h-da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Einführung in die künstliche Intelligenz

EKI03 – ML Regression

Prof. Dr. A. del Pino

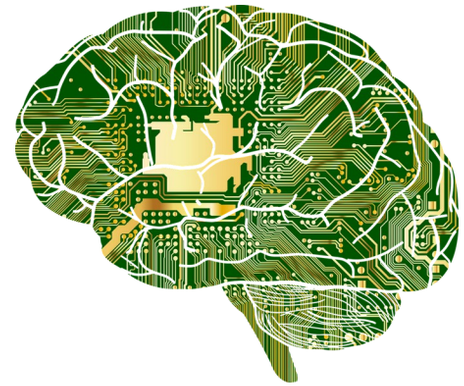


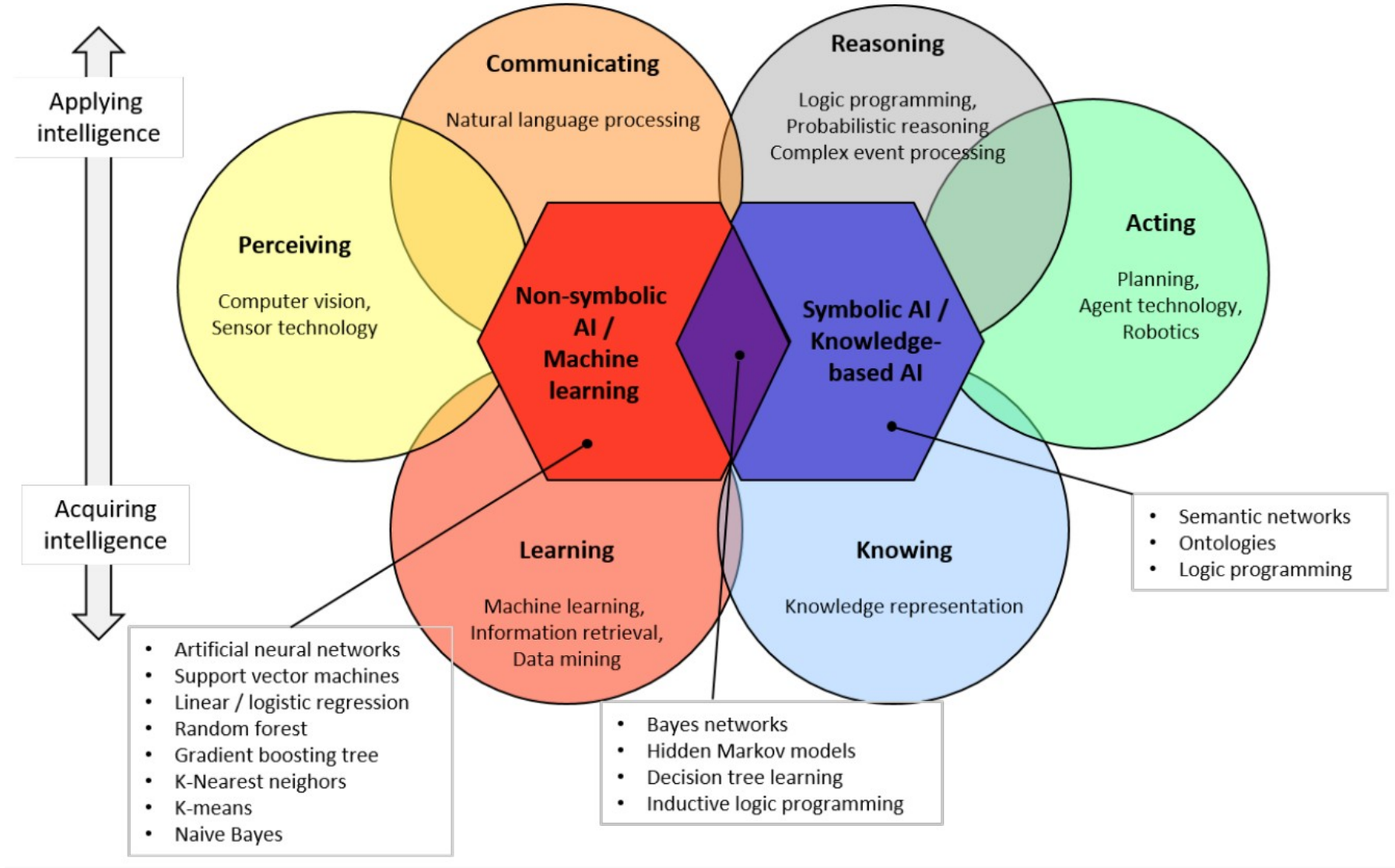
Bild: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Anatomy-1751201_1280.png

Dieser Foliensatz basiert teilweise auf dem “Introduction to Artificial Intelligence”-Kursmaterial von Prof. Dr. B. Humm

Aufbau

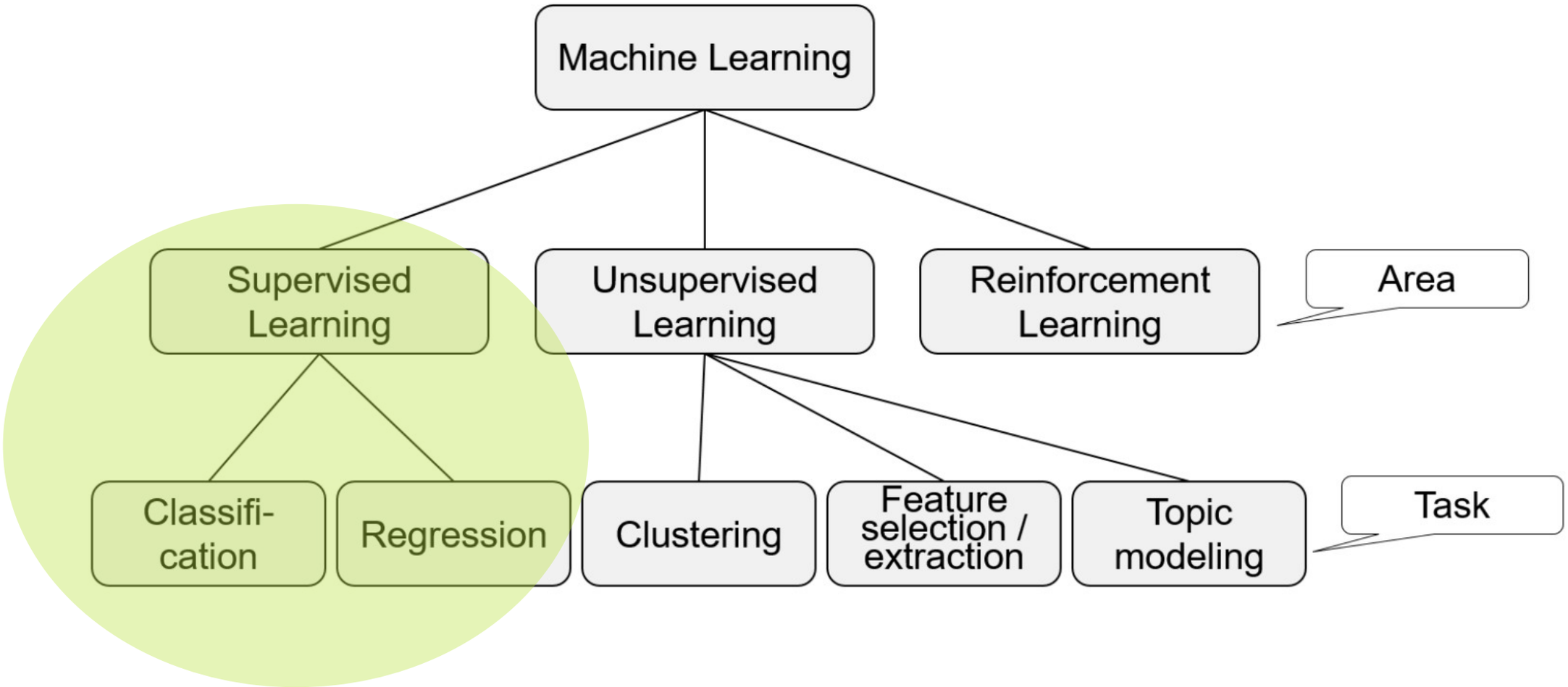
- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

Die KI-Landkarte



Bildquelle: B. Humm “Applied Artificial Intelligence”, S. 4

Kategorien von ML Aufgaben



Bildquelle: B. Humm “Applied Artificial Intelligence”, S. 15

Aufgaben des Überwachten Lernens (supervised learning)

Klassifikation

- Gegeben: Datensätze als Eingabe für das Training, welche in zwei oder mehrere Klassen (“Kategorien”) eingeteilt sind.
- Ziel: Ein Modell erzeugen, welches neue, bisher unbekannte Datensätze klassifizieren kann.
- Beispiele: Spam-Filter, Fraud-Detection,…

Regression

- Gegeben: Datensätze als Eingabe für das Training, welche einen numerischen Ausgabewert besitzen.
- Ziel: Ein Modell erzeugen, welches für neue, bisher unbekannte Datensätze den Ausgabewert vorhersagen kann.
- Beispiel: Aktienkurs-Vorhersage

