



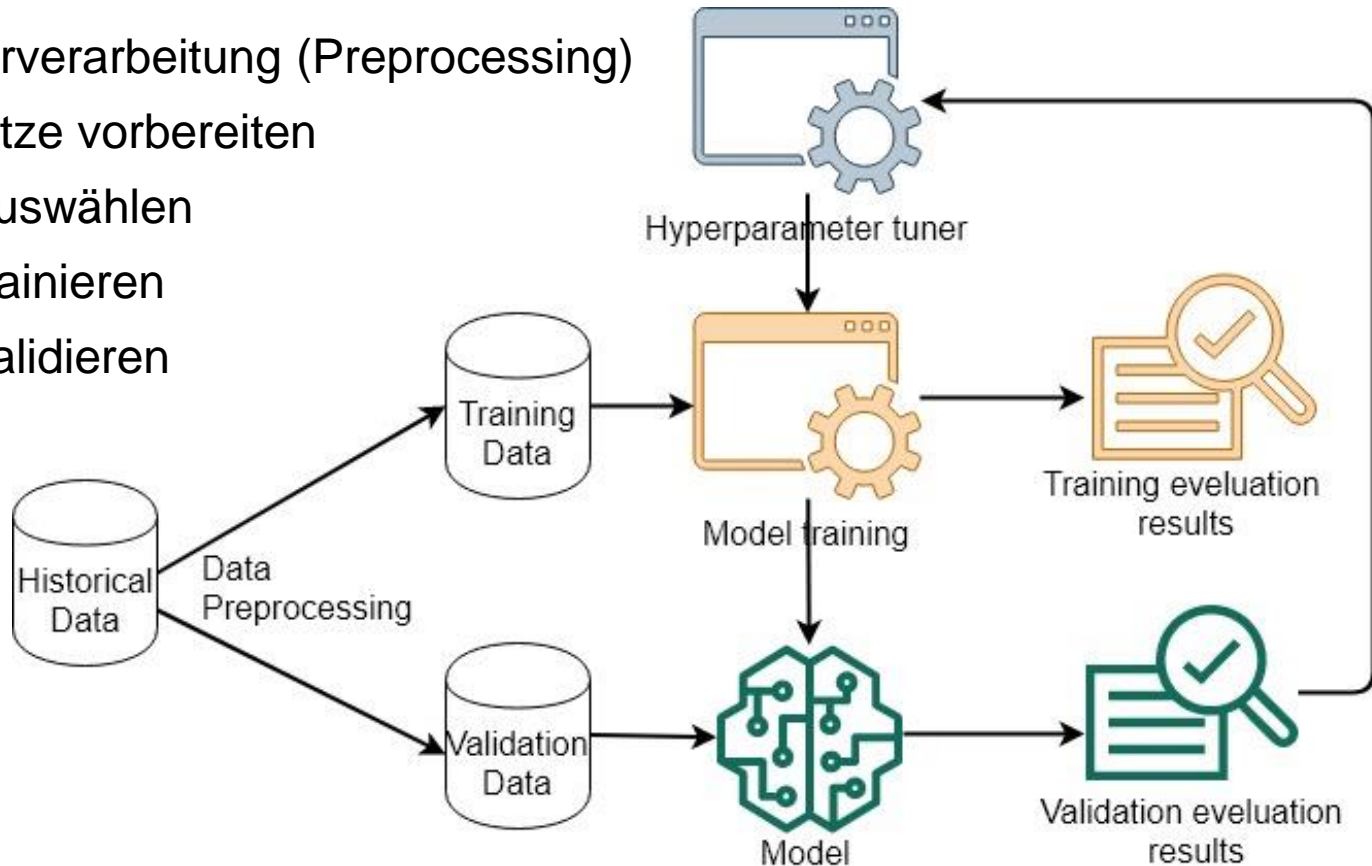
Machines Lernen in der Ingenieur Anwendung (3. Hörsaalübung)

1. Trainingsprozess
2. Python Packages: Numpy, Pandas, Scikit-learn
3. Demo
4. Aufgabestellung für die 2. Übung

Prof. Dr.-Ing. Uwe Rüppel
Meiling Shi, M.Sc.

1. Trainingsprozess

- Datenvorverarbeitung (Preprocessing)
- Datensätze vorbereiten
- Model auswählen
- Model trainieren
- Model validieren
- Testen



Quelle: Eigene Darstellung basiert auf Evaluating Machine Learning Models [Buch]

Was sind Daten?

Sammlung von Datenobjekten und deren Attributen (Features)






- Attribut: Eigenschaft oder Charakteristik eines Objekts (auch als Variable, Feld, Merkmal, Merkmal, Dimension oder Merkmal bezeichnet)
- Eine Sammlung von Attributen beschreibt ein Objekt
- Objekt wird auch als Datensatz, Punkt, Fall, Probe, Entität oder Instanz bezeichnet

Attributes




Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Arten von „Features“

- Nominal 
 - Beispiele: ID-Nummern, Augenfarbe, Postleitzahlen
- Ordinal 
 - Beispiele: Rankings (z.B. Geschmack von Kartoffelchips auf einer Skala von 1-10), Noten, Höhe {hoch, mittel, kurz}.
- Intervall 
 - Diskret  & kontinuierlich 
 - Beispiele: Kalenderdaten, Temperaturen in Celsius oder Fahrenheit.
- Verhältnis
 - Beispiele: Temperatur in Kelvin, Länge, Zeit, Anzahl der Zählungen



Kategorisierung der Features nach S. S. Stevens

		Attribute Type	Description	Examples	Operations
Categorical Qualitative		Nominal	Nominal attribute values only distinguish.  ($=$, \neq)	zip codes, employee ID numbers, eye color, sex: { <i>male</i> , <i>female</i> }	mode, entropy, contingency correlation, χ^2 test
		Ordinal	Ordinal attribute values also order objects. ($<$, $>$)	hardness of minerals, { <i>good</i> , <i>better</i> , <i>best</i> }, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric Quantitative		Interval	For interval attributes,  differences between values are meaningful. ($+$, $-$)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
		Ratio 	For ratio variables, both differences and ratios are meaningful. ($*$, $/$)	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation




Kategorisierung der Features nach S. S. Stevens

		Attribute Type	Transformation	Comments
Categorical Qualitative	Nominal		Any permutation of values 	If all employee ID numbers were reassigned, would it make any difference?
	Ordinal		An order preserving change of values, i.e., $new_value = f(old_value)$ where f is a monotonic function 	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Numeric Quantitative	Interval		$new_value = a * old_value + b$ where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
	Ratio		$new_value = a * old_value$	Length can be measured in meters or feet.


- Schlechte Datenqualität wirkt sich bei vielen Datenverarbeitungsaufwänden negativ aus
- Welche Arten von Datenqualitätsproblemen gibt es?
 - Lärm und Ausreißer
 - Fehlende Werte
 - Doppelte Daten
 - Falsche Daten
- Wie können wir Probleme mit den Daten erkennen?
- Was können wir gegen diese Probleme tun?

