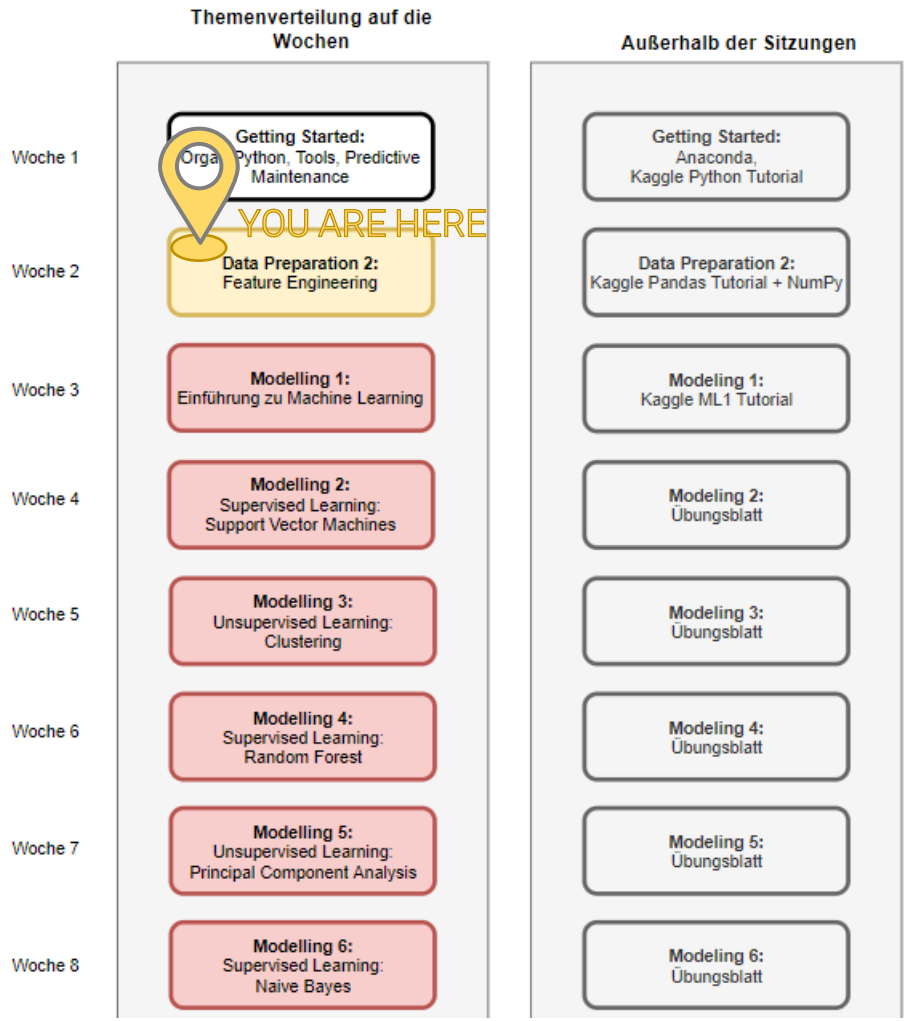
Data Preparation

Feature Engineering: Making data available to models

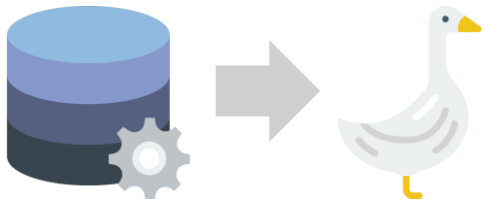
Wo sind wir?


Themenverteilung auf die Wochen
Außerhalb der Sitzungen
Woche 1
Woche 2
Woche 3
Woche 4
Woche 5
Woche 6
Woche 7
Woche 8
Getting Started: Orga, Python, Tools, Predictive Maintenance
YOU ARE HERE
Data Preparation 2: Feature Engineering
Modelling 1: Einführung zu Machine Learning
Modelling 2: Supervised Learning: Support Vector Machines
Modelling 3: Unsupervised Learning: Clustering
Modelling 4: Supervised Learning: Random Forest
Modelling 5: Unsupervised Learning: Principal Component Analysis
Modelling 6: Supervised Learning: Naive Bayes
Getting Started: Anaconda, Kaggle Python Tutorial
Data Preparation 2: Kaggle Pandas Tutorial + NumPy
Modeling 1: Kaggle ML1 Tutorial
Modeling 2: Übungsblatt
Modeling 3: Übungsblatt
Modeling 4: Übungsblatt
Modeling 5: Übungsblatt
Modeling 6: Übungsblatt

Agenda

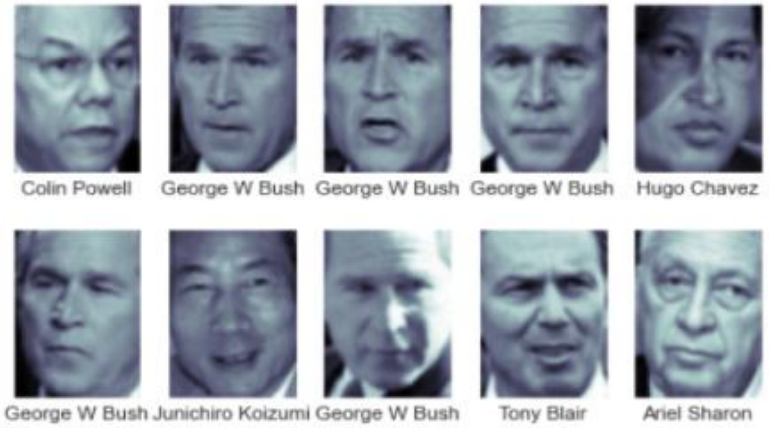
1. Was ist ein Feature? Vom Daten- in den Feature-Raum
2. Feature-Extraktion
3. Feature-Selektion
4. Feature Scaling und Normalisierung

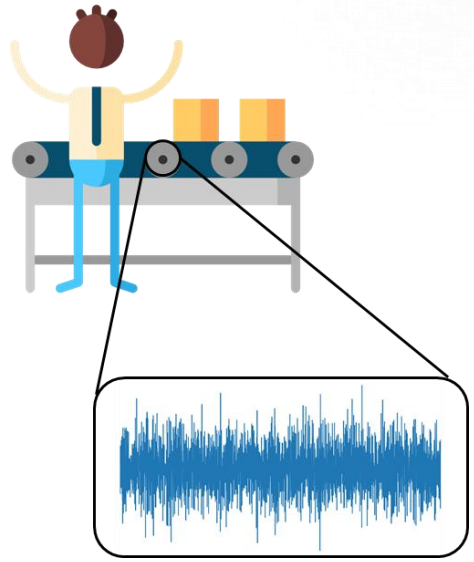
Von Daten zu Features



- Wenn wir in das Gebiet Machine Learning einsteigen, dann werden wir den Begriff „Feature“ sehr häufig hören
- Sie können sich schon jetzt merken: ein Machine Learning Modell nutzt **Features**, um etwas – die sog. **Labels** – vorherzusagen
- Sie werden jetzt dann sehen, dass Features im Grunde nichts anderes sind als **Arrays bzw. DataFrames**

Ihre Intuition: was sind hier „Features“?

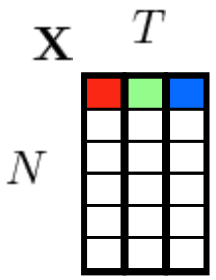




So what?

Features sind also (auch übersetzt) Merkmale – etwas, das zusammenfassend und charakterisierend bzgl. der zugrundeliegenden Daten ist

Was ist ein Feature?



Was denken Sie?

Welche Eigenschaften müssen Features haben?

In Python würden wir für die Dimensionen unserer Feature-Matrix schreiben:
`[n_samples, n_features]`

So what?
- Den meisten Aufwand im Data Science Prozess hat man mit Datenbereinigung und Feature-Engineering
- Modellierung macht gefühlte 20% aus

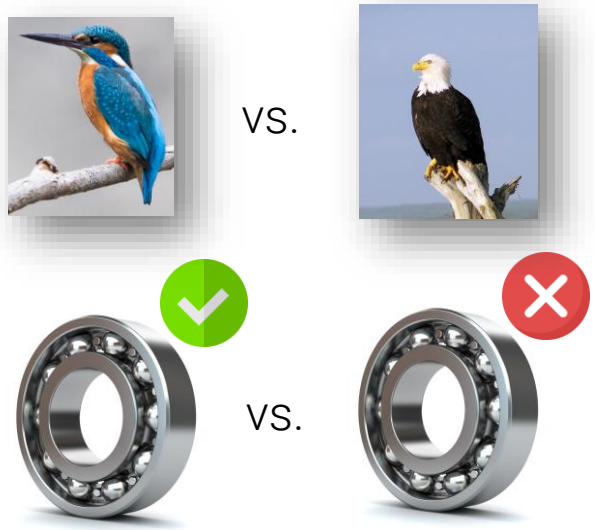
Definition

"In machine learning and pattern recognition, a **feature** is an **individual measurable property or characteristic of a phenomenon** being observed."
→ **Operationalisierung!**

- Features können also sowohl die **Daten an sich** sein, als auch davon **abgeleitete Größen**
- Meist spricht man aber von Features, wenn die Daten auf eine bestimmte Weise **aggregiert** wurden → in der Regel **verringert** man durch Feature-Engineering schon die **Dimensionalität** des Problems!
- Ganz strikt formuliert: Feature-Engineering bildet eine messbare Eigenschaft auf einen **Skalar** ab
- Daher ist der Begriff Feature oft nicht scharf abzugrenzen
- Man könnte auch sagen: aus den Rohdaten erzeugt man im Prozess des **Feature-Engineering** eine **Feature-Matrix** mit sauberem Input für unser Modell
- Der Vorgang des Feature-Engineering ist kritisch für die anschließende **Modellgüte**

Welche Eigenschaften müssen Features haben?

