

Einführung in die künstliche Intelligenz

EKI03 – ML Regression

Prof. Dr. A. del Pino

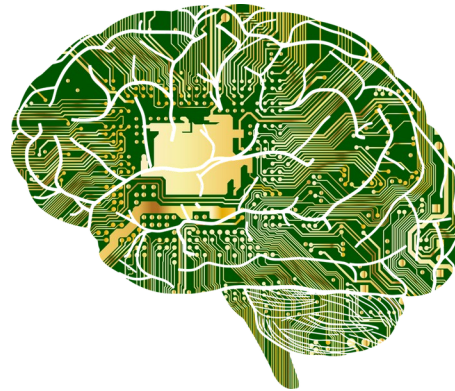
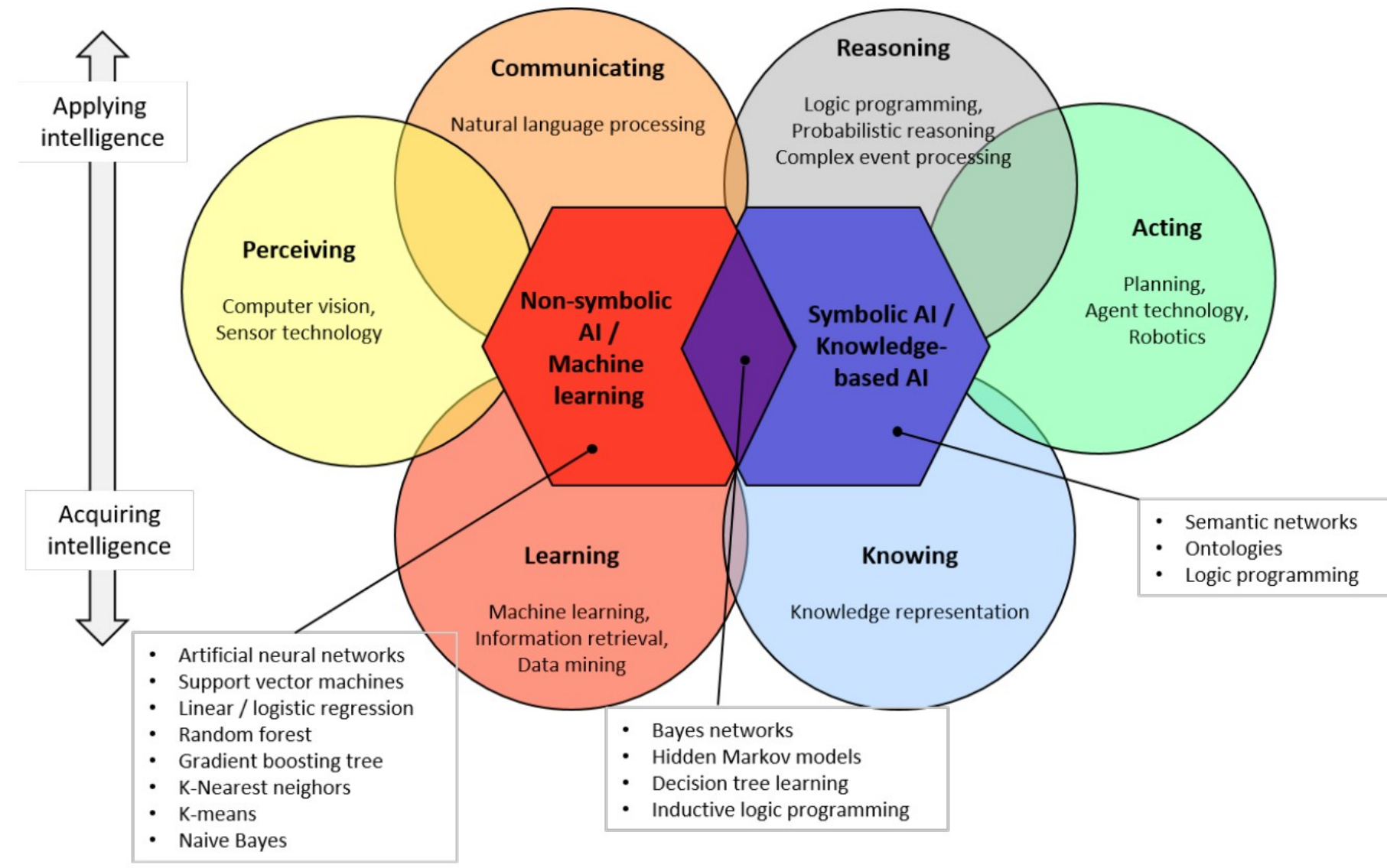


