Freitext eingeben, was wir als Code benötigen.
- Am Start jedes Cases laden wir diese Libraries. In der Python-Programmierumgebung gilt:
  - Lila markierter Text sind Standard-Befehle der Programmiersprache Python
  - Weiss markierter Text sind Variablen oder Libraries
  - Grün markierter Text sind Kommentare und werden mit einer Raute eingeleitet.

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 2: Get the Data

- Öffnen https://colab.research.google.com/
- File → New Notebook
- Benennen der Datei, z.B. Krankenhaus
- + Code → Eingabe Code, + Text → Eingabe erklärender Text (was macht der untenstehende Code) zum Nachvollziehen

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 2: Get the Data

Laden der wichtigsten Libraries für Data Engineering und Rechenoperationen:

```
import pandas as pd

import numpy as np

from google.colab import files
```

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 2: Get the Data

1) Öffnen https://colab.research.google.com/

2) File → New Notebook

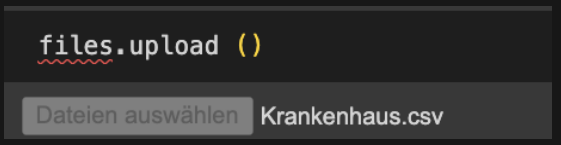
3) Benennen der Datei, z.B. Krankenhaus

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 2: Get the Data

- Datei „Krankenhaus.csv“ hochladen

```
files.upload ()
```



- Datei in Dataframe einlesen

```
Patientendaten_df = pd.read_csv('Krankenhaus.csv', sep=',')
```

Case Study Supervised Learning: Einweisung Krankenhaus

Vorgehen Case Study

Ask an interesting question

Get the Data

Explore the Data

- Daten ansehen
- Muster erkennbar?
- Fehler erkennbar?

Model the Data

Communicate/Visualize the Results

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

Patientendaten_df

	HAEMATOCRIT	HAEMOGLOBINS	ERYTHROCYTE	LEUCOCYTE	THROMBOCYTE	MCH	MCHC	MCV	AGE	SEX	SOURCE
0	35.1	11.8	4.65	6.3	310	25.4	33.6	75.5	1	F	out
1	43.5	14.8	5.39	12.7	334	27.5	34.0	80.7	1	F	out
2	33.5	11.3	4.74	13.2	305	23.8	33.7	70.7	1	F	out
3	39.1	13.7	4.98	10.5	366	27.5	35.0	78.5	1	F	out
4	30.9	9.9	4.23	22.1	333	23.4	32.0	73.0	1	M	out
...	...	...	...	...	...	...	...	...	...	...	...
4407	32.8	10.4	3.49	8.1	72	29.8	31.7	94.0	92	F	in
4408	33.7	10.8	3.67	6.7	70	29.4	32.0	91.8	92	F	in
4409	33.2	11.2	3.47	7.2	235	32.3	33.7	95.7	93	F	out
4410	31.5	10.4	3.15	9.1	187	33.0	33.0	100.0	98	F	in
4411	33.5	10.9	3.44	5.8	275	31.7	32.5	97.4	99	F	out

4412 rows × 11 columns

Case Study Supervised Learning: Einweisung Krankenhaus

Was für Daten haben wir?

- **ERYTHROZYTEN:** ZELLULÄRE ELEMENTE DES MENSCHLICHEN BLUTS, DIE BLUTFARBSTOFF HÄMOGLOBIN ENTHALTEN UND ALS HAUPTAUFGABE DEN TRANSPORT VON SAUERSTOFF IM BLUT HABEN.
- **LEUKOZYTEN:** KERNHALTIGE ZELLEN DES MENSCHLICHEN BLUTS, DIE KEINEN BLUTFARBSTOFF (HÄMOGLOBIN) TRAGEN. MAN NENNT SIE DESHALB AUCH *WEIßE* BLUTKÖRPERCHEN. LEUKOZYTEN SIND DIE ZELLULÄREN ELEMENTE DES MENSCHLICHEN IMMUNSYSTEMS. IHRE AUFGABE BESTEHT UNTER ANDEREM IN DER ERKENNUNG KÖRPER-EIGENER UND KÖRPERFREMDER STRUKTUREN, DER BILDUNG VON ANTIKÖRPERN UND DER PHAGOZYTOSE VON KRANKHEITSERREGERN UND KÖRPEREIGENEN ABBAUPRODUKTEN.
- **THROMBOZYTEN:** FLACHE, UNREGELMÄßIG RUNDLICHE BLUTBESTANDTEILE, DIE IM AKTIVIERTEN ZUSTAND STOFFE FREISETZEN, DIE FÜR DIE BLUTSTILLUNG NOTWENDIG SIND. BEI EINER VERLETZUNG, UND EINER DAMIT ERFOLGENDEN ERÖFFNUNG DER BLUTGEFÄßE, KOMMT ES INNERHALB KURZER ZEIT DURCH DIE ADHÄSION UND AGGREGATION VON THROMBOZYTEN UND DER ANSCHLIEßENDEN FIBRINBILDUNG ZUR BLUTSTILLUNG.
- **HÄMATOKRIT:** VOLUMENANTEIL DER ZELLULÄREN ELEMENTE IM BLUT. ERYTHROZYTEN MACHEN MIT 96 % GRÖßTEN ANTEIL DAVON AUS. EIN HOHER HÄMATOKRIT KANN BSPW. DURCH EINEN HOHEN ANTEIL AN ERYTHROZYTEN IM BLUT (DOPING, HÖHENANPASSUNG, LUNGENERKRANKUNGEN, STARKES RAUCHEN) ODER DEHYDRATATION DES PATIENTEN ZUSTANDE KOMMEN. EIN NIEDRIGER HÄMATOKRIT KANN DURCH STÖRUNGEN DER ERYTHROPOESE, CHRONISCHE NIERENINSUFFIZIENZ (MANGEL AN ERYTHROPOIETIN) ODER DURCH EINE VOLUMENTHERAPIE (Z.B. DURCH GABE VON KOCHSALZINFUSIONEN STATT BLUTKONSERVEN BEI BLUTVERLUST) BEDINGT SEIN.
- **HÄMOGLOBIN:** EISENHALTIGER ROTER BLUTFARBSTOFF IN DEN ERYTHROZYTEN (*ROTEN BLUTKÖRPERCHEN*) DER WIRBELTIERE UND SEINE VARIANTEN. ES ERMÖGLICHT DEN SAUERSTOFF-TRANSPORT IM KÖRPER ÜBER DEN BLUTKREISLAUF.
- DIE INDIZES **MCH, MCV UND MCHC** SIND WEITERE WICHTIGE KENNWERTE ZUR CHARAKTERISIERUNG VON ERYTHROZYTEN (HÄMOGLOBIN-GEHALT, ERYTHORZYTEN-VOLUMEN, HÄMOGLOBIN-KONZENTRATION).

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

Was fällt Ihnen auf?

```
Patientendaten_df.describe(include="all")
```

	HAEMATOCRIT	HAEMOGLOBINS	ERYTHROCYTE	LEUCOCYTE	THROMBOCYTE	MCH	MCHC	MCV	AGE	SEX	SOURCE
count	4412.000000	4412.000000	4412.000000	4412.000000	4412.000000	4412.000000	4412.000000	4412.000000	4412.000000	4412	4412
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	2
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	M	out
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2290	2628
mean	38.197688	12.741727	4.541260	8.718608	257.524479	28.234701	33.343042	84.612942	46.626473	NaN	NaN
std	5.974784	2.079903	0.784091	5.049041	113.972365	2.672639	1.228664	6.859101	21.731218	NaN	NaN
min	13.700000	3.800000	1.480000	1.100000	8.000000	14.900000	26.000000	54.000000	1.000000	NaN	NaN
25%	34.375000	11.400000	4.040000	5.675000	188.000000	27.200000	32.700000	81.500000	29.000000	NaN	NaN
50%	38.600000	12.900000	4.570000	7.600000	256.000000	28.700000	33.400000	85.400000	47.000000	NaN	NaN
75%	42.500000	14.200000	5.050000	10.300000	321.000000	29.800000	34.100000	88.700000	64.000000	NaN	NaN
max	69.000000	18.900000	7.860000	76.600000	1183.000000	40.800000	39.000000	115.600000	99.000000	NaN	NaN

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

```
Patientendaten_df.dtypes
```

```
HAEMATOCRIT     float64
HAEMOGLOBINS    float64
ERYTHROCYTE     float64
LEUCOCYTE       float64
THROMBOCYTE       int64
MCH             float64
MCHC            float64
MCV             float64
AGE               int64
SEX              object
SOURCE           object
dtype: object
```

Machine Learning kann nur mit Zahlen arbeiten.
Wir müssen also Spalte Sex und Source umwandeln

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

```
(Patientendaten_df['SOURCE'].unique())
```

```
array(['out', 'in'], dtype=object)
```

```
source_mapping = {"in": 1, "out": 0}
```

```
Patientendaten_df['TARGET']= Patientendaten_df['SOURCE'].map(source_mapping)
```

```
Patientendaten_df = Patientendaten_df.drop(['SOURCE'], axis=1)
```

VORGEHEN (SPALTE Source):

1. Anschauen aller eindeutigen, vorhandenen Werte
2. Definieren einer Abbildung der existierenden Werte auf Zahlenwerte
3. Anwenden Abbildung auf Dataframe (Spalte source) und Umbenennen Spalte Target
4. Löschen alte Spalte Source, weil sonst doppelt.

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

```
sex_mapping = {"F": 1, "M": 0}

Patientendaten_df['GESCHLECHT']= Patientendaten_df['SEX'].map(sex_mapping)

Patientendaten_df = Patientendaten_df.drop(['SEX'], axis=1)
```

Vorgehen (spalte sex):

1. Anschauen aller eindeutigen, vorhandenen Werte
2. Definieren einer Abbildung der existierenden Werte auf Zahlenwerte
3. Anwenden Abbildung auf Dataframe (spalte sex) und umbenennen Spalte (Geschlecht)
4. Löschen alte Spalte (sex)

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 3: Explore the data

```
Patientendaten_df.dtypes
```

```
HAEMATOCRIT      float64
HAEMOGLOBINS     float64
ERYTHROCYTE      float64
LEUCOCYTE        float64
THROMBOCYTE        int64
MCH              float64
MCHC             float64
MCV              float64
ALTER              int64
GESCHLECHT         int64
TARGET             int64
dtype: object
```

Nur noch Zahlentypen vorhanden, also fertig!

Case Study Supervised Learning: Einweisung Krankenhaus

Vorgehen Case Study

Ask an interesting question

Get the Data

Explore the Data

Model the Data

- Modell erstellen
- Modell trainieren auf Daten (“fitten”)
- Modell validieren

Communicate/Visualize the Results

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Übersicht

Trainieren Machine Learning Model (vereinfacht)

```
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

random_forest = RandomForestClassifier(n_estimators=200)
random_forest.fit(X_train, y_train)

Y_prediction = random_forest.predict(X_test)
```

Messen Modellgüte per Metriken

Einsatz Confusion Matrix als Metrik für Klassifikation:

		Predicted	
		Ja	**Nein**
Tatsächlich	**Ja**	True Positive	False Negative
	Nein	False Positiv	True Negative

Accuracy = $\left(\frac{\text{korrekt vorhergesagt}}{\text{Gesamtzahl}}\right) = 97.49\%$

Anwenden Modell

```
[ ]   # notwendigen Features sind: PassengerClass, Sex (0 Frau, 1 Mann), Age,
      # SiblingSpousesPresent, ParentsChildrenPresent, fare,  AgeGroup
      Kate = [[1, 0, 17., 0, 1, 150., 2]]
      Leo = [[3, 1, 20., 0, 0, 15., 3]]
      Billy = [[1, 1, 30., 0, 0, 150., 3]]

      # make a prediction
      print("Prädiktion Überlebenschance Kate:", logmodel.predict(Kate))
      print("Prädiktion Überlebenschance Leo:", logmodel.predict(Leo))
      print("Prädiktion Überlebenschance Billy:", logmodel.predict(Billy))

    Prädiktion Überlebenschance Kate: [1]
    Prädiktion Überlebenschance Leo: [0]
    Prädiktion Überlebenschance Billy: [1]
```

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Daten für ML vorbereiten

Bevor wir mit Machine Learning anfangen können, müssen wir die Daten noch in ein bestimmtes Format umwandeln:

1. Aufteilen der Daten in
   - eine Spalte (Zielvektor) y, mit den vorherzusagenden Werten (Label)
   - Eine Matrix X mit allen Spalten/ Features, die zur Vorhersage eingesetzt werden sollen

2. Aufteilen in Trainings- und Testmenge:
   - Trainingsmenge: Lernen des Modells
   - Testmenge: Testen, wie gut das Modell ist

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Daten für ML vorbereiten

1. Aufteilen der Daten in
   - eine Spalte (Zielvektor) y, mit den vorherzusagenden Werten.
   - Eine Matrix X mit allen Spalten/ Features, die zur Vorhersage eingesetzt werden sollen

```
[18]  # speichere die Spalte Target in einen Vektor y
      y = Patientendaten_df['TARGET']

[19]  # wir löschen die Spalte Target aus den Daten;
      Patientendaten_df = Patientendaten_df.drop(['TARGET'], axis=1)

      # und weisen die Daten der Matrix X zu.
      # ACHTUNG: die Zielgröße darf nie in X sein!!!!! Sonst kennen wir ja immer die Lösung....
      X = Patientendaten_df

[20]  print (y.shape)
      print(X.shape)

      (4412,)
      (4412, 10)
```

Hier können wir sehen, ob die Aufteilung funktioniert hat (1 Spalte, dann die restlichen Features)

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Daten für ML vorbereiten

2. Aufteilen in Trainings- und Testmenge:
   - Trainingsmenge: Lernen des Modells
   - Testmenge: Testen, wie gut das Modell ist

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

print(x_train.shape, x_test.shape)