Frequenzspektrum



Features müssen **informativ** sein bzw. genügend Informationen über das zugrundeliegende Phänomen tragen

Features müssen **diskriminativ** sein und somit Unterschiede bzw. Strukturen in den Daten erklären können

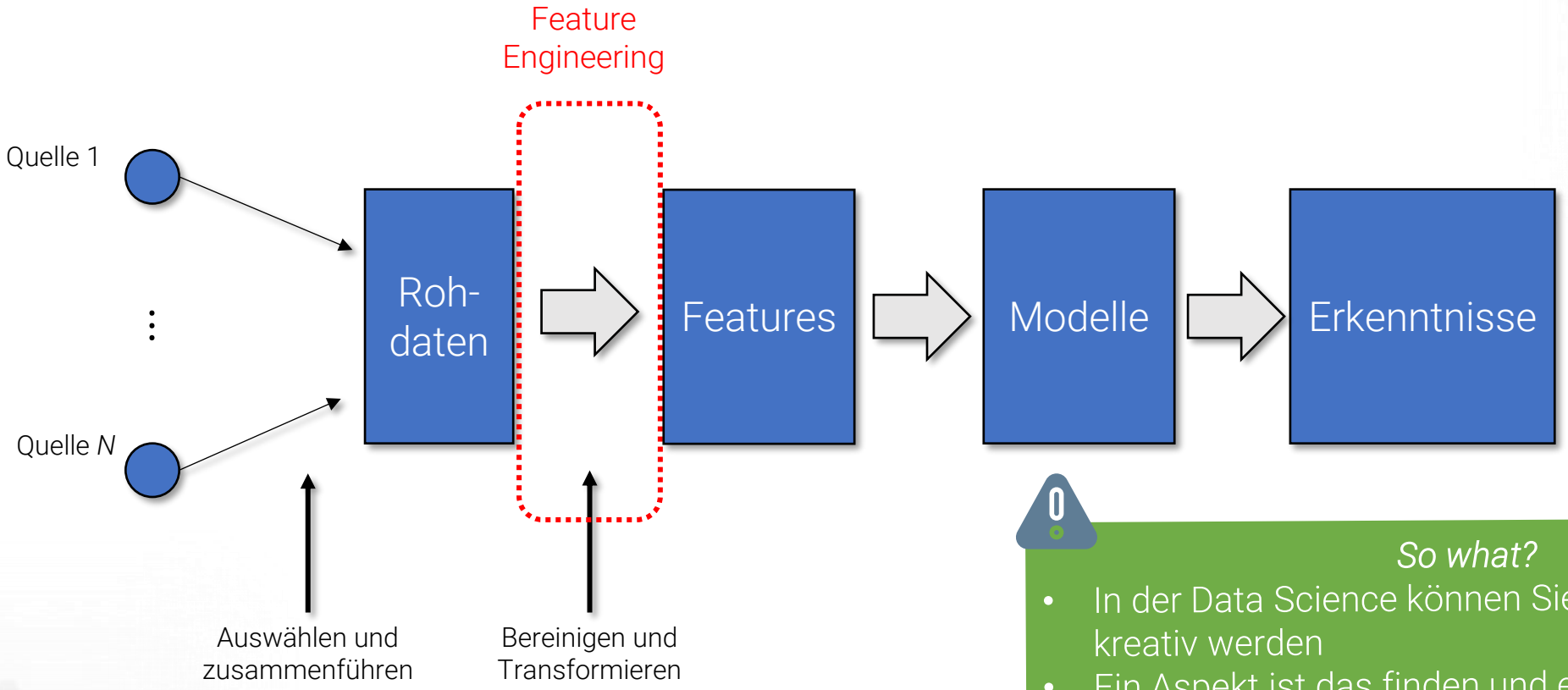
Features dürfen **nicht** zu stark miteinander **korrelieren**, um Redundanz zu vermeiden

So what?

Auch die **Anzahl** an genutzten Features ist wichtig:

- Zu wenig Features: das Modell kann das Phänomen nicht beschreiben
- Zu viele Features: das Modell ist schwierig anzulernen

Feature Engineering: Bindeglied zwischen Daten und Erkenntnissen

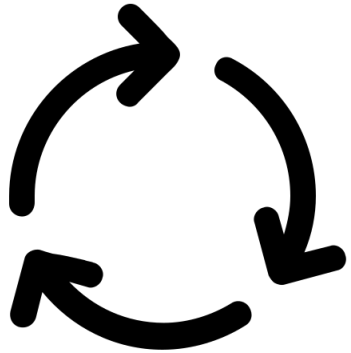


So what?

- In der Data Science können Sie an vielen Stellen kreativ werden
- Ein Aspekt ist das finden und erzeugen von geeigneten Features
- Im Erzeugen von Features steckt Domänenwissen → Wir unterstützen das Modell durch unser Wissen

Iterativer Prozess: Modell vs. Features

Iterativer Prozess



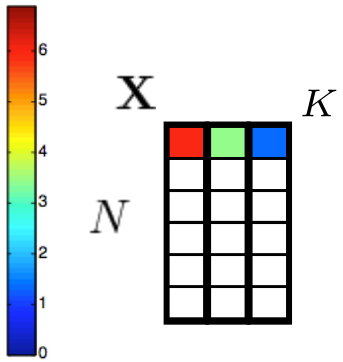
Features

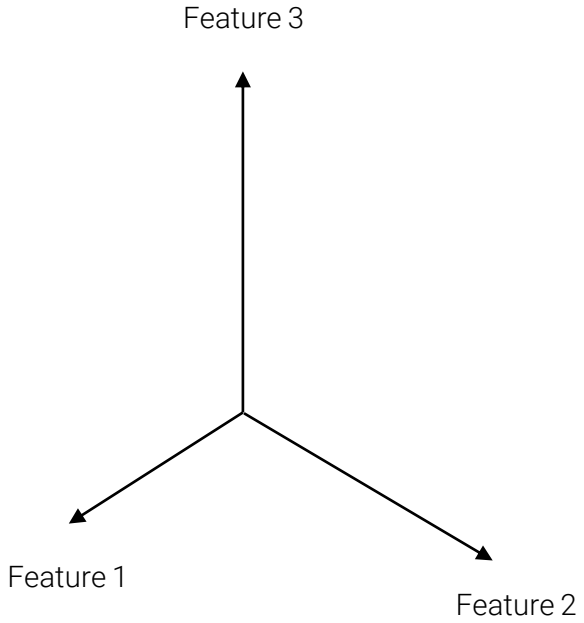
- Features bzw. Feature-Engineering sind nicht leicht zu generalisieren
- Geeignete Features hängen sowohl vom zu beschreibenden Phänomen als auch vom gewählten Modell ab

Modell

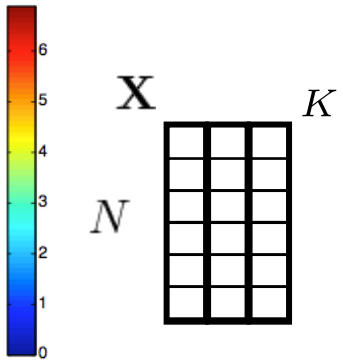
- Die Güte des gewählten Modells wiederum hängt maßgeblich von den gewählten Features und deren Anzahl ab
- Modell und Features beeinflussen sich also gegenseitig

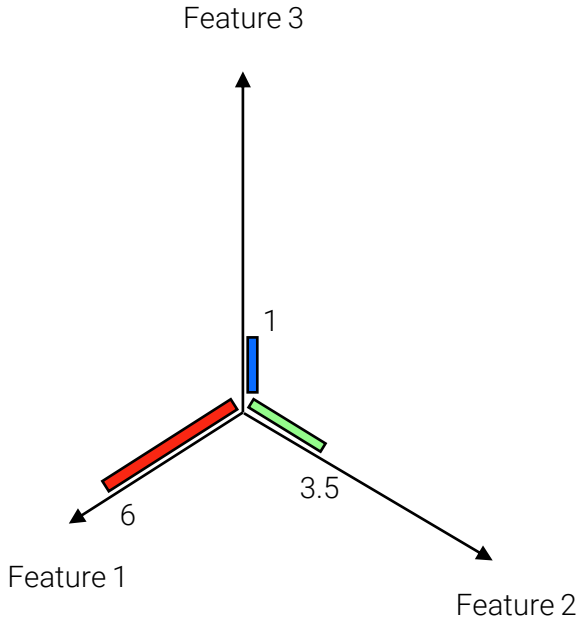
Von der Feature-Matrix in den Feature-Raum – und zurück