Bild: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Anatomy-1751201_1280.png

Aufbau

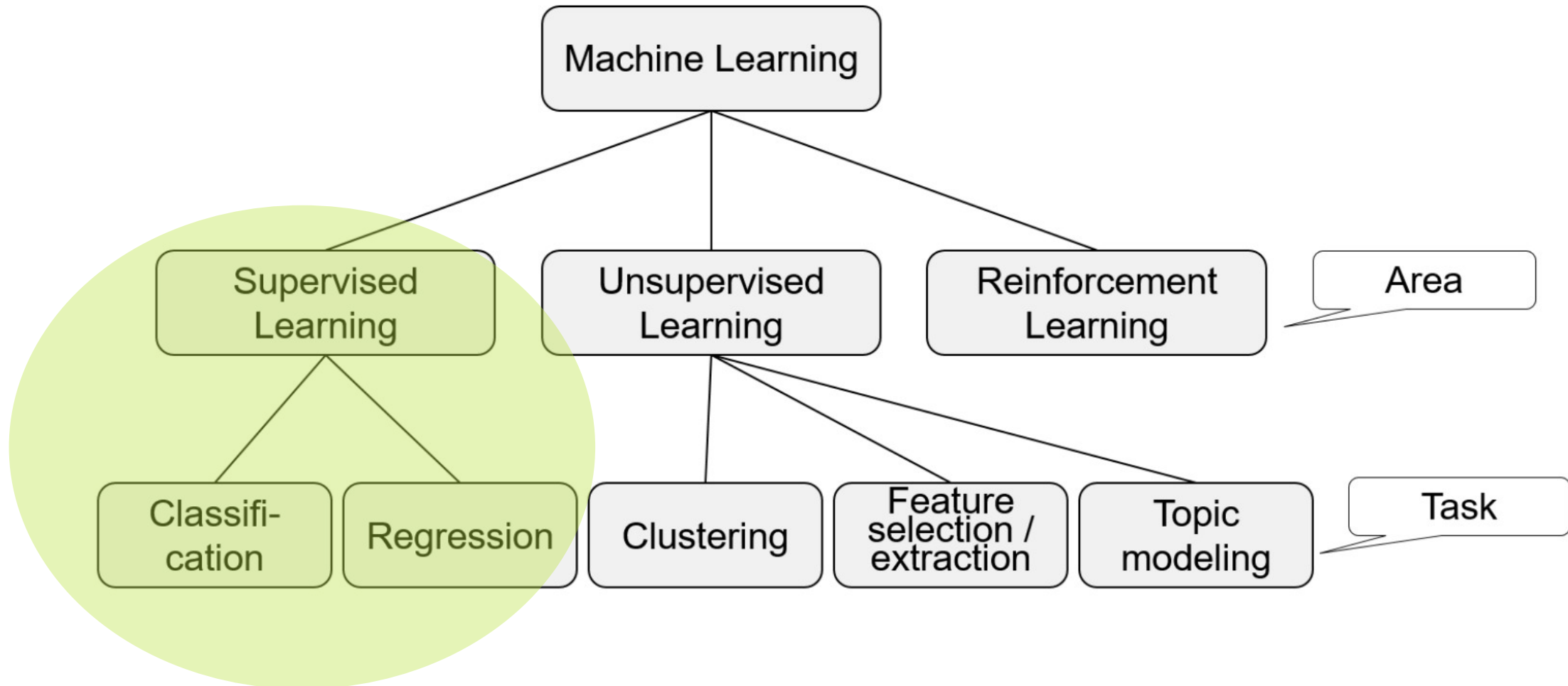
- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