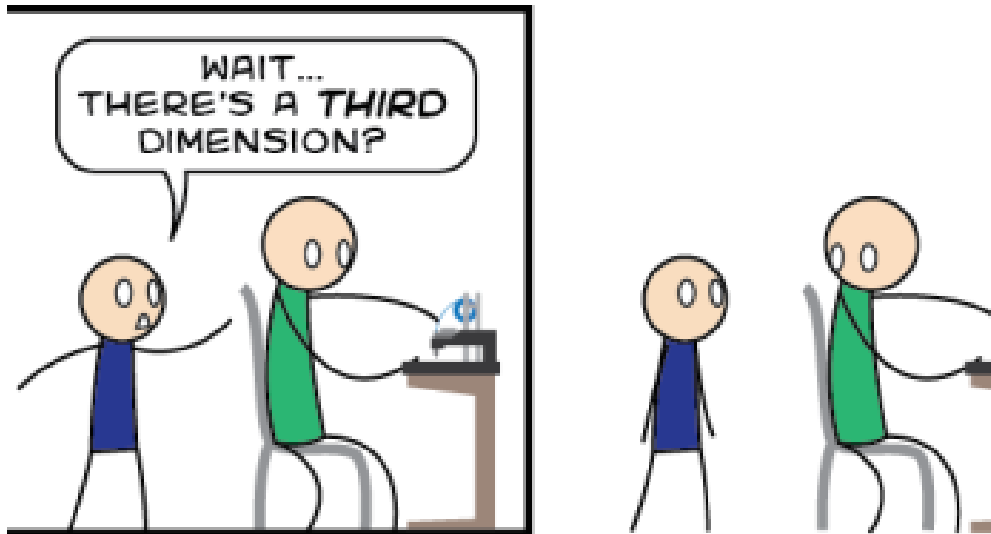
- Kombinieren von zwei oder mehr Attributen (oder Objekten) zu einem einzigen Attribut (oder Objekt)
- Zweck
 - Datenreduktion: Reduzierung der Anzahl der Attribute oder Objekte
 - Skalenwechsel:  Städte, die zu Regionen, Staaten, Ländern usw. zusammengefasst sind; Tage, die zu Wochen, Monaten oder Jahren zusammengefasst werden.
 - Mehr "stabile" Daten: Aggregierte Daten weisen in der Regel eine geringere Variabilität auf. 

Data Preprocessing: Reduzierung der Dimension

- Probleme: Curse of dimensionality 
 - Wenn die Dimensionalität zunimmt, werden die Daten in dem Raum, den sie einnehmen, immer spärlicher.
 - Definitionen von Dichte und Abstand zwischen Punkten, die für die Clusterbildung und Ausreißererkennung entscheidend sind, werden weniger aussagekräftig. 
- Zweck:
 - Zeit- und Speicherbedarf von Data-Mining-Algorithmen reduzieren
 - Visualisierung der Daten vereinfachen
 - Irrelevante Merkmale zu eliminieren  oder Rauschen zu reduzieren

Data Preprocessing: Reduzierung der Dimension

- Techniken
 - Hauptkomponentenanalyse (PCA)
 - Singuläre Wertzerlegung
 - Sonstiges: Beaufsichtigte und nichtlineare Techniken 




Quelle: <https://awkwardturtleworld.com/2015/08/16/comic-ception/>

Data Preprocessing: Auswahl der Feature-Subsets

- Eine weitere Möglichkeit, die **Dimension von Daten** zu reduzieren.
- Redundante Funktionen
 - Beispiel: Kaufpreis eines Produkts und Höhe der gezahlten Umsatzsteuer
- Irrelevante Merkmale
 - Beispiel: Der Studentenausweis ist oft irrelevant für die Aufgabe, das GPA der Studenten

Data Preprocessing: Feature erstellen

- Neue Features erstellen, die die **wichtigen Informationen**  **in einem Datensatz** viel effizienter erfassen können als die ursprünglichen Attribute.
- Allgemeine Methoden:
 - **Merkmalsextraktion**: z.B. Extrahieren von Kanten aus Bildern
 - **Feature-Konstruktion**: z.B. Teilen der Masse durch das Volumen, um die Dichte zu erhalten.

Data Preprocessing: Diskretisierung und Binarisierung

- Diskretisierung ist der Prozess der **Umwandlung eines kontinuierlichen Attributs** in ein **ordinales Attribut**
- Binarisierung bildet ein **kontinuierliches oder kategorisches Attribut** in **eine oder mehrere binäre Variablen** ab
 - Z.B. „Regnen“ -> „1“ und nicht regnen -> „0“

- Feature Transformation ist eine Funktion, die **den gesamten Wertebereich** eines bestimmten Features auf einen **neuen Wertebereich** abbildet, so dass **jeder alte Wert mit einem der neuen Werte identifiziert** werden kann.
- Einfache Funktionen: x_k , $\log(x)$, e^x , $|x|$
- Normalisierung: Bezieht sich auf verschiedene Techniken, um sich an **Unterschiede zwischen den Features** in Bezug auf Häufigkeit des **Auftretens, Mittelwert, Varianz, Bereich**.
- In der Statistik bezieht sich die **Standardisierung auf die Subtraktion der Mittel** und die **Division durch die Standardabweichung**.

Gängige Lernverfahren und ihre Aufgabe

Voraussetzung: Daten in großer Menge

Klassifikation

Regression

Clustering

Gruppen
existieren

Neuer
Datenpunkt in
welcher
Gruppe?

Identifizierung
Trends in den
Datensätze


Keine Gruppe
existieren

Erzeugen
Gruppen durch
Daten die
„ähnlich sind“

Trainingsphase: Die Phase, wo Featuresets, Models, Model Parameters umgestellt werden

Features (dt. Merkmale): individuell messbare Eigenschaft, Variable oder Charakteristik eines zu beobachtenden Data

Target Variable:  die Variable, die die Ausgabe ist oder sein sollte

Hyperparameter:  Werte, die außerhalb des Trainingsverfahrens angegeben werden müssen. Hyperparameter kann nicht aus den Trainingsdaten geschätzt werden.

Modelparameter: Modellinterne Parameter. Sie wird aus den Trainingsdaten geschätzt. "Training a model" beinhaltet die Anwendung eines Optimierungsverfahrens zur Bestimmung des besten Modellparameters, der zu den Daten "passt".

Model validieren und testen

Das Modell ist abhängig von der Größe Ihrer Daten, dem Wert, den Sie vorhersagen möchten, Eingabe usw.

Validierungs-/Testphase: Um abzuschätzen, **wie gut das Modell trainiert wurde** und um **Modelleigenschaften abzuschätzen** (mittlerer Fehler für numerische Prädiktoren, Klassifizierungsfehler für Klassifikatoren usw.)

Wenn wir mit dem ausgewählten Modelltyp und den Hyperparametern zufrieden sind, sollte ein **neues Modell** auf den gesamten **verfügbaren Datensatz** mit den **besten gefundenen Hyperparametern** zu trainieren. Dies sollte auch alle Daten umfassen, die zuvor für die Validierung reserviert waren.