Von der Feature-Matrix in den Feature-Raum – und zurück





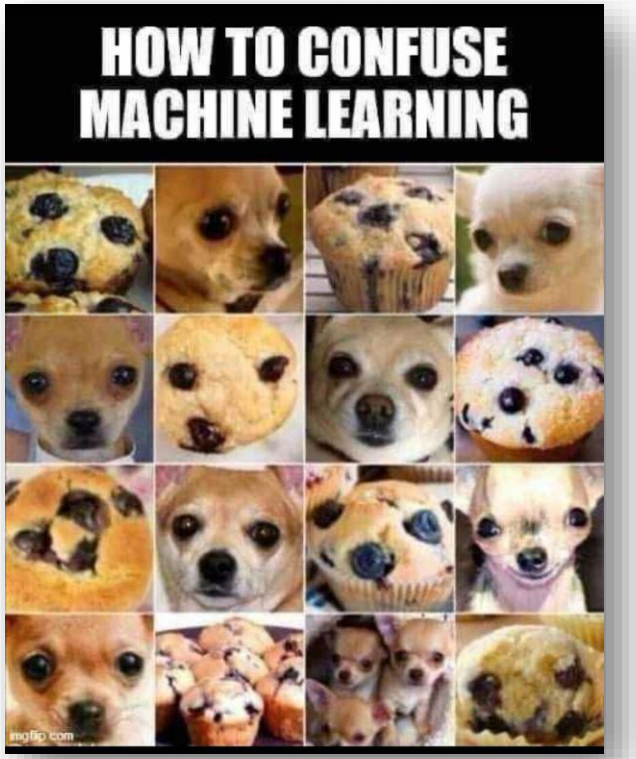
So what?

Vorsicht Verwechslungsgefahr: jetzt wird aus einer Datenmatrix eine Feature-Matrix – oft verwendet man für beides in der Schreibweise ein großes, dick gedrucktes X!

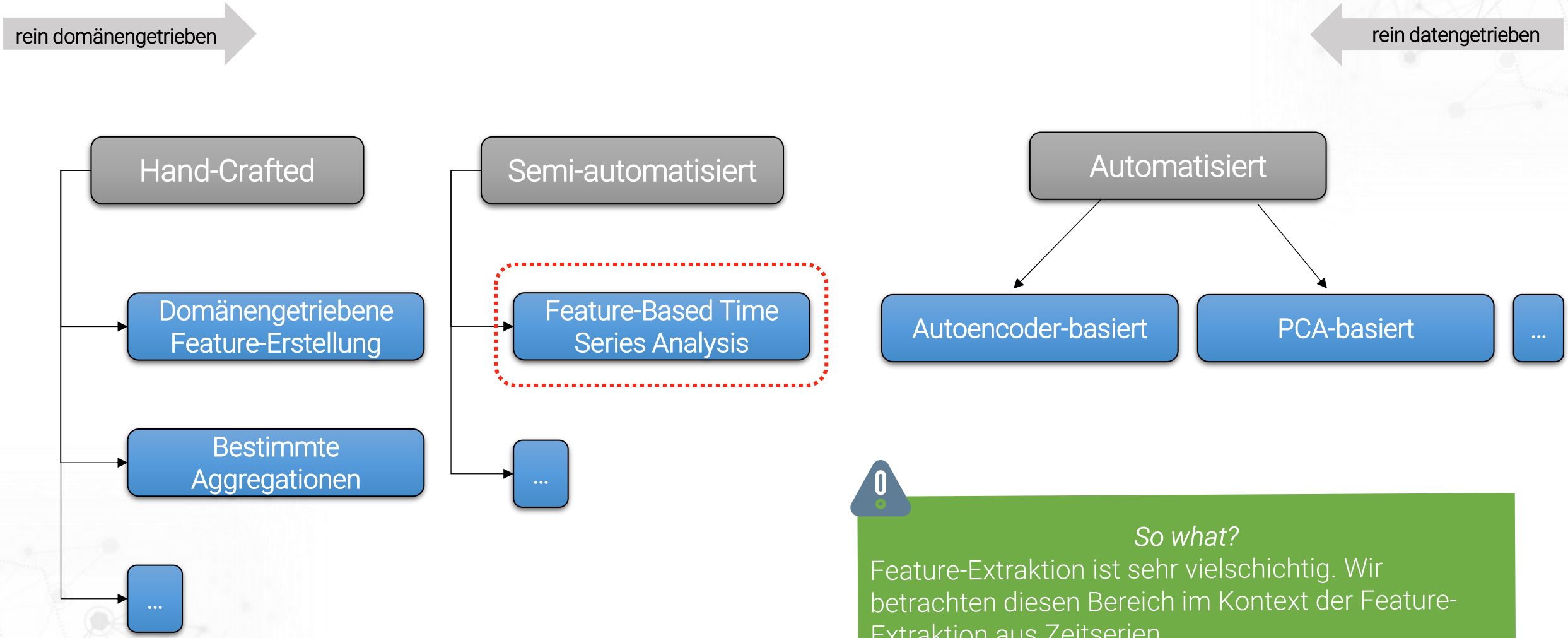
Feature Extraction

Phänomen → Skalar

Daten → Skalar



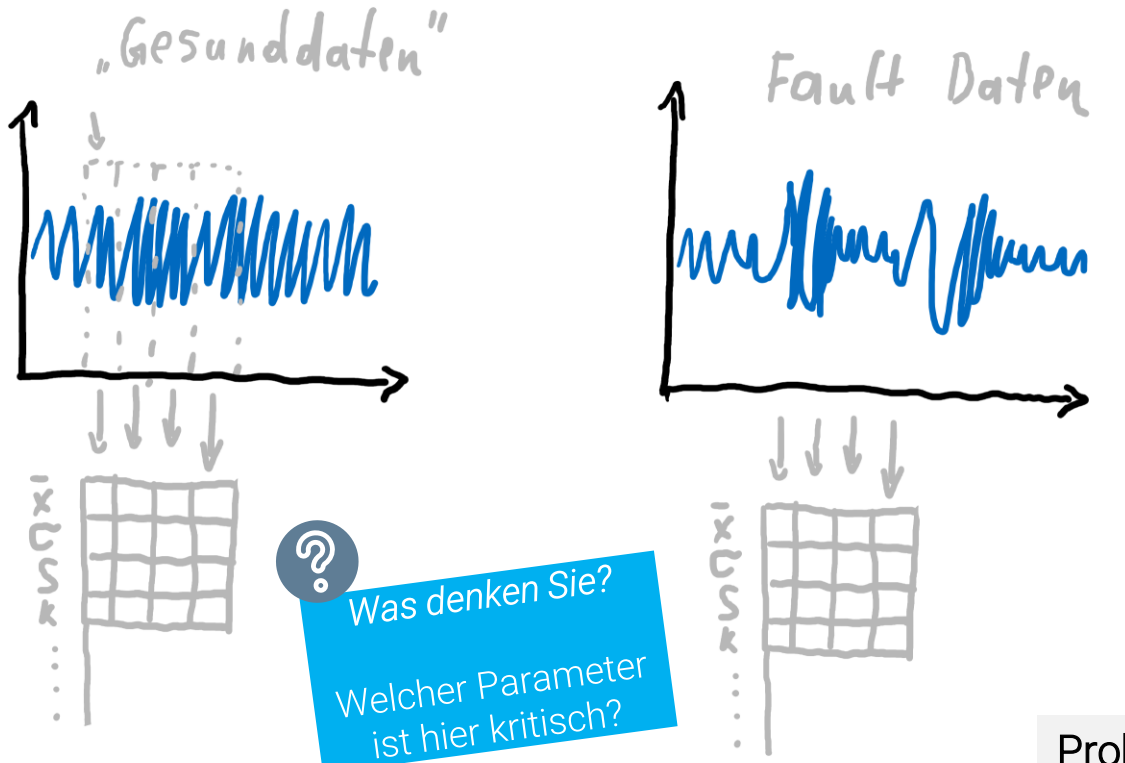
Arten der Feature-Extraktion und Features



So what?

Feature-Extraktion ist sehr vielschichtig. Wir betrachten diesen Bereich im Kontext der Feature-Extraktion aus Zeitserien.

Feature-Extraktion aus Zeitserien



Was denken Sie?

Welcher Parameter ist hier kritisch?

- Stellen Sie sich vor, Sie wollen unterscheiden, ob ein Motor einen Schaden hat oder nicht
- Das tun Sie z.B. anhand von aufgezeichneten Stromdaten
- Wir könnten zum einen Charakteristiken der **gesamten** Zeitserien extrahieren
- Oder von **Intervallen** der jeweiligen Zeitserien

Was denken Sie?