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

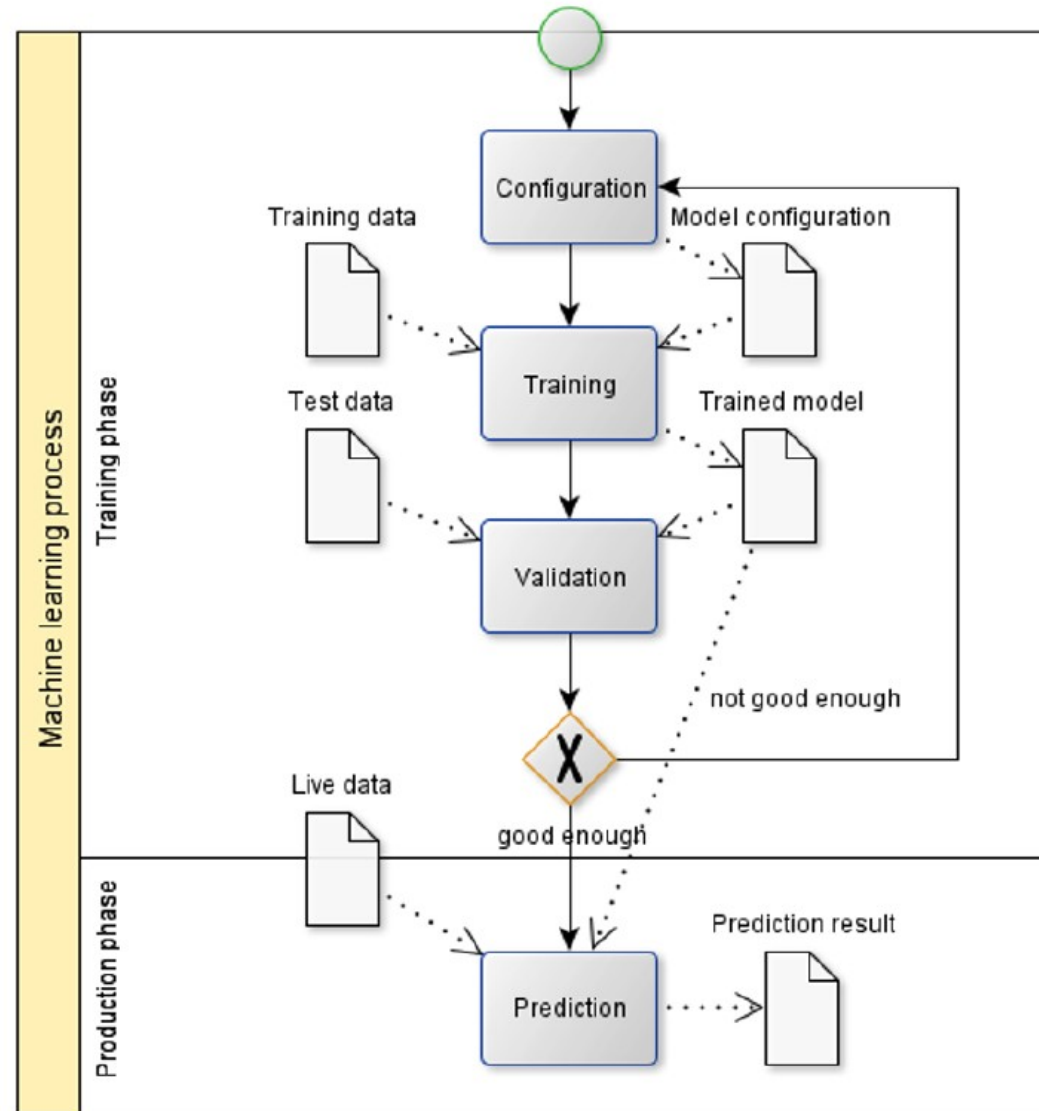
- Überblick
- ML Validierung
- Beispiel zu Regression: Ticketverkauf-Vorhersage mit TMDB
- Cheat sheets
- Mini-Test

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

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.