(3088, 10) (1324, 10)
```

test_size=0.3 legt fest, dass 30 % der Daten zum Testen verwendet werden, und random_state=0 gewährleistet die Reproduzierbarkeit der Aufteilung der Daten (die Aufteilung ist bei jedem Durchlauf des Codes gleich).

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Daten für ML vorbereiten

2. Aufteilen in Trainings- und Testmenge:
   - Trainingsmenge: Lernen des Modells
   - Testmenge: Testen, wie gut das Modell ist

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

print(x_train.shape, x_test.shape)
```

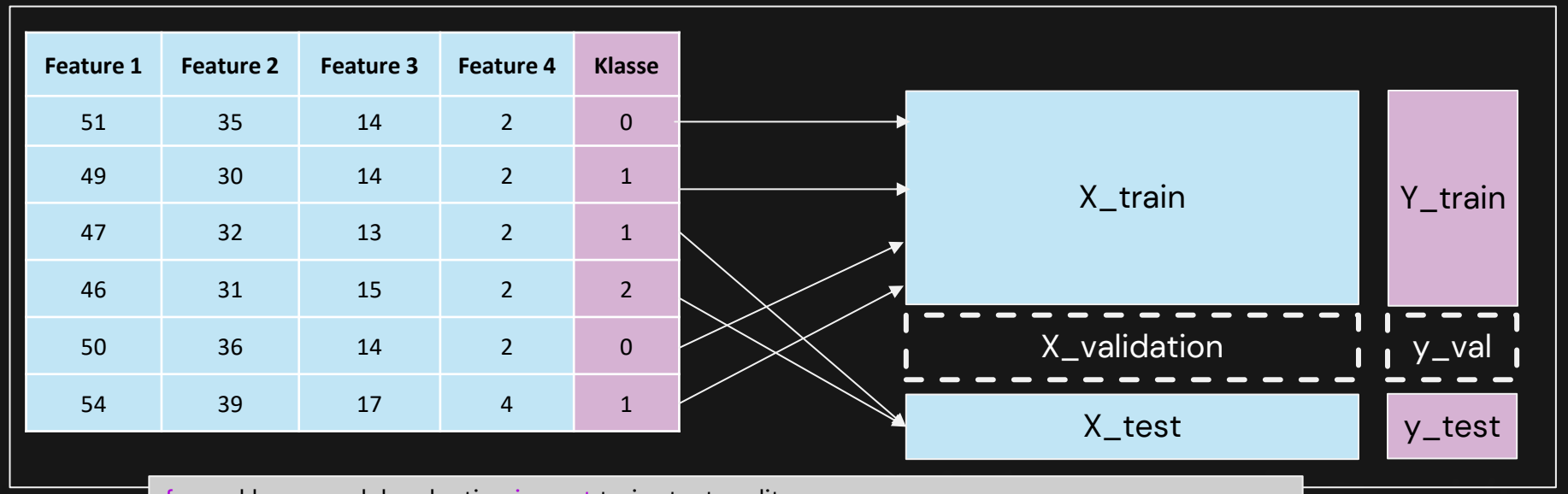
```
(3088, 10) (1324, 10)
```

test_size=0.3 legt fest, dass 30 % der Daten zum Testen verwendet werden, und random_state=0 gewährleistet die Reproduzierbarkeit der Aufteilung der Daten (die Aufteilung ist bei jedem Durchlauf des Codes gleich).

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Detaillierung Datenvorbereitung

Feature 1	Feature 2	Feature 3	Feature 4	Klasse
51	35	14	2	0
49	30	14	2	1
47	32	13	2	1
46	31	15	2	2
50	36	14	2	0
54	39	17	4	1




```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Test_size: Verhältnis Trainings-/u Testmenge
Random_state = ermöglicht reproduzierbare zufällige Verteilung.

- Aufgabe **Trainingsset**: Lernen/ Optimieren der Parameter des Machine Learning Modells.
- Aufgabe **Testset**: Evaluation, wie gut die Parameter und somit das Modell sind.
- **Zufällige Aufteilung** in Trainings- & Testmenge stellt ähnliche statistische Eigenschaften sicher (sortierte Ausgangsdaten!)
- Oft wird **Validierungsset** zur Optimierung der Parameter des Trainingsprozesses (**Hyperparameter**) gebildet.
- Verhältnis Trainings- zu Testmenge: oft 70% zu 30%. Bei Einsatz Validierungsmenge: oft 70%-20%-10%.

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Modell wählen und initialisieren.

1. **Wahl Machine Learning-Modell** abhängig von:
   - Datentyp: Zahlen, Bilder, Sprache, ….
   - Verfügbaren Ressourcen: Rechen-Power, Zeit, Kosten, …
   - Datenmenge: viele oder wenig Daten vorhanden?
   - Qualität Daten: fehlende Werte/ Ausreißer stören manche Lernverfahren.
   - Anforderungen an Modellgüte.
   - Komplexität Algorithmus: komplexe Algorithmen neigen zu hoher Streuung, simple zu hoher Verzerrung.

Bias-/ Variance Tradeoff

2. **Initialisierung Parameter**
   - Hyperparameter, d.h. Parameter für Trainingsprozess (bspw. Lernrate): erst Einsatz Standardwerte, dann Tuning (oft erfahrungsbasiert oder ausprobieren)
   - Modellparameter (Weights w_i und Bias b): Initialisierung mit zufälligen Werten (für schnelleres Lernen).

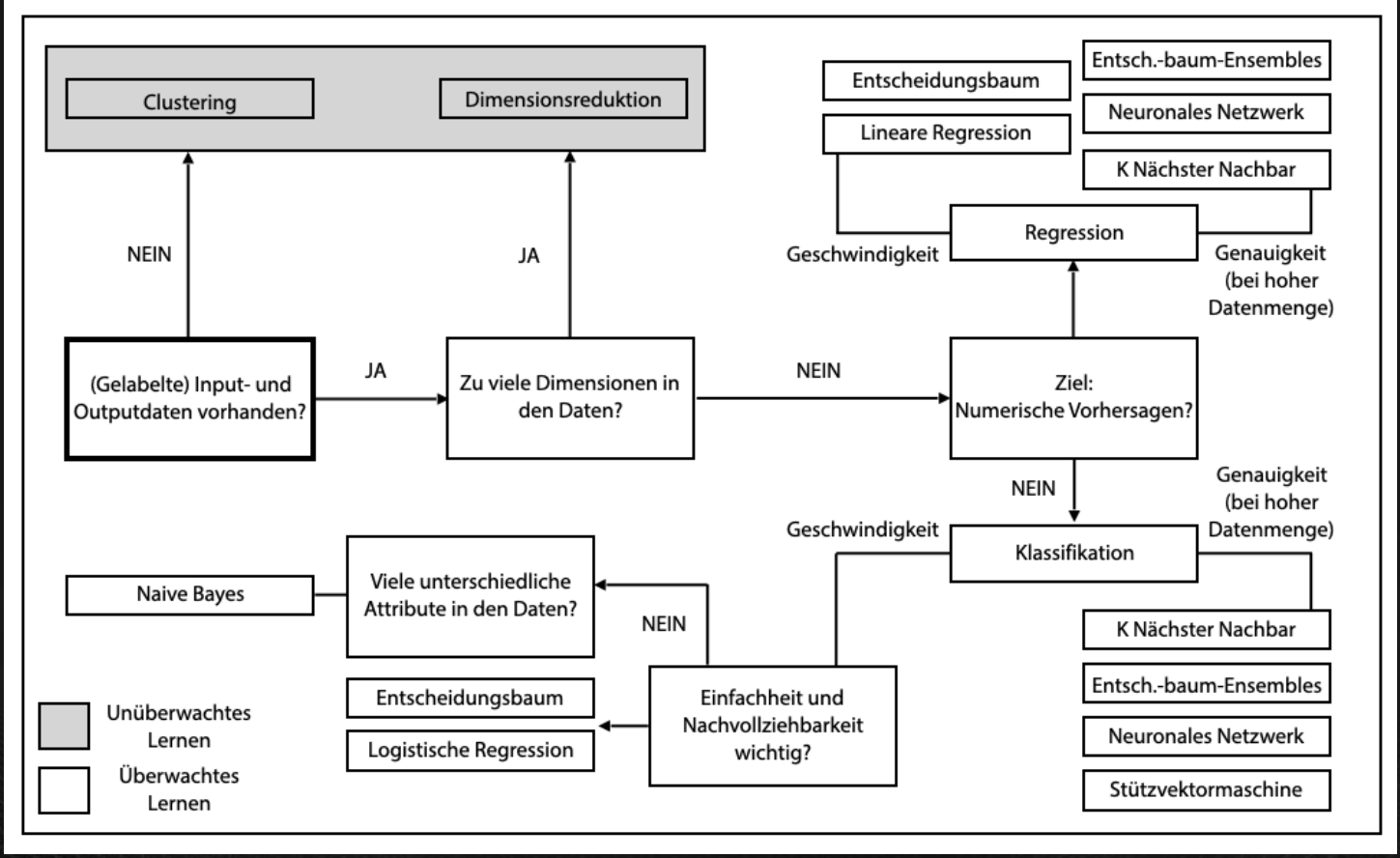
3. **Wahl Loss-Funktion:** Bewertungsfunktion für Genauigkeit des Machine Learning Modells (bspw. MSE[1], Cross-Entropy[2], …)

Auswahl ist erfahrungsbasierend. Die meisten Machine Learning Bibliotheken treffen aber standardmäßig gute Vorauswahlen.

Quelle: 1 Mean Squared Error
2 Rubinstein, R.Y. and Kroese, D.P., The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning, 2004.

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Modell wählen und initialisieren.

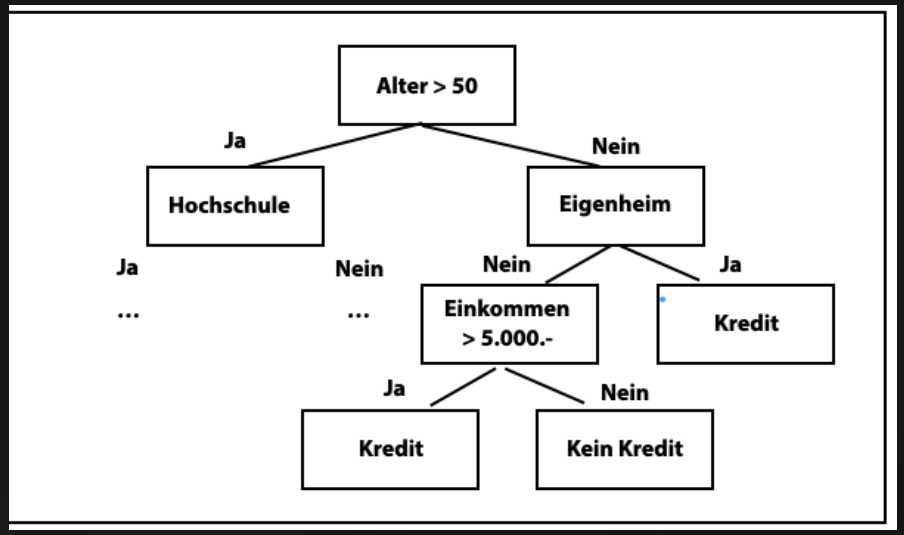


Quelle: Gröner & Heinecke, „Kollege KI“

Case Study Supervised Learning: Einweisung Krankenhaus

Entscheidungsbaum

- Konstruiert Baum, bei dem die Knoten die Merkmalstests und die Blätter die Klassenbezeichnungen oder kontinuierlichen Werte (für Regression) darstellen.
- Nicht-linear: Kann nicht-lineare Beziehungen in den Daten erfassen.
- Interpretierbar: Der Baum kann visualisiert werden und ist im Vergleich zu einigen anderen Modellen relativ leicht zu verstehen.
- ABER: Anfällig für Überanpassung (OVERFITTING), d.h. ohne Einschränkungen kann er sich dem Rauschen in den Daten anpassen und sehr komplex werden.
- FEATURE IMPORTANCE: Kann verwendet werden, um die Bedeutung verschiedener Merkmale bei der Erstellung von Vorhersagen zu verstehen.



Quelle: Gröner & Heinecke, „Kollege KI“

Case Study Supervised Learning: Einweisung Krankenhaus

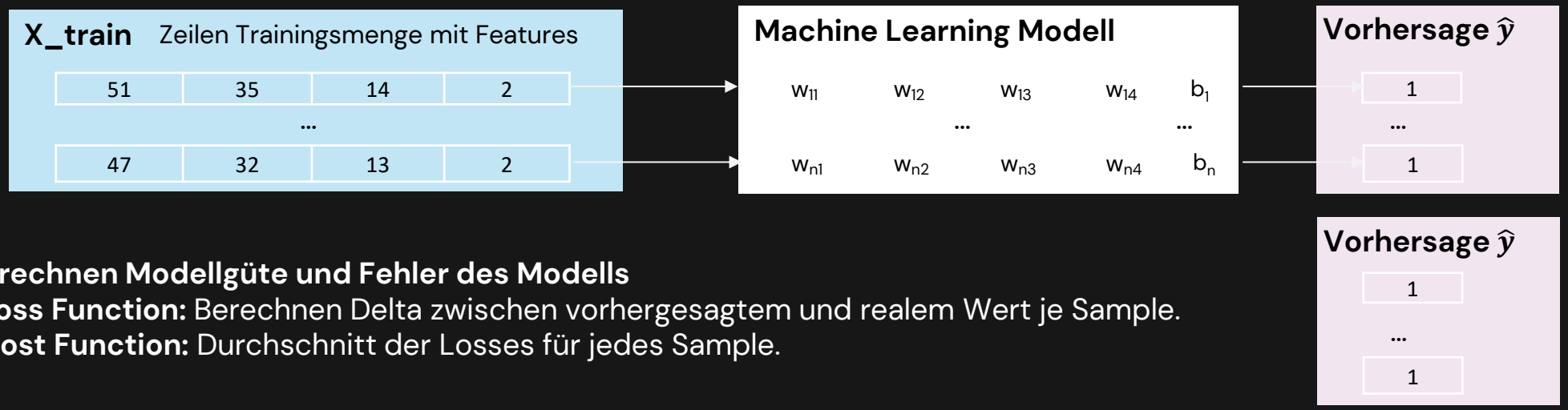
Demo Decision Tree

Link: http://www.r2d3.us/visuelle-einfuehrung-ins-maschinelle-lernen-teil-1/

Case Study Supervised Learning: Einweisung Krankenhaus

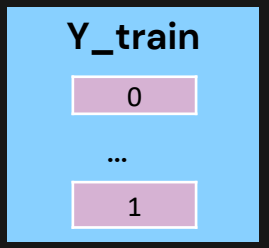
Schritt 4: Training

1. Feed Forward: Vorhersage Wert je **Sample** (individuelle Zeile Datensatz)



2. Berechnen Modellgüte und Fehler des Modells

- **Loss Function:** Berechnen Delta zwischen vorhergesagtem und realem Wert je Sample.
- **Cost Function:** Durchschnitt der Losses für jedes Sample.



3. Minimierung Fehler und Anpassen Modellparameter:

- Aufstellen Gleichung für Minimierung Cost Function (erste Ableitung gleich 0 setzen).
- Lösen Gleichung für die Modellparameter.
- Aktualisieren Modellparameter (Δw_{ij} und Δb_i), um so im nächsten Trainingslauf näher am realen Y-Wert zu landen.

→ Einsatz iterativer Algorithmus bei hochdimensionalen Daten (bspw. Gradient Descent).

Aktualisiertes ML-Modell mit geänderten Gewichten/ Bias

Δw_{11} Δw_{12} Δw_{13} Δw_{14} Δb_1
...
Δw_{n1} Δw_{n2} Δw_{n3} Δw_{n4} Δb_n

Durchführen Schritte 1-3 inkl. Update Modellparameter für gesamtes Trainingsset heißt **Epoch**, für Teile Trainingsset **Batch.**
Anzahl Durchläufe je Batch/ Epoch erfahrungsabhängig inkl. Abbruchkriterien (ausreichende oder stagnierende Güte)

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Trainieren des Modells (hier mit Entscheidungsbaum)

1. Lade Bibliothek
2. Definiert einen untrainierten Entscheidungsbaum und weist ihn der Variable Model_Decision_Tree zu.
3. Trainieren des Modells mit Trainingsdaten (x_train für Features und y_train für Labels).
4. Verwendung des trainierten Modells, um Vorhersagen für die x_test-Daten zu treffen und speichert die VORHERSAGE-Ergebnisse in y_pred_DECISION_TREE.