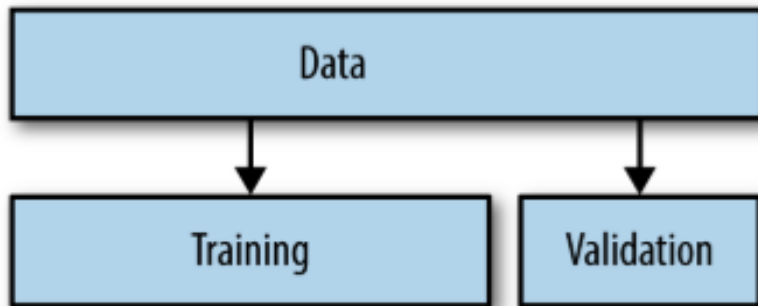
Danach findet das Testen statt. Training, Validierung und Test sollten **an verschiedenen Datensätzen** stattfinden

- **Trainingsdatensatz:** Die Stichprobe der Daten, die zur Anpassung an das Modell verwendet werden.
- **Validierungsdatensatz:** Die Stichprobe von Daten, die verwendet wird, um eine **unvoreingenommene (unbiased) Bewertung eines Modells** zu ermöglichen, das **auf den Trainingsdatensatz passt**, während die Hyperparameter des Modells angepasst werden. **Die Bewertung wird verzerrter, da die Kenntnisse über den Validierungsdatensatz in die Modellkonfiguration einfließen.**
- **Testdatensatz:** Die Stichprobe von Daten, die verwendet werden, um eine unvoreingenommene (unbiased) Bewertung **eines endgültigen Modells** zu ermöglichen, das auf den Trainingsdatensatz passt.

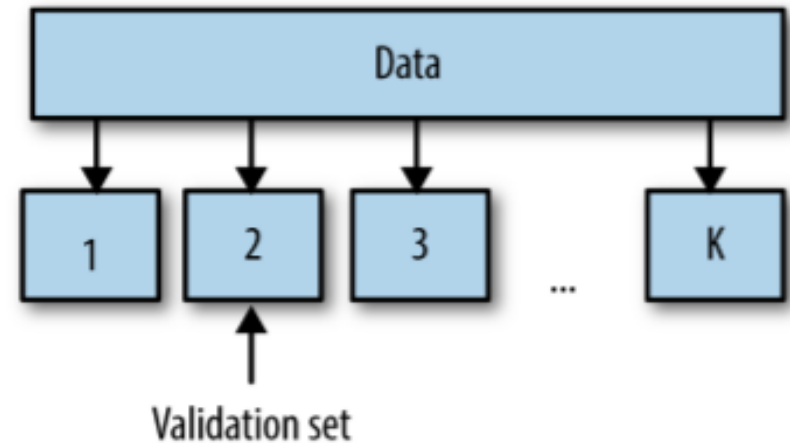
1. Hold-Out Validation: z.B. 2/3 für Trainieren und 1/3 für Testen
2. Cross Validation: Datenmenge in k Teilmenge (T_1, \dots, T_k) aufteilen; starten k Testdurchläufe, bei denen die jeweils i -te teilmenge T_i als Testmenge und die verbleibenden $(k-1)$ als Trainingsmengen verwenden.
3. Random Subsampling: Wiederholte Holdout
4. Bootstrap: Eine Resampling-Technik. Es erzeugt mehrere Datensätze, indem es von einem einzigen, ursprünglichen Datensatz abliest.

Methode für Modell Validierung

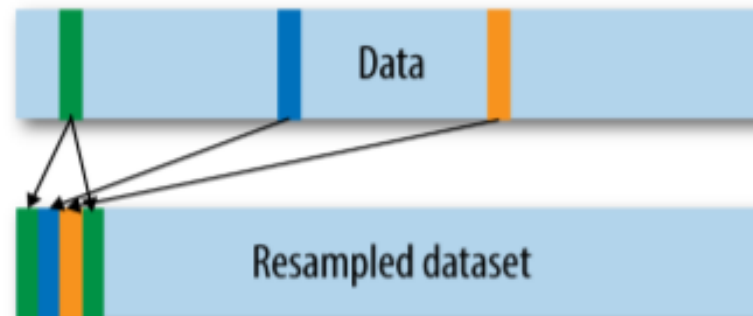
Hold-out validation



K-fold cross validation



Bootstrap resampling



Quelle: Evaluating Machine Learning Models [Buch]

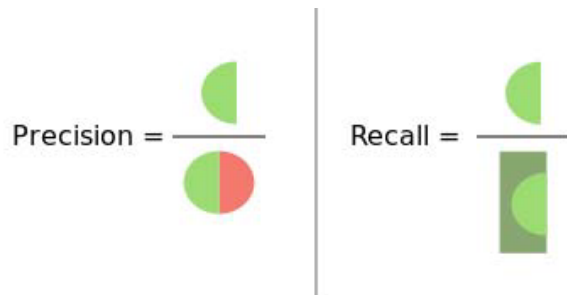
Model Evaluation: Klassifizierung

Metrics:

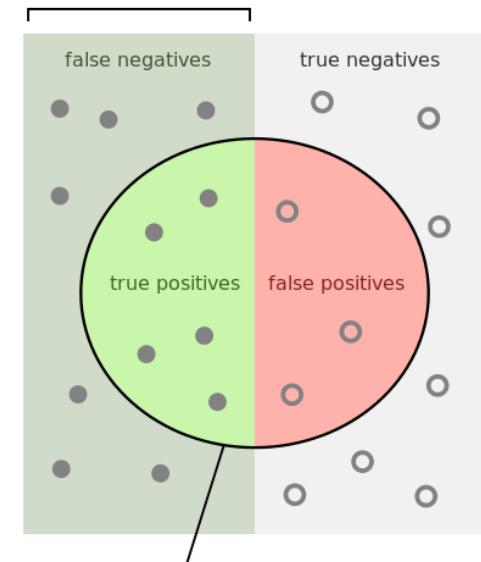
Konfusion Matrix	Predicted Class		
		Yes	No
Actual Class	Yes	True Positives (<i>tp</i>)	False Negatives (<i>fn</i>)
	No	False Positives (<i>fp</i>)	True Negatives (<i>tn</i>)

Model Evaluation: Klassifizierung

- $Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$ (am meisten verwendet)
- $Precision = \frac{tp}{tp+fp}$
- $Recall = \frac{tp}{tp+fn}$
- $Specificity = \frac{tn}{tn+fp}$



Alle Positives



Klassifiziert als Positive

Source: Walber

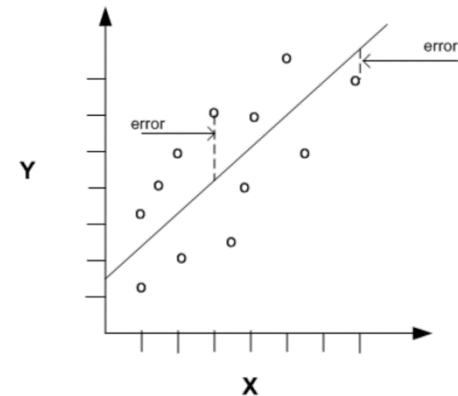
Model Evaluation: Regression

RMSE: Root-mean-square error 

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

MAPE: Median absolute percentage

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$



2. Python Packages

Numpy: ermöglicht eine einfache Handhabung von Vektoren, Matrizen oder generell großen mehrdimensionalen Arrays