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

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

Im Beispiel:

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

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

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

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

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

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

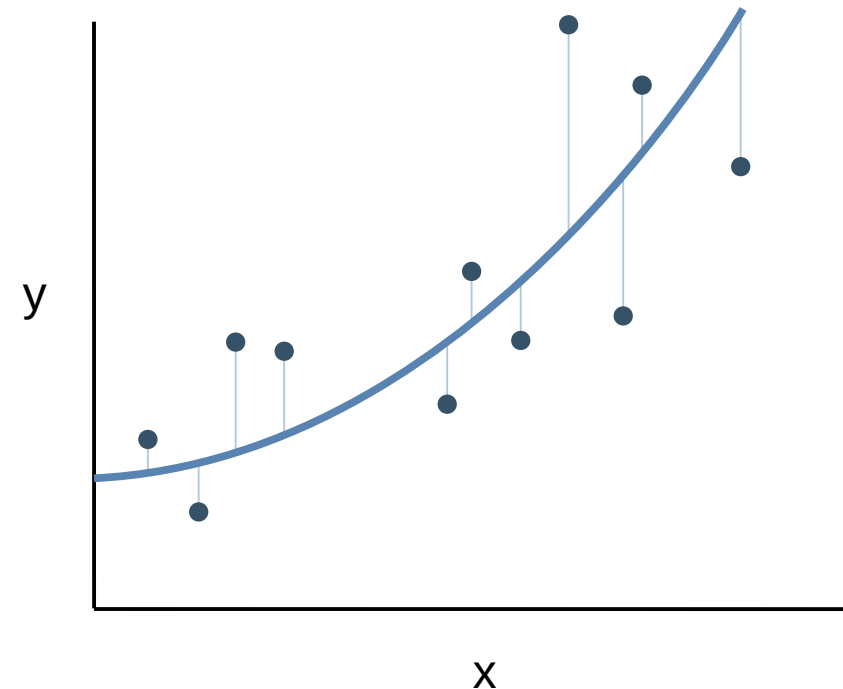
Validierung der Regression: MAE, MSE, RMSE

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

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^n |\hat{f}_i - y_i| \quad \text{Mean Absolute Error}$$