```
from sklearn.tree import DecisionTreeClassifier

Model_Decision_Tree = DecisionTreeClassifier()

Model_Decision_Tree.fit(x_train, y_train)
```

```
y_pred_decision_tree = Model_Decision_Tree.predict(x_test)
```

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Hilfsfunktion für Modellgüte

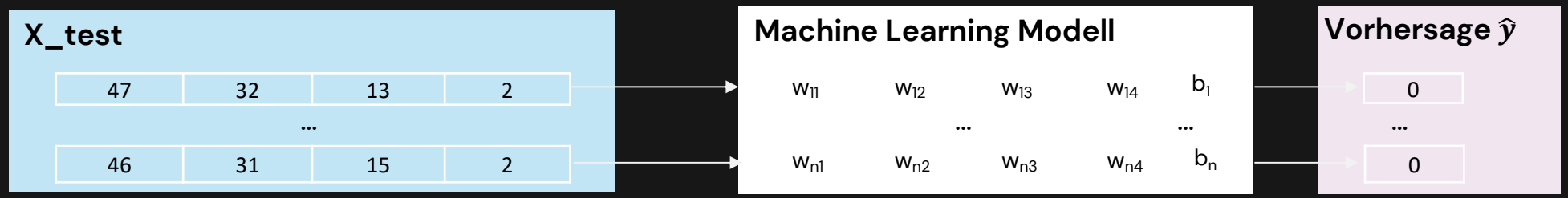
```
def Model_accuracy(predicted_y_values, real_y_values):
  print(confusion_matrix(real_y_values, predicted_y_values))
  print(classification_report(real_y_values, predicted_y_values))
  print('Die Genauigkeit ist: ',accuracy_score(predicted_y_values, real_y_values))
```

Der Code definiert eine Funktion Model_accuracy, die vorhergesagte und tatsächliche Werte aufnimmt und dann die Konfusionsmatrix, den Klassifizierungsbericht und die Genauigkeit der Vorhersagen ausgibt.

Case Study Supervised Learning: Einweisung Krankenhaus

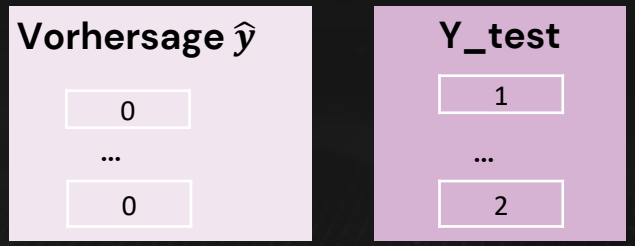
Schritt 4: Modellgüte prüfen

1. Feed Forward: Vorhersage Wert je **Sample**, d.h. eine individuelle Zeile des Datensatzes



2. Berechnen Modellgüte und Fehler des Modells

- **Validation Accuracy:** Berechnen Delta zwischen realem und vorhergesagtem Wert
- **Klassifikationsmetriken:** Confusion Matrix
- **Regressionsmetriken**: MSE, RSME, …



Durchführen Schritte 1-3 inkl. Update Modellparameter für gesamtes Trainingsset heißt **Epoch**, für Teile Trainingsset **Batch.**
Anzahl Durchläufe je Batch/ Epoch erfahrungsabhängig inkl. Abbruchkriterien (ausreichende oder stagnierende Güte)

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 4: Betrachten Ergebnisse (hier mit Entscheidungsbaum)

```
Model_accuracy(y_pred_SVM, y_test)
```

```
[[714  52]
 [351 207]]
              precision    recall  f1-score   support

           0       0.67      0.93      0.78       766
           1       0.80      0.37      0.51       558

    accuracy                           0.70      1324
   macro avg       0.73      0.65      0.64      1324
weighted avg       0.72      0.70      0.66      1324

Die Genauigkeit ist:  0.695619335347432
```

Verwenden der Hilfsfunktion

69% Genauigkeit ist nicht gut....

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 5: Modellgüte validieren - wie bewertet man die Güte eines Modells?

Klassifizierungsmetriken

Confusion Matrix

		Vorhergesagt	
		Ja	**Nein**
Tatsächlich	**Ja**	True Positive	False Negative
	Nein	False Positive	True Negative

```
from sklearn.metrics import confusion_matrix
confusion_matrix(real_y, predicted_y)
```

Accuracy

$$\frac{Anzahl\ richtige\ Vorhersagen}{Anzahl\ aller\ Vorhersagen}$$

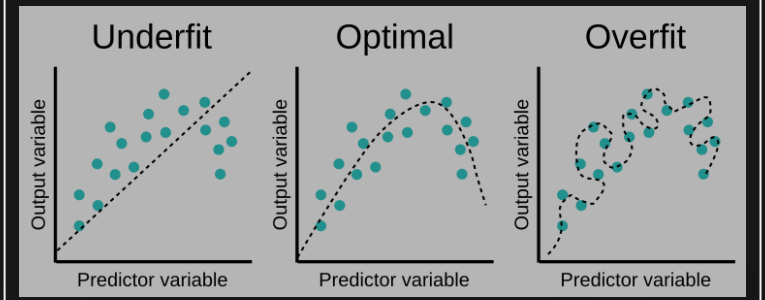
```
from sklearn.metrics import accuracy_score
accuracy_score(predicted_y, real_y)
```

Daraus abgeleitete Metriken:
- F1 Score
- Precision
- Recall
- ROC/ AUC

Regressionsmetriken

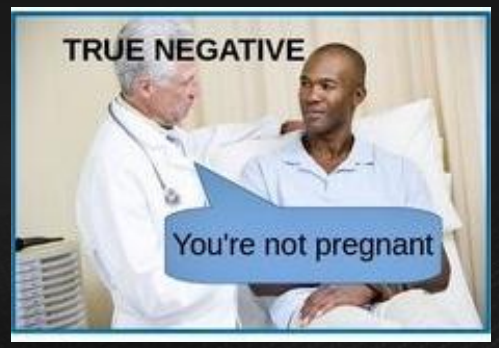
- **MAE (Mean absolute error)**
- **MSE**
- **RSME**
- **R-squared**
- Adjusted R squared
- Explained Variance

Over- und Underfitting



Case Study Supervised Learning: Einweisung Krankenhaus

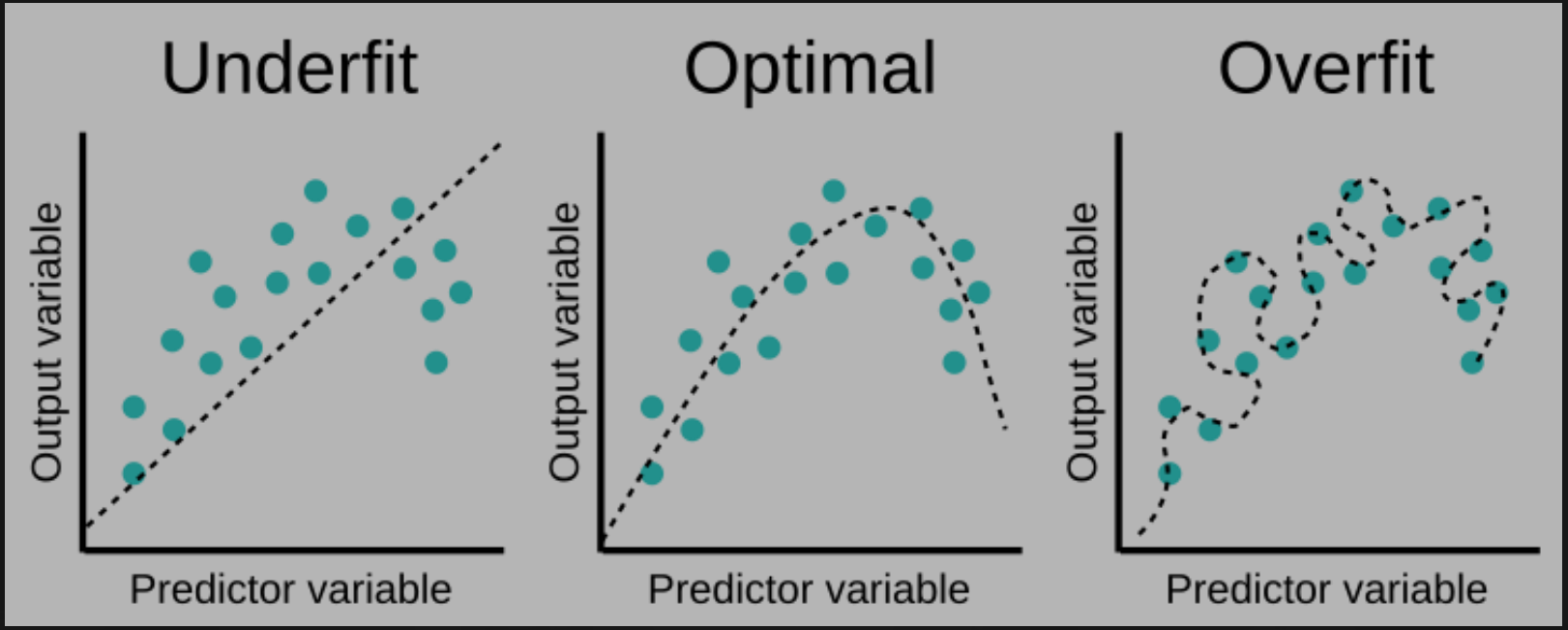
Schritt 5: Modellgüte validieren – Confusion Matrix

		Vorhersage
	schwanger	nicht schwanger
schwanger		
Tatsächlich nicht schwanger		

Welche Fehlerklasse ist schlimmer?

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 5: Modellgüte validieren – over and underfitting



Underfitting (Hoher bias/ Verzerrung):
wichtige Abhängigkeiten werden nicht erkannt, da Modell zu simpel.
→ Erkennbar an: schlechter Accuracy Trainings- und Testdaten.

Overfitting (Hohe Varianz):
nicht relevante Eigenheiten Trainingsdaten werden gelernt („Auswendiglernen"), Modell zu komplex
→ Erkennbar an: Trainings-Accuracy steigt und ist höher als die Test-Accuracy, Test Accuracy stagniert.

Eselsbrücke anhand Prüfungsvorbereitung:
- Lernten Sie zu wenig, dann haben Sie sowohl bei Übungsprüfungen als auch bei realer Prüfung viele Fehler (Sie wissen nicht, was wichtig ist).
- Lernten Sie sehr viel anhand von Übungsprüfungen, dann hilft das nur, falls die der realen Prüfung ähneln (Sie lernten nicht, was dran kam).

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 5: Modellgüte validieren – Woran erkenne ich, was optimal ist?

- Bau einer Baseline und Vergleich mit Model Performance:
  - Wie gut ist ein ganz simples Verfahren? (Bspw. Raten bei 2 Zielklassen ist 50% Chance).
  - Wie gut ist ein ganz simples ML-Verfahren (bspw. lineare Regression?)
  - Wie gut ist ein Mensch (Human baseline, bspw. Arzt bei Diagnose)

„Optimales“ Model: Modell und Hyperparameter verbessern, bis bessere Performance als Mensch und kein Overfitting!

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 5: Modellgüte optimieren

- Optimieren durch Wahl anderer ML-Verfahren
- Optimieren durch Ändern Parameter des Modells (Hyperparameter-Tuning)
- Automatisiertes Ausprobieren (Bayesian Model Averaging, CV-Grid, …)

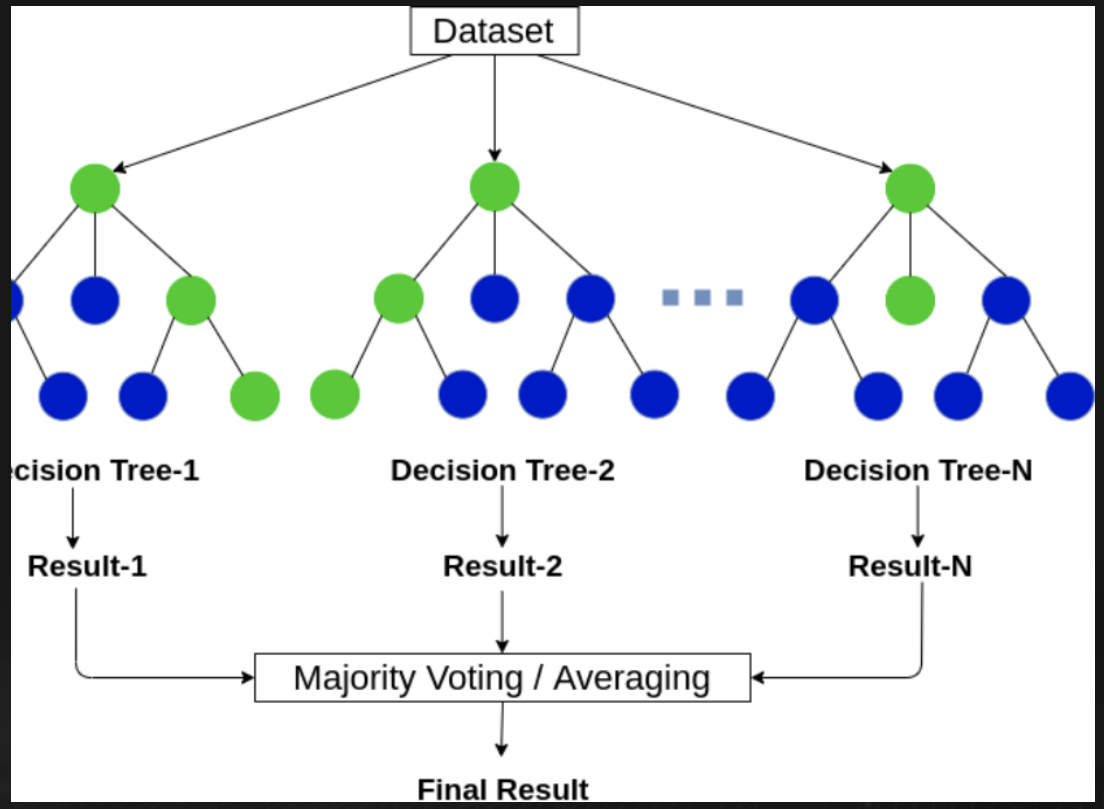
Nicht im Fokus Vorlesung

Case Study Supervised Learning: Einweisung Krankenhaus

Schritt 5: Modellgüte optimieren

Random Forest (Ensemble Learning):

- Kombiniert mehrere Entscheidungsbäume, um robusteres & genaueres Modell zu lernen.
- Jeder Baum wird auf anderen Teilmenge Daten trainiert, die ergebnisse der einzelnen bäume werden gleich gewichtet zu einem finalen Ergebnis kombiniert („Bagging“).
- Zufälligkeit Merkmale: Aufteilungs-Entscheidungen jedes Baums werden anhand zufälligeR Teilmenge Merkmale getroffen.
- Verringert Overfitting: Kombination vieler Bäume verringert Risiko Überanpassung im Vergleich zu einzelnem Entscheidungsbaum.
- FEATURE IMPORTANCE: Wie bei Entscheidungs-bäumen kann Wichtigkeit jedes Merkmals bei Erstellung von Vorhersagen gemessen werden.



Case Study Supervised Learning: Einweisung Krankenhaus

Random Forest trainieren

1. Lade Bibliothek
2. initialisiert (untrainiertes) Random Forest Modell und weist der Variablen Model_RANDOM_FOREST zu.
3. Trainieren des Modells mit Trainingsdaten (x_train für Features und y_train für Labels).
4. Verwendung des trainierten Modells, um Vorhersagen für die x_test-Daten zu treffen und speichert die VORHERSAGE-Ergebnisse in y_pred_random_forest.