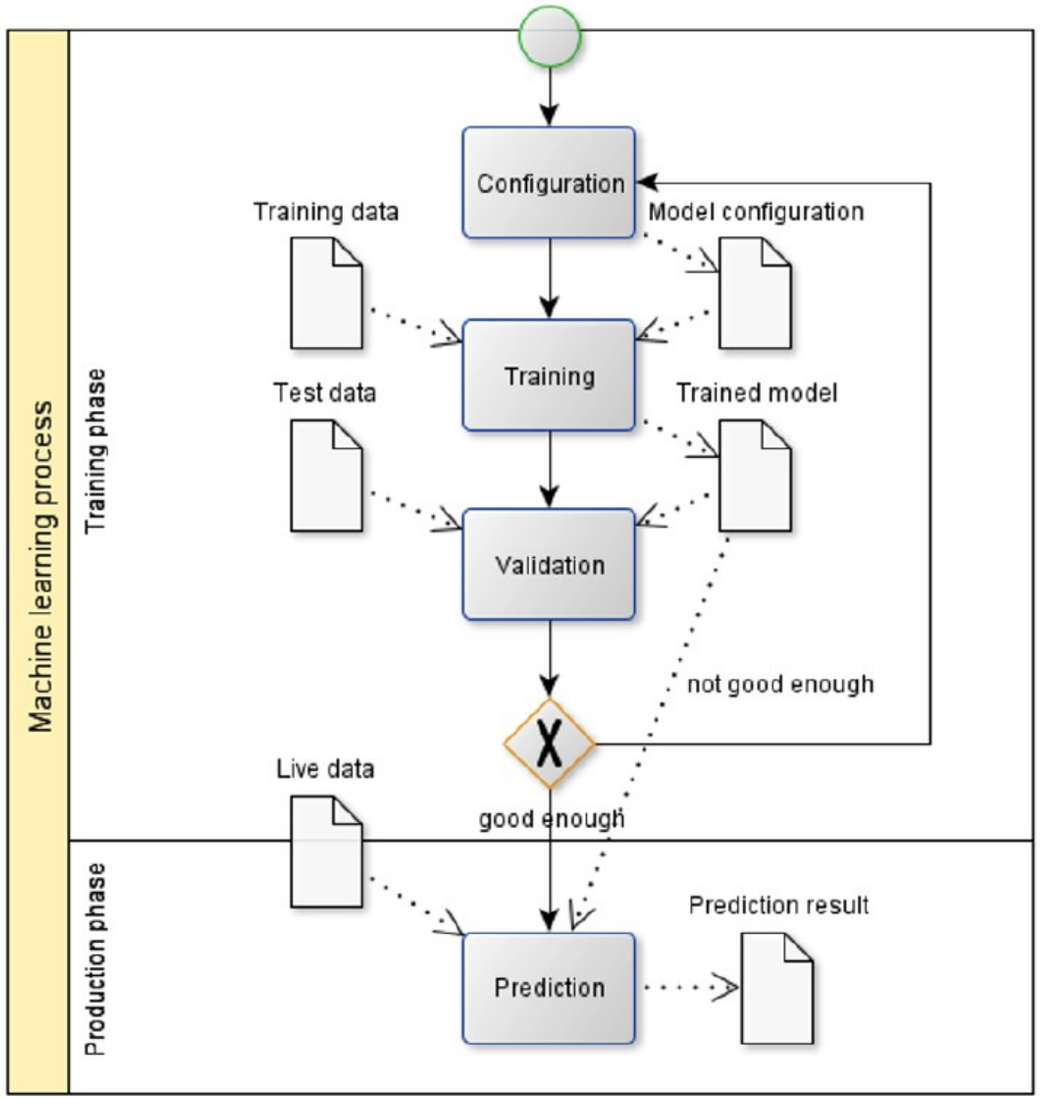
Aufbau

- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

Woher weiss man, ob das ML Modell gut genug ist?

Bildquelle: https://cdn.pixabay.com/photo/2015/11/03/08/56/question-mark-1019820_960_720.jpg

Der ML Entwicklungsprozess: Training und Validierung vor dem produktiven Einsatz



Bildquelle: B. Humm “Applied Artificial Intelligence”, S. 28

Validierung der Klassifikation: Konfusionsmatrix (confusion matrix)

Beispiel: Klassifikation von medizinischen Daten, ob eine bestimmte Krankheit vorliegt oder nicht.

	Krankheit liegt tatsächlich vor (krank)	Krankheit liegt nicht vor (gesund)
Vorhersage der Klassifikation: KRANK	TP True positive	FP False positive Type I Error
Vorhersage der Klassifikation: GESUND	FN False negative Type II Error	TN True negative

Treffergenauigkeit (accuracy)

Definition: $$Treffergenauigkeit = \frac{Anzahl\ der\ korrekten\ Vorhersagen}{Gesamtanzahl\ der\ Vorhersagen}$$

Bei binärer Klassifikation (2 Klassen) ergibt sich: $$Treffergenauigkeit = \frac{TP + TN}{TP + TN + FP + FN}$$

Beispiel: Klassifikation von Tumorbildern als bösartig (positiv) oder gutartig (negativ)

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

$$Treffergenauigkeit = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1 + 90}{1 + 90 + 1 + 8} = 0.91$$

Quelle: https://developers.google.com/machine-learning/crash-course/classification/accuracy

Positiver Vorhersagewert (precision, positive predictive value, PPV)

Anteil der Personen, die als krank klassifiziert wurden und auch tatsächlich krank sind.

$$precision = \frac{TP}{TP + FP}$$

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

Im Beispiel:

$$precision = \frac{TP}{TP + FP} = \frac{1}{1+1} = 0.5$$

D.h. nur jede zweite als krank klassifizierte Person ist tatsächlich krank. Konsequenz=?

Trefferquote (recall, true positive rate, TPR)

Mit welcher Wahrscheinlichkeit wird ein Kranker auch erkannt?

$$recall = \frac{TP}{TP + FN}$$

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

Im Beispiel:

$$recall = \frac{TP}{TP + FN} = \frac{1}{1+8} = 1/9 = 0.111$$

D.h. nur jede(r) neunte Kranke wird auch als krank klassifiziert. Konsequenz = ?

F Score

Der F-Score (manchmal auch: F_1 Score) verbindet *precision* und *recall* durch das harmonische Mittel.

$$F\ Score = \frac{2}{(\frac{1}{precision}) + (\frac{1}{recall})} = 2 * \frac{precision * recall}{precision + recall}$$

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

Im Beispiel:

$$F\ Score = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{0.5 * 0.111}{0.5 + 0.111} = 0.181$$

Der niedrige F Score (0.18) reflektiert in diesem Fall die schlechte Qualität des Klassifikators weitaus besser als seine gute Treffergenauigkeit (0.91).

Achtung! Benutzen Sie die Treffergenauigkeit nur bei ausgewogenen Datensätzen

- Tumor-Klassifikations-Beispiel: Treffergenauigkeit = 0.91
- Das sieht auf dem ersten Blick gut aus aber 8 aus 9 Fälle werden falsch klassifiziert. Das ist also sehr schlecht!
- Der F Score (manchmal auch F_1 Score genannt) ist für nicht ausgewogene Datensätze besser geeignet.
- Je nach Anwendung gibt es weitere Scores, siehe auch https://en.wikipedia.org/wiki/Confusion_matrix

$$Negative\ Predictive\ Value\ NPV = \frac{TN}{FN + TN}$$

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

$$specificity = \frac{TN}{FP + TN}$$

TP=1 True positive	FP=1 False positive
FN=8 False negative	TN=90 True negative

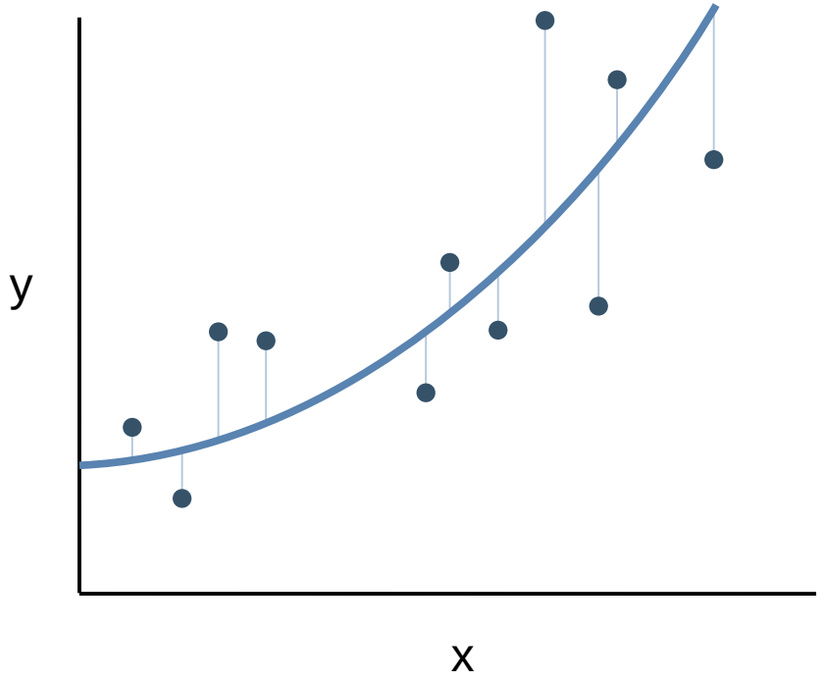
Validierung der Regression: MAE, MSE, RMSE

- Validierung der Regression mit Hilfe der vorhergesagten Werte $\hat{f}_i$ und den tatsächlichen (erwarteten) Werten y_i

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |\hat{f}_i - y_i|$$ **M**ean **A**bsolute **E**rror

