HERAUSFORDERUNGEN

Joern Ploennigs · AI4SC

Fortschritt

Grundlagen

- Programmierung
- Lineare Algebra
- Statistik
- Data cleaning

Supervised Learning

- Regression
- Klassifikation

Unsupervised Learning

- Clustering
- Dimensionsreduktion

Neuronal & Reinforcement Learning

- Neuronale Netzwerke
- Reinforcement Learning

Herausforderungen

- Skallierbarkeit
- Datenschutz
- Fairness
- Ethik

Evaluation

Füllt bitte die Evaluation aus und helft die Vorlesung zu verbessern

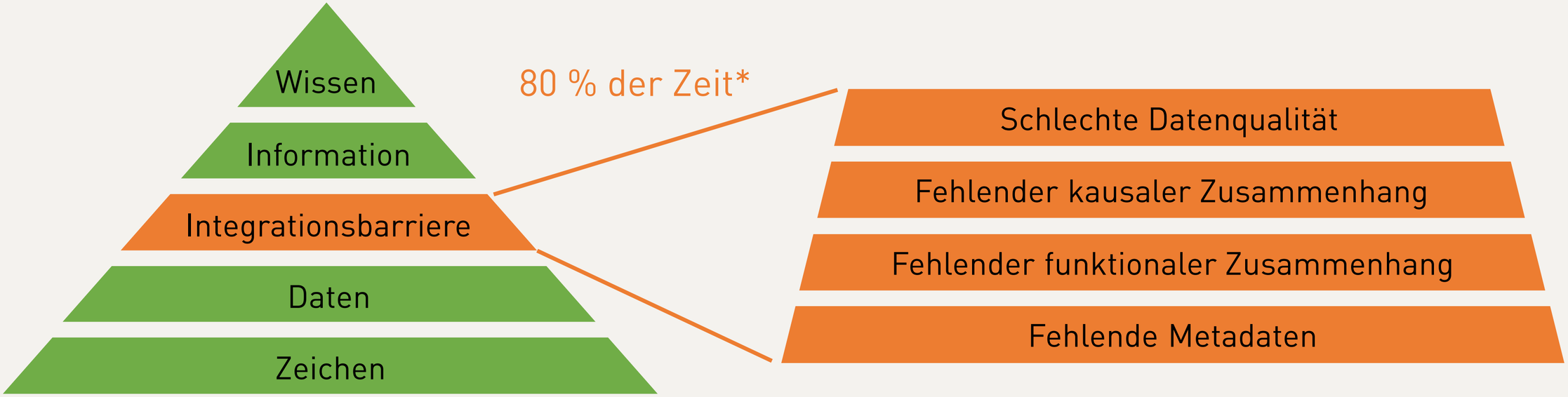
Hörsaalfrage

Hörsaalfrage

Die Herausforderung von ML

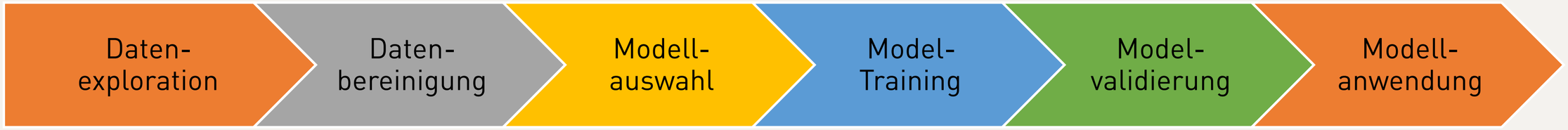
Datenaufbereitung ist der zeitaufwändigste (80%) und wenig ~~herausforderndste~~ *langweiligste* (76%) Schritt bei der Datenanalyse.[1]

Die digitale Transformation besteht zu 80 % aus Menschen, zu 20 % aus Technologie: Domänenwissen ist von entscheidender Bedeutung, und daher der Bedarf an Experten, die das Geschäft verstehen und über das notwendige Wissen verfügen, um die richtigen Fragen zu stellen und Antworten zu kontextualisieren.[2]



01 Forbes: Cleaning Big Data (2016)

Vorgehensweise im Maschinellem Lernen



Typische Vorgehensweise beim Maschinellem Lernen

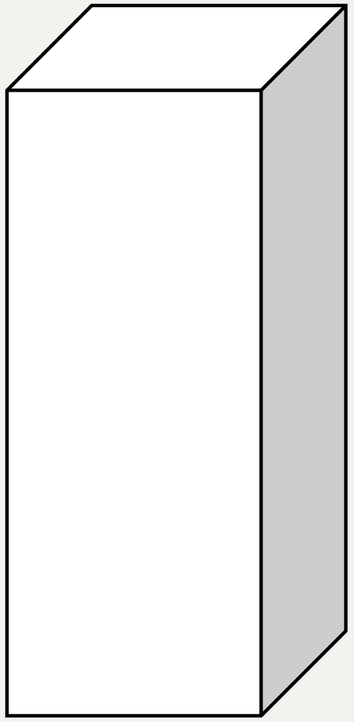
Die Datenanalyse in der Praxis geschieht meist nicht sequentiell, sondern man iteriert meist agil zu der richtigen Lösung. Das umfasst mehrmaliges Bereinigen der Daten, Überdenken der Modellauswahl und mehrmaliges Modell-Training und Modellvalidierung bevor das richtige Modell gefunden wird.

Beispielhafter Ablauf in relativer Zeit mit den Farben für die Schritte oben

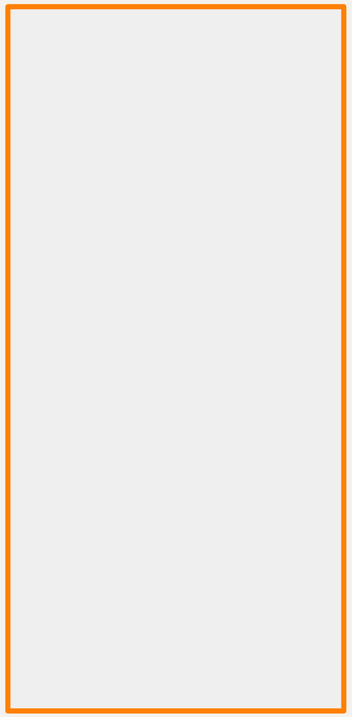
SKALIERBARKEIT

Skalierbarkeitsprobleme

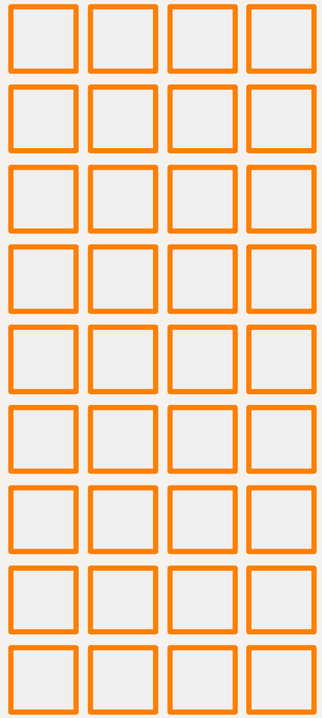
Big Data: Es werden mehr Daten erzeugt als gespeichert kann (z.B. Telekommunikation), bzw. bewegt werden kann (z.B. Satelliten).



Big Model: Modelle (z.B. LLM) sind zu groß, um sie zu bewegen bzw. die Berechnung zu aufwändig für einzelne Rechner.