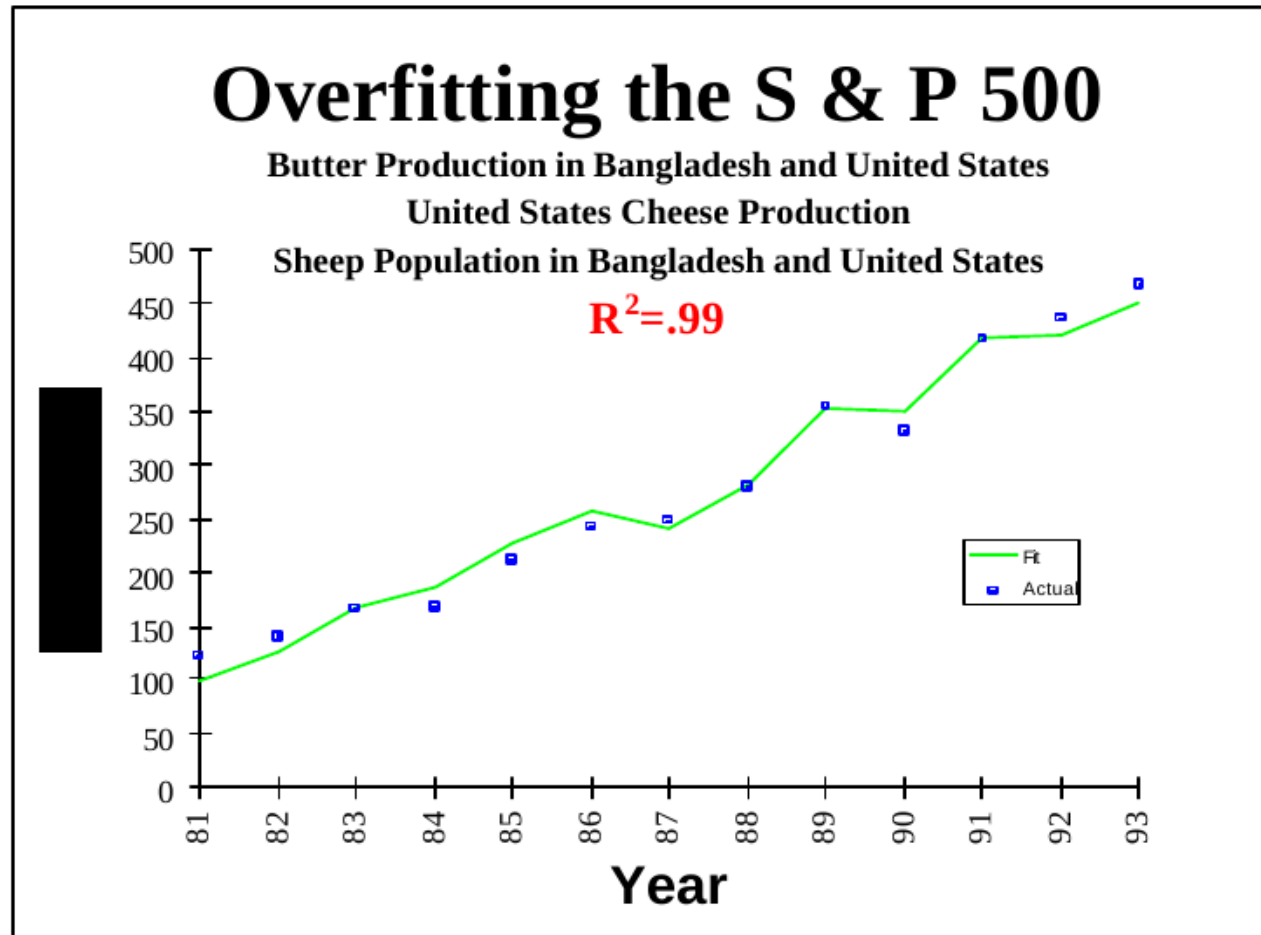
$$MSE = \frac{1}{n} \cdot \sum_{i=1}^n (\hat{f}_i - y_i)^2 \quad \text{Mean Squared Error}$$

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n (\hat{f}_i - y_i)^2} \quad \text{Rooted MSE}$$