```
from sklearn.ensemble import RandomForestClassifier

Model_Random_Forest = RandomForestClassifier()

Model_Random_Forest.fit(x_train, y_train)
```

```
y_pred_random_forest = Model_Random_Forest.predict(x_test)
```

Case Study Supervised Learning: Einweisung Krankenhaus

Random Forest evaluieren

```
Model_accuracy(y_pred_random_forest, y_test)
```

```
    [[657 109]
     [230 328]]
                  precision    recall  f1-score   support

               0       0.74      0.86      0.79       766
               1       0.75      0.59      0.66       558

        accuracy                           0.74      1324
       macro avg       0.75      0.72      0.73      1324
    weighted avg       0.74      0.74      0.74      1324

    Die Genauigkeit ist:  0.7439577039274925
```

Case Study Supervised Learning: Einweisung Krankenhaus

Random Forest optimieren

- Dokumentation unter https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html zeigt Übersicht aller Parameter des Modelles selbst, die sogenannten Hyperparameter

```
[46]  rf_optimized = RandomForestClassifier(max_depth=2)
      rf_optimized.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=2)
```

```
y_pred_random_forest_optimized = rf_optimized.predict(X_test)
Model_accuracy(y_pred_random_forest_optimized, y_test)
```

```
[[677  89]
 [287 271]]
              precision    recall  f1-score   support

           0       0.70      0.88      0.78       766
           1       0.75      0.49      0.59       558

    accuracy                           0.72      1324
   macro avg       0.73      0.68      0.69      1324
weighted avg       0.72      0.72      0.70      1324

Die Genauigkeit ist:  0.716012084592145
```

Wir sind schlechter geworden ☹

```
rf_optimized = RandomForestClassifier(max_depth=8, n_estimators=300)
rf_optimized.fit(X_train, y_train)
y_pred_random_forest_optimized = rf_optimized.predict(X_test)
Model_accuracy(y_pred_random_forest_optimized, y_test)
```

```
[[678  88]
 [239 319]]
              precision    recall  f1-score   support

           0       0.74      0.89      0.81       766
           1       0.78      0.57      0.66       558

    accuracy                           0.75      1324
   macro avg       0.76      0.73      0.73      1324
weighted avg       0.76      0.75      0.74      1324

Die Genauigkeit ist:  0.7530211480362538
```

Mit 2 Parametern sind wir nicht viel besser geworden

Generischer Workflow Supervised Learning

Schritt 5: Was sind die wichtigsten Eigenschaften für die Vorhersage?

hole die für den Algorithmus wichtigsten Features

```
import matplotlib.pyplot as plt
import seaborn as sns

feature_importances = Model_Random_Forest.feature_importances_
col_names = ['HAEMATOCRIT', 'HAEMOGLOBINS', 'ERYTHROCYTE', 'LEUCOCYTE', 'THROMBOCYTE',  'MCH', 'MCHC', 'MCV', 'ALTER','GESCHLECHT']
indices = np.argsort(feature_importances) # sortiere die Feature Importances absteig

plt.title('Feature Importances') # Titel des Bildes
plt.barh(range(len(indices)), feature_importances[indices], color='b', align='center') # plotte die einzelnen Feature Importances
plt.yticks(range(len(indices)), [col_names[i] for i in indices]) # schrifte die einzelnen Werte auf der Y-Achse
plt.xlabel('Relative Importance')
plt.show()
```

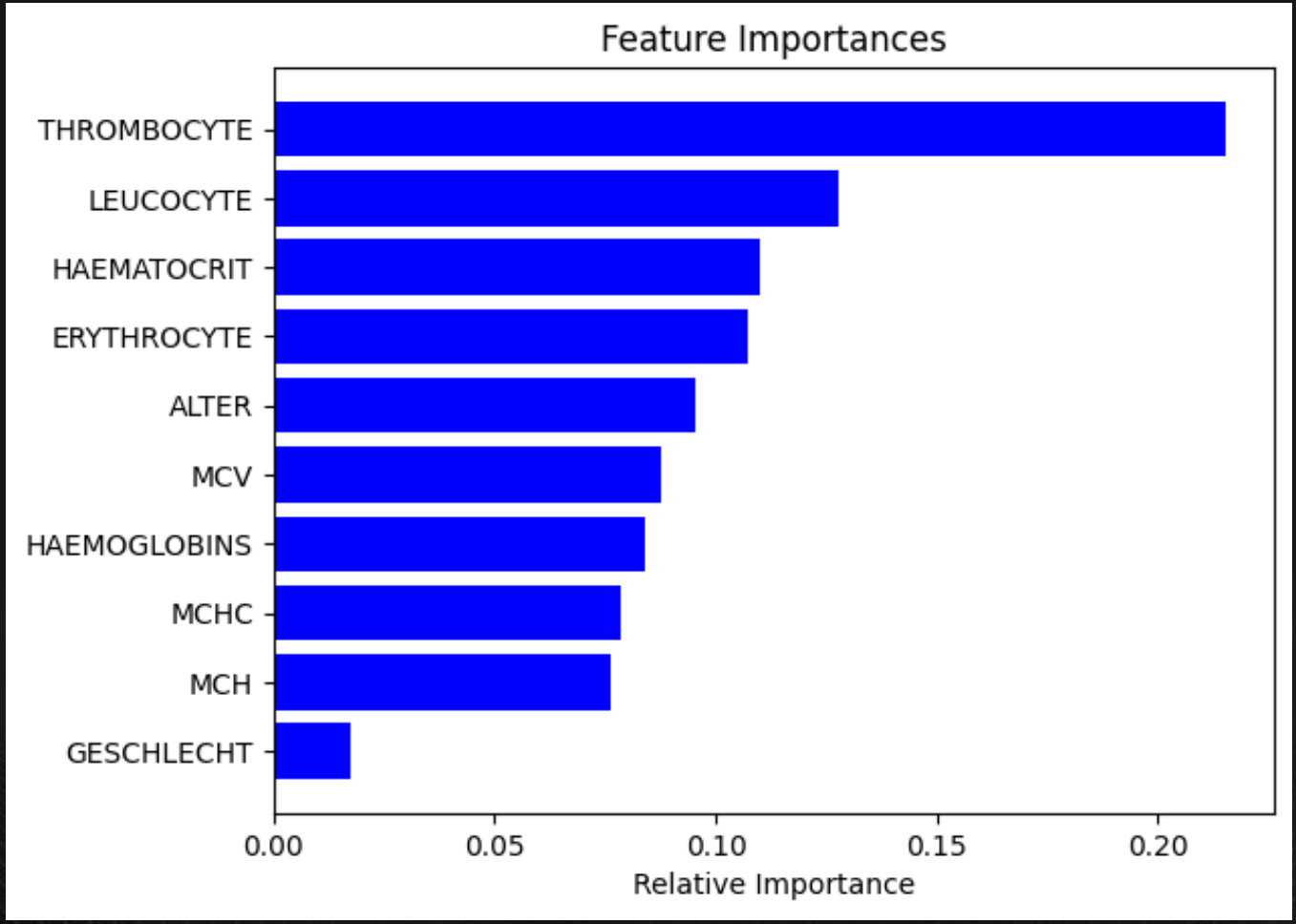
speichern der Features in ein Array. Wir benötigen das für die Beschriftung der y-Achse des Plots später

sortiert mit absteigender Wichtigkeit und speichert diese Reihenfolge in Indices

plotte die einzelnen Features horizontal (barh!). Reichweite y-Achse wird durch die Anzahl der Features definiert (range(len(indices)), die Werte für y sind Werte der Feature importances in absteigender Wichtigkeit

Generischer Workflow Supervised Learning

Schritt 5: Was sind die wichtigsten Eigenschaften für die Vorhersage?



Case Study Supervised Learning: Einweisung Krankenhaus

Ausblick: automatisiertes Optimieren Hyperparameter

Welche Hyperparameter sollen ausprobiert werden mit welchen Werten?

```python
from sklearn.model_selection import RandomizedSearchCV

rf_params = {
    'n_estimators': [100, 200],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 2, len(X.columns), 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

rf_random_search = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
                                      param_distributions=rf_params,
                                      n_iter=100,
                                      cv=5,
                                      random_state=42)

rf_random_search.fit(X_train, y_train)
```

Kombiniere die Hyperparameter mit den zu testenden Werten zufällig miteinander über 100 Iterationen mit 5mal verschiedener Aufteilung Daten.

Drucke Setup des besten Ergebnis aus

```
[54]  rf_random_search.best_params_

{'n_estimators': 200,
 'min_samples_split': 2,
 'min_samples_leaf': 4,
 'max_features': 'sqrt',
 'max_depth': 10,
 'criterion': 'entropy'}
```

```python
RF_optimized_best_results = RandomForestClassifier(
    n_estimators=200,
    criterion='entropy',
    max_depth=10,
    min_samples_split=2,
    min_samples_leaf=4,
    max_features='sqrt')

RF_optimized_best_results.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf=4,
                       n_estimators=200)
```

Definiere Random Forest mit besten Parametern

Case Study Supervised Learning: Einweisung Krankenhaus

Ausblick: automatisiertes Optimieren Hyperparameter

```
RF_optimized_best_results = RandomForestClassifier(
    n_estimators=200,
    criterion='entropy',
    max_depth=10,
    min_samples_split=2,
    min_samples_leaf=4,
    max_features='sqrt')

RF_optimized_best_results.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf=4,
                       n_estimators=200)
```

```
y_pred_random_forest_optimized_best_results = RF_optimized_best_results.predict(X_test)
Model_accuracy(y_pred_random_forest_optimized_best_results, y_test)
```

```
[[678  88]
 [231 327]]
              precision    recall  f1-score   support

           0       0.75      0.89      0.81       766
           1       0.79      0.59      0.67       558

    accuracy                           0.76      1324
   macro avg       0.77      0.74      0.74      1324
weighted avg       0.76      0.76      0.75      1324