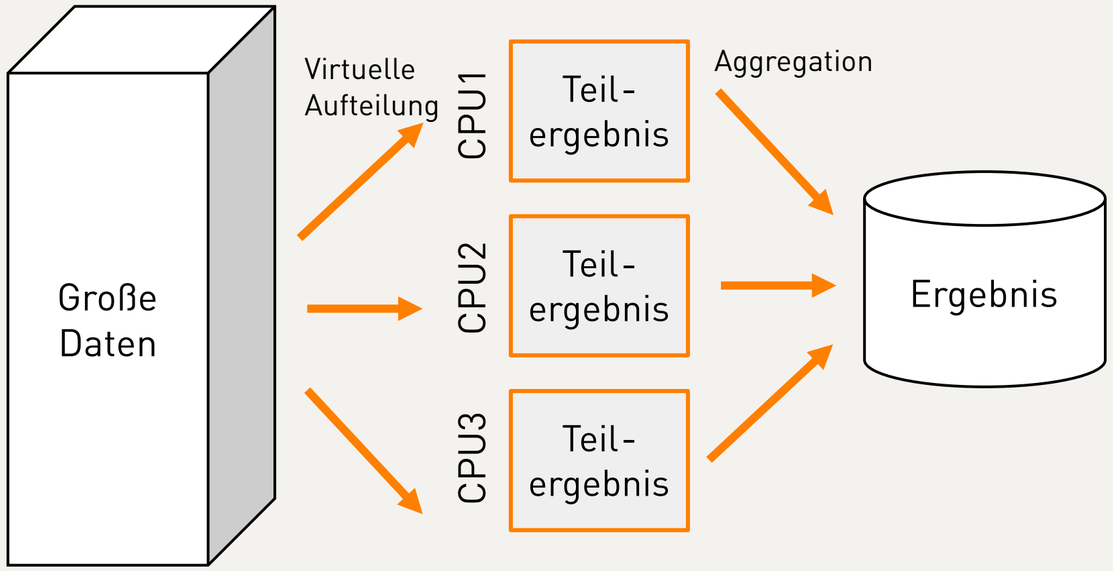
Many Models: Es werden sehr viele spezifische Modelle erzeugt (z.B. Energievorhersage, Anomalierkennung).



Partitionierung

Bei der *Partitionierung* wird eine Berechnung in kleinere, überschaubare Teilberechnungen zerlegt, die unabhängig voneinander (parallel) auf unterschiedlichen Teilen des Datensatzes berechnet werden. Dies ist besonders nützlich für Algorithmen die in Teilen berechnet und dann Aggregiert werden können (Statistiken, Lineare Regression, k-Means, Random Forest, SVM, PCA, KNN).

Wichtige Frameworks zur Parallelisierung sind dask und Spark.

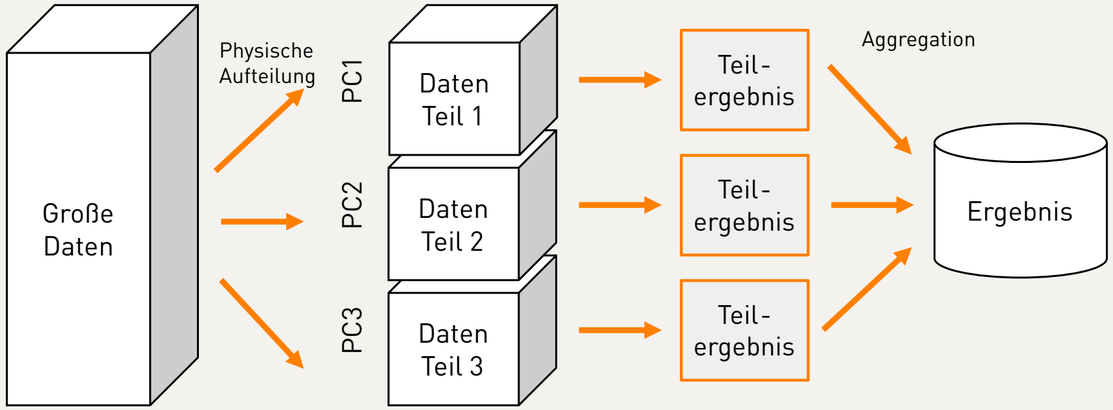



```python
import dask.dataframe as dd

# Erstellen eines Dask DataFrames aus CSV
df = dd.read_csv('large_dataset.csv')

# Verarbeiten des DataFrames in Partitionen
result = df.groupby('column').mean().compute()
print(result)
```

Verteiltes Rechnen grosser Modelle

Verteiltes Rechnen eines großen Modells ist eine Erweiterung der Partitionierung, bei der die Ausführung auf mehreren Rechenknoten (z.B. Cloud-PCs) verteilt werden, um die Effizienz und Geschwindigkeit zu erhöhen. Das bedeutet auch dass die Daten mit der Berechnung eines großen Modells auf die Rechenknoten verteilt werden müssen, was wiederum Performance-Nachteile mit sich bringen kann.

Typische Frameworks sind Hadoop und Spark die den MapReduce-Algorithmus [Dean & Ghemawat, 2008] nutzen.

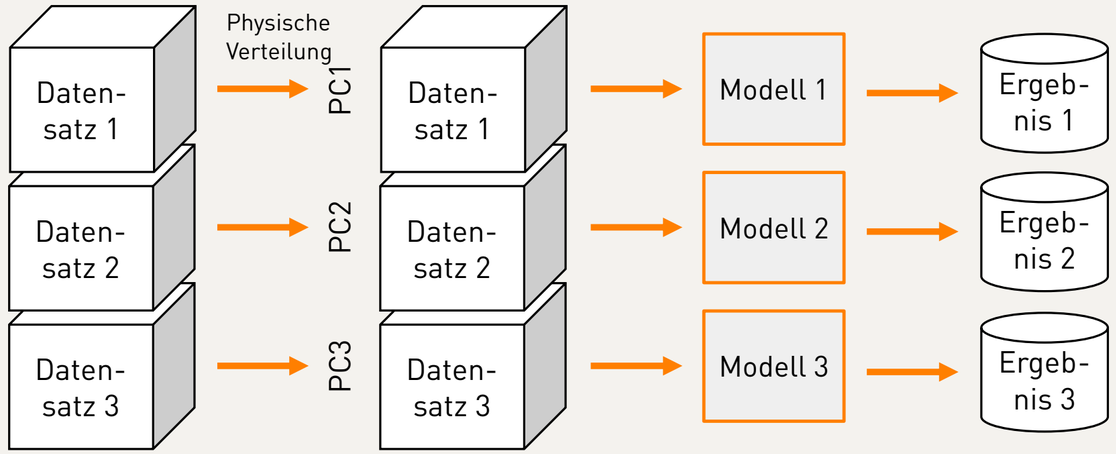



```python
from pyspark.sql import SparkSession

# SparkSession erstellen
spark = SparkSession.builder.getOrCreate()

# DataFrame laden
df = spark.read.csv('large_dataset.csv')

# Verarbeiten des DataFrames
result = df.groupBy('column').mean().collect()
print(result)
```

Verteiltes Rechnen vieler Modelle

Beim *verteilten Rechnen vieler Modelle* werden mehrere Modelle parallel trainiert. Diese nutzen meist unterschiedliche, z.T. überschneidende Datensätze. Die Aufgabe besteht darin, die Daten und die Berechnungen über mehrere Rechenknoten in der Cloud zu verteilen. Zur Lastverteilung werden meist Joblisten genutzt.

Frameworks zum verteilten Rechnen sind Ray für Python Modelle, was wir bereits beim Reinforcement Learning genutzt haben, oder Airflow.