Pandas: Basic Anwendung für Data Konvertierung und  Tabularische Darstellung

Scikit-learn: Python Package für klassische Machine Learning Algorithmen


Matplotlib/Seaborn: **graphische Darstellung** der Trainingsprozesse und –ergebnisse








DemoCode in GitLab

- GitLab Repo: <https://github.com/MeilingShi-iib/conda>
- Binder Adresse:
<https://gke.mybinder.org/v2/gh/MeilingShi-iib/conda/master>

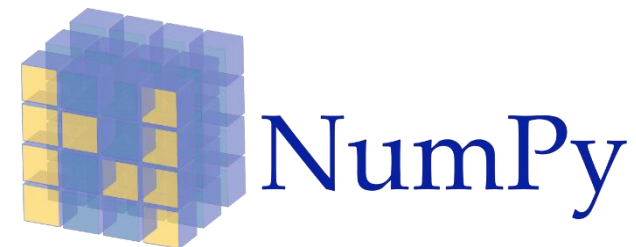
Branch: master ▼ New pull request

This branch is 11 commits ahead of binder-examples:master.

 MeilingShi-iib Add files via upload

 Concrete	Add files via upload
 Rain	Add files via upload
 basicTutorial	Add files via upload
 LICENSE	Create LICENSE
 README.md	Update README.md
 environment.yml	Update environment.yml
 index.ipynb	updating to new syntax

- Hauptobjekt :
 - Narray: das Homogene multidimensionale Array
- Wichtige Attribute:
 - ndarray.ndim: die Anzahl der Achsen (Abmessungen) des Arrays
 - ndarray.shape: Die Dimension des Arrays
 - ndarray.size: die Gesamtzahl der Elemente des Arrays
 - ndarray.dtype: beschreibt den Typ der Elemente in dem Array. Man kann dtype's mit Standard-Python-Typen erstellen oder spezifizieren.



Numpy Beispiel



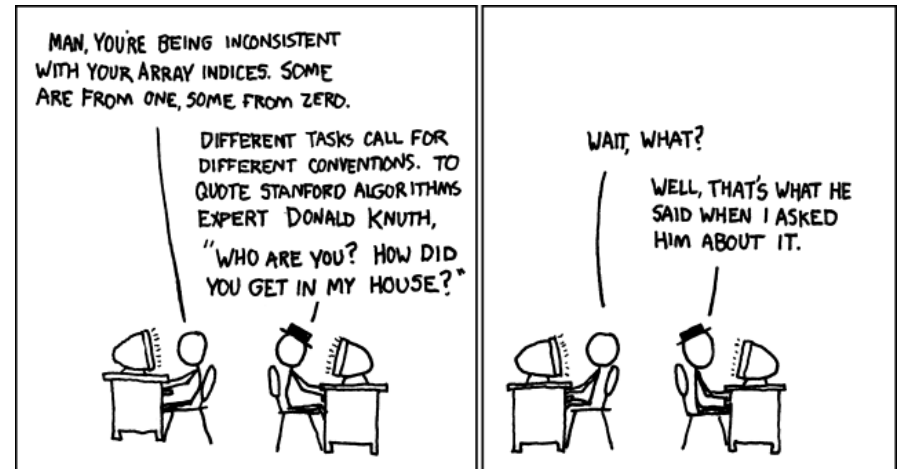
TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
import numpy as np
a = np.array([[1,2,3,4], [5,6,7,8]])
print( "Type:", type(a), "\n", " a: ", a, " shape: ",
      a.shape, " Dimension: ", a.ndim, " Datatype: ",
      a.dtype.name, "Data Size: ", a.size)
```

Type: <class 'numpy.ndarray'>

a: [[1 2 3 4]

[5 6 7 8]] shape: (2, 4) Dimension: 2 Datatype: int32 Data Size: 8



<https://xkcd.com/163/>

- Arithmetische Operatoren auf Arrays wenden **elementweise** an.
Ein neues Array wird erstellt und mit dem Ergebnis gefüllt.

```
a = np.array( [20, 30, 40, 50] )  
b = np.arange( 4 )  
print (b)  
c = a-b  
print (c)  
print (b**2)  
print (a<35)
```

```
[0 1 2 3]  
[20 29 38 47]  
[0 1 4 9]  
[ True  True False False]
```

Achtung: „*****“ wird elementweise
in Numpy Arrays gearbeitet

Das Matrixprodukt kann mit:

- **@**-Operator (in Python
>=3,5) oder
- „**dot**“ funktion

```
A = np.array( [[1,1],  
               [0,1]] )  
B = np.array( [[2,0],  
               [3,4]] )  
print ( "A*B = ", A*B )  
print ( "A@B = ", A@B )  
print ( "A.dot(B) = ", A.dot(B) )
```

```
A*B =  [[2 0]  
        [0 4]]  
A@B =  [[5 4]  
        [3 4]]  
A.dot(B) =  [[5 4]  
             [3 4]]
```

Ein Array hat eine „Shape“, die durch die Anzahl der Elemente entlang jeder Achse gegeben ist

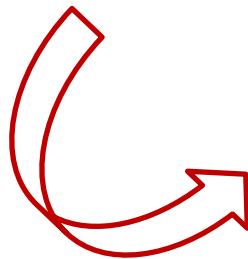
- `Narray.shape`: Die Dimension des Arrays
- `Narray.ravel()`: gibt das Array zurück, abgeflacht
- `Narray.reshape()` & `Narray.resize()`: Array Dimension ändern. Die Funktion `reshape()` gibt ihr Argument mit einer modifizierten Form zurück, während die Methode `ndarray.resize()` das Array selbst modifiziert
- `Narray.T`: Array tranponieren

Numpy Shape Manipulation Beispiel



TECHNISCHE
UNIVERSITÄT
DARMSTADT

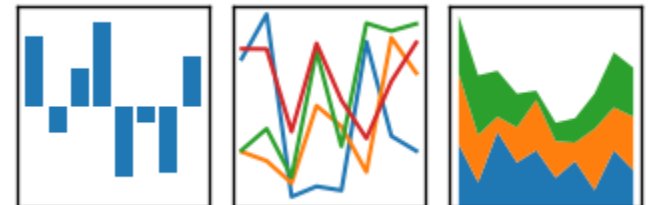
```
a = np.array([[ 2.,  8.,  0.,  6.],
              [ 4.,  5.,  1.,  1.],
              [ 8.,  9.,  3.,  6.]])
print ("a Shape: ", a.shape)
print("a revald: ",a.ravel())
print("a reshape to 6,2: ", a.reshape(6,2))
print ("a shape after reshape: ", a.shape)
print ("a transposed: ", a.T)
print ("a transposed shape: ", a.T.shape)
a.resize(6,2)
print ("a shape after resize: ", a.shape)
```