Wie könnten wir nun Charakteristiken dieser Zeitserien erzeugen?

So what?

Achtung: es gibt (mind.) zwei große Gruppen an Zeitseriendaten.

- Event-related/triggered: es existieren ausgezeichnete Zeitpunkte anhand derer referenziert werden kann
- Kontinuierliche: fehlen solcher Zeitpunkte

Probleme, die dadurch behoben werden:

- Domänenwissen fehlt → nach was soll ich suchen?
- Datenumfang zu groß
- In welchen **Segmenten** der Zeitserien befinden sich die **Charakteristika**?
- Bei Zeitserien: ein Startpunkt liegt in den wenigsten Fällen vor! → die extrahierten Charakteristika sollen zu **beliebigen Zeitpunkten diskriminativ** sein!

Mathematische Formulierung

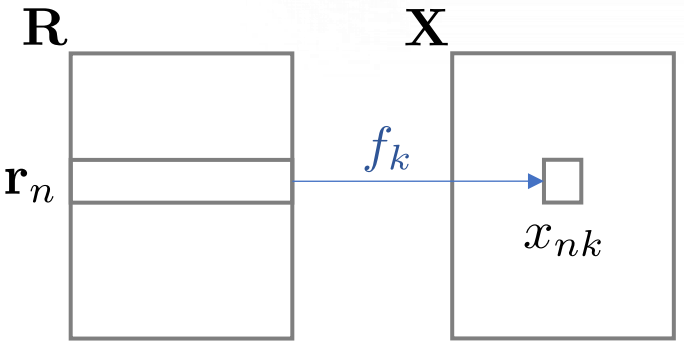
Was denken Sie?

Wie könnte man die Transformation von Rohdaten in Features generisch formulieren?

Rohdaten $\mathcal{R} := \{\mathbf{r}_1, \ldots, \mathbf{r}_N\}$ $\mathbf{r}_n \in \mathbb{R}^T$ $\mathbf{R} \in \mathbb{R}^{N \times T}$

Abbildung $\mathbf{f} : \mathbb{R}^T \to \mathbb{R}^K$ $f_k : \mathbb{R}^T \to \mathbb{R}$

Featurevektoren $\mathcal{X} := \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ $\mathbf{x}_n \in \mathbb{R}^K$ $\mathbf{X} \in \mathbb{X}^{N \times K}$

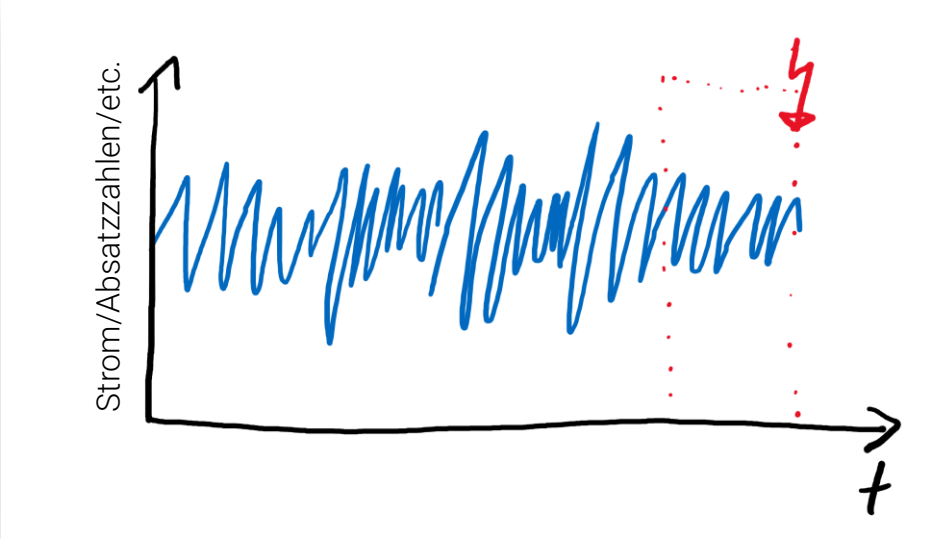


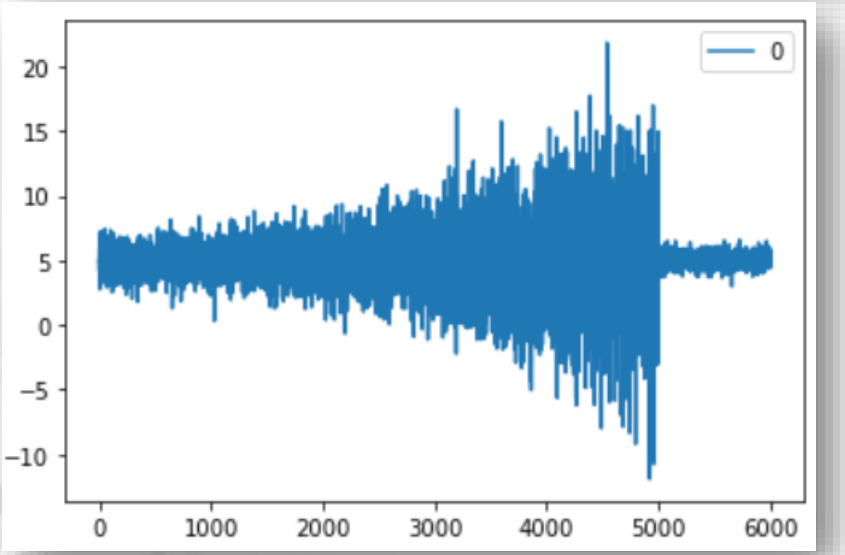
So what?

Hinter Feature Extraction steht also eine **vektorwertige Funktion**, die vom Raum der Rohdaten in den Feature-Raum abbildet.

Stift, Zettel, kurz mit Nachbarn austauschen

Anwendungsfall: Vorhersage Motorausfall, Einbruch Absatz eines Produkts





- Neben dieser Klassifikationsaufgabe, können auch kontinuierliche Labels – also Regressionsprobleme mittels dieser Feature-Extraktion aus Zeitserien angegangen werden

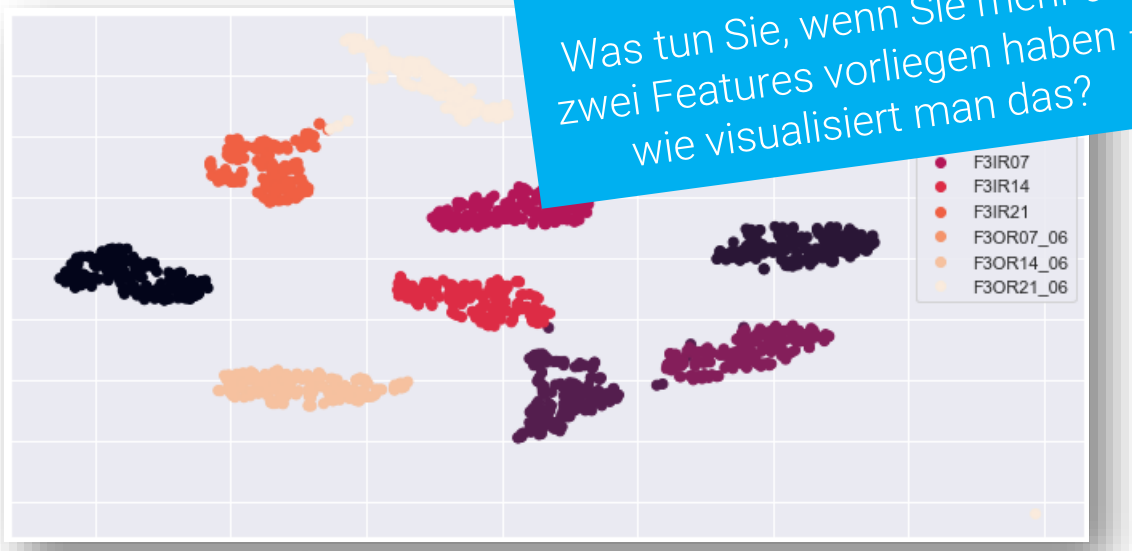
- Wenn man z.B. eine Zeitserie vor und nach einem Event (Motorausfall, Einbruch im Absatzverhalten, etc.) vorliegen hat → Extrahierte Features können Aufschluss auf das Event geben!

Demo BirdNET

Visualisierung anhand t-SNE

Was denken Sie?

Was tun Sie, wenn Sie mehr als zwei Features vorliegen haben – wie visualisiert man das?