Die Genauigkeit ist:  0.7590634441087614
```

75,9%...damit sind wir um 0,6% besser geworden

Literatur

Künstliche Intelligenz:

- Gröner, Heinecke: Kollege KI
- Burkov: The Hundred-Page Machine Learning Book, online verfügbar unter Link
- Nielsen: Neural Networks and Deep Learning, online verfügbar unter Link
- Russel, Norvig: Artificial Intelligence – a modern approach
- Produktentwicklung mit AI:
  - Ameisen: Building Machine Learning Powered Applications: Going from Idea to Product
  - Ng: Machine Learning Yearning, online verfügbar unter Link

- Wie können Sie Ihr Modell optimieren? https://github.com/google-research/tuning_playbook

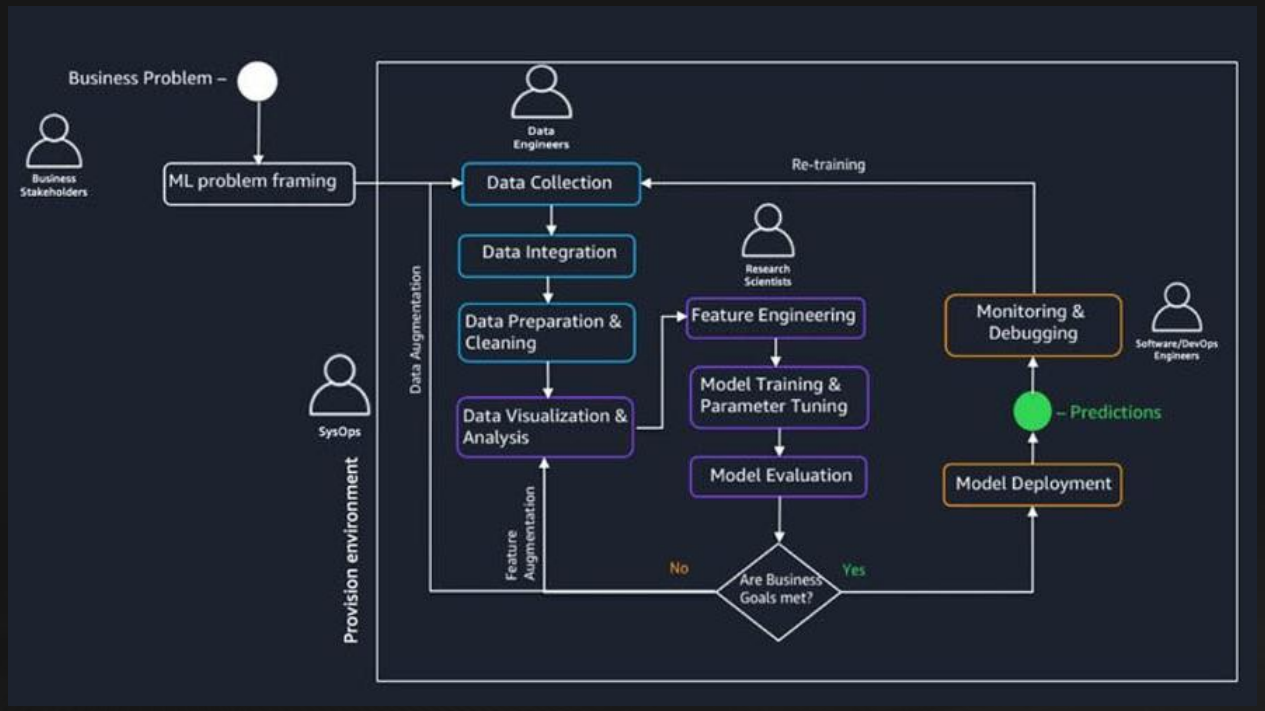
Kostenfreie Online-Kurse (bei Interesse):

- Python-Kurse
  - Python for Everybody (Link)
  - Udacity Python Course (Link)
  - **Coursera Course Deep Learning** (Link)
  - FAST AI (Link)

BACKUP

Generischer Workflow Supervised Learning

Übersicht



1. Customer/ Business Problem und Metriken definieren
2. Daten organisieren, aufbereiten, säubern, ggf. neue Features bauen
3. Machine Learning Vorbereiten:
   - Aufteilen Daten aufteilen in Trainings- und Testmenge (und ggf. Validierungsmenge).
   - Algorithmus und Kosten-/Optimierungsfunktion wählen und Modellparameter initialisieren.
4. Training: schrittweise Optimierung Modellparameter bis Modell möglichst gute Performance für Trainingsmenge hat.
5. Modell(-güte) validieren mit Testmenge, dessen Elemente Modell nicht kannte. Falls Modellgüte i.O., weiter zu Schritt 6. Sonst zu Schritt 5.
6. Deployment: Modell „live“ vor Kunde schalten.
7. Monitoring: kontinuierliches Überprüfen Modellgüte [„Modelldrift“] und Modellinput [„Parameterdrift“]

Ziel: Lernen eines möglichst genauen Modells H(Input) = Zielgröße

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 1: BUSINESS/ CUSTOMER PROBLEM STATEMENT.

- Allgemein:
  - Welches Business/ Customer problem wollen Sie mit AI lösen?
  - Woran erkennen Sie, ob und wie gut Sie das Problem lösen?
  - Woher bekommen Sie die Daten?

  → Dies kann bspw. mit einer Business Canvas oder Ansatz „Working backwards from Customer" erfolgen.

- In unserem Fall:
  - Business Problem: Automatisierung Kreditvergabe inkl. Vermeiden von Kreditausfall.
  - Kundenproblem: langfristiger Prozess Kreditvergabe.
  - Problem wird gut gelöst, wenn schnell eine Aussage über Kreditwürdigkeit erfolgt bei Minimierung Kreditausfälle.
  - Woher kommen Daten: Beispielhafter Datensatz.

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 2.1: DATEN ORGANISIEREN.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	Semiurban	N
10	LP001024	Male	Yes	2	Graduate	No	3200	700.0	70.0	360.0	1.0	Urban	Y
11	LP001027	Male	Yes	2	Graduate	NaN	2500	1840.0	109.0	360.0	1.0	Urban	Y
12	LP001028	Male	Yes	2	Graduate	No	3073	8106.0	200.0	360.0	1.0	Urban	Y
13	LP001029	Male	No	0	Graduate	No	1853	2840.0	114.0	360.0	1.0	Rural	N
14	LP001030	Male	Yes	2	Graduate	No	1299	1086.0	17.0	120.0	1.0	Urban	Y
15	LP001032	Male	No	0	Graduate	No	4950	0.0	125.0	360.0	1.0	Urban	Y
16	LP001034	Male	No	1	Not Graduate	No	3596	0.0	100.0	240.0	NaN	Urban	Y
17	LP001036	Female	No	0	Graduate	No	3510	0.0	76.0	360.0	0.0	Urban	N
18	LP001038	Male	Yes	0	Not Graduate	No	4887	0.0	133.0	360.0	1.0	Rural	N
19	LP001041	Male	Yes	0	Graduate	NaN	2600	3500.0	115.0	NaN	1.0	Urban	Y

Quelle: Link

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 2.2: DATEN AUFBEREITEN.

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0	0	0
2	1	1	0	0	1	3000	0.0	66.0	360.0	1.0	2	1
3	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0	2	1
4	1	0	0	0	0	6000	0.0	141.0	360.0	1.0	2	1
5	1	1	2	0	1	5417	4196.0	267.0	360.0	1.0	2	1
6	1	1	0	1	0	2333	1516.0	95.0	360.0	1.0	2	1
7	1	1	3	0	0	3036	2504.0	158.0	360.0	0.0	1	0
8	1	1	2	0	0	4006	1526.0	168.0	360.0	1.0	2	1
9	1	1	1	0	0	12841	10968.0	349.0	360.0	1.0	1	0
10	1	1	2	0	0	3200	700.0	70.0	360.0	1.0	2	1
12	1	1	2	0	0	3073	8106.0	200.0	360.0	1.0	2	1
13	1	0	0	0	0	1853	2840.0	114.0	360.0	1.0	0	0
14	1	1	2	0	0	1299	1086.0	17.0	120.0	1.0	2	1
15	1	0	0	0	0	4950	0.0	125.0	360.0	1.0	2	1
17	0	0	0	0	0	3510	0.0	76.0	360.0	0.0	2	0
18	1	1	0	1	0	4887	0.0	133.0	360.0	1.0	0	0
20	1	1	0	1	0	7660	0.0	104.0	360.0	0.0	2	0
21	1	1	1	0	0	5955	5625.0	315.0	360.0	1.0	2	1
22	1	1	0	1	0	2600	1911.0	116.0	360.0	0.0	1	0
25	1	1	0	0	1	9560	0.0	191.0	360.0	1.0	1	1

Eingesetzte Data Engineering Operationen:
- Löschen Spalte Loan_id
- Label Encoding: jeder Wert eines Features erhält eine eindeutige Zahl

Saubere, viele Daten sind der wichtigste Faktor um ein möglichst gutes Modell zu erhalten!

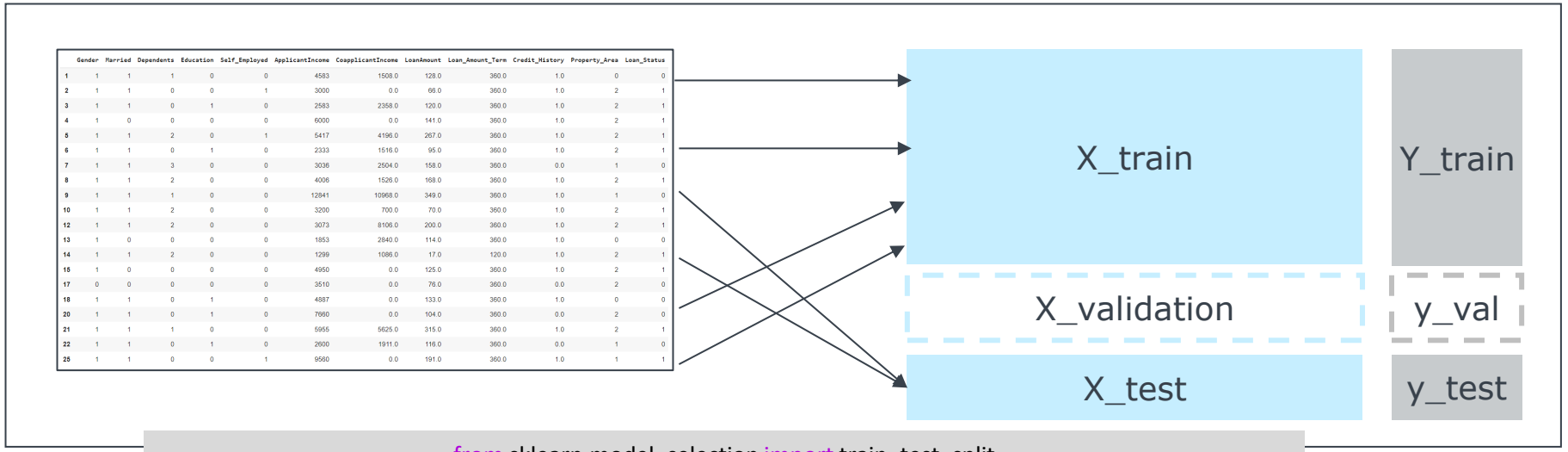
Ziel der Datenaufbereitung ist das Verbessern der Datenqualität um somit bessere Ergebnisse beim Machine Learning zu erhalten. Dazu gehört z.B. das Beseitigen von Fehlern, Datentypumwandlung, Erstellen neuer Features und Bestimmen Zielklassen („labeln“).

Quelle: Link

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 2.3: DATENAUFTEILUNG.

Matrix mit Features/ Eigenschaften

Vektor mit den Zielwerten



	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0	0	0
2	1	1	0	0	1	3000	0.0	66.0	360.0	1.0	2	1
3	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0	2	1
4	1	0	0	0	0	6000	0.0	141.0	360.0	1.0	2	1
5	1	1	2	0	1	5417	4196.0	267.0	360.0	1.0	2	1
6	1	1	0	1	0	2333	1516.0	95.0	360.0	1.0	2	1
7	1	1	3	0	0	3036	2504.0	158.0	360.0	0.0	1	0
8	1	1	2	0	0	4006	1526.0	168.0	360.0	1.0	2	1
9	1	1	1	0	0	12841	10968.0	349.0	360.0	1.0	1	0
10	1	1	2	0	0	3200	700.0	70.0	360.0	1.0	2	1
12	1	1	2	0	0	3073	8106.0	200.0	360.0	1.0	2	1
13	1	0	0	0	0	1853	2840.0	114.0	360.0	1.0	0	0
14	1	1	2	0	0	1299	1086.0	17.0	120.0	1.0	2	1
15	1	0	0	0	0	4950	0.0	125.0	360.0	1.0	2	1
17	0	0	0	0	0	3510	0.0	76.0	360.0	0.0	2	0
18	1	1	0	1	0	4887	0.0	133.0	360.0	1.0	0	0
20	1	1	0	1	0	7660	0.0	104.0	360.0	0.0	2	0
21	1	1	1	0	0	5955	5625.0	315.0	360.0	1.0	2	1
22	1	1	0	1	0	2600	1911.0	116.0	360.0	0.0	1	0
25	1	1	0	0	1	9560	0.0	191.0	360.0	1.0	1	1

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Test_size: Verhältnis Trainings-/ zu Testmenge
Random_state = ermöglicht reproduzierbare zufällige Verteilung.

- Aufgabe **Trainingsset**: Lernen/ Optimieren der Parameter des Machine Learning Modells.
- Aufgabe **Testset**: Evaluation, wie gut die Parameter und somit das Modell sind.
- **Zufällige Aufteilung** in Trainings- und Testmenge stellt ähnliche statistische Eigenschaften sicher (sortierte Ausgangsdaten!)
- Oft wird **Validierungsset** zur Optimierung der Parameter des Trainingsprozesses (**Hyperparameter**) gebildet.
- Verhältnis Trainings- zu Testmenge: oft 70% zu 30%. Bei Einsatz Validierungsmenge: oft 60%-20%-20%.

Eselsbrücke anhand Prüfungsvorbereitung (z.B. Führerschein):
- Zur Prüfungsvorbereitung lernen Sie anhand Prüfungsbögen und vergleichen ihre Antworten mit vorgegebenen Musterantworten.
- Sie sehen ihre Fehler und üben an den Prüfungsbögen solange, bis Sie die Prüfungsbögen „gut genug" können.
- Bei der realen Prüfung kennen Sie die Antworten nicht und sehen dann, wie gut Sie den Stoff können.

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 3: MODELL WÄHLEN UND INITIALISIEREN.

1. **Wahl Machine Learning-Modell** abhängig von:
   - Datentyp: Zahlen, Bilder, Sprache, …. → tabellarische Daten
   - Verfügbaren Ressourcen: Rechen-Power, Zeit, Kosten, …
   - Datenmenge: viele oder wenig Daten vorhanden? → wenige Daten (Testdatensatz)
   - Qualität Daten: fehlende Werte/ Ausreißer stören manche Lernverfahren. → einige fehlende Werte
   - Anforderungen an Modellgüte → sehr hoch
   - Komplexität Algorithmus: komplexe Algorithmen neigen zu hoher Streuung, simple zu hoher Verzerrung.

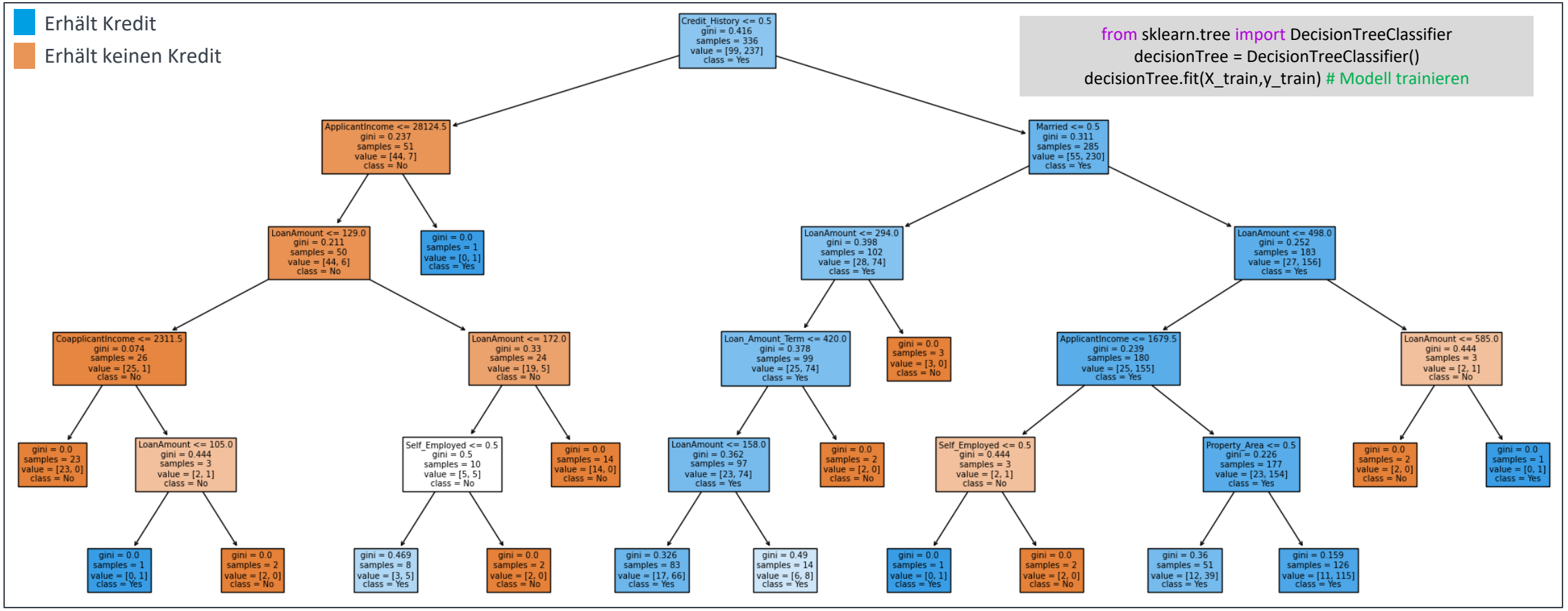
Zur Veranschaulichung nehmen wir Decision Tree.

2. **Initialisierung Parameter**
   - Hyperparameter, d.h. Parameter für Trainingsprozess: bspw. „maximale Tiefe Baum"
   - Modellparameter: werden gelernt.

3. **Wahl Loss-Funktion:** Bewertungsfunktion f. Genauigkeit Modell → Anzahl Fehlklassifikationen.

Die Auswahl Machine Learning Verfahren ist abhängig von diesen Fragen und ist erfahrungsbasierend.
Es gibt für die meisten Datentypen jedoch Standardverfahren, die auch für die Parameter gute Standardwerte treffen.

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 4: TRAINING

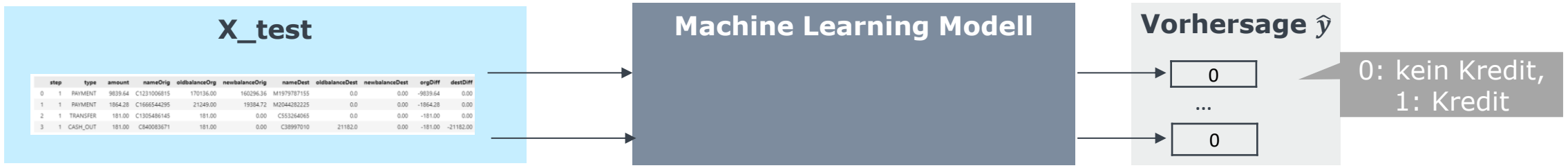


Klassifikation: wiederholtes Aufteilen Datenset in zwei Untermengen anhand des Features mit höchstem Informationsgehalt.

Quelle: Link

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 5: MODELL TESTEN.

1. Feed Forward: Vorhersage Wert je **Sample**, d.h. eine individuelle Zeile des Datensatzes



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	orgDiff	destDiff
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	-9839.64	0.00
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	-1864.28	0.00
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	-181.00	0.00
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	-181.00	-21182.00