Validierung der Regression

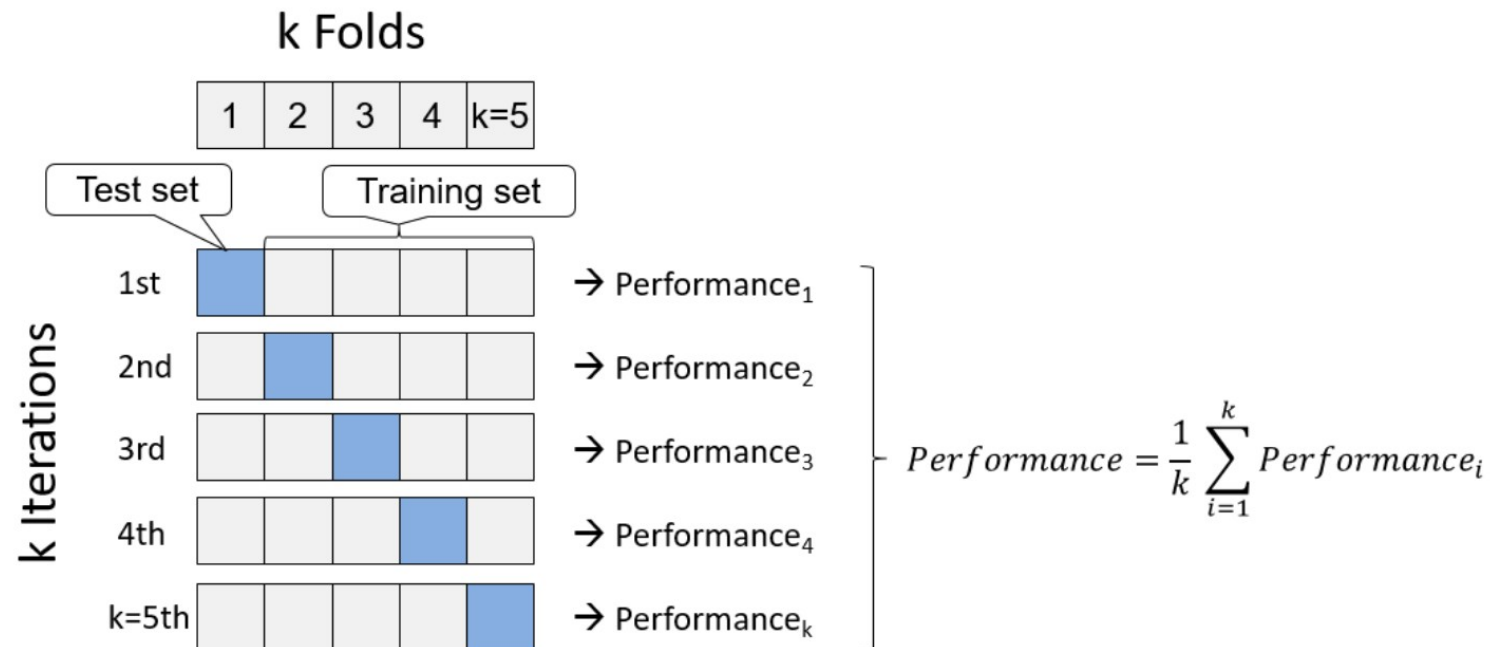
Achtung! Regressionen müssen einen Sinn ergeben, eine gute Validierung alleine ist nicht ausreichend!



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

 Playground Prediction Competition

TMDb Box Office Prediction

Can you predict a movie's worldwide box office revenue?

 Kaggle · 1,395 teams · 4 years ago

[Overview](#)
[Data](#)
[Code](#)
[Discussion](#)
[Leaderboard](#)
[Rules](#)

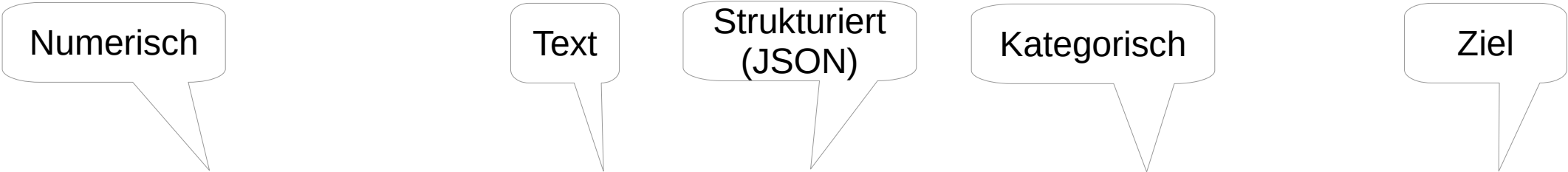
Late Submission

Overview

Description	We're going to make you an offer you can't refuse: a Kaggle competition!
Evaluation	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."
Timeline	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.
Prizes	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>