Grafik erstellt durch ein Team der Projektarbeit „Predictive Maintenance“

- t-SNE: t-Distributed Stochastic Neighbor Embedding
- Nicht-lineare Methode zur Datenexploration
- Projektion hochdimensionaler Daten in den zwei- oder dreidimensionalen Raum
- Grobe Funktionsweise: die Dichten der Daten vom hochdimensionalen Raum müssen auch im niedrigdimensionalen erhalten bleiben
- Abbildung aus Projektarbeit Predictive Maintenance: → 30-dim. Feature-Raum von Vibrationsdaten an einem Motor

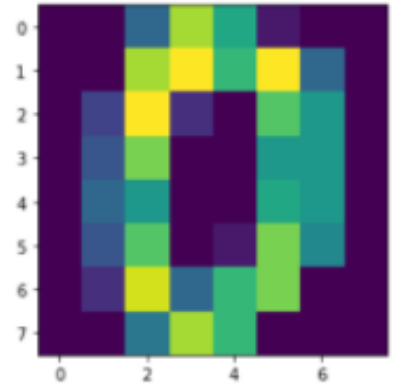
Demo
Tensorflow projector

https://projector.tensorflow.org/

https://distill.pub/2016/misread-tsne/

So what?
Eines der wichtigsten Verständnisse – bzw. die wichtigste Perspektive:
Daten im Feature-Raum!

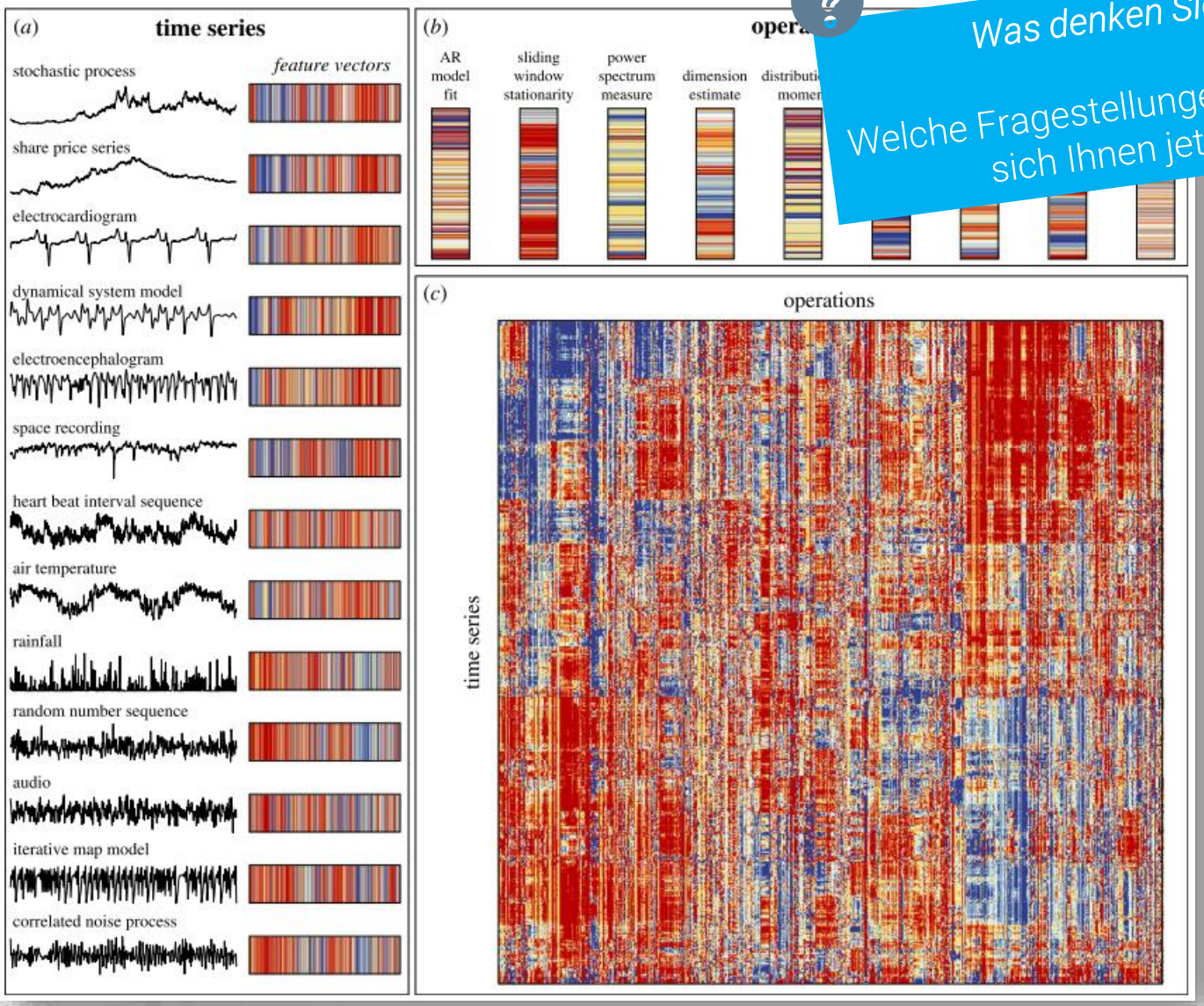
Beispiel: Embedding Digits



In `sklearn` gibt es ein bekanntes Dataset - `load_digits()`. In ihm liegen uns handgeschriebene Zahlen vor. Jede Zahl wird ursprünglich durch ein Array der Dimension `(8, 8)` repräsentiert. Im Datensatz liegen die Zahlen als Zeilenvektoren in einer großen Datenmatrix vor. Wir wollen uns in diesem Beispiel diesen 64-dimensionalen Datensatz in den zweidimensionalen Raum mittels `TSNE` projizieren.

Exkurs: Feature-Based Time Series Analysis

(Fulcher et al., 2013)



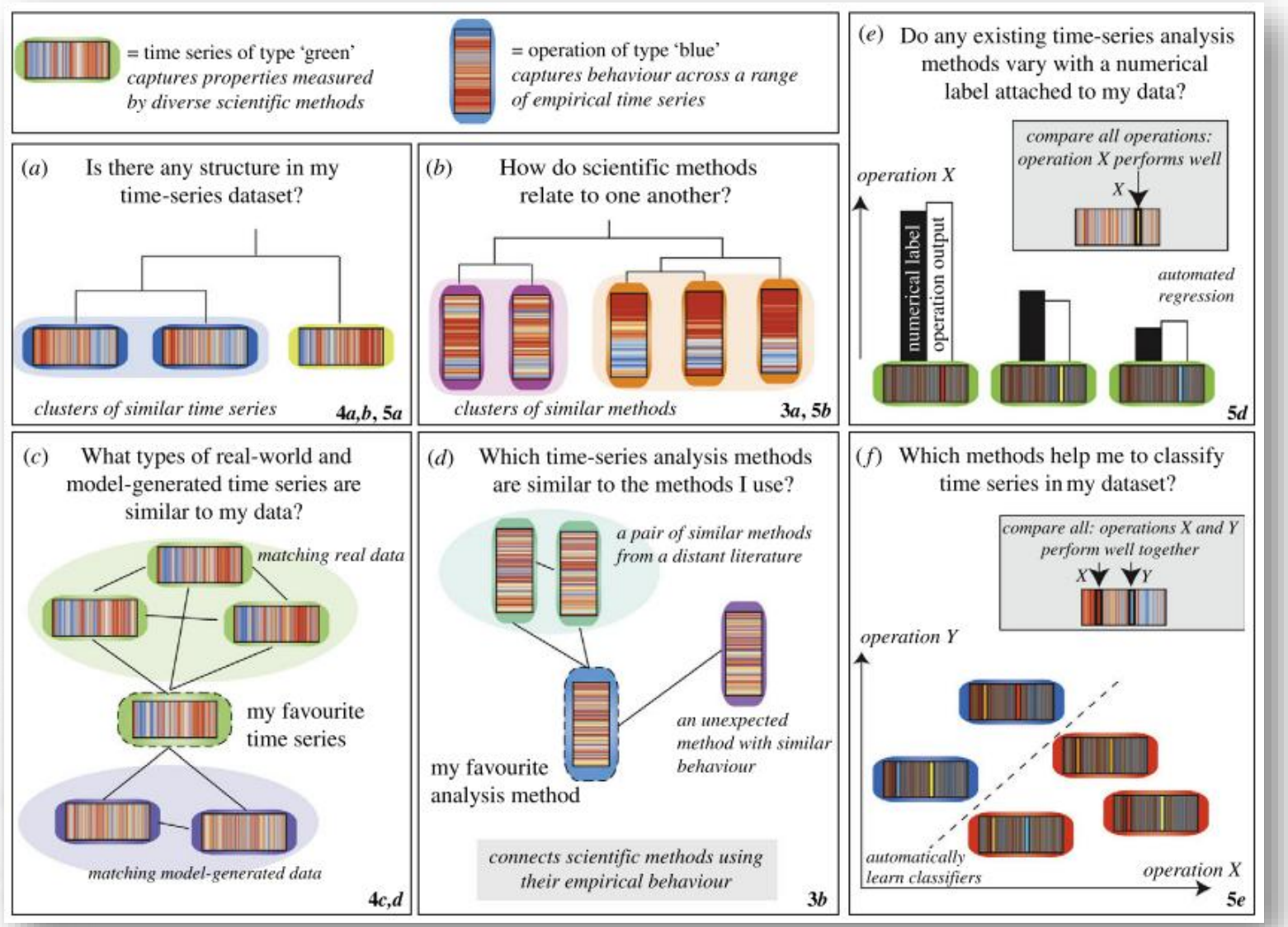
Was denken Sie?