```
Y_pred_Tree = decisionTree.predict(X_test) # Modell trainieren
```

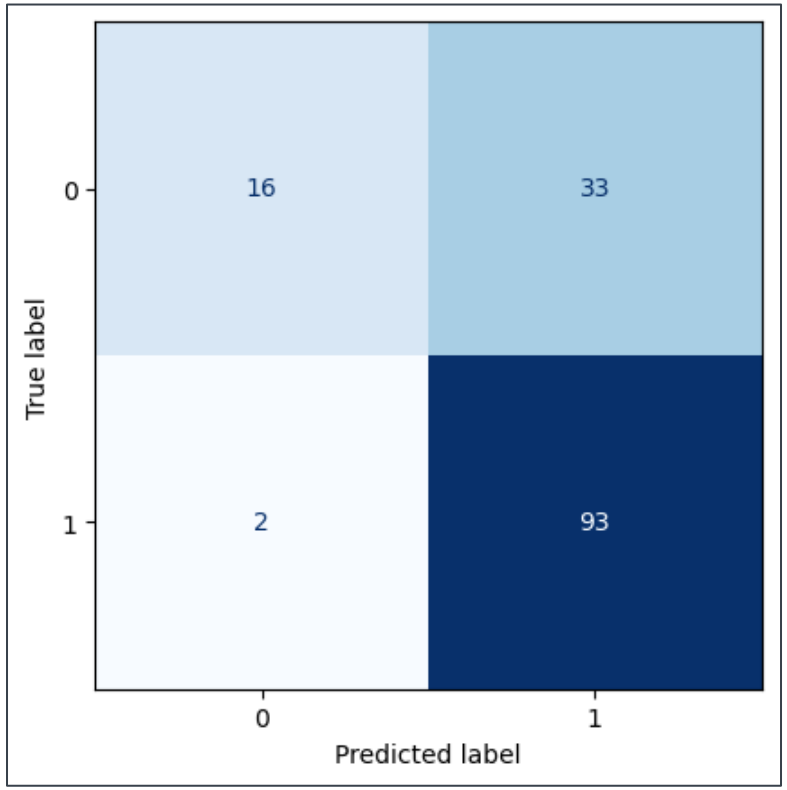
2. Berechnen Modellgüte und Fehler des Modells

- **Validation Accuracy:** Berechnen Delta realer zu vorhergesagtem Wert
- **Klassifikationsmetriken:** Confusion Matrix → nächste Seite
- **Regressionsmetriken**: MSE, RSME, …

Eselsbrücke anhand Prüfungsvorbereitung (z.B. Führerschein):
- Zur Prüfungsvorbereitung lernen Sie anhand Prüfungsbögen und vergleichen ihre Antworten mit vorgegebenen.
- Sie machen das solange, bis Sie die Prüfungsbögen „gut genug" können.
- Bei der realen Prüfung kennen Sie die Antworten nicht und sehen dann, wie gut Sie den Stoff können.

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 5: MODELL TESTEN: ERGEBNISSE.

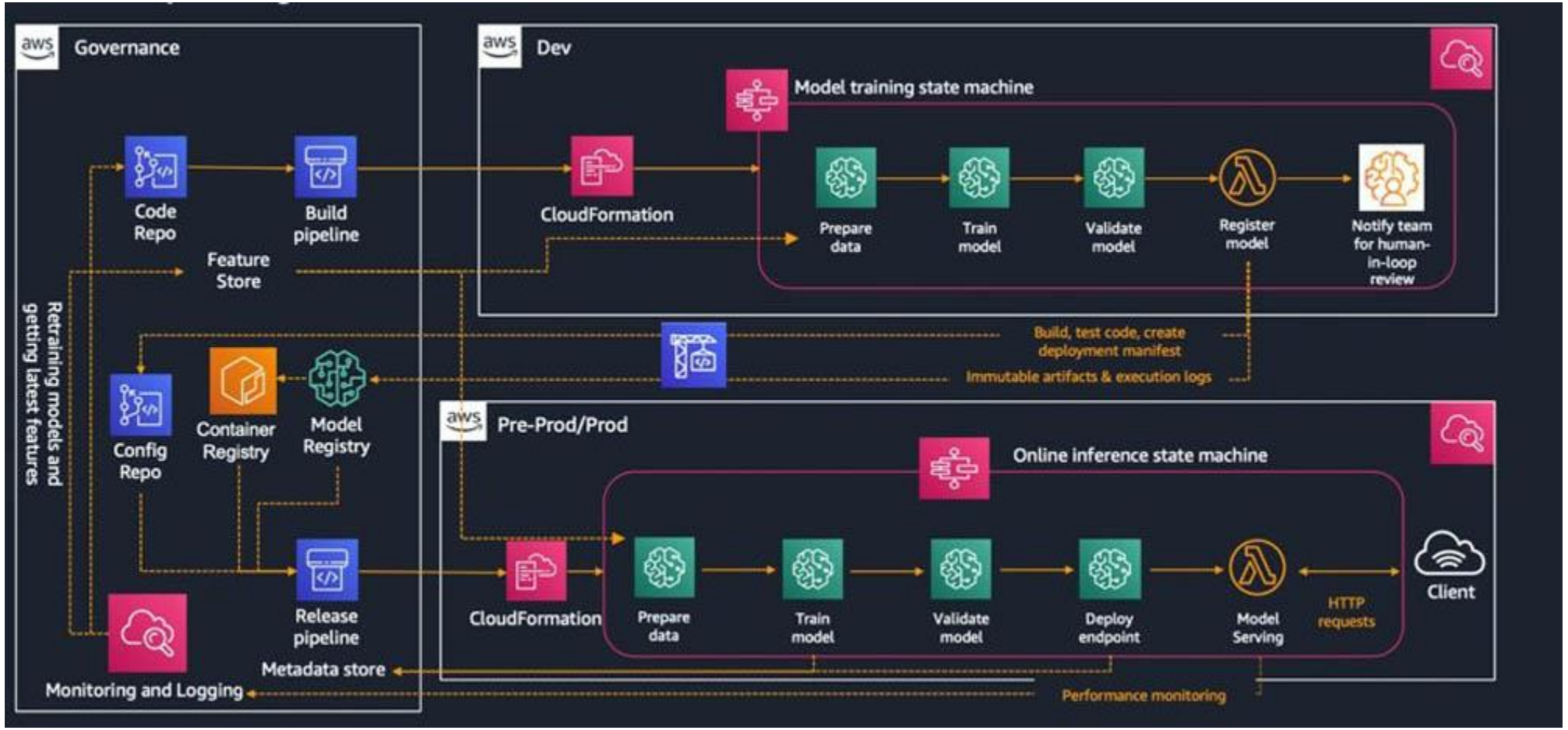
$$\frac{Anzahl\ richtige\ Vorhersagen}{Anzahl\ aller\ Vorhersagen} = 74\%$$