```
a Shape: (3, 4)
a revald: [2. 8. 0. 6. 4. 5. 1. 1. 8. 9. 3. 6.]
a reshape to 6,2: [[2. 8.]
 [0. 6.]
 [4. 5.]
 [1. 1.]
 [8. 9.]
 [3. 6.]]
a shape after reshape: (3, 4)
a transposed: [[2. 4. 8.]
 [8. 5. 9.]
 [0. 1. 3.]
 [6. 1. 6.]]
a transposed shape: (4, 3)
a shape after resize: (6, 2)
```

- Pandas ist auf NumPy aufgebaut.
- Die Hilfsmittel für die Verwaltung von Daten und deren Analyse.
- Insbesondere enthält Pandas Datenstrukturen und Operatoren für den Zugriff auf numerische Tabellen und Zeitreihen.
- Primären Datenstrukturen:
 - Series (1-dimensional)
 - DataFrame (2-dimensional)

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$


- Series ist ein eindimensional beschriftetes Array, das jeden Datentyp (ganze Zahlen, Zeichenketten, Gleitkommazahlen, Python-Objekte usw.) aufnehmen kann. Die Achsenbeschriftungen werden als Index bezeichnet.
- Aufruf: `Pandas.series(data, index = index)`
- Hier können „**data**“ viele verschiedene Datentypen sein:
 - Python Dictionary
 - Numpy ndarray
 - Skalar (wie 5)

Series Beispiel

```
s = pd.Series(np.random.randn(5),  
              index = ['a','b','c','d','e'] )  
print ("s with giving Index: \n ", s)  
print ("s: ", s.index)  
print ("s without giving index: \n",  
        pd.Series(np.random.randn(5)))
```


```
s with giving Index:  
a    -1.529503  
b     0.394126  
c     0.307152  
d    -0.478542  
e    -0.021849  
dtype: float64  
s:  Index(['a', 'b', 'c', 'd', 'e'], dtype='object')  
s without giving index:  
0     0.559753  
1     0.696184  
2    -1.803624  
3     0.031206  
4     0.013759  
dtype: float64
```

Series instanziiert aus
narray

Series instanziiert aus dict

```
d = {'b': 1, 'a': 0, 'c': 2}  
pd.Series(d)  
  
b    1  
a    0  
c    2  
dtype: int64
```

Eine 2-dimensional beschriftete Datenstruktur mit Spalten unterschiedlicher Art. Wie die Serie akzeptiert auch DataFrame viele verschiedene Arten von Eingaben:

- Diktat  von 1D ndarrays, Listen, Dikaten oder Serien
- 2-D numpy.ndarray.
- Strukturiertes oder aufgezeichnetes ndarray
- Pandas.Series
- Ein andere DataFrame
- Aus .csv Datei

Pandas: DataFrame Beispiel

```
import pandas as pd
d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
     'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
print ("Dictionary d: \n ", d)
df = pd.DataFrame(d)
print("DataFrame df: \n", df)
```

Dictionary d:

```
  {'one': a    1.0
b     2.0
c     3.0
dtype: float64, 'two': a    1.0
b     2.0
c     3.0
d     4.0
dtype: float64}
```

DataFrame df:

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0



Quelle: <https://realpython.com/python-pandas-tricks/>

Pandas: DataFrame Beispiel

```
df = pd.read_csv('./weatherAUS.csv')
print('Size of weather data frame is :', df.shape)
#Let us see how our data looks like!
df[0:5]
```

DataFrame instanziiert aus .csv

Size of weather data frame is : (142193, 24)

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	Wi
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	

5 rows × 24 columns

DataFrame: Spalte abrufen, hinzufügen, löschen

Ein DataFrame kann semantisch wie ein „dict“ von gleich indizierten Objekten der Serie behandeln. Das Abrufen, Setzen und Löschen von Spalten funktioniert mit der gleichen Syntax wie bei den analogen „dict“.

```
df['one']
```

Spalten abrufen

```
a    1.0  
b    2.0  
c    3.0  
d    NaN  
Name: one, dtype: float64
```

Spalten-Name
abrufen mit
Spalte-Index

```
df.columns[0:3]
```

```
Index(['one', 'three', 'foo'], dtype='object')
```

```
df['three'] = df['one'] * df['two']  
df
```

Spalte hinzufügen

	one	two	three
a	1.0	1.0	1.0
b	2.0	2.0	4.0
c	3.0	3.0	9.0
d	NaN	4.0	NaN

```
#Columns can be deleted or  
#popped like with a dict:  
del df['two']  
df
```

Spalte löschen

	one	three
a	1.0	1.0
b	2.0	4.0
c	3.0	9.0
d	NaN	NaN

Zeile und Spalte Indexing



Operation	Syntax	Result
Select column	<code>df[col]</code>	Series
Select row by label	<code>df.loc[label]</code>	Series
Select row by integer location	<code>df.iloc[loc]</code>	Series
Slice rows	<code>df[5:10]</code>	DataFrame
Select rows by boolean vector	<code>df[bool_vec]</code>	DataFrame

```
print (df)
print (" Row b : ")
df.loc['b']
```

```
   one  three  foo
a  1.0    1.0  bar
b  2.0    4.0  bar
c  3.0    9.0  bar
d  NaN    NaN  bar
Row b :

one      2
three    4
foo      bar
Name: b, dtype: object
```

```
df.iloc[2]
```

```
one      3
three    9
foo      bar
Name: c, dtype: object
```

```
df[1:3]
```

```
   one  three  foo
b  2.0    4.0  bar
c  3.0    9.0  bar
```

Spalte umgestalten

DataFrame[[*neu umgestellte Spalte Name*]]

```
df['three'] = df['one'] * df['two']  
df
```

	one	two	three
a	1.0	1.0	1.0
b	2.0	2.0	4.0
c	3.0	3.0	9.0
d	NaN	4.0	NaN



```
df[['foo', 'three']]
```

	foo	three
a	bar	1.0
b	bar	4.0
c	bar	9.0
d	bar	NaN

Andere hilfreiche Funktionen

`DataFrame.info()`: Gibt Spalte Information und Datengröße zurück

`DataFrame.dropna(how='any')`: löschen alle Zeile die eine Null Wert hat


`DataFrame.replace()`: Werte des Datenrahmens werden dynamisch durch andere Werte ersetzt.

`DataFrame.T`: Transponing

`DataFrame.describe()`: Generieren deskriptive Statistiken (minimale und maximale Werte, zentrale Tendenz, Verteilung, NaN-Werten zusammenfassen)

```
dataSet = pd.read_csv("Concrete_Data")
print(dataSet.info())
# Rename the columns
dataSet.columns = ['cement', 'slag',
```