Welche Fragestellungen eröffnen sich Ihnen jetzt?

- 38190 univariate Zeitserien aus realen (klimatisch, medizinisch, audio, (astro-)pyhsikalisch, Finanzen, etc.) und modellierten Systemen
- 9613 Methoden zur Feature-Extraktion
- Automatisiert – kein Domänenwissen nötig!

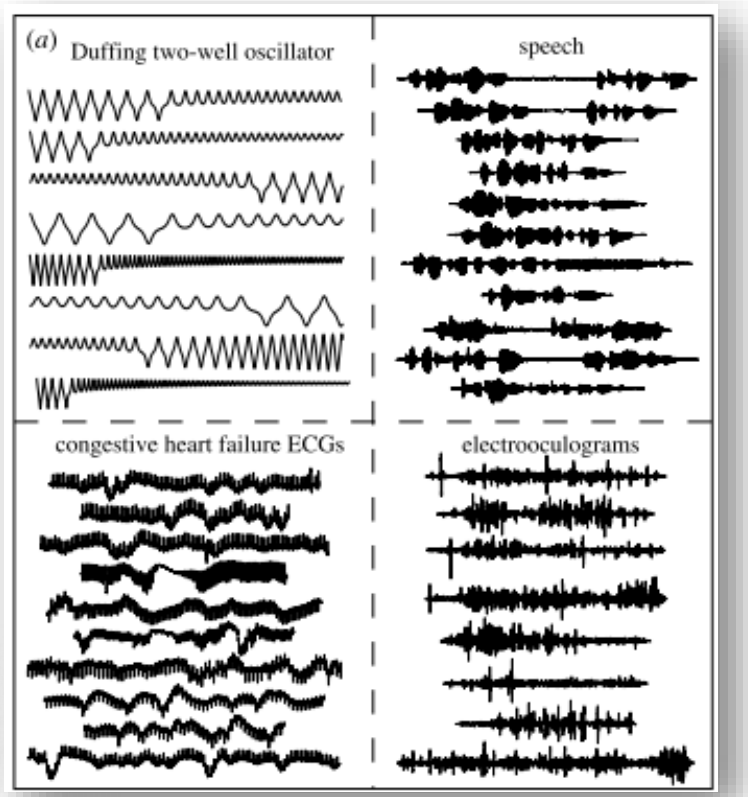
Exkurs: Feature-Based Time Series Analysis

(Fulcher et al., 2013)

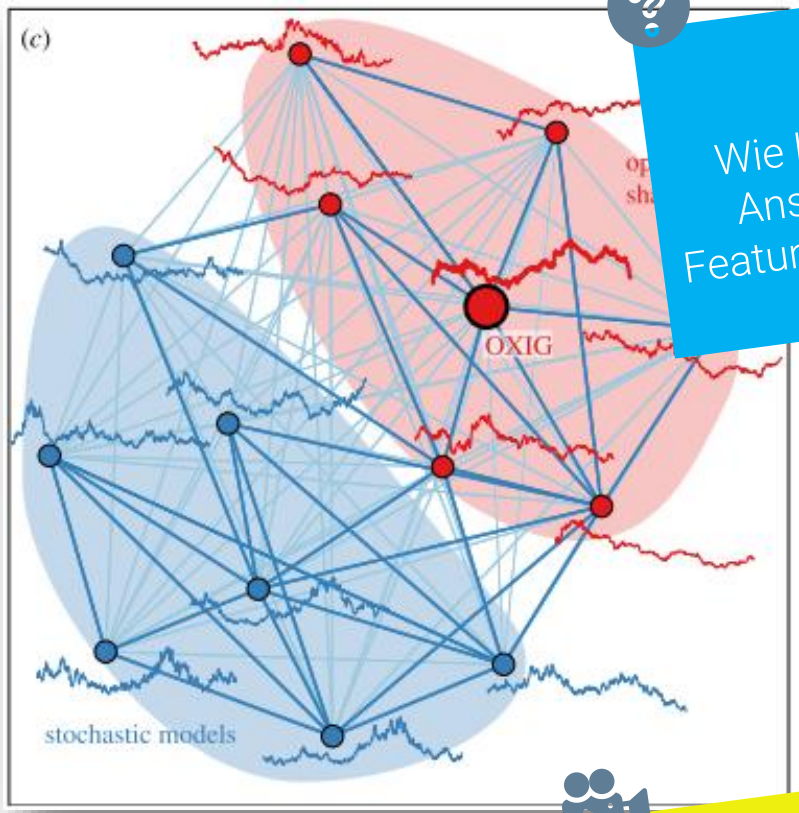


http://gph.is/2cpbE6Y

Exkurs: Feature-Based Time Series Analysis



(Fulcher et al., 2013)



(Fulcher et al., 2013)

Was denken Sie?

Wie könnte man aus diesem Ansatz eine automatisierte Feature-Extraction und -Selection bauen?

Demo CompEngine

https://www.comp-engine.org/#!

Weitere Feature-Arten: Label- und One-Hot-Encoding

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	
Broccoli	3	

One Hot E

Apple	Chicken	Br	
1	0	0	
0	1	0	
0	0	1	50

Wikipedia

Was denken Sie?

Was ist das Problem, wenn man Kategorien einfach nur (aufsteigende) Zahlen zuordnet?

- Beim One Hot Encoding wird für **jede Kategorie** einer kategorialen Variable eine **Spalte erzeugt**
- Eine „1“ bedeutet: trifft zu, eine „0“ bedeutet. Trifft nicht zu
- **Warum tut man das?**
  - Ausprägungen kategorialer Variablen haben oft keine hierarchische Beziehung zueinander
    Apple > Chicken ?
  - Bei Label Encoding würde man diese aber erzeugen!
  - One Hot Encoding verhindert das

- Liegen **kategoriale** Variablen vor, so ist das sog. *One Hot Encoding* – neben dem *Label Encoding* - eine übliche Methode, um Features zu bilden
- Einschub kategoriale Variable:
  *„a categorical variable is a variable that can take on **one of a limited**, and usually fixed, number of possible values, assigning each individual or other unit of observation to a particular group or nominal category on the basis of some qualitative property.”*
- **Beispiele**:
  - Machinentyp
  - Namen
  - Geschlecht
  - Blutgruppe
  - Lager-ID eines Kunden
  - etc.

Weitere Feature-Arten: Label- und One-Hot-Encoding

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Categories
categories = np.array(['apple', 'apple', 'milk', 'banana', 'coffee'])

# Creating instance of labelencoder
labelencoder = LabelEncoder()

# Label encoding
label_encoded = labelencoder.fit_transform(categories)
label_encoded
```

```python
# Creating instance of one-hot-encoder
ohe = OneHotEncoder(handle_unknown='ignore')

# One Hot Encoding with numerical labels
ohe.fit_transform(label_encoded.reshape(-1, 1)).toarray()
```

```python
# One Hot Encoding with string labels
enc.fit_transform(categories.reshape(-1, 1)).toarray()
```

- `sklearn` bietet sowohl einen `LabelEncoder`, als auch einen `OneHotEncoder`
- Beide nehmen Arrays an Ausprägungen kategorialer Variablen auf (String, numerisch, …)
- Und transformieren diese dann entweder in **Numerische Labels** oder **One Hot Encoded** Vektoren

So what?

Wir kommen hier schon in Kontakt mit der allgemeingültigen Syntax von sklearn

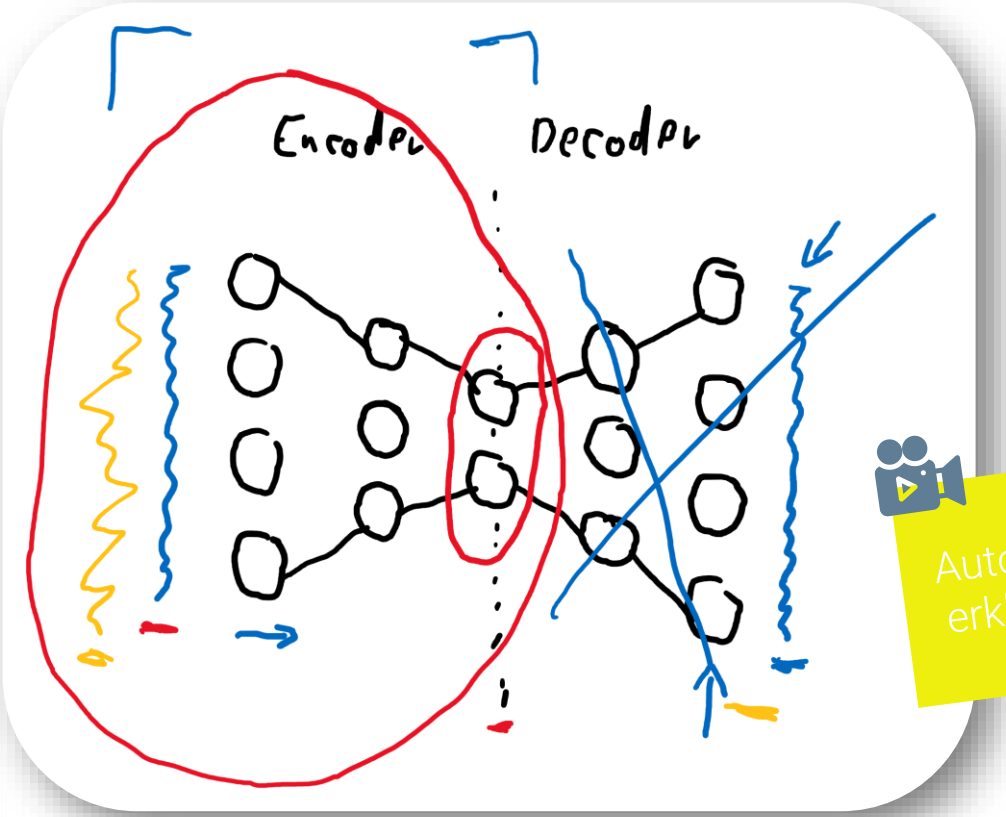
→ `.fit()`
→ `.transform()`
→ `.fit_transform()`

Weitere Feature-Arten: Bag-of-Words