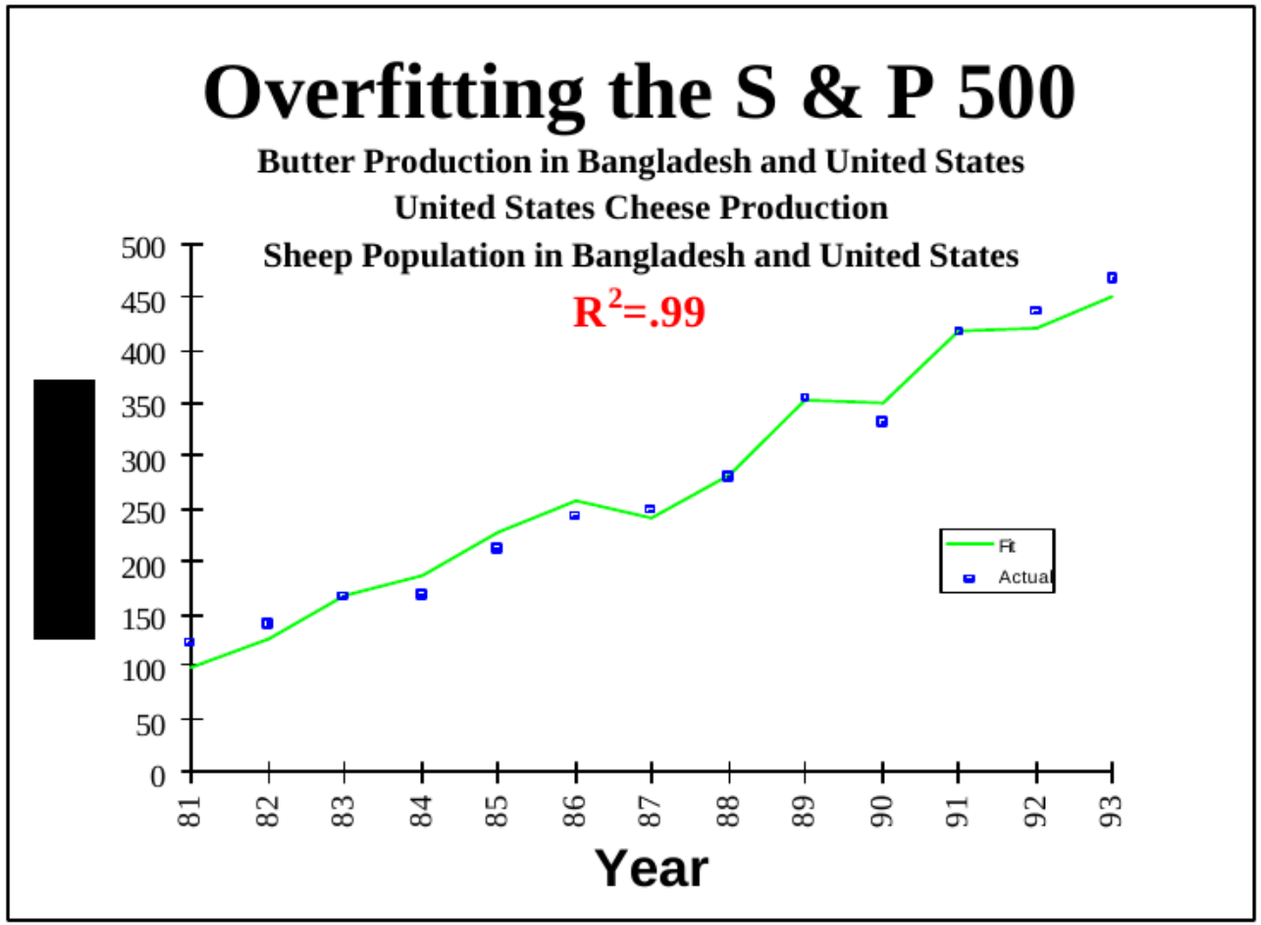
$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (\hat{f}_i - y_i)^2$$ **M**ean **S**quared **E**rror

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (\hat{f}_i - y_i)^2}$$ **R**ooted **MSE**



Validierung der Regression

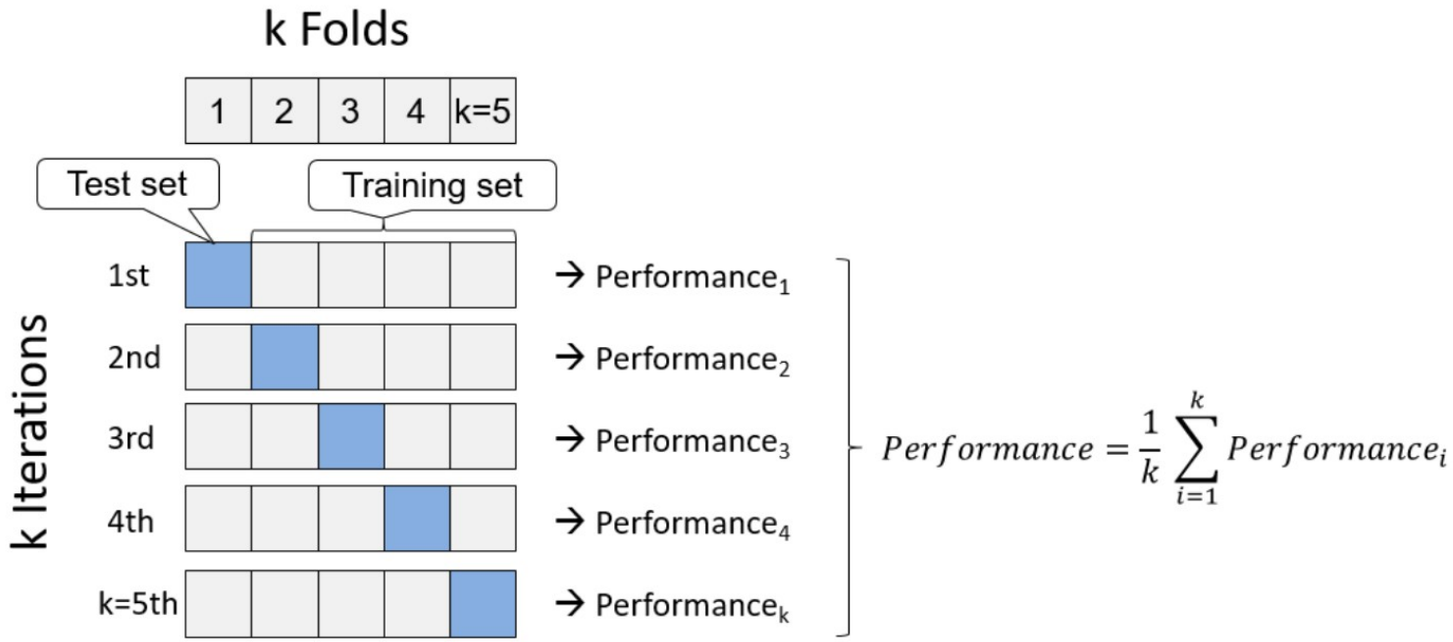
Achtung! Regressionen müssen einen Sinn ergeben, eine gute Validierung alleine ist nicht ausreichend!



Quelle: https://www.researchgate.net/publication/247907373_Stupid_Data_Miner_Tricks_Overfitting_the_SP_500

K-fache Kreuzvalidierung (k fold cross validation)

1. Spalte den Datensatz in *k* Teildatensätze auf, z.B. *k*=5
2. Führe das Training mit *k*-1 Teildatensätzen durch und benutze 1 Teildatensatz für die Validierung
3. Iteriere *k* mal mit jeweils einem anderen Teildatensatz für die Validierung
4. Berechne das durchschnittliche Ergebnis von allen Iterationen



Bildquelle: B. Humm “Applied Artificial Intelligence”, S. 35

Kreuzvalidierung mit scikit-learn

```
from sklearn.model_selection import cross_val_score


# Evaluiere das Modell
scores = cross_val_score(model, X, y)
print(f'Accuracy: {scores.mean()}')
```

Führt eine Kreuzvalidierung mit dem Modell, dem Datensatz X und den Labels y durch.
Default: accuracy, k=5
Viele andere Optionen möglich

Formatierte Ausgabe: Gebe den mean der resultierenden scores aus

Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Aufbau

- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

Kaggle Wettbewerb zu Regression: TMDB Box Office Prediction

Overview

Description
Evaluation
Timeline
Prizes

We're going to make you an offer you can't refuse: a Kaggle competition!

In a world... where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. But what movies make the most money at the box office? How much does a director matter? Or the budget? For some movies, it's "You had me at 'Hello.'" For others, the trailer falls short of expectations and you think "What we have here is a failure to communicate."

In this competition, you're presented with metadata on over 7,000 past films from The Movie Database to try and predict their overall worldwide box office revenue. Data points provided include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. You can collect other publicly available data to use in your model predictions, but in the spirit of this competition, use only data that would have been available before a movie's release.

Join in, "make our day", and then "you've got to ask yourself one question: 'Do I feel lucky?'"

Quelle: https://www.kaggle.com/c/tmdb-box-office-prediction

Trainingsdaten: Merkmale (features) und Ziel (target)

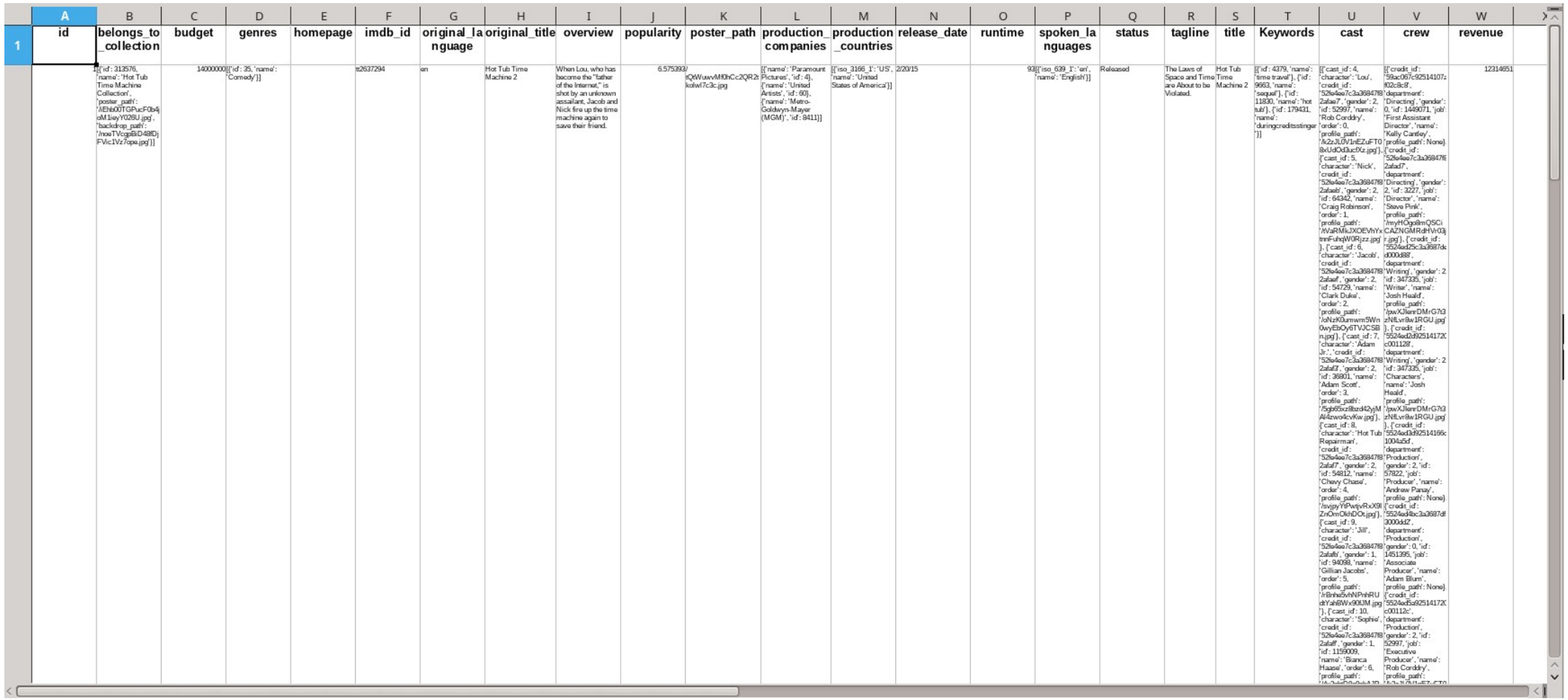
Numerisch

Text

Strukturiert (JSON)