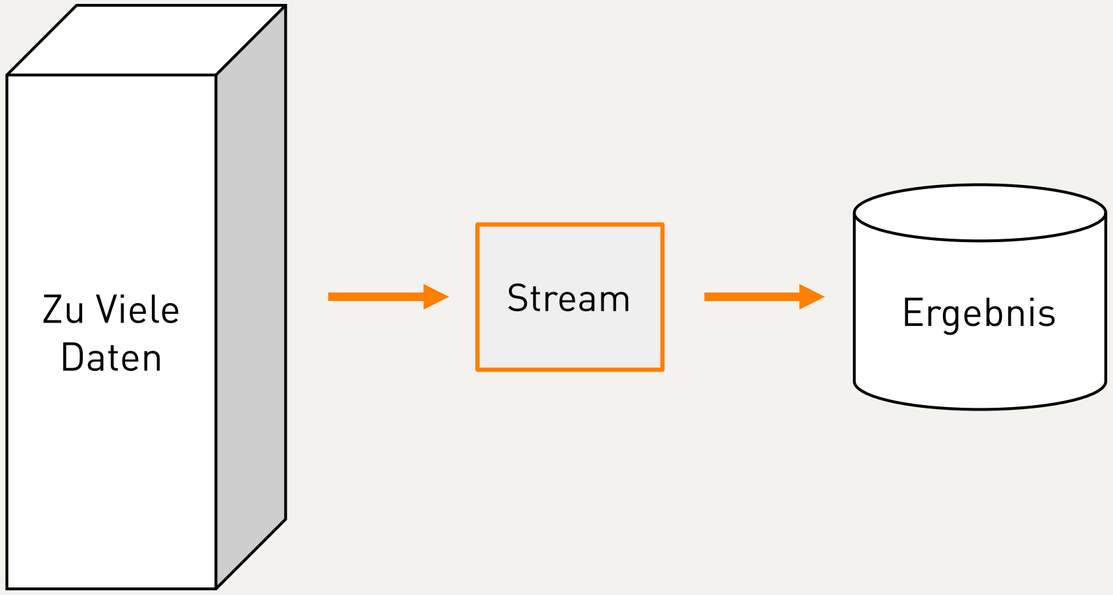



```python
import ray

# Einfache parallele Funktion
@ray.remote
def sum(a, b):
    return a + b

# Parallel mehrere Instanzen der Funktion ausführen
ray.init()
res = [sum.remote(i, i + 1) for i in range(10)]

# Ergebnisse sammeln und anzeigen
res = ray.get(res)
print("Ergebnisse:", res)

# Ray beenden
ray.shutdown()
```

Stream-Processing

Sind die Daten zu groß zum Speichern, nutzt man meist *Stream-Processing*. Dabei werden die Datenströme in Echtzeit kontinuierlich verarbeitet und nur die Ergebnisse und Modelle gespeichert. Werden die Modelle dabei kontinuierlich aktualisiert, so spricht man auch vom `Online Learning`. Das eignet sich besonders zur Datenaggregation (Statistiken), Mustererkennung (Anomalien), oder Vorhersagen mit hohen Echtzeitanforderungen (Short-Term-Trading).

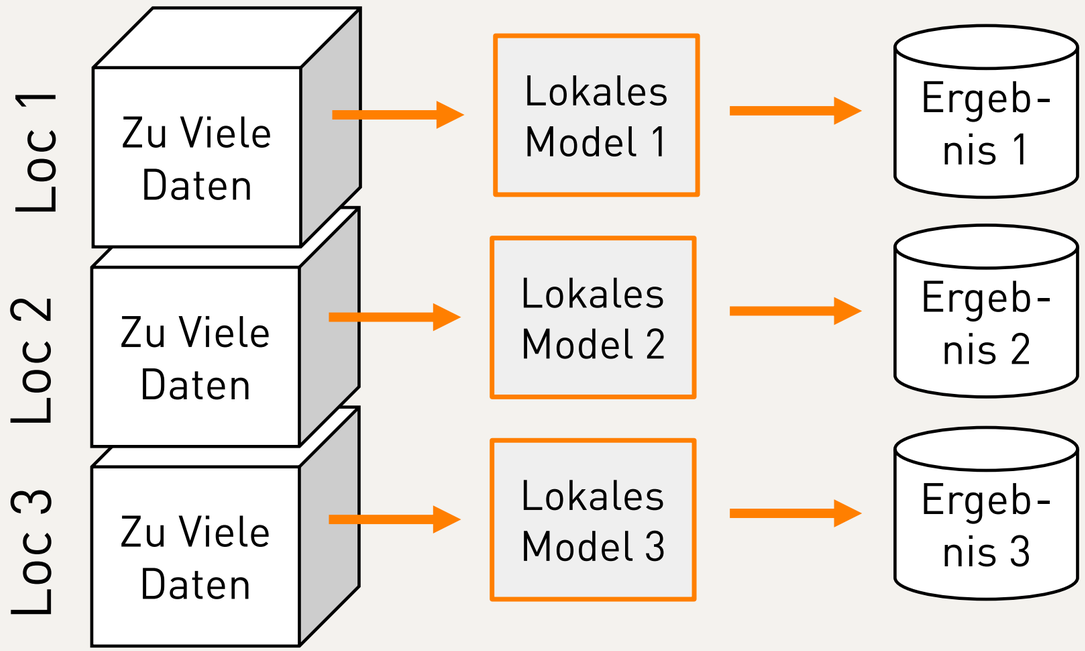
Ein typisches Framework für Stream-Processing ist Kafka oder Flink.




```python
from kafka import KafkaConsumer
import json

# Kafka-Consumer erstellen
consumer = KafkaConsumer('test-topic')

# Nachrichten vom Topic lesen und verarbeiten
for message in consumer:
    # Verarbeitung der empfangenen Nachrichten
    print(f"Received message: {message.value}")

# Kafka-Consumer schließen
consumer.close()
```

Edge Computing

Beim *Edge Computing* werden die Daten bereits dort verarbeitet, wo sie entstehen, also direkt auf den lokalen Geräten. Dadurch hat man schnellere Ergebnisse und höheren Datenschutz. Das eignet sich insbesondere für Probleme, bei denen nur lokale Modelle (z.B. Energievorhersage, Anomalieerkennung, etc.) benötigt werden und globale Modelle vereinfacht nur angewendet werden, z.B. mit Tensorflow Lite.

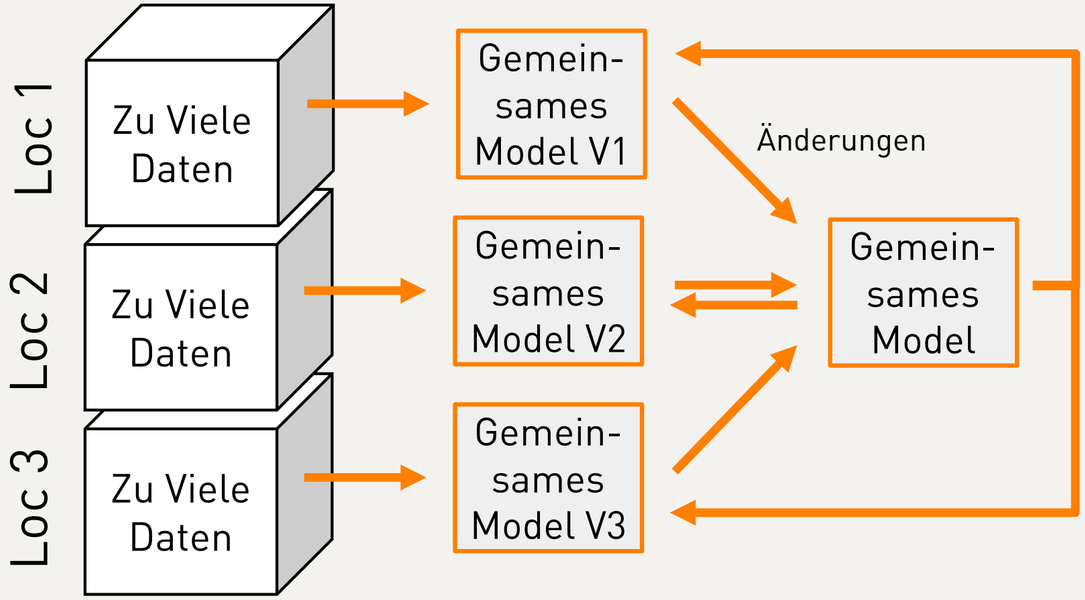



```python
import tensorflow as tf

# Modell für Edge Computing konvertieren
# converter = tf.lite.TFLiteConverter.from_keras_model(model)
# tflite_model = converter.convert()
# with open('model.tflite', 'wb') as f: f.write(tflite_model)

# TFLite-Modell auf Edge-Geräten ausführen
interpreter = tf.lite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

# Input und Output Tensor definieren
input_tensor = interpreter.tensor(interpreter.get_input_details()[0]['index'])
output_tensor = interpreter.tensor(interpreter.get_output_details()[0]['index'])

# Beispiel-Inferenz auf Edge-Gerät durchführen
input_tensor()[0] = input_data
interpreter.invoke()
print(output_tensor()[0])
```

Federated Learning

Federated Learning erweitert das Edge Computing zum Training von gemeinsamen Modellen über verteilte Geräte, ohne dass die Daten zentralisiert werden müssen. Dabei wird lokal ein gemeinsames Modell weiter trainiert und Änderungen synchronisiert, so dass das gemeinsame Modell sich verbessert.