```python
# Import
from sklearn.feature_extraction.text import CountVectorizer

# Our documents
corpus = [
        'This is the first document.',
        'This is the second second document.',
        'And the third one.',
        'Is this the first document?',
        ]

# Create instance
vectorizer = CountVectorizer()

# Count words
count_vectors = vectorizer.fit_transform(corpus)
count_vectors.toarray()
```

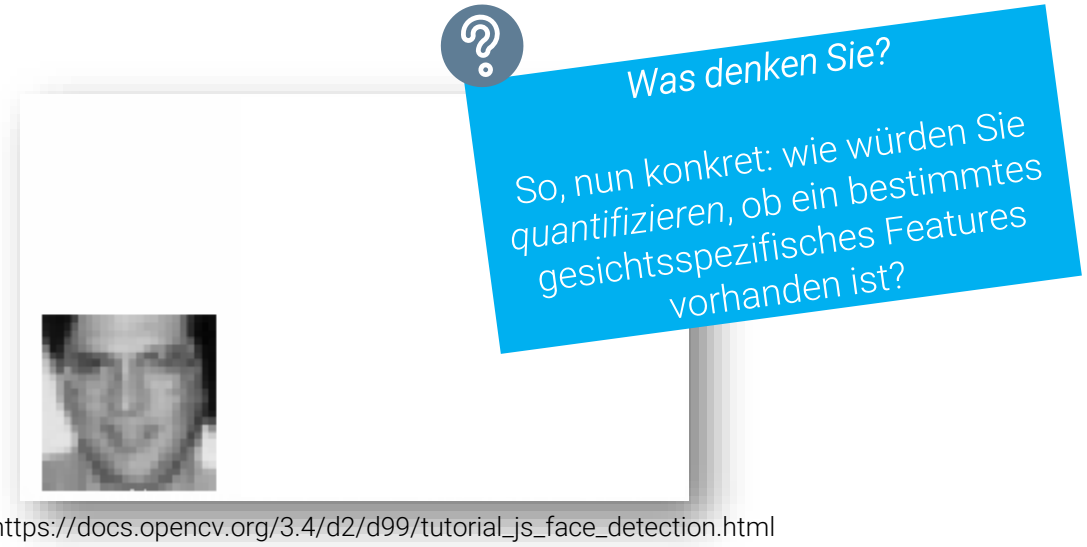
- Eine typische Methode, um Texte/Dokumente für Machine Learning Modelle verfügbar zu machen ist der sog. **Bag-of-Words** Ansatz
- Jedes Dokument wird einem **Vektor** zugewiesen
- Die **Einträge** in den Vektoren entsprechen den **Worthäufigkeiten**
- Die **Struktur** der Texte geht verloren!
- Diese Feature-Art genügt jedoch häufig bei der Analyse von Texten mittels Machine Learning

Weitere Feature-Arten: Autoencoder-basiert



Demo
Autoencoder-basiert erklären wir uns am Whiteboard

Weitere Feature-Arten: Haar Features from Images



https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html

0	0	255	255
0	0	255	255
0	0	255	255
0	0	255	255

Ideal: Kante ist sicher vorhanden

255 – 0 = 255 (Feature-Wert)

1	14	202	252
4	9	200	245
2	23	220	251
5	12	212	250

Realistisch: Kante ist wahrscheinlich vorhanden

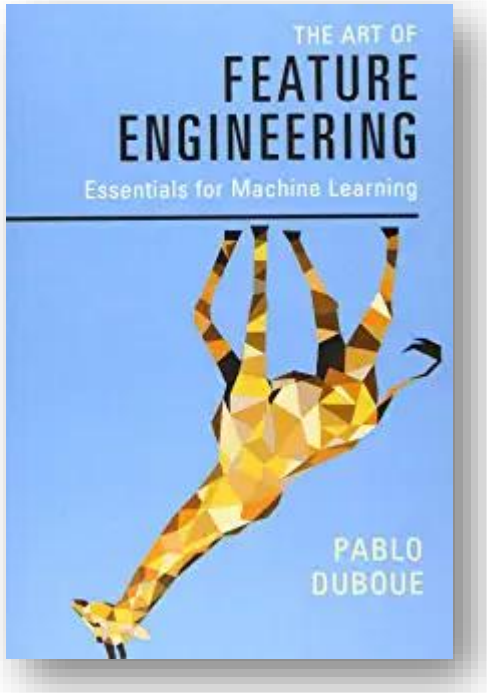
229 – 8.75 = 220.25 (Feature-Wert)

- Eine typische Feature-Art in der Bildverarbeitung bzw. Gesichtserkennung sind sog. **Haar-Features**
- Sie werden genutzt, um Aussagen darüber treffen zu können, ob **bestimmte Aspekte** in einem Bild vorhanden sind (z.B. Kanten, Ecken, Nase, Augenpartie, etc.)
- Haar-Features bestehen aus **schwarzen und weißen Regionen**
- Der mittlere Intensitätswert des schwarzen Bereichs wird von dem des weißen Bereichs abgezogen
  → Abbildung auf einen **Skalar**!
- Hohe Werte sprechen dafür, dass der Aspekt, den das spezifische Haar-Feature abbilden will, in einem Bild vorhanden ist

Weitere Feature-Arten: …

… ein gefühlt unerschöpfliches Feld

Es gibt nicht umsonst Bücher wie z.B.



Feature Selection

Let‘s avoid „garbage in, garbage out.“

Warum überhaupt?

Was denken Sie?

Warum füttern wir nicht einfach alle Features, die wir extrahieren, in den ML Algorithmus?

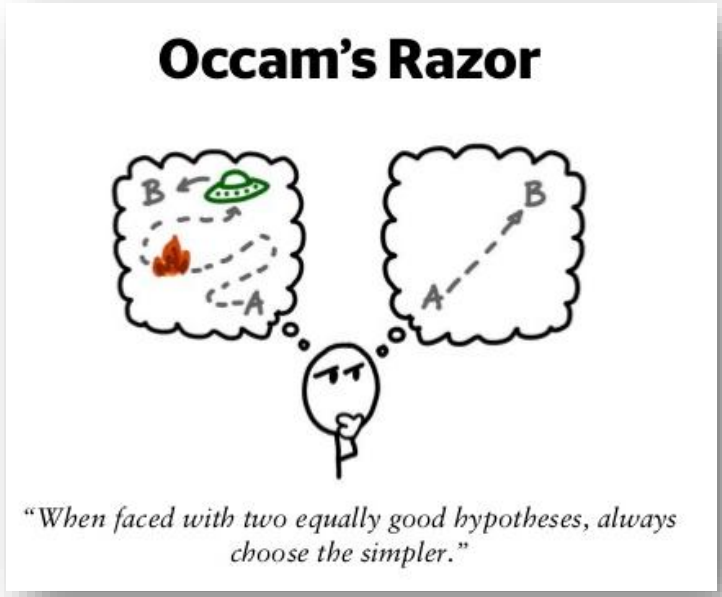
http://gph.is/1rjAV43

Curse of Dimensionality

Dimensionalität erhöht sich
→ Volumen erhöht sich so stark, dass Datensätze *sparse* werden.
→ Strukturen können nicht mehr statistisch haltbar abgebildet warden
→ Daten nähern sich schnell einer Gleichverteilung

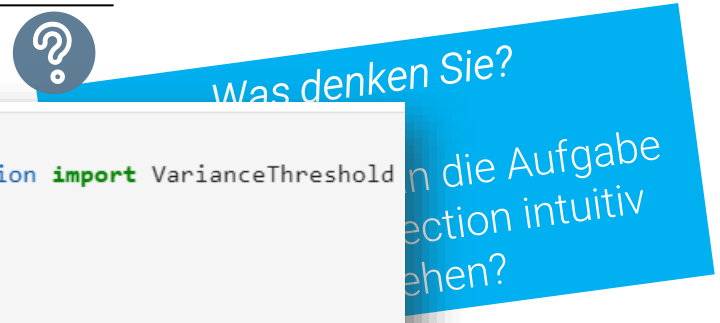
https://i1.wp.com/cdn.shortpixel.ai/client/to_webp,q_glossy,ret_img,w_1000,h_400/https://www.dataquest.io/wp-content/uploads/2019/08/garbage-in-garbage-out.jpg?w=450&ssl=1

"Also, a large number of features make a model bulky, time-taking, and harder to implement in production."



https://automaticaddison.com/wp-content/uploads/2019/06/1.jpg

Einfache Feature Selection