Kategorisch

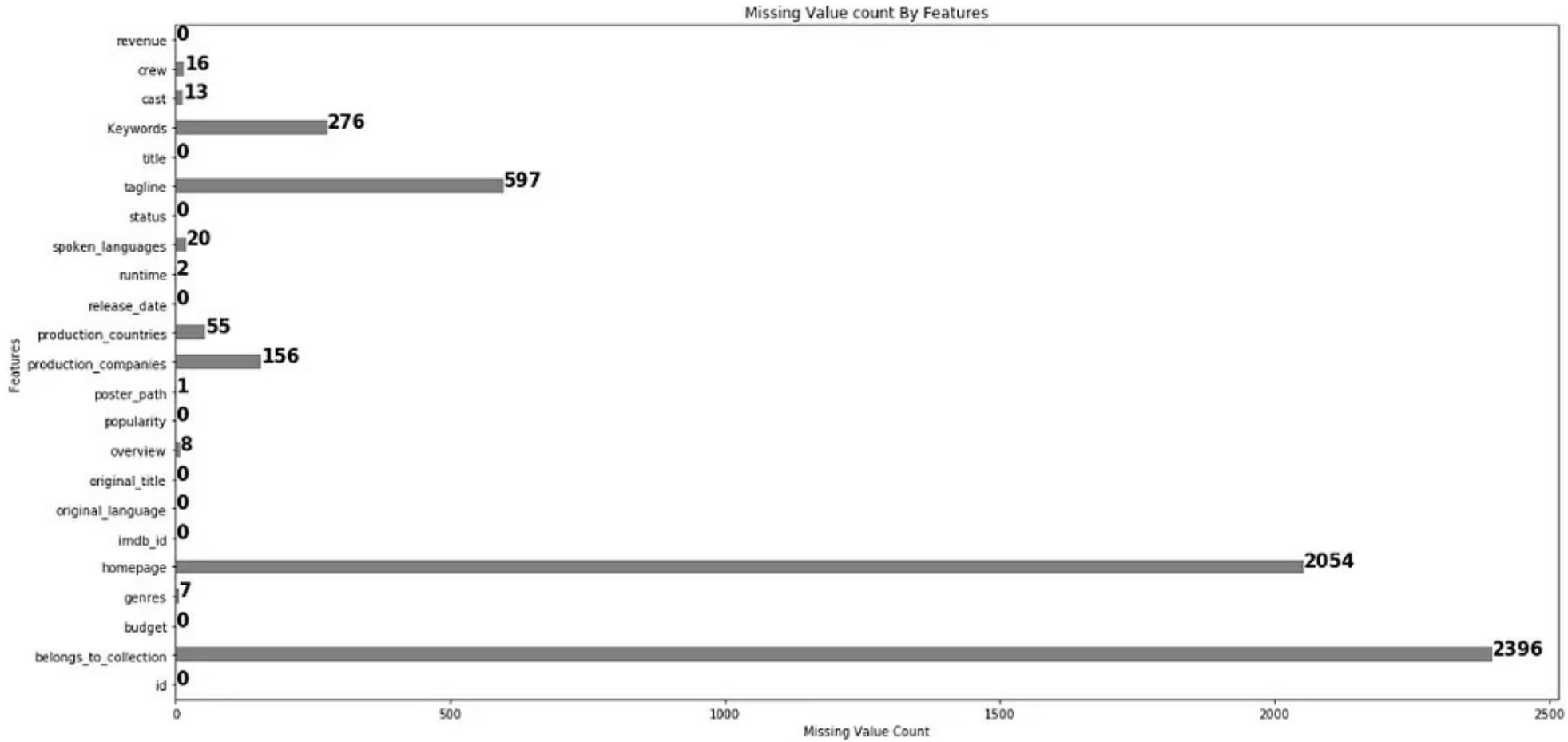
Ziel



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_title	overview	popularity	poster_path	production_companies	production_countries	release_date	runtime	spoken_languages	status	tagline	title	Keywords	cast	crew	revenue
		[{'id': 313576, 'name': 'Hot Tub Time Machine Collection', 'poster_path': '/iEhb00TGPucF0b4oM1kyY026U.jpg', 'backdrop_path': '/noeTVcgpBiD48fDjFVic1Vz7opu.jpg'}]	14000000	[{'id': 35, 'name': 'Comedy'}]		tt2637294	en	Hot Tub Time Machine 2	When Lou, who has become the "father of the Internet," is shot by an unknown assailant, Jacob and Nick fire up the time machine again to save their friend.	6.575393	/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg	[{'name': 'Paramount Pictures', 'id': 4}, {'name': 'United Artists', 'id': 60}, {'name': 'Metro-Goldwyn-Mayer (MGM)', 'id': 8411}]	[{'iso_3166_1': 'US', 'name': 'United States of America'}]	2/20/15		93	[{'iso_639_1': 'en', 'name': 'English'}]	Released	The Laws of Space and Time are About to be Violated.	Hot Tub Time Machine 2	[{'id': 4379, 'name': 'time travel'}, {'id': 9663, 'name': 'sequel'}, {'id': 11830, 'name': 'hot tub'}, {'id': 179431, 'name': 'duringcreditsstinger'}]	[{'cast_id': 4, 'character': 'Lou', 'credit_id': '52fe4ae7c3a368478	[{'credit_id': '59ac067c92514107

Datenquelle: https://www.kaggle.com/competitions/tmdb-box-office-prediction/data

Fehlende Werte



Quelle: https://saicharanars.medium.com/tmdb-box-office-prediction-fb9ac6371603

Format der Einreichung in den Kaggle Wettbewerb

	A	B
1	id	revenue
2	3001	15339775.0916079
3	3002	17437353.8394148
4	3003	12356487.361688
5	3004	14928576.6293748
6	3005	14288547.2801949
7	3006	16727556.0869667
8	3007	16729628.7974298
9	3008	14125483.3840709
10	3009	15064986.9095512
11	3010	12043777.9266763
12	3011	14666377.4588518
13	3012	14510651.7069058
14	3013	14325339.6364499
15	3014	16205363.2200799
16	3015	15138516.4545314
17	3016	13787177.4175561
18	3017	14632605.741609
19	3018	16562329.0352228
20	3019	14129622.1219946
21	3020	16392099.8046748
22	3021	17119993.3656217
23	3022	13206085.2248491
24	3023	15110069.4993374
25	3024	13368591.8081153
26	3025	14769735.4214566
27	3026	15127844.8605547
28	3027	14691436.1069797
29	3028	17161238.3840275
30	3029	13611696.9311902

sample_submission

DataFrame aus einer CSV-Datei laden

```
import pandas as pd


train_data = pd.read_csv('data/train.csv')

test_data = pd.read_csv('data/test.csv')
```

Liest eine CSV-Datei (relativer Pfad zum Notebook) und speichert es in einem pd.DataFrame.
Viele optionale Parameter z.B. encoding, delimiter, quotechar,...

Quelle: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Datenvorverarbeitung: Extrahieren der relevanten Spalten

Extrahieren das Regressionsziel

```
# Labels extrahieren

y = train_data ["revenue"]
```

```
# Relevante Spalten (features) extrahieren (wird später erweitert...)

features = ["budget"]

X = train_data [features]
```

Extrahieren der relevanten Features

Andere Ideen für die Datenvorverarbeitung?

Bildquelle: https://cdn.pixabay.com/photo/2015/11/03/08/56/question-mark-1019820_960_720.jpg