```
<class 'pandas.core.frame.DataFrame':
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
cement                1030 non-null fl
slag                  1030 non-null fl
flyash                1030 non-null fl
water                 1030 non-null fl
superplasticizer      1030 non-null fl
coarseaggregate       1030 non-null fl
fineaggregate         1030 non-null fl
age                   1030 non-null in
csMPa                  1030 non-null fl
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
None
```

- Einfaches Handling von **fehlenden Daten** (dargestellt als NaN)
- Spalten können eingefügt und gelöscht werden.
- Robuste IO-Tools zum Laden von Daten aus **Flat Files (CSV und delimited), Excel-Dateien** etc.
- Intelligentes Label-basiertes Slicing, ausgefallenes Indexing und  Subsetting von großen Datensätzen
- Hierarchische Beschriftung der Achsen (mehrere Beschriftungen pro Tick möglich)
- Vereinfachung der Konvertierung von zerklüfteten, unterschiedlich indizierten Daten in anderen Python- und NumPy-Datenstrukturen in DataFrame-Objekte.

Eine freie Software-Bibliothek zum maschinellen Lernen in Python.

Algorithmen und Modules:

- Classification: SVM, nearest neighbors, random forest ...
- Regression: SVR, ridge regression, Lasso...
- Clustering: K-Means, spectral clustering, mean-shift, ...
- Dimensionality reduction: PCA, feature selection, non-negative matrix factorization
- Model Selection: Grid search, cross validation, metrics
- Preprocessing: Preprocessing, feature extraction



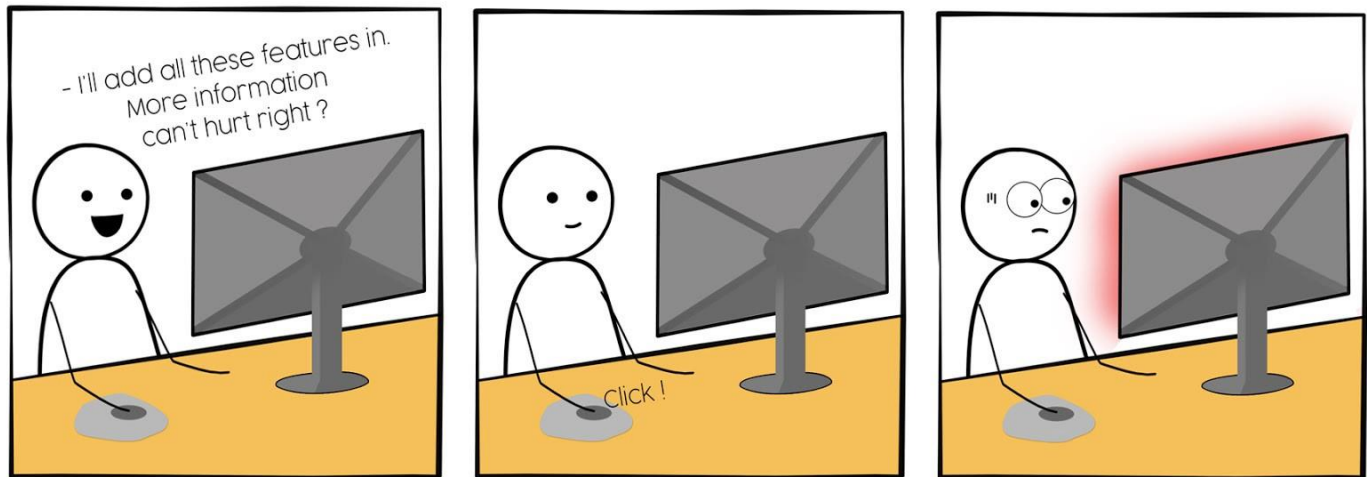
Standardisierung ist eine Transformation, die die **Daten zentriert**, indem sie den **Mittelwert jedes Merkmals entfernt** und dann **skaliert**, indem sie **Merkmale (Features) durch ihre Standardabweichung teilt**. Nach der Standardisierung der Daten ist der **Mittelwert = 0** und die **Standardabweichung = 1**.

Funktionen:

- `preprocessing.MinMaxScaler()`: $x_scaled = (x - u) / s$
- `Preprocessing.StandardScaler()`: $x_scaled = (x - \min(x)) / (\max(x) - \min(x))$
- `Preprocessing.MaxAbsScaler()`: $x_scaled = x / \max(\text{abs}(x))$
- `Preprocessing.RobustScaler()`

Feature Selection (Auswahl)

Die Klassen im Modul `sklearn.feature_selection` können zur **Merkmalsauswahl/Dimensionalitätsreduzierung** bei Stichprobensätzen verwendet werden, entweder um die **Genauigkeitswerte der Schätzer** zu verbessern oder um ihre **Leistung bei sehr hochdimensionalen Datensätzen** zu steigern.




Quelle: <https://deezer.io/deezer-kaggledays-paris-b686362d00fb>

Feature Selection (Auswahl)

Removing features with low variance:

`sklearn.feature_selection.VarianceThreshold(threshold=0.0)`

Univariate feature selection  (1/2): Untersucht jedes Merkmal einzeln mit univariaten statistischen Tests, um **die Stärke der Beziehung des „Features“ zur Target-Variablen zu bestimmen.**

- einfach anzuwenden
- eignen sich gut für ein besseres Verständnis von Daten
- nicht unbedingt für die Optimierung des „Feature-Sets“ zur besseren Verallgemeinerung.

(2/2):

Funktionen:

- SelectKBest: entfernt alle bis auf die ***k*** besten „Features“
- SelectPercentile: entfernt alle bis auf einen benutzerdefinierten Prozentsatz der höchsten Punktzahl der Features.
- GenericUnivariateSelect: Führt eine univariate Featureauswahl mit einer konfigurierbaren Strategie durch. Dies ermöglicht die **Auswahl der besten univariaten Auswahlstrategie** mit dem Hyperparameter-Suchschätzer.
- Univariaten statistischen Tests
 - Für Regression: `f_regression`, `mutual_info_regression`
 - Für die Klassifizierung: `chi2`, `f_classif`, `mutual_info_classif`

Trainingsdaten und Testdaten splitten

Splitter Funktionen:

[model_selection.check_cv](#)([cv, y, classifier])

Input checker utility for building a cross-validator

[model_selection.train_test_split](#)(*arrays, ...)

Split arrays or matrices into random train and test subsets



Validierungsdaten splitten

Splitter Klasse:

<code>model_selection.GroupKFold ([n_splits])</code>	K-fold iterator variant with non-overlapping groups.
<code>model_selection.GroupShuffleSplit ([...])</code>	Shuffle-Group(s)-Out cross-validation iterator
<code>model_selection.KFold ([n_splits, shuffle, ...])</code>	K-Folds cross-validator
<code>model_selection.LeaveOneGroupOut</code>	Leave One Group Out cross-validator
<code>model_selection.LeavePGroupsOut (n_groups)</code>	Leave P Group(s) Out cross-validator
<code>model_selection.LeaveOneOut</code>	Leave-One-Out cross-validator
<code>model_selection.LeavePOut (p)</code>	Leave-P-Out cross-validator
<code>model_selection.PredefinedSplit (test_fold)</code>	Predefined split cross-validator
<code>model_selection.RepeatedKFold ([n_splits, ...])</code>	Repeated K-Fold cross validator.
<code>model_selection.RepeatedStratifiedKFold ([...])</code>	Repeated Stratified K-Fold cross validator.
<code>model_selection.ShuffleSplit ([n_splits, ...])</code>	Random permutation cross-validator
<code>model_selection.StratifiedKFold ([n_splits, ...])</code>	Stratified K-Folds cross-validator
<code>model_selection.StratifiedShuffleSplit ([...])</code>	Stratified ShuffleSplit cross-validator
<code>model_selection.TimeSeriesSplit ([n_splits, ...])</code>	Time Series cross-validator

Trainingsmodell: Klassifizierung

`sklearn.linear_model.LogisticRegression()`

`sklearn.ensemble.RandomForestClassifier()`

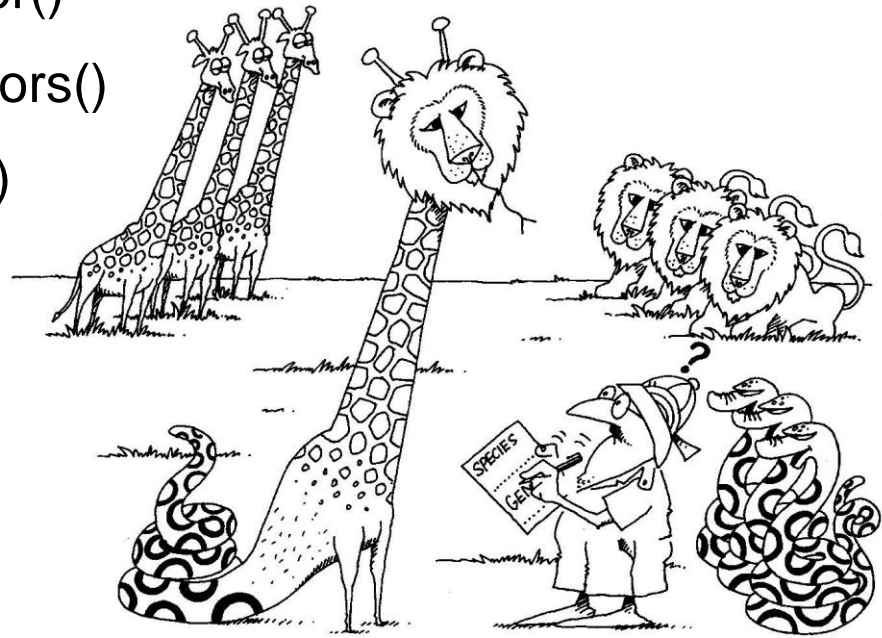
`Sklearn.tree.DecisionTreeClassifier()`

`sklearn.neighbors.Nearest Neighbors()`

`sklearn.naive_bayes.BernoulliNB()`

`sklearn.svm.SVC()`

...




Quelle: <http://www.wiki-hilfe.de/wiki/Kategorisierung>

Trainingsmodell: Regression

- `sklearn.linear_model.LinearRegression()`,
`sklearn.linear_model.Ridge()`,
`sklearn.linear_model.Lasso()`,
- `sklearn.neighbors.KNeighborsRegressor()`,
- `sklearn.tree.DecisionTreeRegressor()`,
- `sklearn.ensemble.RandomForestRegressor()`,
`sklearn.ensemble.GradientBoostingRegressor()`,
`sklearn.ensemble.AdaBoostRegressor()`

Modell validieren

[model_selection.cross_validate](#)
(estimator, X)

Evaluate metric(s) by cross-validation and also record fit/score times. 

[model_selection.cross_val_predict](#)