```python
# Import
from sklearn.feature_selection import VarianceThreshold

# Data
X = np.array([[0, 0, 1],
              [0, 1, 0],
              [1, 0, 0],
              [0, 1, 1],
              [0, 1, 0],
              [0, 1, 1]])

# Variance
X.var(axis=0)
```

```
array([0.13888889, 0.22222222, 0.25      ])
```

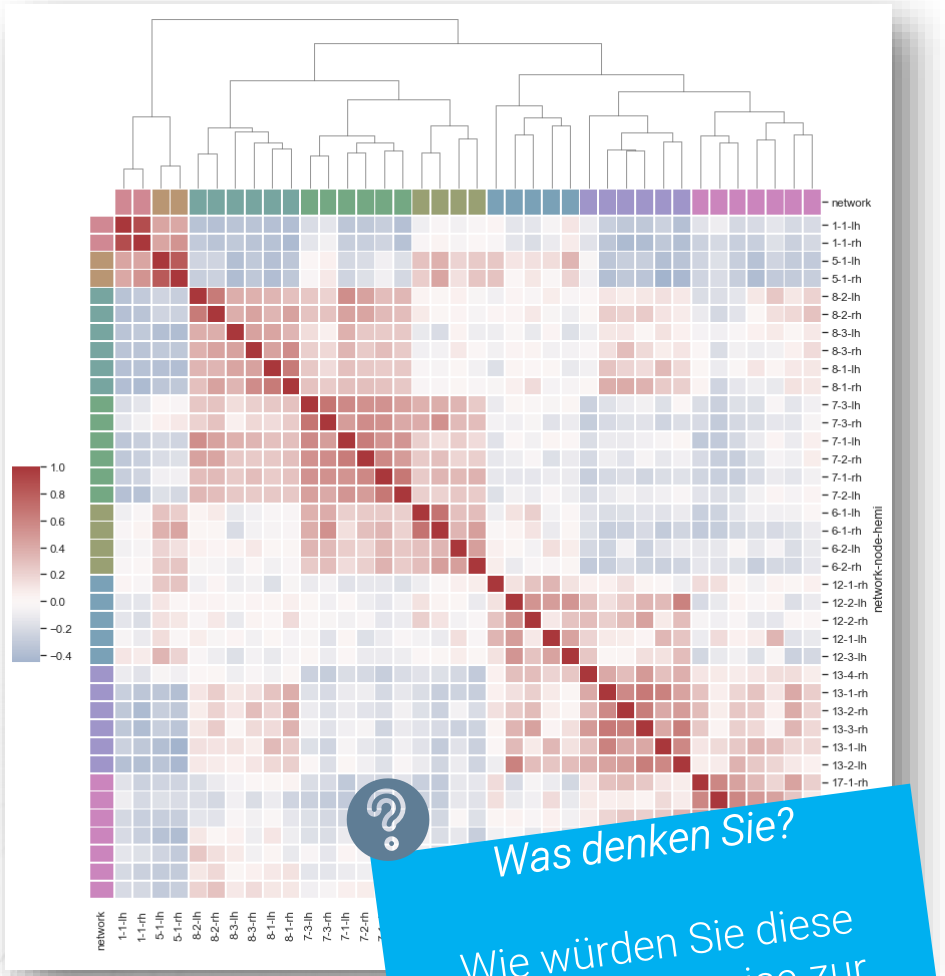
```python
# Select features based on variance threshold
sel = VarianceThreshold(threshold=.15)
sel.fit_transform(X)
```

% Missing Values: Features werden für unbrauchbar erklärt, die mehr als einen bestimmten Prozentsatz an **fehlenden** Datenpunkten beinhalten

Constant Values: Features werden für unbrauchbar erklärt, die mehr als einen bestimmten Prozentsatz an **konstanten** Datenpunkten beinhalten

Low Variance: weist ein Feature eine zu niedrige Varianz auf, so wird es ausgeschlossen

Feature Selection mittels Korrelation



Was denken Sie?

Wie würden Sie diese Darstellungsweise zur Feature Selection nutzen?

Was denken Sie?

Welche Möglichkeiten sehen Sie?

Es gibt zwei Möglichkeiten Features basierend auf ihrer Korrelationsstruktur zu filtern.

- Korrelation von Features untereinander:
  - Features die stark miteinander korrelieren tragen ähnliche/gleiche Information
  - Um Redundanz zu vermeiden und Dimensionalität zu verringern → Filtern
  - Zwei Möglichkeiten:
    - Paarweises Ausschließen
    - Filtern basierend auf Clustermap-Analyse

- Korrelation von Features mit dem Target:
  - Korrelation eines Features mit dem Target liefert einen Hinweis auf den prädiktiven Gehalt eines Features
  - Herangehensweise: aus einem Satz an Features nur die top *n* korrelierenden Features behalten

SelectKBest

```python
# Import
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, mutual_info_classif

# Data
X, y = load_digits(return_X_y=True)
X.shape
```

```
(1797, 64)
```

```python
# Select k best
X_new = SelectKBest(mutual_info_classif, k=20).fit_transform(X, y)
X_new.shape
```

```
(1797, 20)
```

- `sklearn` hat eine Klasse `SelectKBest`, mit der man Features bzgl. verschiedener Scores auswählen kann

  „Select features according to the k highest scores."

- Auch hier werden Maße zwischen Features und Target berechnet und die Top Scoring Features ausgewählt

See also:

`f_classif`
ANOVA F-value between label/feature for classification tasks.

`mutual_info_classif`
Mutual information for a discrete target.

`chi2`
Chi-squared stats of non-negative features for classification tasks.

`f_regression`
F-value between label/feature for regression tasks.

`mutual_info_regression`
Mutual information for a continuous target.

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

Feature Scaling und Normalisierung

Abbildung des Wertebereichs von Features auf bestimmte Zahlenbereiche

Warum?

→ Manche Modelle benötigen bestimmte Wertebereiche, Skalierungen, etc.

→ Um Features gleich zu gewichten

MinMax-Transformation

```python
# Data
X = np.random.randn(10, 5)
```

```python
# MinMax Transformation from scratch
max_value = 5
min_value = -3
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max_value - min_value) + min_value
```

```python
# MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(min_value, max_value))
scaler.fit_transform(X)
```

- In der sog. MinMax-Transformation werden die Feature-Werte auf ein bestimmtes Intervall – häufig `[0;1]` skaliert
- Hierzu gibt es eine `sklearn` Klasse `MinMaxScaler`
- Die zugrundeliegende Transformation ist auch einfach durch wenige Zeilen Code darstellbar

Standardisierung

```python
# Import
from sklearn.preprocessing import StandardScaler

# Data
X = np.random.rand(200, 1)

# Mean, std
print(X.mean())
print(X.std())
```

```
0.5366466358055808
0.2882643838253411
```

```python
# Scale
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Mean, std
print(X.mean())
print(X.std())
```

```
1.9539925233402755e-16
1.0
```

- Bei der Standardisierung von Features werden die Feature-Werte so skaliert, dass sie einen Mittelwert von null und eine Standardabweichung von eins aufweisen

$$Z = \frac{X - \mu}{\sigma}$$

- Man bringt dadurch die Features näher an eine Standardnormalverteilung heran
- Auch hierzu gibt es wiederum eine `sklearn` Klasse `StandardScaler`

"Standardization of a dataset is a common requirement for many machine learning estimators: they might behave **badly** if the individual features do not more or less look like **standard normally distributed** data (e.g. Gaussian with 0 mean and unit variance).

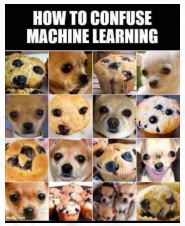
For instance many elements used in the **objective function** of a learning algorithm assume that all features are **centered around 0 and have variance in the same order**. If a feature has a variance that is **orders of magnitude larger** than others, it might **dominate** the objective function and make the estimator unable to learn from other features correctly as expected."

Quellen

Icons made by https://smashicons.com/ from https://www.flaticon.com/

Icons made by https://smashicons.com/ from https://www.flaticon.com/

Von Matt Tillett from Cumberland, MD, USA - House Wren, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=30614136



https://pbs.twimg.com/media/EcXxYFPXgAEDo6Z.jpg