ML Training

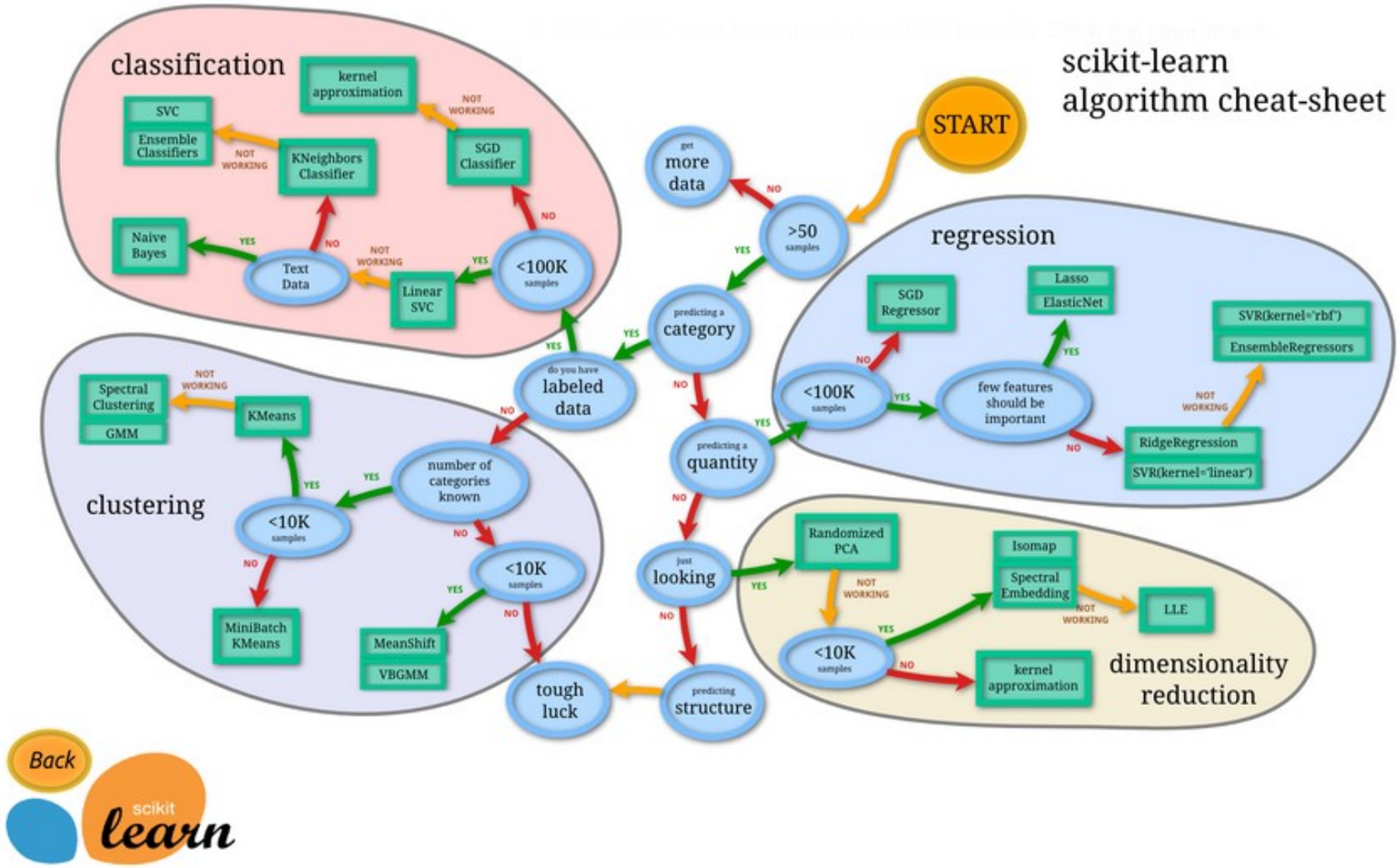
```
from sklearn.svm import SVR


# Erzeuge mittles ML ein Regressionsmodell
model = SVR()
# Trainiere das Regressionsmodell
model.fit(X, y)
```

Gewählter ML Ansatz:
Support Vector Regression

Training (“fitting”) des gewählten ML Modells mit den Merkmalen (X) und Zielen (y)

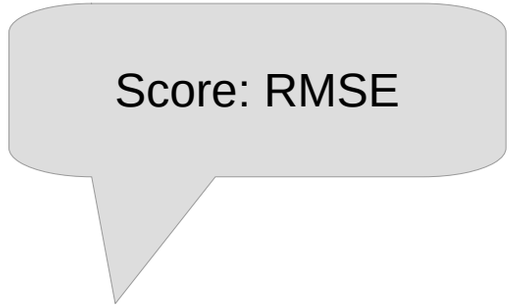
Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Ideen für weitere ML Ansätze: Das ML algorithm cheat sheet



Quelle:https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Validierung




```
# Evaluiere das Modell
scores = cross_val_score(model, X, y, scoring='neg_root_mean_squared_error')
print(f'RMSE: {scores.mean()}')
```

Erzeuge die Datei für die Einreichung zum Kaggle-Wettbewerb

```
# Vorhersage der Labels für den Test-Datensatz
predictions = model.predict(X_test)


# Bereite die Einreichung vor
output = pd.DataFrame({'id': test_data.id, 'revenue': predictions})
output.to_csv('data/submission.csv', index=False)
```

Vorhersage des Umsatzes des Kaggle Testdatensatzes (er muss vorverarbeitet worden sein)

DataFrame mit zwei Spalten: id und revenue

Schreibt den DataFrame in eine CSV-Datei

Aufbau

- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

Cheat Sheets: Überblick und Hilfestellung für ML

Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data

The Most Complete List of Best AI Cheat Sheets

Stefan Kojouharov · Follow

Published in Becoming Human: Artificial Intelligence Magazine · 8 min read · Jul 9, 2017

68K 166

Over the past few months, I have been collecting AI cheat sheets. From time to time I share them with friends and colleagues and recently I have been getting asked a lot, so I decided to organize and share the entire collection. To make things more interesting and give context, I added descriptions and/or excerpts for each major topic.

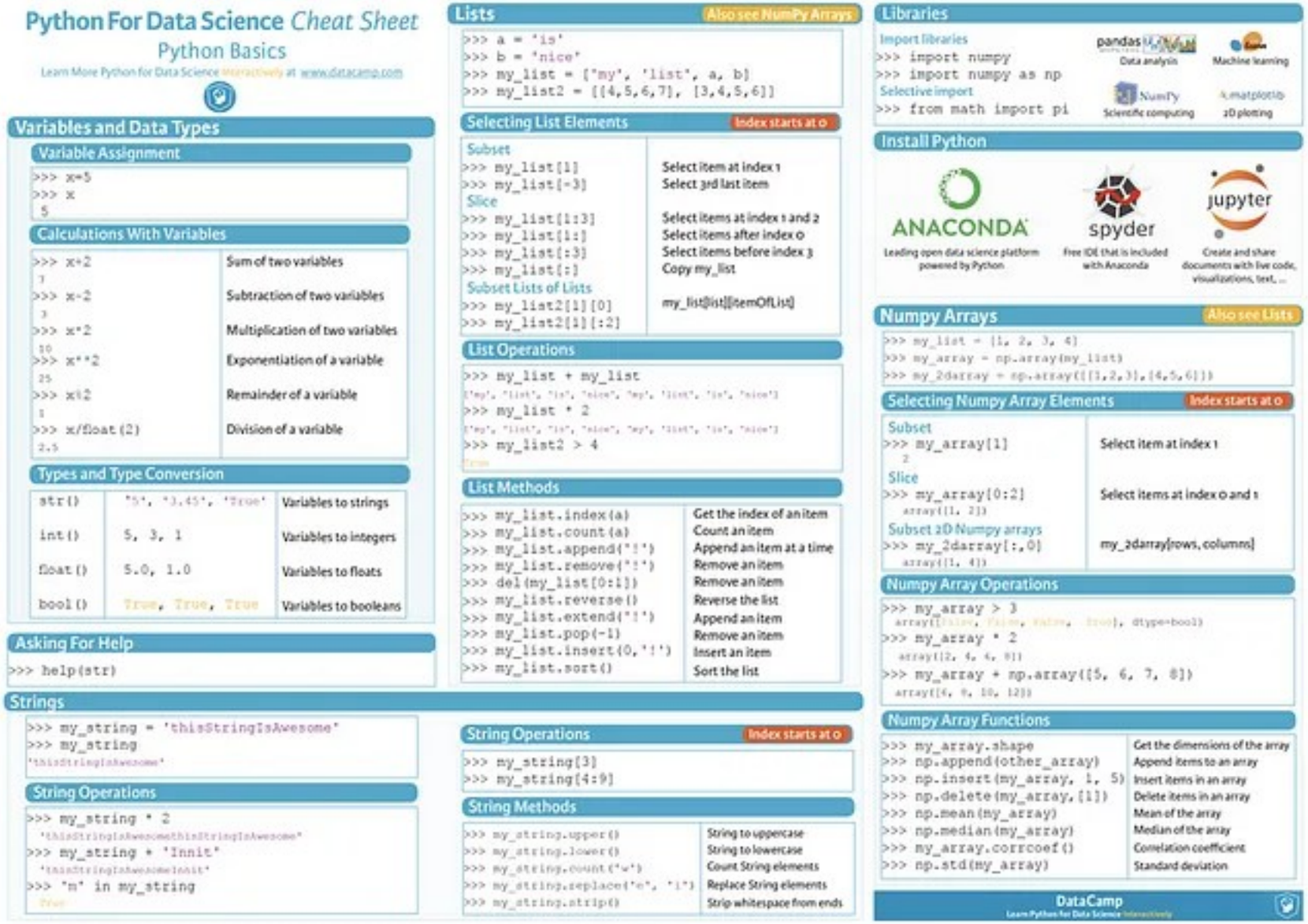
This is the most complete list and the Big-O is at the very end, enjoy...

>>> Update: We have recently redesigned these cheat sheets into a Super High Definition PDF. Check them out below:

Get AI Cheat Sheets PDF

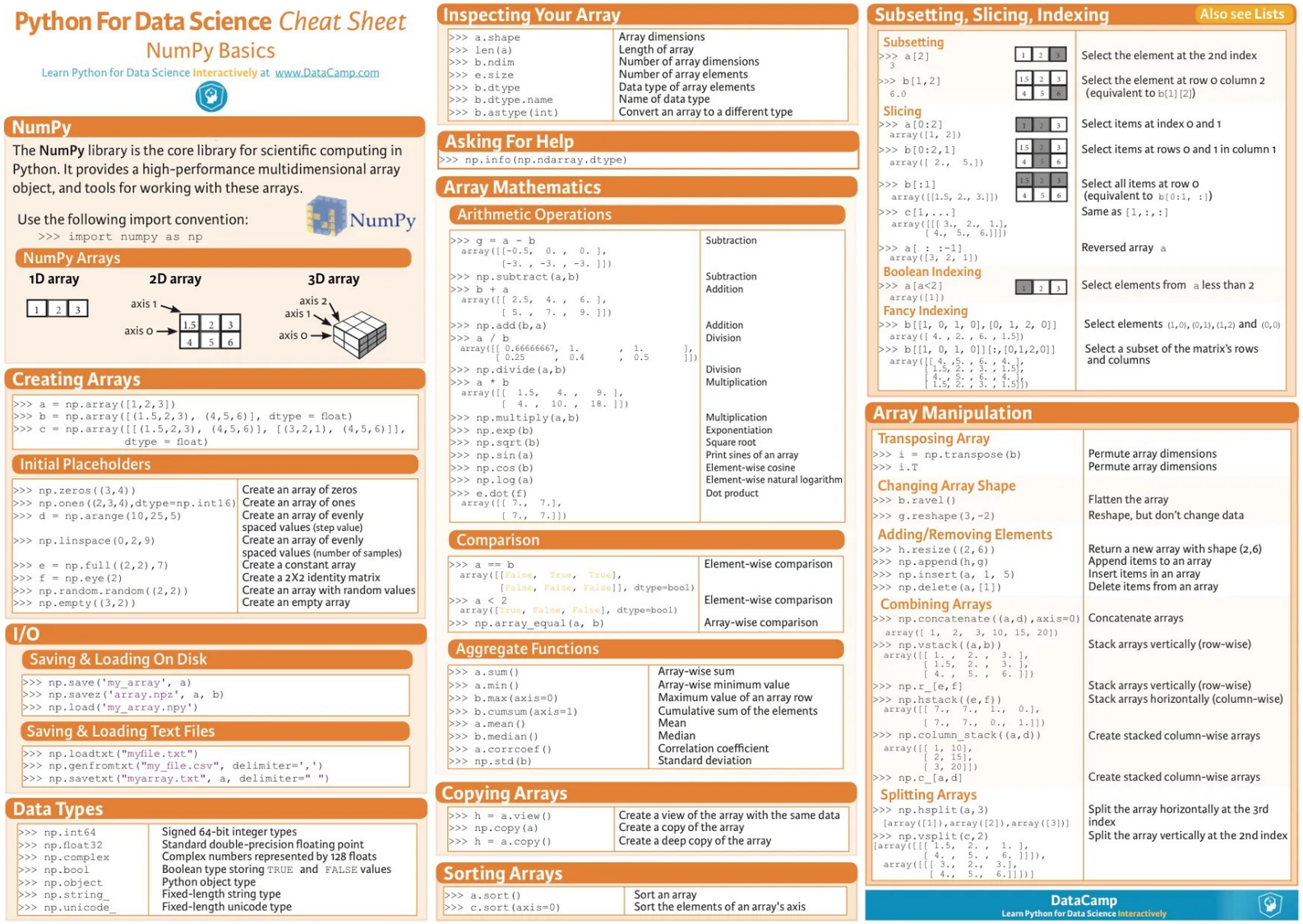
Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Python Basics: Python For Data Science Cheat Sheet



Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

NumPy Basics: Python For Data Science Cheat Sheet



Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array

2D array

3D array

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

`>>> np.zeros((3,4))`	Create an array of zeros
`>>> np.ones((2,3,4),dtype=np.int16)`	Create an array of ones
`>>> d = np.arange(10,25,5)`	Create an array of evenly spaced values (step value)
`>>> np.linspace(0,2,9)`	Create an array of evenly spaced values (number of samples)
`>>> e = np.full((2,2),7)`	Create a constant array
`>>> f = np.eye(2)`	Create a 2X2 identity matrix
`>>> np.random.random((2,2))`	Create an array with random values
`>>> np.empty((3,2))`	Create an empty array

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

`>>> np.int64`	Signed 64-bit integer types
`>>> np.float32`	Standard double-precision floating point
`>>> np.complex`	Complex numbers represented by 128 floats
`>>> np.bool`	Boolean type storing `TRUE` and `FALSE` values
`>>> np.object`	Python object type
`>>> np.string_`	Fixed-length string type
`>>> np.unicode_`	Fixed-length unicode type

Inspecting Your Array

`>>> a.shape`	Array dimensions
`>>> len(a)`	Length of array
`>>> b.ndim`	Number of array dimensions
`>>> e.size`	Number of array elements
`>>> b.dtype`	Data type of array elements
`>>> b.dtype.name`	Name of data type
`>>> b.astype(int)`	Convert an array to a different type

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

`>>> g = a - b`
`array([[-0.5, 0. , 0. ],`
`[-3. , -3. , -3. ]])`
	Subtraction

`>>> np.subtract(a,b)`	Subtraction

`>>> b + a`
`array([[ 2.5, 4. , 6. ],`
`[ 5. , 7. , 9. ]])`
	Addition

`>>> np.add(b,a)`	Addition

`>>> a / b`
`array([[ 0.66666667, 1. , 1. ],`
`[ 0.25 , 0.4 , 0.5 ]])`
	Division

`>>> np.divide(a,b)`	Division

`>>> a * b`
`array([[ 1.5, 4. , 9. ],`
`[ 4. , 10. , 18. ]])`
	Multiplication

`>>> np.multiply(a,b)`	Multiplication
`>>> np.exp(b)`	Exponentiation
`>>> np.sqrt(b)`	Square root
`>>> np.sin(a)`	Print sines of an array
`>>> np.cos(b)`	Element-wise cosine
`>>> np.log(a)`	Element-wise natural logarithm

`>>> e.dot(f)`
`array([[ 7., 7.],`
`[ 7., 7.]])`
	Dot product

Comparison

`>>> a == b`
`array([[False, True, True],`
`[False, False, False]], dtype=bool)`
	Element-wise comparison

`>>> a < 2` `array([True, False, False], dtype=bool)`	Element-wise comparison
`>>> np.array_equal(a, b)`	Array-wise comparison

Aggregate Functions

`>>> a.sum()`	Array-wise sum
`>>> a.min()`	Array-wise minimum value
`>>> b.max(axis=0)`	Maximum value of an array row
`>>> b.cumsum(axis=1)`	Cumulative sum of the elements
`>>> a.mean()`	Mean
`>>> b.median()`	Median
`>>> a.corrcoef()`	Correlation coefficient
`>>> np.std(b)`	Standard deviation

Copying Arrays

`>>> h = a.view()`	Create a view of the array with the same data
`>>> np.copy(a)`	Create a copy of the array
`>>> h = a.copy()`	Create a deep copy of the array

Sorting Arrays

`>>> a.sort()`	Sort an array
`>>> c.sort(axis=0)`	Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
3
```
Select the element at the 2nd index

```
>>> b[1,2]
6.0
```
Select the element at row 0 column 2 (equivalent to `b[1][2]`)

Slicing

```
>>> a[0:2]
array([1, 2])
```
Select items at index 0 and 1

```
>>> b[0:2,1]
array([ 2.,  5.])
```
Select items at rows 0 and 1 in column 1

```
>>> b[:1]
array([[1.5, 2., 3.]])
```
Select all items at row 0 (equivalent to `b[0:1, :]`)

```
>>> c[1,...]
array([[[ 3.,  2.,  1.],
        [ 4.,  5.,  6.]]])
```
Same as `[1,:,:]`

```
>>> a[ : :-1]
array([3, 2, 1])
```
Reversed array `a`

Boolean Indexing

```
>>> a[a<2]
array([1])
```
Select elements from `a` less than 2

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
array([ 4. , 2. , 6. , 1.5])
```
Select elements `(1,0)`, `(0,1)`, `(1,2)` and `(0,0)`

```
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
array([[ 4. , 2. , 6. , 4. ],
       [ 1.5, 2. , 3. , 1.5],
       [ 4. , 5. , 6. , 4. ],
       [ 1.5, 2. , 3. , 1.5]])
```
Select a subset of the matrix's rows and columns

Array Manipulation

Transposing Array

`>>> i = np.transpose(b)`	Permute array dimensions
`>>> i.T`	Permute array dimensions

Changing Array Shape

`>>> b.ravel()`	Flatten the array
`>>> g.reshape(3,-2)`	Reshape, but don't change data

Adding/Removing Elements

`>>> h.resize((2,6))`	Return a new array with shape (2,6)
`>>> np.append(h,g)`	Append items to an array
`>>> np.insert(a, 1, 5)`	Insert items in an array
`>>> np.delete(a,[1])`	Delete items from an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
array([ 1,  2,  3, 10, 15, 20])
```
Concatenate arrays

```
>>> np.vstack((a,b))
array([[ 1. ,  2. ,  3. ],
       [ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```
Stack arrays vertically (row-wise)

```
>>> np.r_[e,f]
```
Stack arrays vertically (row-wise)

```
>>> np.hstack((e,f))
array([[ 7.,  7.,  1.,  0.],
       [ 7.,  7.,  0.,  1.]])
```
Stack arrays horizontally (column-wise)

```
>>> np.column_stack((a,d))
array([[ 1, 10],
       [ 2, 15],
       [ 3, 20]])
```
Create stacked column-wise arrays

```
>>> np.c_[a,d]
```
Create stacked column-wise arrays