Hier können Frameworks wie PySyft oder Tensorflow Federated genutzt werden.




```python
import tensorflow as tf
import tensorflow_federated.learning as tffl

# Federated Learning-Modell definieren
def create_federated_model():
    return tf.keras.models.Sequential([
        tf.keras.layers.Dense(10),
        tf.keras.layers.Dense(1)
    ])

# TensorFlow Federated Konfiguration erstellen
fed_avg = tffl.build_federated_averaging_process(
    model_fn=create_federated_model)

# Modell trainieren
state = fed_avg.initialize()
for round_num in range(10):
    state, metrics = fed_avg.next(state, client_data)
    print('Round {}: loss={}'.format(round_num, metrics.loss))
```

DATENSCHUTZ


© Midjourney: Agent Smith protecting a database

Allgemeines Persönlichkeitsrecht

- Das *Allgemeine Persönlichkeitsrecht* (APR) leiten sich nach der Rechtsprechung des BVerfG aus der freien Entfaltung der Persönlichkeit (Art. 2 Abs. 1 GG) und der Menschenwürde (Art. 1 Abs. 1 GG) ab.
- *Recht auf Selbstbestimmung*: Dies beinhaltet unter anderem die Kontrolle über die eigenen Daten, die Entscheidungsfreiheit über die eigene Lebensgestaltung und das Recht auf informationelle Selbstbestimmung.
- *Recht auf Selbstbewahrung*: Hierzu zählen der Schutz der körperlichen und geistigen Integrität, der Schutz der Privatsphäre und das Recht auf informationelle Selbstbestimmung.
- *Recht auf Selbstdarstellung*: Dies umfasst das Recht auf freie Meinungsäußerung, das Recht am eigenen Bild und das Recht auf Privatsphäre.

Personenbezogene Daten

Definition: Personenbezogene Daten

Personenbezogene Daten [sind] alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen.

- *Direkte Identifikatoren*: Name, Adresse, Telefonnummer, E-Mail-Adresse, Geburtsdatum, Geburtsort
- *Indirekte Identifikatoren*: Kennnummern, Finanzinformationen, IP-Adressen, Cookies
- *Sensible Daten*: Gesundheitsdaten, Biometrische Daten, Genetische Daten, Ethnische Herkunft, Politische Meinungen, Religiöse oder philosophische Überzeugungen, Gewerkschaftszugehörigkeit, Sexuelle Orientierung
- *Technische Daten*: Standortdaten, Online-Kennungen

Grundsätze des Datenschutz I

- *Rechtmäßigkeit*: Die Verarbeitung personenbezogener Daten muss rechtmäßig erfolgen und auf einer zulässigen Rechtsgrundlage basieren, z. B. Einwilligung, Vertragserfüllung oder berechtigtes Interesse.
- *Verarbeitung nach Treu und Glauben*: Die Verarbeitung muss in einer Weise erfolgen, die den Betroffenen gegenüber transparent ist.
- *Transparenz*: Der Verantwortliche muss in der Lage sein, die Einhaltung der Grundsätze nachzuweisen.
- *Zweckbindung*: Daten dürfen nur für festgelegte, eindeutige und legitime Zwecke erhoben und verarbeitet werden.
- *Datenminimierung*: Es sollen nur die Daten erhoben werden, die für den Zweck wirklich notwendig sind. Unnötige Daten müssen vermieden werden.

Grundsätze des Datenschutz II

- *Richtigkeit*: Personenbezogene Daten müssen sachlich richtig sein und erforderlichenfalls aktualisiert werden.
- *Speicherbegrenzung*: Daten sollen nur so lange gespeichert werden, wie es für den Zweck erforderlich ist. Daten müssen nach Ablauf der Speicherfrist gelöscht oder anonymisiert werden.
- *Integrität und Vertraulichkeit*: Daten müssen vor unbefugtem Zugriff, Verlust oder Zerstörung technisch und organisatorisch geschützt werden.
- *Rechenschaftspflicht*: Der Verantwortliche muss die Einhaltung der Grundsätze nachweisen können.

Anforderungen an den technischen Datenschutz

- *Verfügbarkeit*: Möglichkeit eines gesicherten Zugriff auf die Daten innerhalb festgelegter Zeit, um diese ordnungsgemäß zu verwenden.
- *Integrität*: Ein System erfüllt ausschließlich seine zweckbestimmte Funktion wobei personenbezogene Daten unversehrt, vollständig und aktuell bleiben.
- *Vertraulichkeit*: Nicht zuständige Dritte erhalten keine Möglichkeit, die Daten zu bekommen oder das System einzusehen.
- *Transparenz*: Es muss klar sein, welche Daten erhoben werden, wohin sie fließen, wie und von wem sie verarbeitet werden, wann sie gelöscht werden.
- *Intervenierbarkeit*: Der Betroffene muss die Erhebung und Kontrolle seiner Daten beeinflussen, abschalten und löschen können.
- *Nichtverkettbarkeit*: Unmöglichkeit der Verkettung von Daten und Entitäten untereinander und miteinander.

ANONYMISIERUNG


© Midjourney: Anonymisierung

Anonymisierung

Definition: Anonymisierung

Die Anonymisierung von Daten ist ein Verfahren, um personenbezogene Daten so zu verändern, dass die betroffene Person nicht mehr identifiziert werden kann. Es gibt verschiedene Methoden und Techniken zur Anonymisierung, die je nach Anwendungsfall und erforderlichem Schutzgrad eingesetzt werden.

Beispieldatensatz mit verschiedenen personenbezogenen Daten

```
# Beispieldaten erstellen
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Geburt': ['1990-01-01', '1985-05-23', '1979-07-14', '1992-12-30', '1988-03-09'],
    'Telefon': ['0123/4567890', '0234/5678901', '0345/6789012', '0456/7890123', '0567/8901234'],
    'Gehalt': [50000, 60000, 55000, 65000, 70000],
    'PLZ': ['10115', '10243', '10365', '10437', '10555']
})
print(df)
```

Verfahren I

Pseudonymisierung

Beschreibung: Daten werden durch ein Pseudonym ersetzt, das die direkte Identifikation der Person verhindert, aber erlaubt die ursprünglichen Daten wiederherzustellen.

Anwendung: Häufig verwendet, um Daten zu schützen, die ggf. wiederhergestellt werden sollen.

Beispiel: Ersetzung eines Namens durch eine Kennnummer.

```
# Namen durch Pseudonyme ersetzen
df['Pseudonym'] = ['P1', 'P2', 'P3', 'P4', 'P5']
df = df.drop(columns=['Name'])
print(df)
```

Maskierung

Beschreibung: Ersetzen von Daten mit Platzhaltern oder zufälligen Werten, um die Identifikation der betroffenen Person zu verhindern.

Anwendung: Oft verwendet in Entwicklungs- und Testumgebungen, um den Zugriff auf echte personenbezogene Daten zu vermeiden.

Beispiel: Umwandlung von Telefonnummern in “XXXX/XXX4534”.

```
# Telefonnummern maskieren
df['Telefon'] = df['Telefon'].apply(
    lambda x: 'XXXX/XXX' + x.split('/')[1][-4:])
print(df)
```

Verfahren II

Aggregation

Beschreibung: Daten werden nur in zusammengefasster Form dargestellt, sodass nur noch statistische Infos verfügbar sind.

Anwendung: Häufig in Berichten und statistischen Analysen.

Beispiel: Anstatt individuelle Gehälter anzugeben, wird der Durchschnitt genutzt.

```
# Gehälter aggregieren (z.B. Durchschnittsgehalt)
avg_salary = df['Gehalt'].mean()
print("Durchschnittsgehalt:", avg_salary)
```

Datenunterdrückung

Beschreibung: Datenfelder oder Datensätze werden vollständig entfernt, um die Identifikation der betroffenen Person zu verhindern.