```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred_DecTree, y_test)
```

```
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, display_labels=clf.classes_,)
```

FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 6: MODEL AT SCALE.



Quelle: Link

Detaillierung ausgewählter Supervised Algorithmen

DETAILLIERUNG ALGORITHMEN SUPERVISED LEARNING.

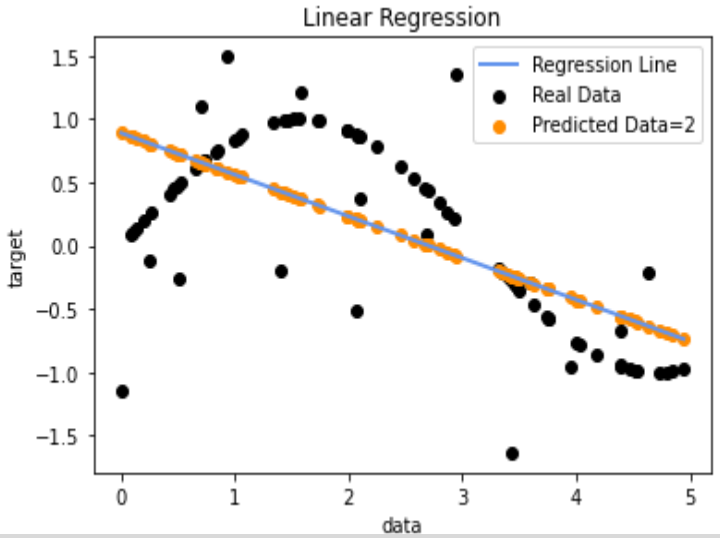
Auf den folgenden Folien werden geläufige Algorithmen an zwei Beispielen veranschaulicht.

- **Regression**: Vorhersage der Werte einer Sinuskurve. Dabei ist Sinuskurve an bestimmten Stellen manuell geändert, um die Vorhersage zu erschweren.
- **Klassifikation**: anhand Iris-Datensatz. Da dieser Datensatz 4 Features/ Dimensionen hat, läßt er sich schwer graphisch darstellen. Zur Veranschaulichung wurde der Datensatz deshalb auf die 2 Sepal-Werte reduziert.

Eine Übersicht bekannter Algorithmen finden Sie bspw. hier: Link

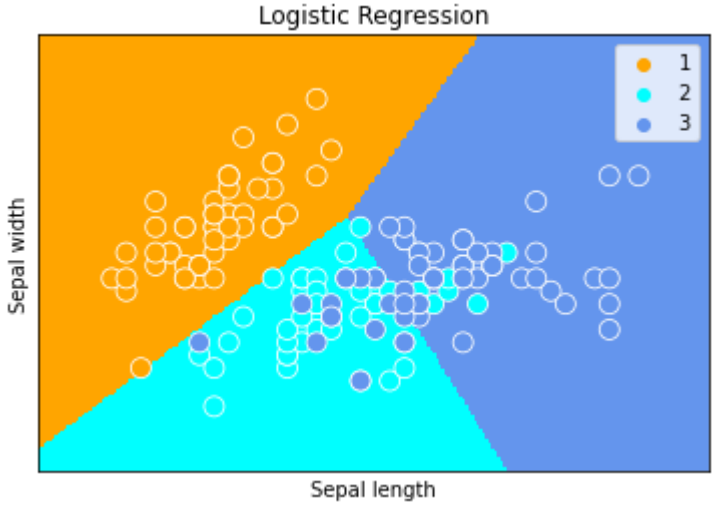
SUPERVISED LEARNING: LINEARE/ LOGISTISCHE REGRESSION.

Regression kontinuierliche Variable




```
from sklearn.linear_model import LinearRegression
linRegressor = LinearRegression() # Modell initialisieren
linRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_LR = linRegressor.predict(X_test) # vorhersagen
```

Klassifizierung diskrete Variable




```
from sklearn.linear_model import LogisticRegression
logRegressor = LogisticRegression() # Modell init.
logRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_LR = logRegressor.predict(X_test) # vorhersagen
```

Vorteile:

- Leicht verständlicher Algorithmus und Ergebnisse.
- Leicht implementierbar.

Nachteile:

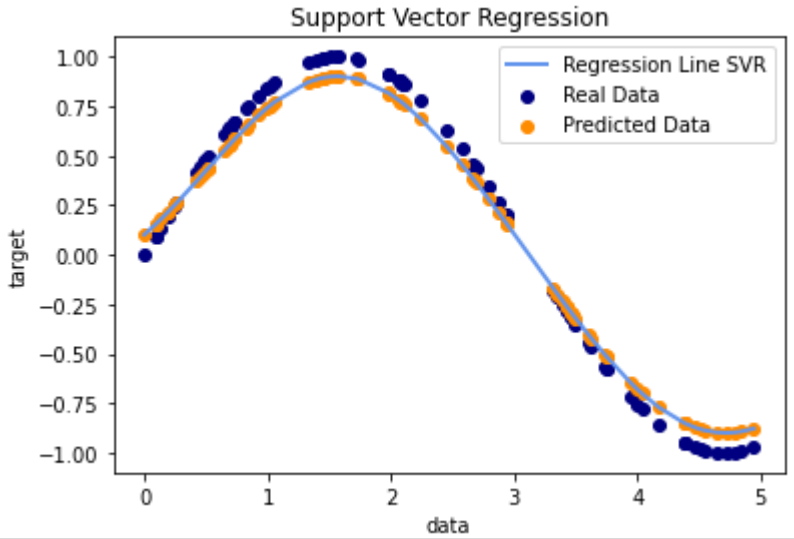
- Probleme mit Ausreißern.
- Benötigt Zusammenhang/ Abhängigkeit zwischen Variablen.
- Probleme mit komplexen/ hochdimensionalen Daten.

Lineare Regression versucht, die Zielvariable durch ein lineares Modell (Gerade) zu beschreiben.
Die Gerade/n ist/sind dabei so konstruiert, daß sie möglichst geringen Abstand von den Punkten des Datensatzes hat.

Quellen: SK-Learn Dokumentation: Link, Link

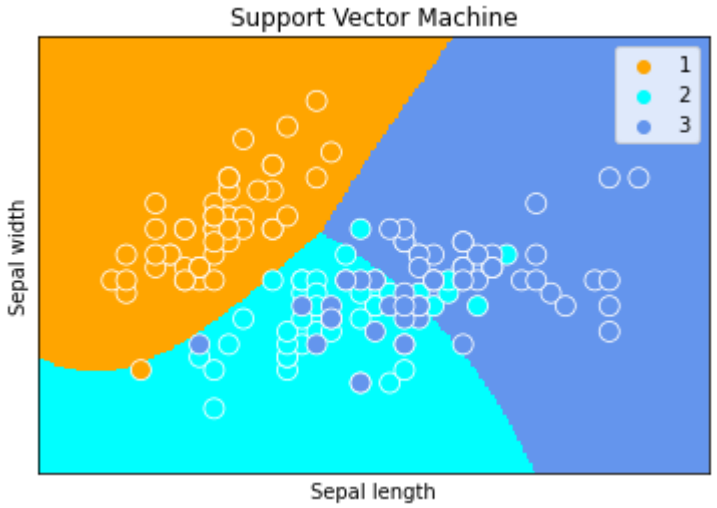
SUPERVISED LEARNING: SUPPORT VECTOR MACHINE.

Regression kontinuierliche Variable




```
from sklearn import svm
SVR_basic = svm.SVR() # Modell initialisieren
SVR_basic.fit(X_train,y_train) # Modell trainieren
Y_pred_SVR = SVR_basic.predict(X_test) # vorhersagen
```

Klassifizierung diskrete Variable




```
from sklearn import svm
SVC_basic = svm.SVC() # Modell initialisieren
SVC_basic.fit(X_train,y_train) # Modell trainieren
Y_pred_SVC = SVC_basic.predict(X_test) # vorhersagen
```

Vorteile:

- Robust auch bei hochdimensionalen Features.
- Einsetzbar für Klassifikation und Regression.
- Einsetzbar auch für Klassifikation Bilder.

Nachteile:

- Viele Hyperparameter, deshalb Tuning notwendig.
- Längere Trainingszeit bei großen Datenmengen.
- Erklärbarkeit Ergebnisse schwierig (Hyperebene n-dimensionaler Raum).

Klassifikation: Algorithmus bestimmt „Hyperebene“[1], die Datensatz am besten in die gewünschte Anzahl Klassen einteilt. Dabei wird die Hyperebene so gelegt, daß die Grenze zwischen den Klassen möglichst breit ist.
Regression: Algorithmus bestimmt „Hyperebene“ und nimmt Punkt, der möglichst nah an der Hyperebene liegt

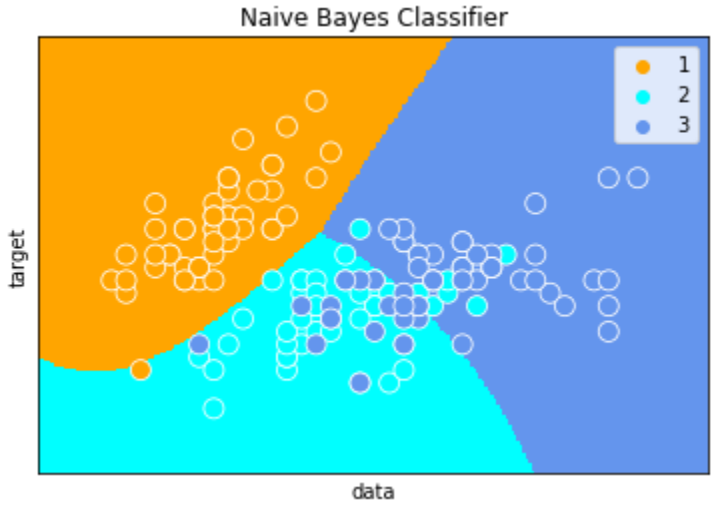
Quelle: Vapnik und Chervonenkis, Theory of Pattern Recognition, 1974.
1 Hyperebene: Ebene mit um 1 geringeren Dimension als Datensatz (Bspw. Gerade für 2 Features oder flaches Rechteck im Dreidimensionalen).

SUPERVISED LEARNING: NAIVE BAYES.

Regression kontinuierliche Variable

-

Klassifizierung diskrete Variable




```
from sklearn.naive_bayes import  GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train) # Modell trainieren
Y_pred_GB = gnb.predict(X_test)
```

Vorteile:

- Kann gut mit hochdimensionalen Daten umgehen.
- Wenig Trainingsdaten notwendig.
- Schnell berechenbarer Algorithmus.
- Oft bessere Genauigkeit als kompliziertere Modelle (aber nur falls Variablen statistisch unabhängig).

Nachteile:

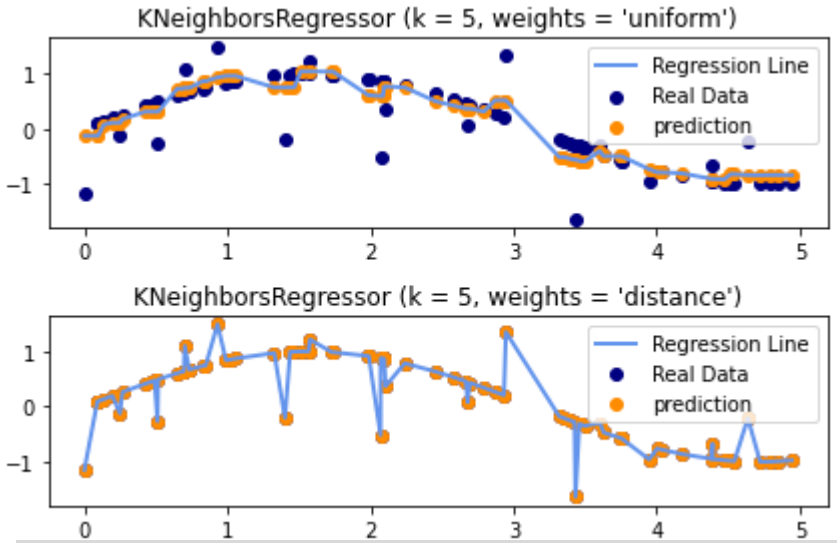
- Annahme bedingte Unabhängigkeit nicht immer gegeben.
- Bei großen Datenmengen liefern andere Algorithmen bessere Ergebnisqualität (wegen Bayes-Regel für Updates).

Nutzt Bayes-Theorem für das Berechnen der Zugehörigkeit eines Features zu einer Klasse und ordnet das Feature der Klasse zu, für die die berechnete Wahrscheinlichkeit am höchsten ist.

Quellen: SK-Learn Dokumentation: Link

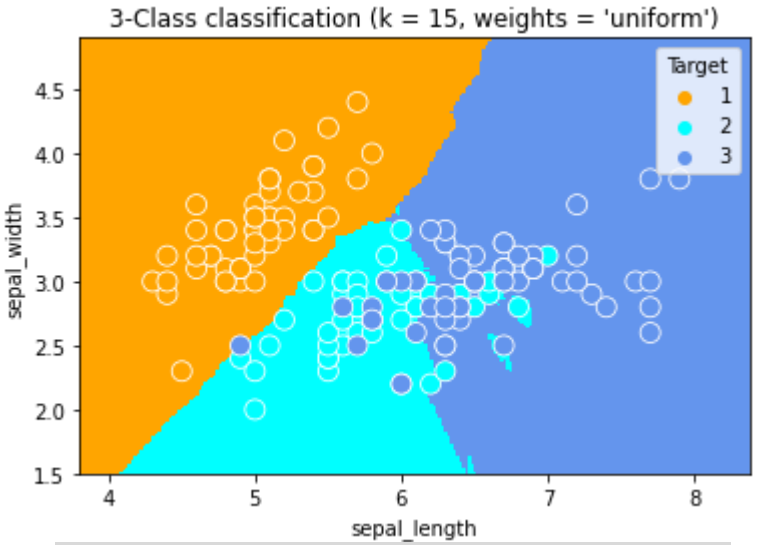
SUPERVISED LEARNING: K NEAREST NEIGHBOURS.

Regression kontinuierliche Variable




```
from sklearn import neighbors
knn = neighbors.KNeighborsRegressor(weights="distance")
knn.fit(X_train,y_train) # Modell trainieren
Y_pred_knn = knn.predict(X_test)
```

Klassifizierung diskrete Variable