(estimator, X)

Generate cross-validated estimates for each input data point 


[model_selection.cross_val_score](#)
(estimator, X)

Evaluate a score by cross-validation

[model_selection.learning_curve](#)
(estimator, X, y)

Learning curve.

[model_selection.permutation_test_score](#) (...)

Evaluate the significance of a cross-validated score with permutations 

[model_selection.validation_curve](#)
(estimator, ...)

Validation curve.

Modell evaluieren: Regression



TECHNISCHE
UNIVERSITÄT
DARMSTADT

'explained_variance'	<u>metrics.explained_variance_score</u>
'max_error'	<u>metrics.max_error</u>
'neg_mean_absolute_error'	<u>metrics.mean_absolute_error</u>
'neg_mean_squared_error'	<u>metrics.mean_squared_error</u>
'neg_median_absolute_error'	<u>metrics.median_absolute_error</u>
'r2'	<u>metrics.r2_score</u>

At the beginning of every evaluation

I know our
project works



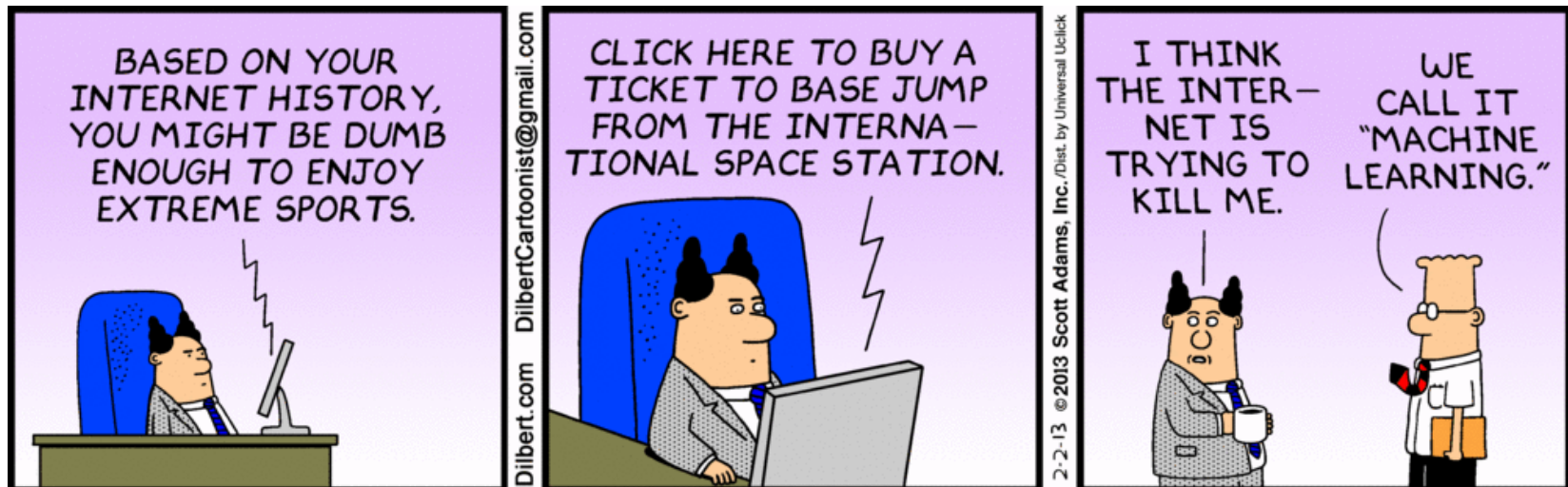
No,
you don't



Quelle: freshspectrum

Modell evaluieren: Klassifizierung

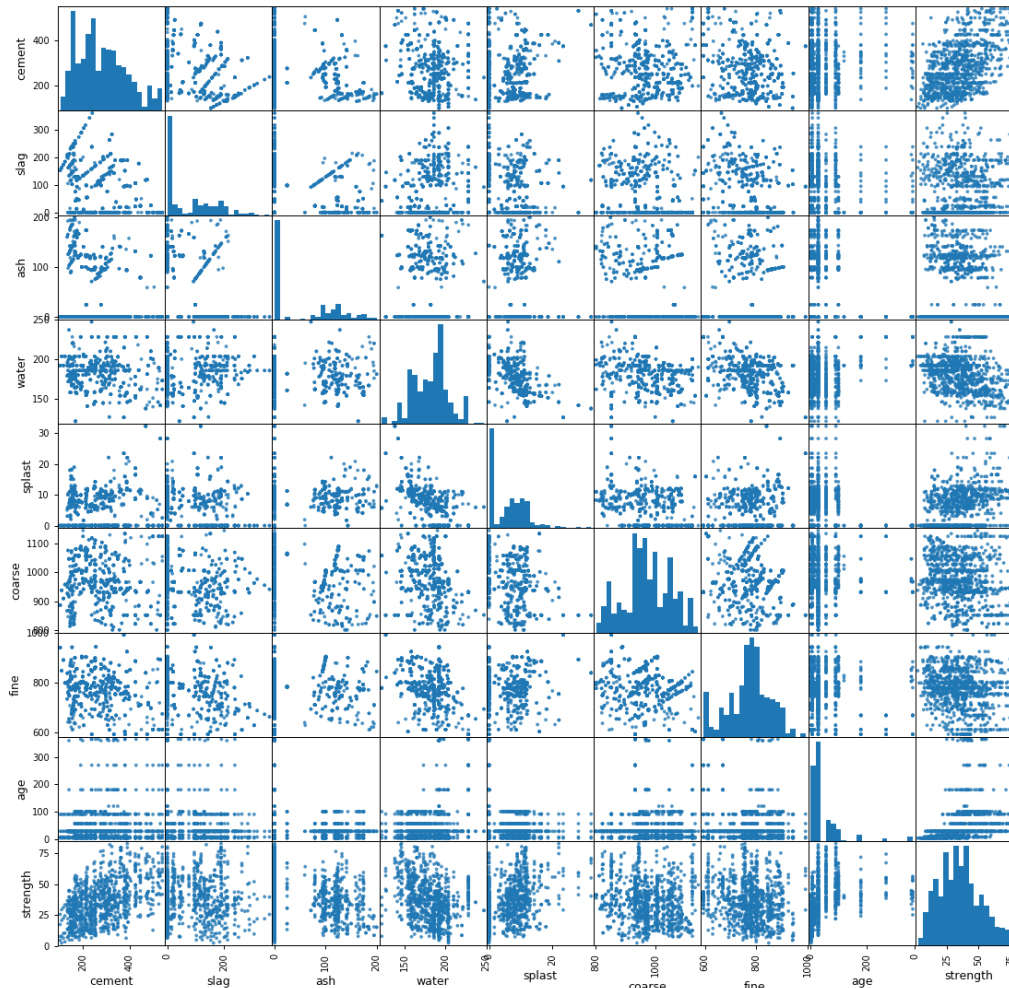
- `sklearn.metrics.accuracy_score`
- In der Multilabel-Klassifizierung berechnet diese Funktion **die Genauigkeit der Teilmengen**: Der für ein Sample vorhergesagte Output muss genau mit dem entsprechenden „Targets“ übereinstimmen.



Um mathematische Darstellungen aller Art anzufertigen
(Siehe Demo)

matplotlib

Demo











DemoCode in GitLab

- GitLab Repo: <https://github.com/MeilingShi-iib/conda>
- Binder Adresse:
<https://gke.mybinder.org/v2/gh/MeilingShi-iib/conda/master>

Branch: master ▼ New pull request

This branch is 11 commits ahead of binder-examples:master.

 MeilingShi-iib Add files via upload

 Concrete	Add files via upload
 Rain	Add files via upload
 basicTutorial	Add files via upload
 LICENSE	Create LICENSE
 README.md	Update README.md
 environment.yml	Update environment.yml
 index.ipynb	updating to new syntax

Aufgabestellung

Aufgabe 1: Prognose der Betonfestigkeit durch gegebene Materialien



Quelle: <http://www.desmedtbeton.be/producten/betonbalken>

Aufgabe 2: Wetter (regen oder nicht) für den nächsten Tag vorhersagen



Quelle: <https://www.wetter.com/videos/wetter-weltweit/>

Sprechstunde



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Do. 11.07.2019 13:00-16:00

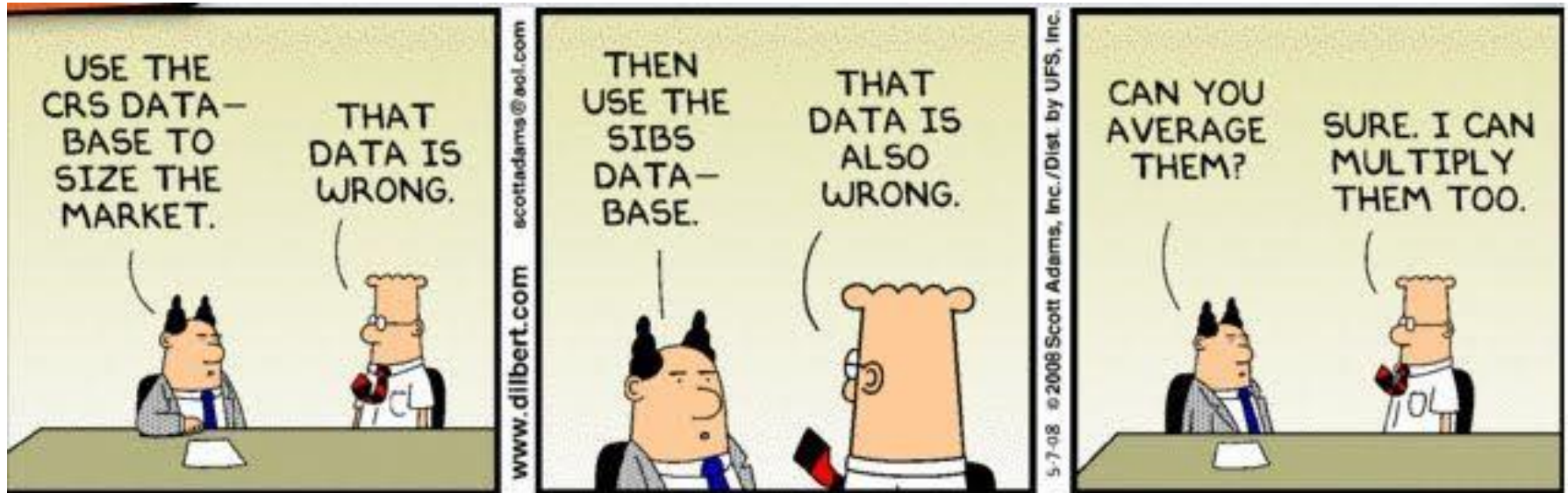
Di. 17.07.2019 14:00-16:00

Fr. 19.07.2019 14:00-16:00

Viel Erfolg!



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Frage zur Klausur?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- <https://www-users.cs.umn.edu/~kumar001/dmbook/index.php>
- <https://www.numpy.org/devdocs/user/quickstart.html>
- https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html
- [Scikit-learn: Machine Learning in Python](#), Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
- <https://matplotlib.org/>
- <https://www.kaggle.com/>
- <https://seaborn.pydata.org/>
- <https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcb9>
- <https://medium.com/@nikolayryabykh/splunk-s01e02-the-one-with-machine-learning-toolkit-e808007736de>
- <https://www.kaggle.com/>