Anwendung: Bei besonders sensiblen Datenfeldern, die nicht für den Analysezweck erforderlich sind.

Beispiel: Entfernen von Geburtsdaten aus einem Datensatz.

```
# Geburtsdatum entfernen
df = df.drop(columns=['Geburt'])
print(df)
```

Verfahren III

Datenverschiebung

Beschreibung: Datenverschiebung vertauscht Datenfelder zufällig, sodass diese nicht mehr den ursprünglichen Personen zugeordnet werden können.

Anwendung: Erlaubt die statistische Analyse zu ermöglichen, ohne dass Datensätze zurückverfolgt werden.

Beispiel: Vertauschen der Postleitzahlen zwischen verschiedenen Datensätzen.

```
# Verschieben der Postleitzahlen
df['PLZ'] = np.random.permutation(df['PLZ'])
print(df)
```

Generalisation

Beschreibung: Die Präzision der Daten wird durch die Verallgemeinerung reduziert, sodass Individuen nicht mehr identifizierbar sind.

Anwendung: Bei der Veröffentlichung von Daten, um detaillierte Informationen zu verbergen.

Beispiel: Anstatt das Gehalt anzugeben, wird die Gehaltsgruppe verwendet.

```
# Generalisieren des Gehalts in Gehaltsgruppen
df['Gehalt'] = pd.cut(df['Gehalt'],
    bins=[0, 50000, 60000, 70000, 75000],
    labels=['0-50k','50k-60k','60k-70k','70k+'])
print(df)
```

Verfahren IV

Perturbation

Beschreibung: Daten werden durch zufällige Veränderungen modifiziert, um die Identifikation zu erschweren, ohne den Mittelwert zu ändern.

Anwendung: In Datenbanken und bei der Veröffentlichung von Mikrodaten.

Beispiel: Hinzufügen eines zufälligen Rauschens zu Gehaltsdaten.

```
# Zufälliges Rauschen zum Gehalt hinzufügen
noise = np.random.normal(0, 1000, df.shape[0])
df['Gehalt'] = df['Gehalt'] + noise
print(df)
```

Verschleierung

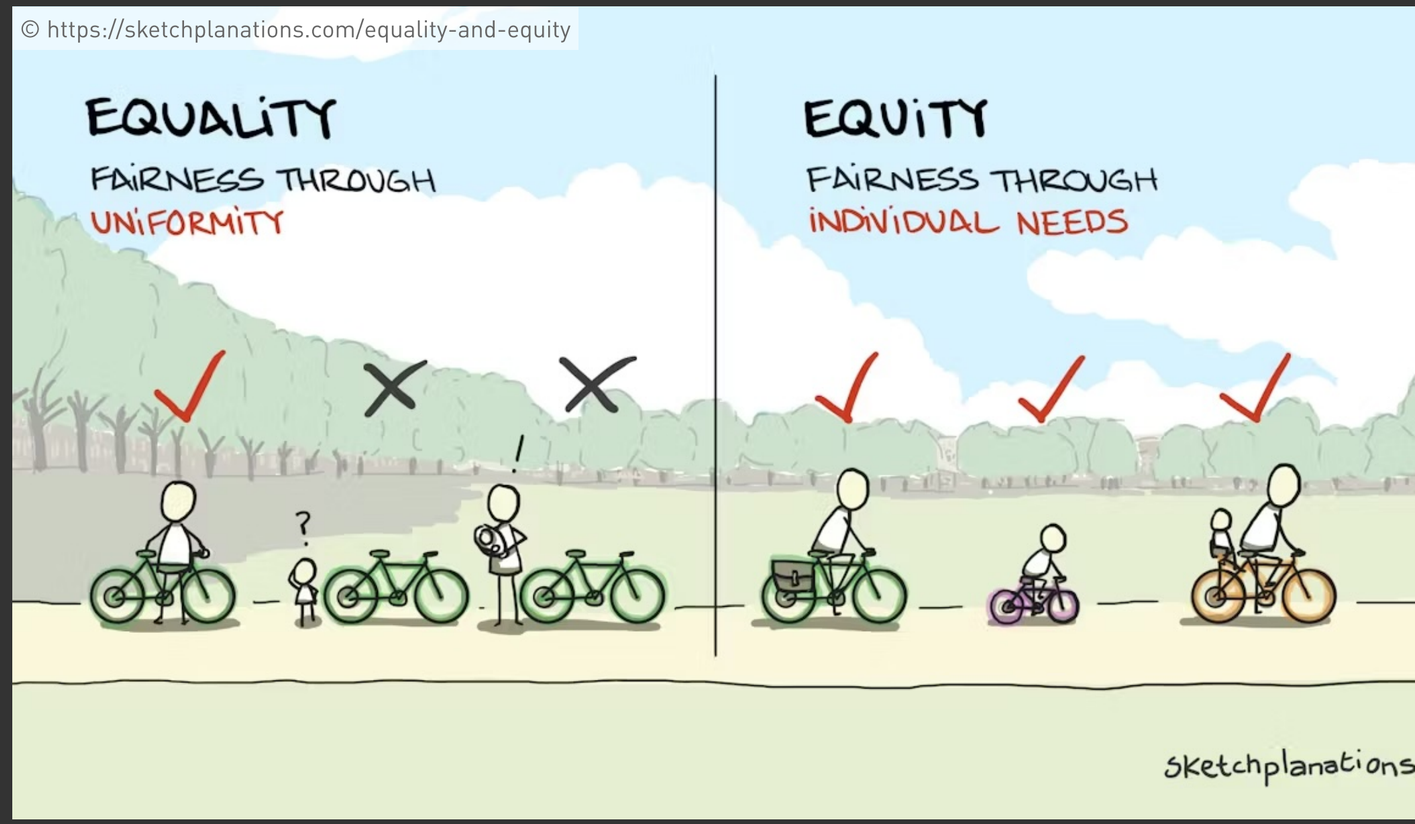
Beschreibung: Verschleierung (Obfuscation) fügt falsche oder ungenaue Daten in den Datensatz hinzu, um die Identifikation zu erschweren.

Anwendung: Schutz besonders sensibler Informationen.

Beispiel: Leichte Veränderung von GPS-Koordinaten.

```
# Leichte Veränderung der PLZ
df['PLZ'] = df['PLZ'].apply(lambda x: x[:2] +
    str(np.random.randint(10, size=3)))
print(df)
```

FAIRNESS


© https://sketchplanations.com/equality-and-equity
EQUALITY
FAIRNESS THROUGH
UNIFORMITY
EQUITY
FAIRNESS THROUGH
INDIVIDUAL NEEDS
?
!
sketchplanations