```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier()
knn.fit(X_train,y_train) # Modell trainieren
Y_pred_knn = knn.predict(X_test)
```

Vorteile:

- Einfach verständlich.
- Einsetzbar für Regression und Klassifikation.
- Ohne Labels und ohne Training einsetzbar.
- Nur 1 Hyperparameter.

Nachteile:

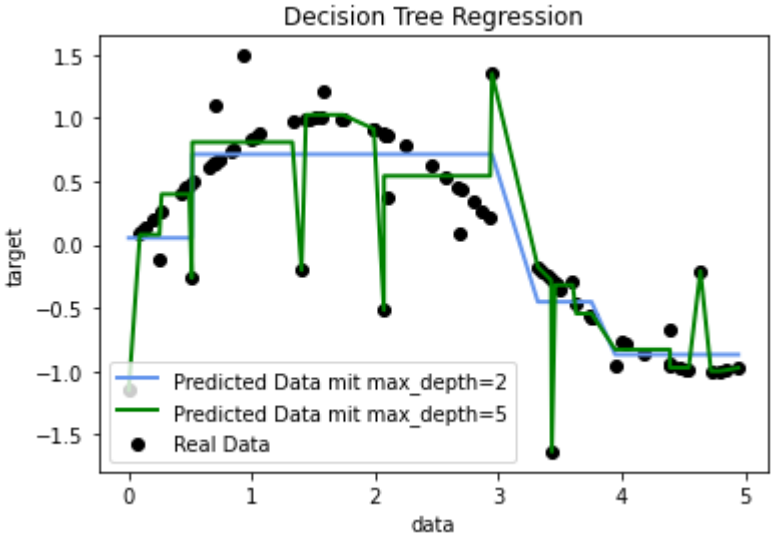
- Lernt nicht aus Trainingsdaten.
- Hoher Speicherbedarf (ganzer Datensatz wird in Speicher abgelegt).
- Ausreißer/ fehlende Daten haben großen Einfluß auf Algorithmus.
- Probleme mit hochdim. Daten.
- Selten eingesetzt für Regression.

Klassifikation eines Punktes erfolgt anhand Mehrheitsvotum seiner k-Nachbarn.
Regression erfolgt anhand Bilden des gewichteten Durchschnitt der Zielwerte der k-Nachbarn.

Quellen: SK-Learn Dokumentation: Link, Link
KNN: Fix, E., Hodges, J.L. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, 1951.

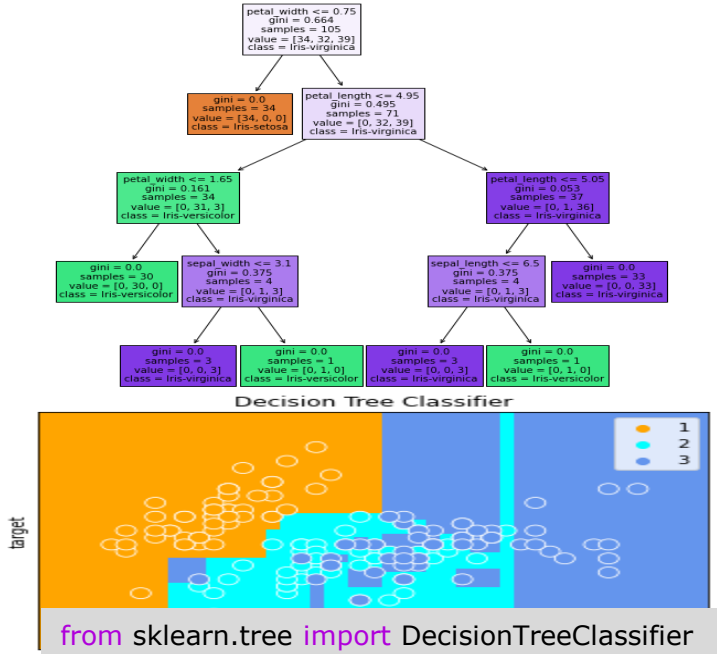
SUPERVISED LEARNING: DECISION TREE.

Regression kontinuierliche Variable




```
from sklearn.tree import DecisionTreeRegressor
TreeRegr = DecisionTreeRegressor(max_depth=5)
TreeRegr.fit(X_train,y_train) # Modell trainieren
Y_pred_Tree = TreeRegr.predict(X_test)
```

Klassifizierung diskrete Variable




```
from sklearn.tree import DecisionTreeClassifier
decisionTree = DecisionTreeClassifier()
decisionTree.fit(X_train,y_train) # Modell trainieren
 Y_pred_Tree = decisionTree.predict(X_test)
```

Vorteile:

- Einfach verständliche Ergebnisse.
- Einsetzbar für Regression und Klassifikation.
- Weniger Aufwand Datenaufbereitung (keine Skalierung/ Normierung).
- Robust bei fehlenden Daten.
- Robuster auch bei vieldimensionalen Features.

Nachteile:

- Nicht stabil gegenüber Updates (dann muß erneut trainiert werden).
- Schlechte Übertragbarkeit auf ähnliche Daten.
  Höherer Zeit-/ Ressourcenbedarf für Training.

Klassifikation: wiederholtes Aufteilen Datenset in zwei Untermengen anhand des Features mit höchstem Informationsgehalt.
Regression: erfolgt durch Bestimmen des Durchschnitts über die y-Werte des finalen Endknoten.

Quellen: SK-Learn Dokumentation: Link
Breiman, Friedman, Olshen, Stone: Classification and regression trees, 1984.

SUPERVISED LEARNING: ENSEMBLE LEARNING

Ziel: Verbesserte Performance durch die Kombination einzelner Lernverfahren (und Ausgleichen deren Schwächen).

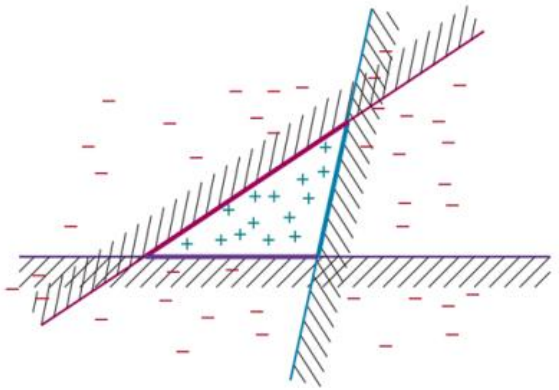
Ermöglicht:

- Höherere Genauigkeit (Reduktion Streuung und Verzerrung).
- Mehr Anwendungsfelder können abgedeckt werden.

Aber: Höherer Ressourcenbedarf für Training.

Häufigst eingesetzte Verfahren:

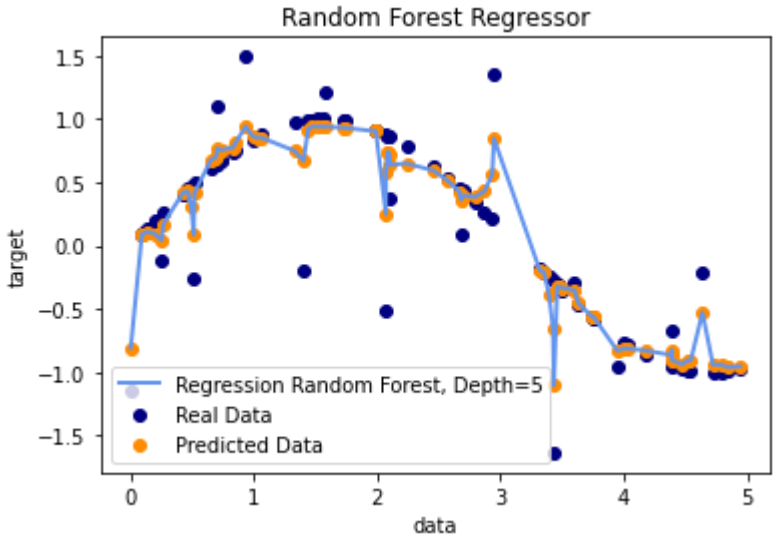
- Bagging: Kombination verschiedener Lernverfahren und **gleiche** Gewichtung (bspw. Durchschnitt) → Random Forest.
- Boosting: Kombination "schwacher" Lernverfahren per **performance-abhängiger Gewichtung** → XGBoost.



Quelle: Russell, Norvig: Artificial Intelligence: a modern approach, Kapitel 19.8

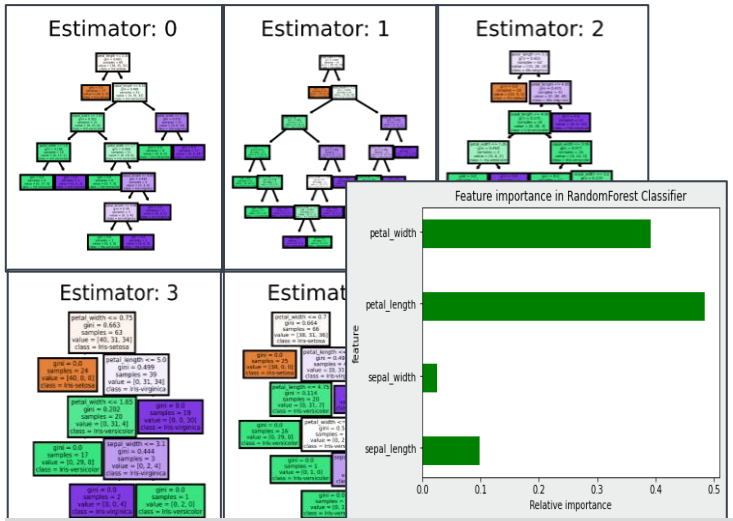
SUPERVISED LEARNING: RANDOM FOREST.

Regression kontinuierliche Variable




```
from sklearn.ensemble import RandomForestRegressor
RFRegressor = RandomForestRegressor ()
RFRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_RFR = RFRegressor.predict(X_test)
```

Für diskrete, mehrere Klassen




```
from sklearn.ensemble import RandomForestClassifier
RandomForest_clf = RandomForestClassifier()
RandomForest.fit(X_train,y_train) # Modell trainieren
Y_pred_RFC = RandomForest.predict(X_test)
```

Vorteile:

- Parallelisierbarkeit Training.
- Feature Importance: Algorithmus liefert Aussage, was die wichtigsten Features für die Vorhersage sind.
- Einsetzbar für hochdimensionale Daten.
- Höhere Genauigkeit als Entscheidungs-bäume (Reduziert Overfitting durch Gewichtung mehrerer Entscheidungsb.).

Nachteile:

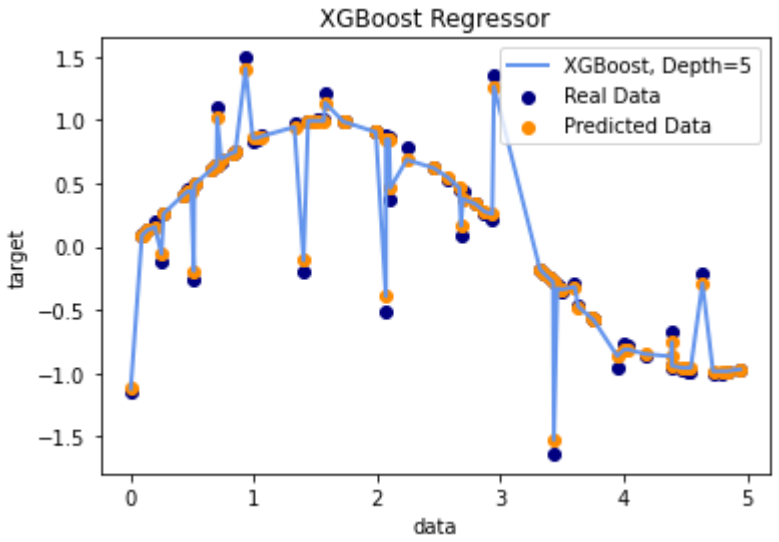
- Komplexer als Entscheidungsbäume.
- Nicht mehr so leicht interpretierbar.
- Schlechtere Performance bei Regression.
- Schlechtere Laufzeit-Performance.

Einsatz verschiedener Entscheidungsbäume und danach Kombination der Entscheidungsbäum.
Dabei wird jeder Entscheidungsbaum gleich gewichtet, d. h. hat gleichen Einfluß auf das Ergebnis.

Quellen: Ho: "Random Decision Forests", 1995.
Breiman, L: "Random Forests". Machine Learning. 45 (1): 5–32, 2001.
SK-Learn Dokumentation: Link

SUPERVISED LEARNING: XGBOOST.

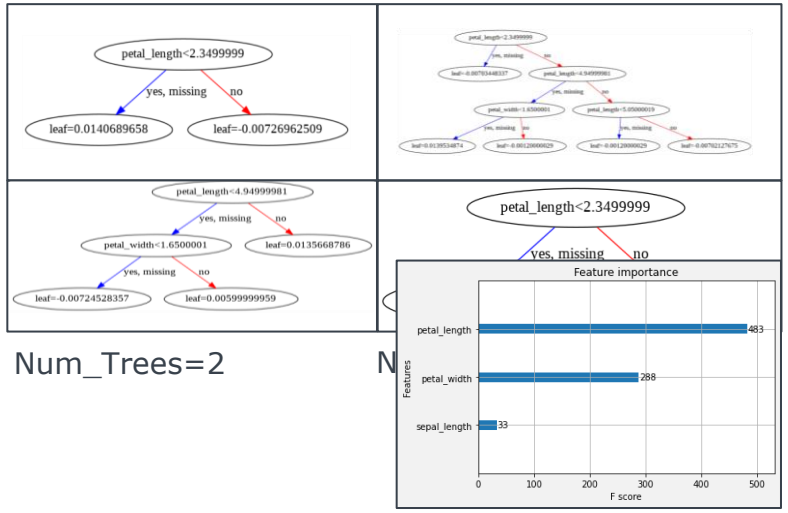
Regression kontinuierliche Variable




```
import xgboost as xgb
XGB_Regressor = xgboost.XGBRegressor()
XGB_Regressor.fit(X_train,y_train) # Modell trainieren
Y_pred_XGBR = XGB_Regressor.predict(X_test)
```

Für diskrete, mehrere Klassen

Num_Trees=0 Num_Trees=1



Num_Trees=2 N

```
import xgboost as xgb
xgb_clf = xgb.XGBClassifier()
xgb_clf.fit(X_train,y_train) # Modell trainieren
Y_pred_XGB_CLF = xgb_clf.predict(X_test)
```

Vorteile:

- Hervorragende Performance für strukturierte Daten.
- Sehr hohe Genauigkeit.
- Parallelisierbarkeit Training.
- Feature Importance (was sind die wichtigsten Features für Vorhersage?).

Nachteile:

- Erklärbarkeit Algorithmus.
- Arbeitet nur mit numerischen Werten (Kategorische Variablen müssen in 1-dimensionale Features je Wert umgewandelt werden).
- Hyperparameter-Tuning notwendig für Vermeiden Overfitting.

Einsatz mehrerer Modelle mit Ziel, durch Hinzunahme zusätzlicher Modelle die Fehlklassifikationen der bisherigen Modelle zu reduzieren (Start mit Modell 1, dann trainiere Modell 2 auf Fehler von Modell 1, …).

Quellen: Chen, Guestrin: „XGBoost: A Scalable Tree Boosting System", 2016.