Splitting Arrays

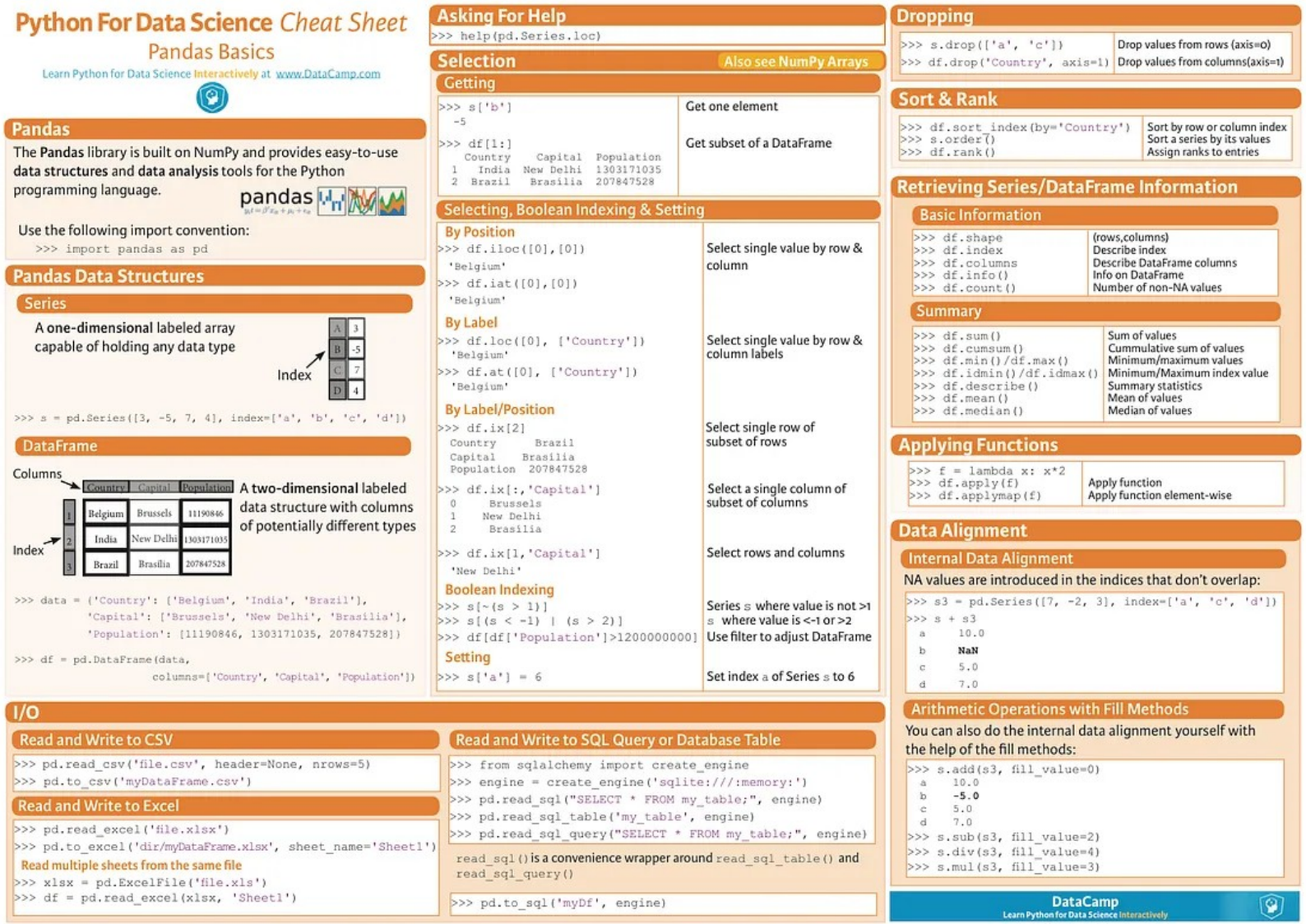
```
>>> np.hsplit(a,3)
[array([1]),array([2]),array([3])]
```
Split the array horizontally at the 3rd index

```
>>> np.vsplit(c,2)
[array([[[ 1.5,  2. ,  1. ],
        [ 4. ,  5. ,  6. ]]]),
 array([[[ 3. ,  2. ,  3. ],
        [ 4. ,  5. ,  6. ]]])]
```
Split the array vertically at the 2nd index

DataCamp
Learn Python for Data Science Interactively

Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Pandas Basics: Python For Data Science Cheat Sheet



Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A **one-dimensional** labeled array capable of holding any data type

Index

A	3
B	-5
C	7
D	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasilia	207847528

Index

A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
-5
```
Get one element

```
>>> df[1:]
   Country     Capital  Population
1   India   New Delhi  1303171035
2  Brazil    Brasilia   207847528
```
Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```
Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```
Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population  207847528
```
Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasilia
```
Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```
Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
```
Series `s` where value is not >1 `s` where value is <-1 or >2

```
>>> df[df['Population']>1200000000]
```
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```
Set index `a` of Series `s` to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```
Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index(by='Country')
>>> s.order()
>>> df.rank()
```
Sort by row or column index
Sort a series by its values
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```
(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idmin()/df.idmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```
Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```
Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
  a    10.0
  b     NaN
  c     5.0
  d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
  a    10.0
  b    -5.0
  c     5.0
  d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

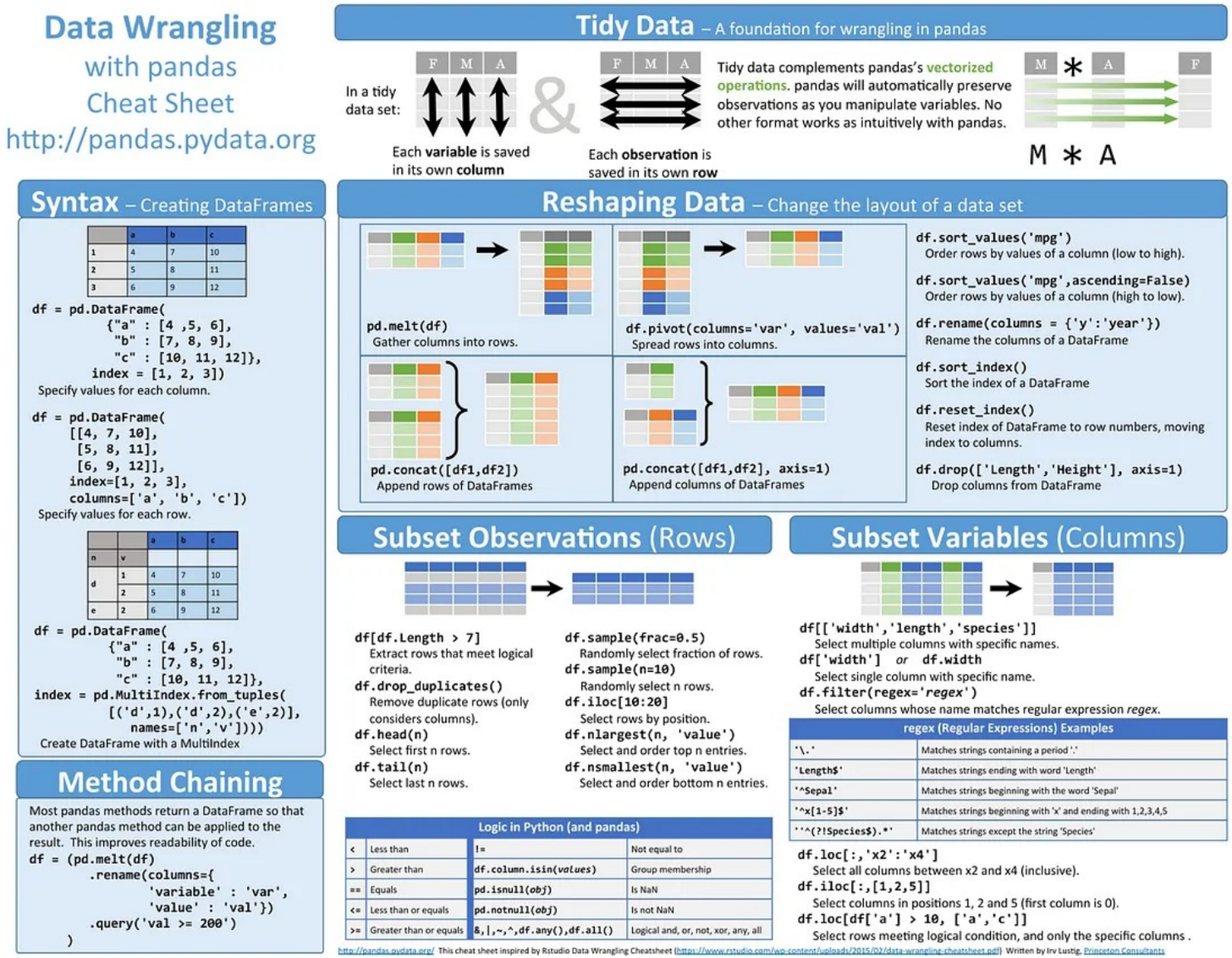
`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> pd.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

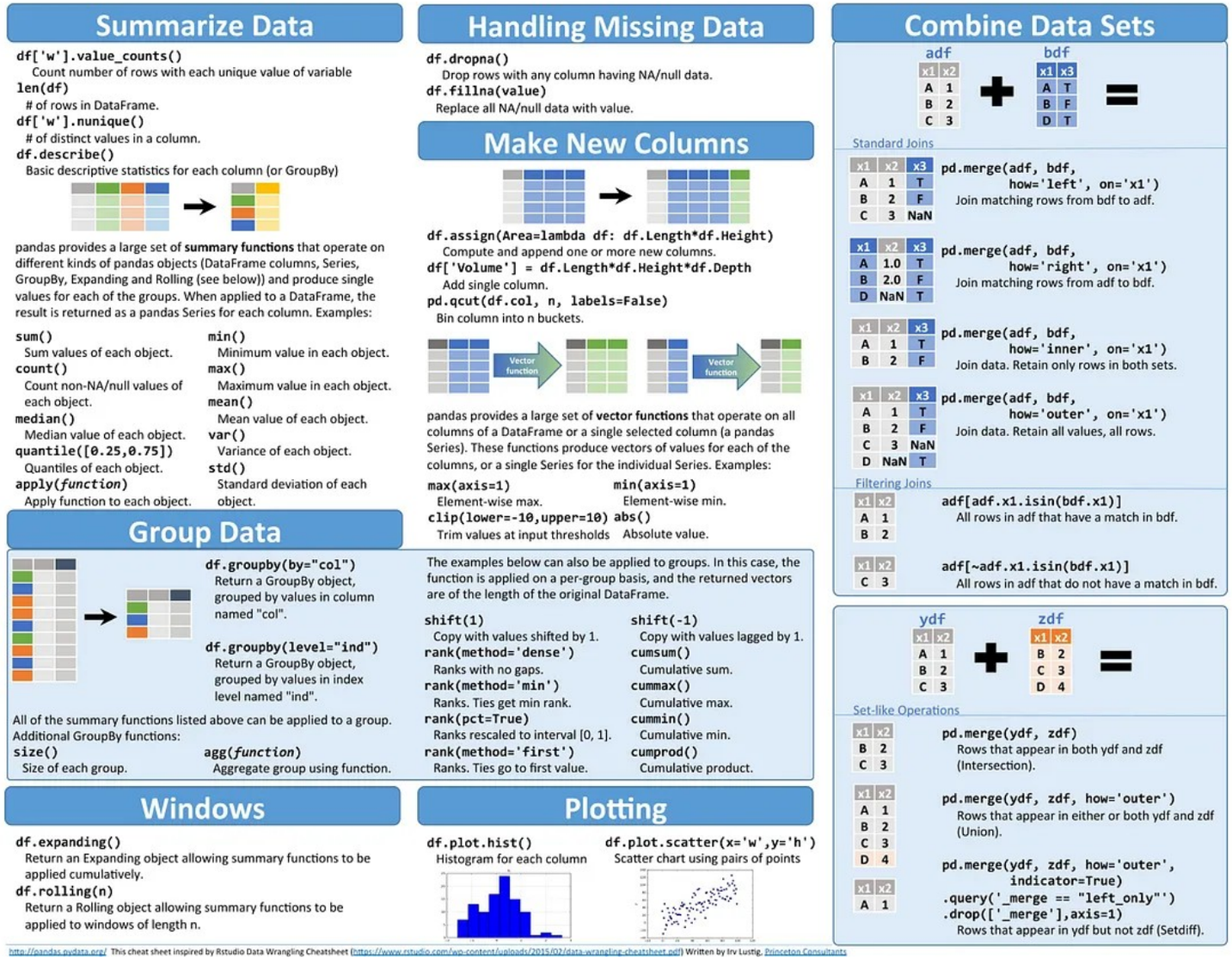
Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Data Wrangling with Pandas (1/2) Cheat Sheet



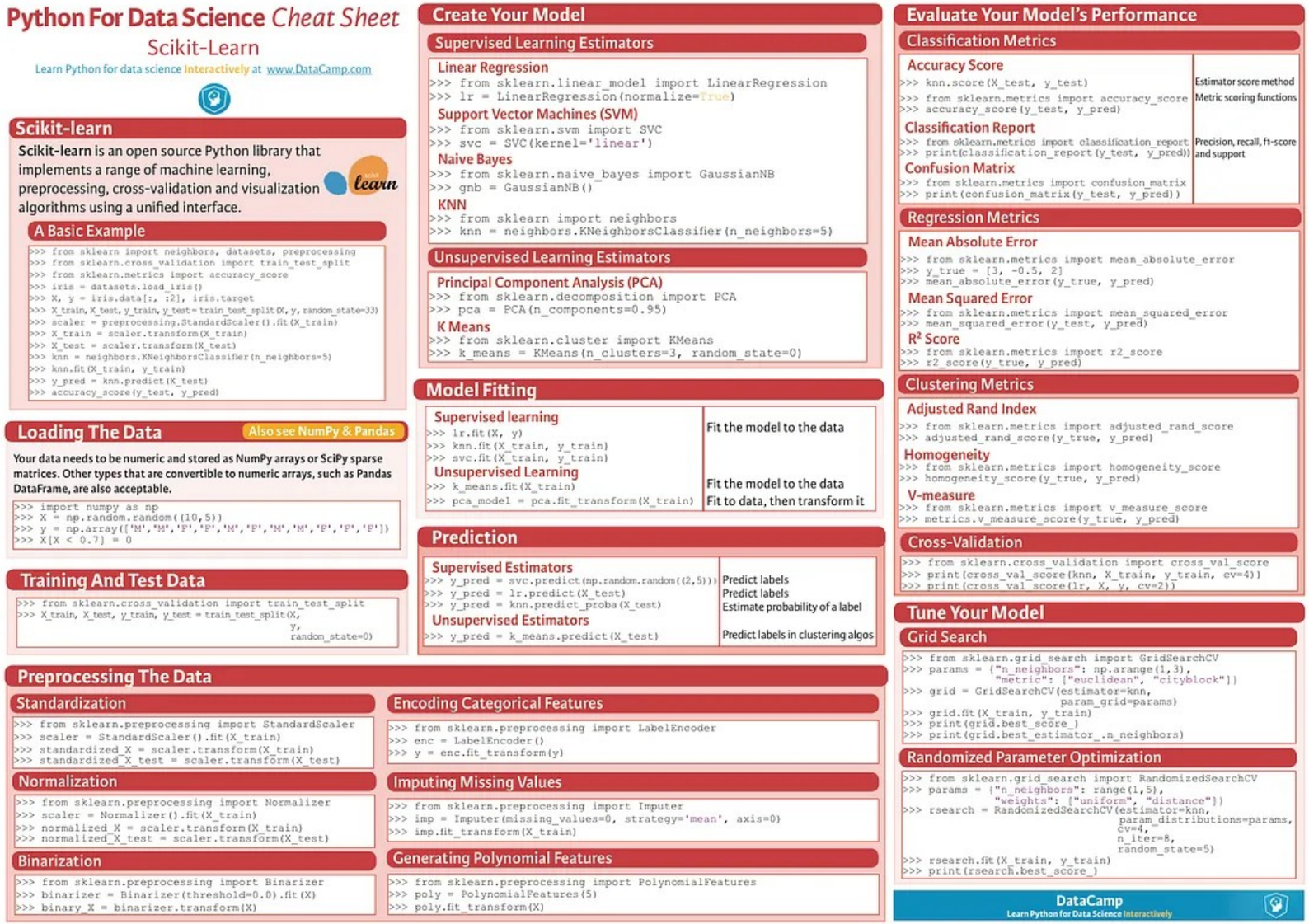
Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Data Wrangling with Pandas (2/2) Cheat Sheet



Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Scikit-Learn: Python For Data Science Cheat Sheet



Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning	
`>>> lr.fit(X, y)`	Fit the model to the data
`>>> knn.fit(X_train, y_train)`	
`>>> svc.fit(X_train, y_train)`	
Unsupervised Learning	
`>>> k_means.fit(X_train)`	Fit the model to the data
`>>> pca_model = pca.fit_transform(X_train)`	Fit to data, then transform it

Prediction

Supervised Estimators	
`>>> y_pred = svc.predict(np.random.random((2,5)))`	Predict labels
`>>> y_pred = lr.predict(X_test)`	Predict labels
`>>> y_pred = knn.predict_proba(X_test)`	Estimate probability of a label
Unsupervised Estimators	
`>>> y_pred = k_means.predict(X_test)`	Predict labels in clustering algos

Preprocessing The Data

Standardization

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

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score — Estimator score method