Algorithmic Fairness

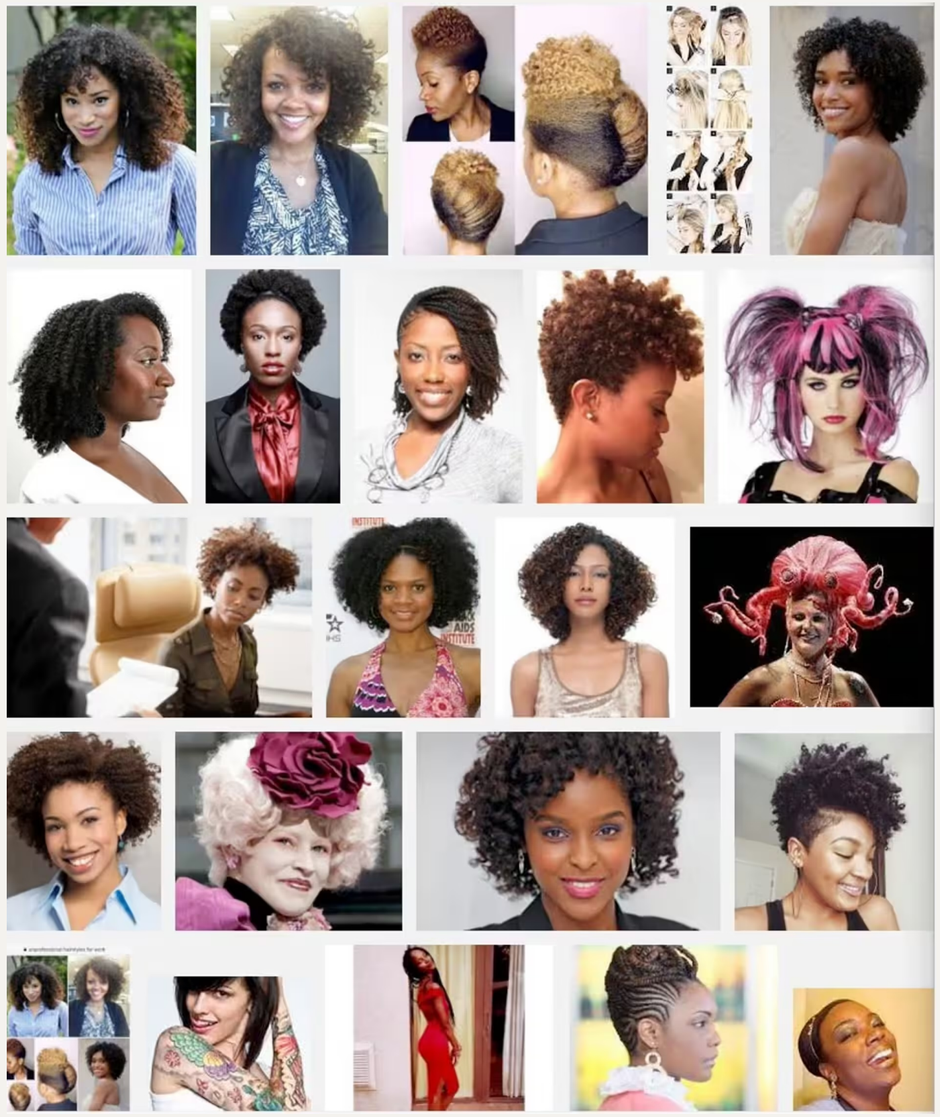
Definition: Algorithmic Fairness

Algorithmen, insbesondere ML Modelle, sollten keine Vorurteile abbilden und nicht bestimmte Gruppen privilegieren oder diskriminieren und somit anti-diskriminierung Gesetzen entsprechen.

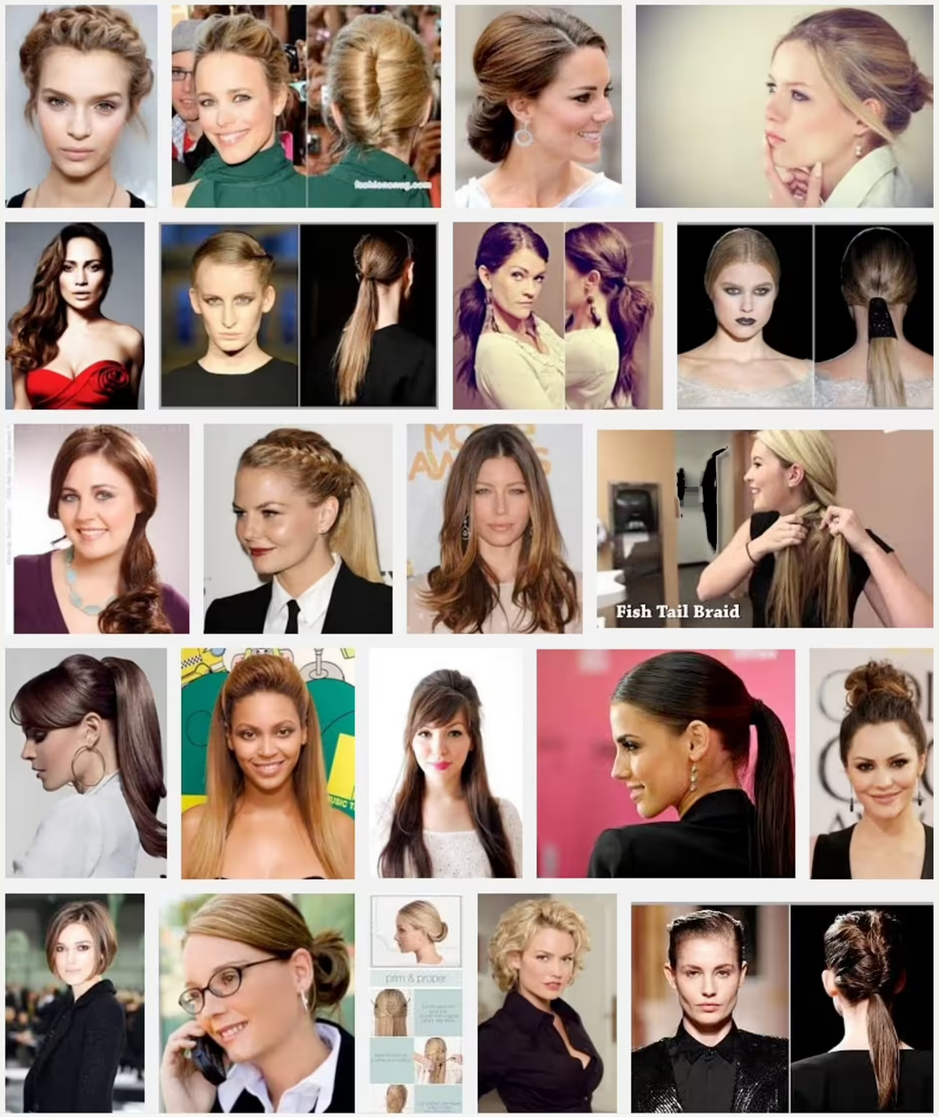
Insbesondere sensibel sind Merkmale bezüglich: - Geschlecht - Sexualität - Rasse - Alter - Religion - Wohnort - Behinderungen

Beispiel: Google image search – Bad vs Good Hair

Unprofessional Hair for Work



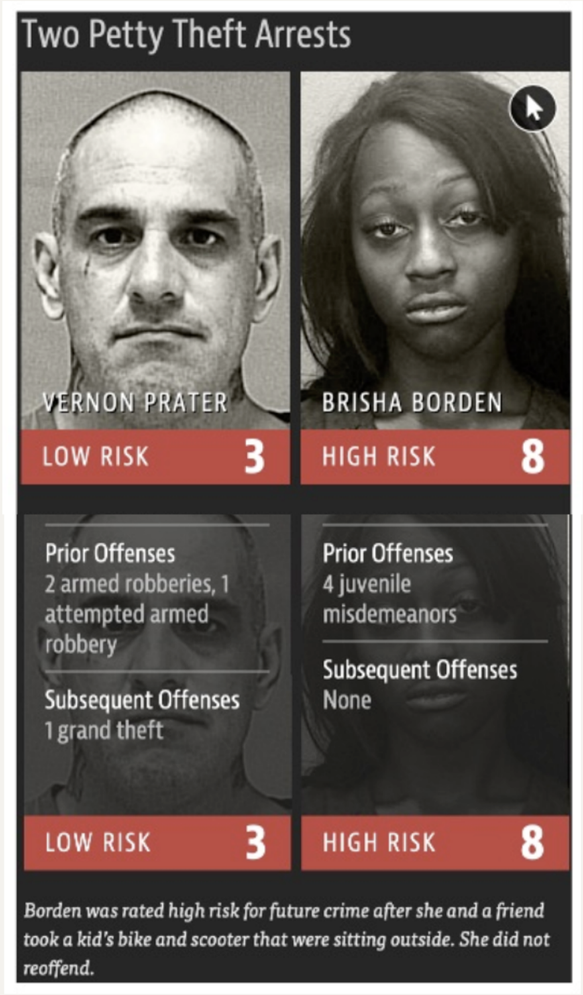
Professional Hair for Work



[Kay et al., 2015]

Beispiel: COMPAS - Rückfälligkeit von Straftätern

- COMPAS wird seit 2004 zur Bewertung der Rückfälligkeit von Straftätern genutzt um Strafhöhe und -dauer festzulegen
- 2016 wiesen KI-Forscher nach, dass die Risikobewertung für dunkelhäutige Menschen meist doppelt so hoch ist [Angwin et al., 2022]
- Algorithmus ist proprietär



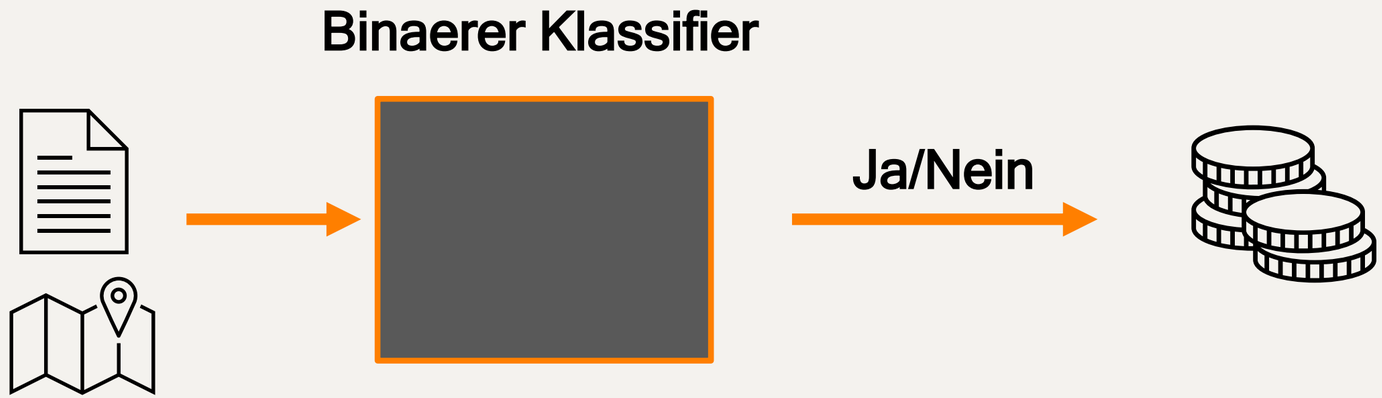
Beispiel: Schufa

- Schufa Score - Schätzwert für die Wahrscheinlichkeit, dass ein Kredit bedient wird
- Algorithmus ist proprietär
- 2018 formte sich OpenSchufa-Projekt welches durch Selbstauskunft versucht hat den Schufa Algorithmus zu testen
- Ergebnisse zeigen, dass Geschlecht, Alter und andere Faktoren den Score beeinflussen – z.B. werden junge Männer eher schlecht bewert, auch ohne negative Geschichte

Digitalisierung der Kreditwürdigkeit

Manueller Ansatz in der Vergangenheit

Digitaler Ansatz



1) Wie fair ist die Datengrundlage?

2) Wie fair ist das ML Modell ?

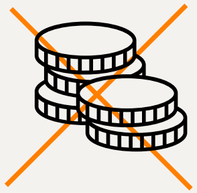
3) Wie fair sind die Ergebnisse?

Frage: Warum?
Antwort: nicht erklärbares black-box ML Model

Am Wohnort hatten 60% der Familien irgendwann ein ein Zahlungsproblem

Binaerer Klassifier

Nein



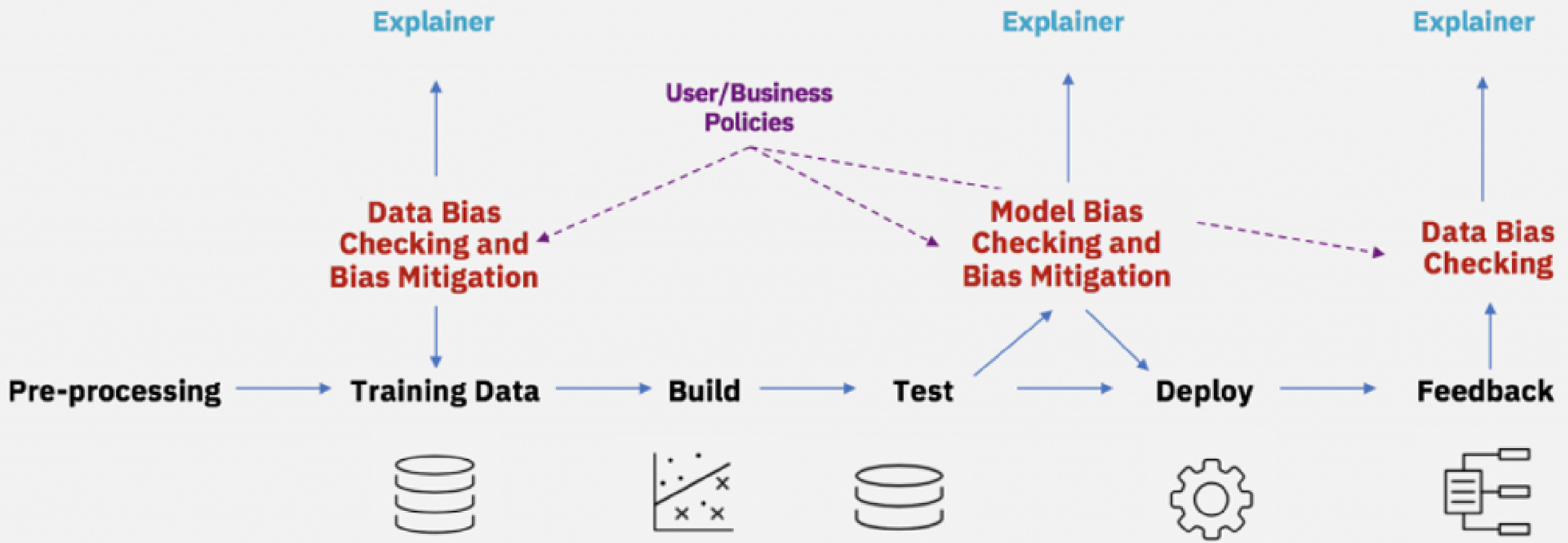
4) Ist der Prozess nachvollziehbar und erklärbar z.B. für Rückfragen?

Die drei Ansätze

Fairness der Daten Durch Vorverarbeitung kann man versuchen gezielt den Bias aus den Trainingsdaten zu entfernen.

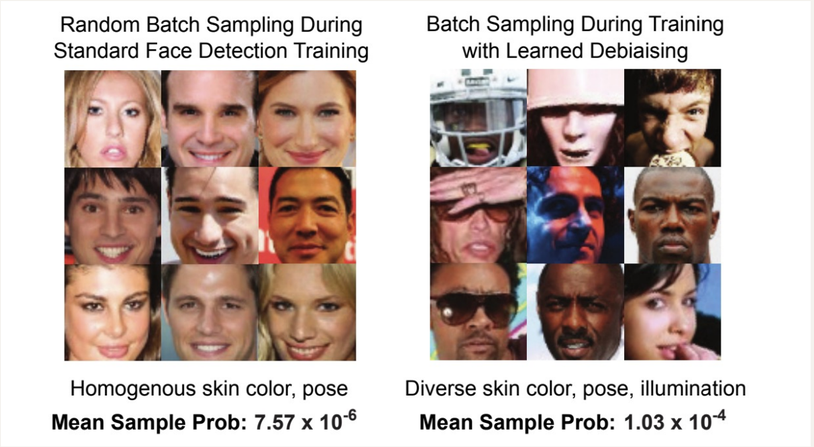
Fairness der ML Modelle Spezielle Modelle erlauben es die sensitiven Daten verstärkt zu ignorieren.

Fairness des Ergebnisse Bestimmte Tests und Algorithmen Bewerten die Fairness von Modellen und ggf. widerrufen Ergebnisse.