```
>>> knn.score(X_test, y_test)
```

Metric scoring functions

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Classification Report — Precision, recall, f1-score and support

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

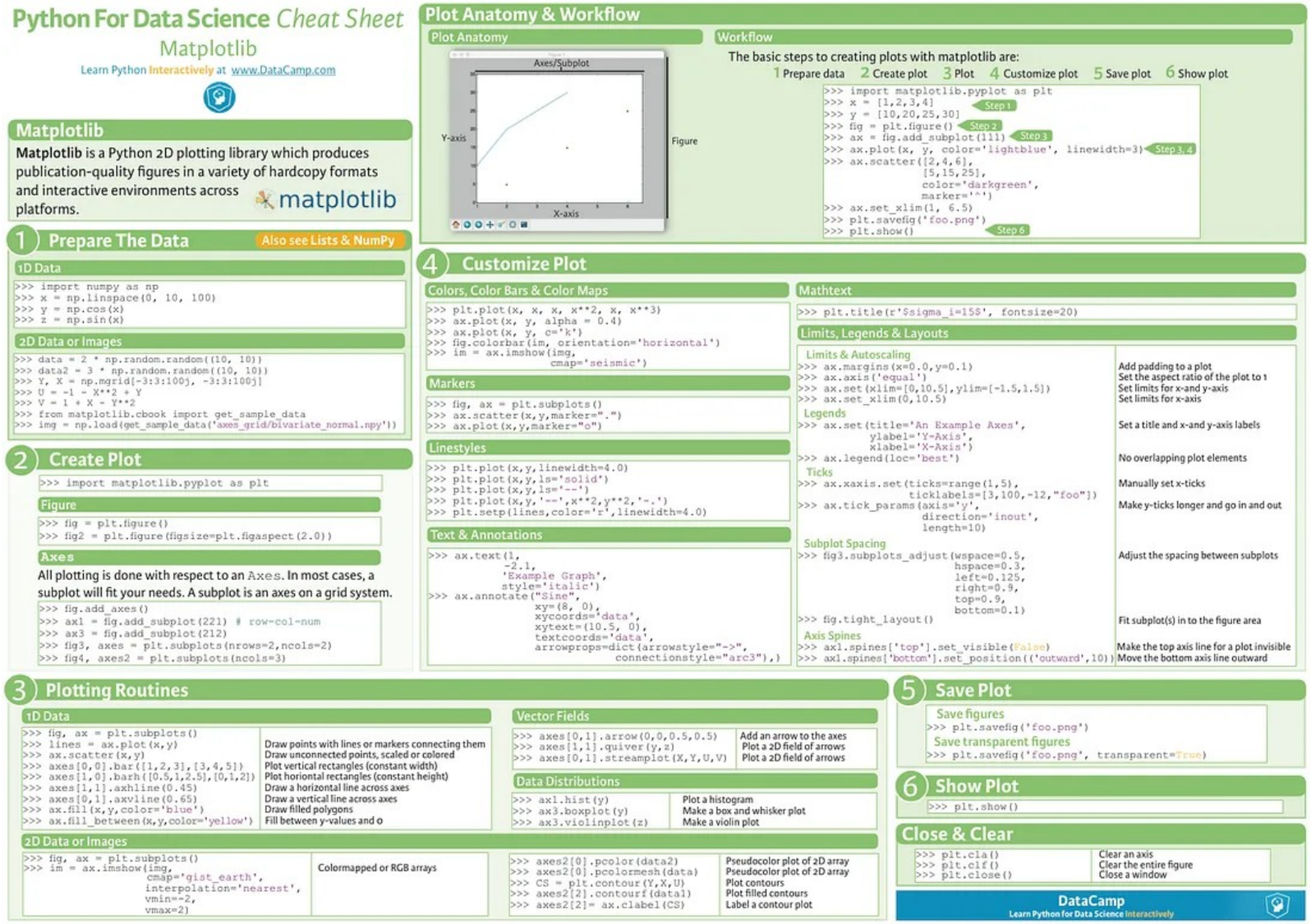
Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

DataCamp
Learn Python for Data Science Interactively

Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Matplotlib: Python For Data Science Cheat Sheet



Quelle:https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Aufbau

- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

Mini-Test “ML Regression”

- Erklären Sie die ML Aufgabe Regression
- Erklären Sie den ML Entwicklungsprozess
- Warum ist es wichtig, ein ML Modell zu validieren?
- Erklären Sie die Konfusionsmatrix
- Erklären Sie Treffergenauigkeit
- Wann ist es sinnvoll, Treffergenauigkeit zu benutzen, wann nicht?
- Welcher Score sollte bei unausgewogenen Datensätzen verwendet werden?
- Welche Scores existieren für die Validierung von Regressionen?
- Erklären Sie die k-fache Kreuzvalidierung
- Welche ML Modelle können für Regression verwendet werden? Wie werden Sie trainiert? Wie werden Sie für die Vorhersage benutzt?