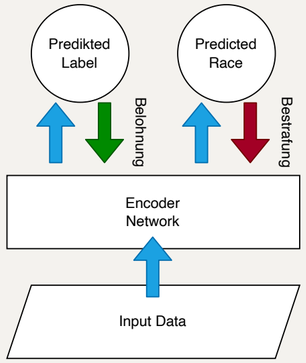
Fairness der Daten verbessern

- *Komplettes Entfernen* sensibler Merkmale wenn möglich. Oft sind Merkmale aber korreliert und schwer zu entfernen.
- Ausprägungen der Merkmale sollte *repräsentativ* sein und alle Varianten enthalten ohne Ausschluß oder Bevorzugung
- *Umgewichtung* von Varianten verhindert, dass Ausprägungen dominieren
- Beim *Sampling* von Traings-daten achtet man darauf, das immer verschiedene Varianten vertreten sind



Fairness der Modelle

- Im Training wird das Modell durch *Kostenfunktionen bestraft*, wenn es sensible Merkmale benutzt.
- Z.B. werden spezielle *Adversarial Debiasing* KNN Modelle eingesetzt, die gezielt darauf trainiert sind eine Vorhersage Y zu treffen ohne die sensiblen Merkmale Z zu lernen



Fairness der Ergebnisse

- Bestimmte *statistische Tests* werden verwendet, um die Fairness von Ergebnissen zu bewerten
- Modelle oder *Ergebnisse werden unterdrückt* wenn sie nicht fair sind (z.B. Ja/Nein/Unbekannt)
- Man nutzt z.B. *Testdatensätze*, die Randgruppen enthalten und so gegen Bias testen
- Datensätze und Modelle werden mit *Datenblätter* beschrieben die Eigenschaften und bekannte Probleme darlegen

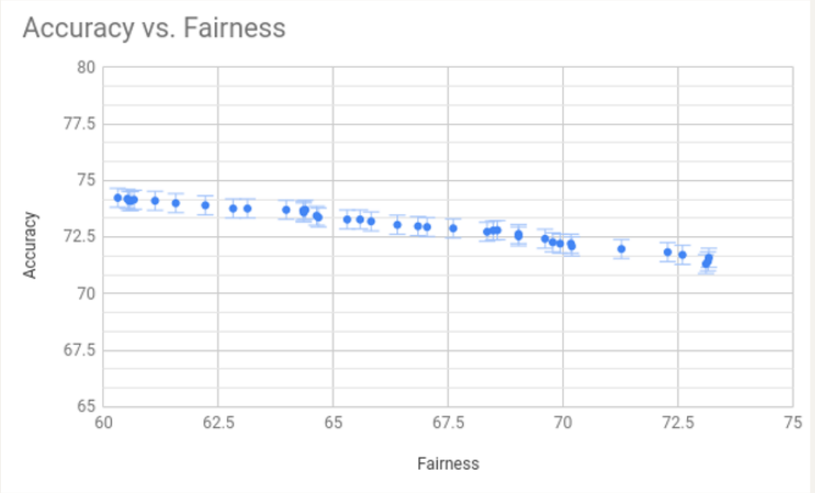
Das Ergebnis $\hat{y}$ des Modells ist unter der Nutzung von Testdatensätzen mit dominantem sensiblen Merkmal A (z.B. nur Männer) nicht signifikant anders als im Test gegen die Gesamtheit.

$$P(\hat{y} \mid A) = P(\hat{y})$$

Genauigkeit vs. Fairness

- Verbesserte Fairness führt normalerweise zu einer reduzierten Vorhersagegenauigkeit der Modelle
- Genauigkeit wird oft überbewertet!
- In vielen praktischen Anwendungen ist es sinnvoll eher etwas Genauigkeit zu verlieren und dafür auf Fairness, Robustheit und Angriffssicherheit der Modelle zu achten.

Beispiel: Ein verbessertes Adversarial Network Modell zum COMPAS Datensatz erlaubte die Fairness um 200% zu verbessern bei einem Verlust von ~6% Genauigkeit

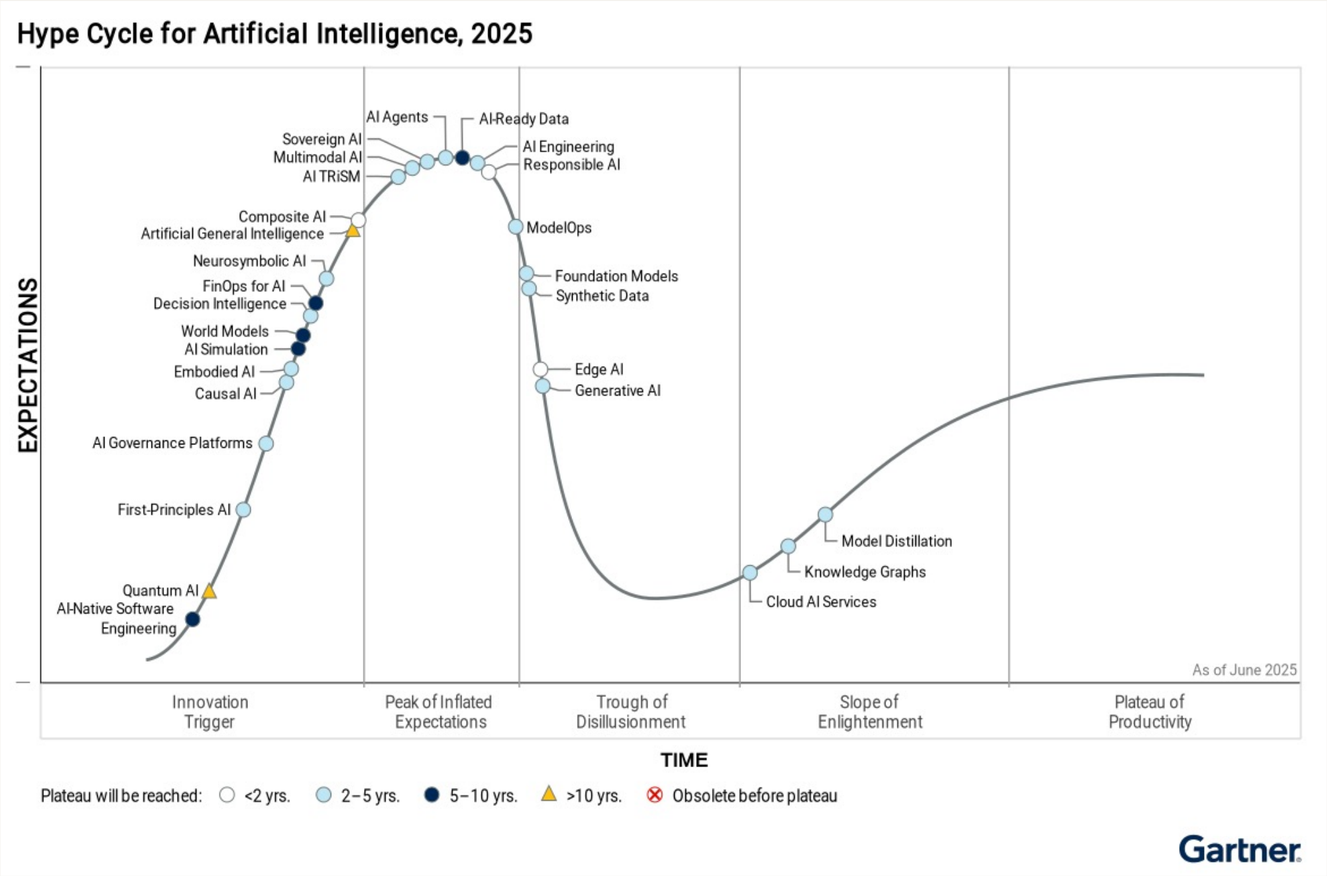


AUSBLICK


© Midjourney: A window into the future

AI Hype

- Generative AI
- Foundation Models
- AI Agents
- Multimodale AI
- Neurosymbolik AI
- Kausale AI



Gartner Hype Cycle KI 2025

Maschinelles Lernen und Künstliche Intelligenz

- Grundlagen Neuraler Netzwerke
- Einführung in Deep Neural Networks
- Architekturen Neuronaler Netzwerke (z.B. ANN, CNN, DNN, LSTM, Transformer, GNN)
- Einführung in Foundation Models

Zeit: Wintersemester 2024


© Midjourney: human with AR glasses controlling construction robots

Referenzen

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2022). Machine bias. In, *Ethics of data and analytics* (pp. 254–264). Auerbach Publications.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), 107–113.

Kay, M., Matuszek, C., & Munson, S. A. (2015). Unequal representation and gender stereotypes in image search results for occupations. *Proc. An. ACM Conf. on human factors in computing systems*.

QUESTIONS