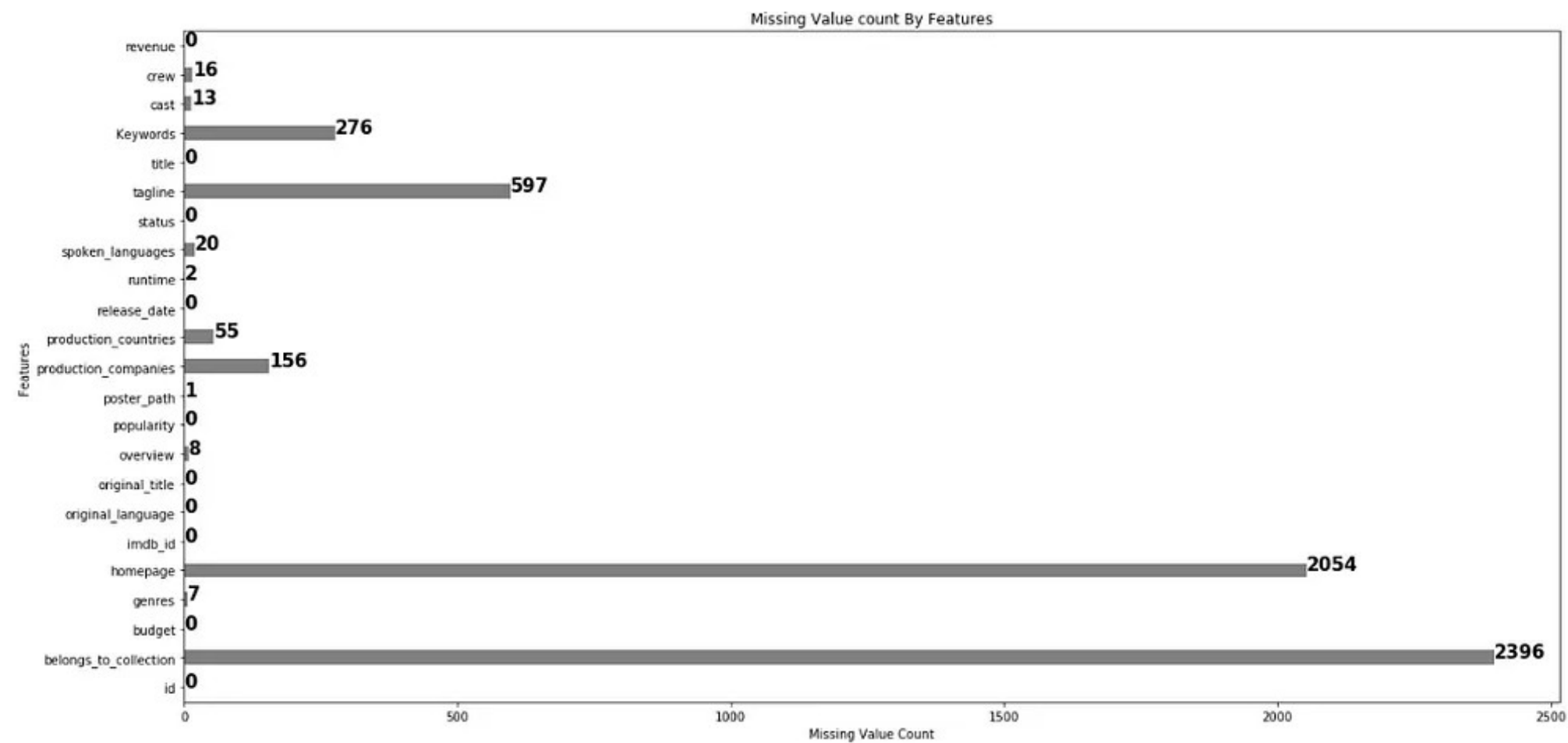
Trainingsdaten: Merkmale (features) und Ziel (target)



	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": 312576, "name": "Hot Tub Time Machine Collection", "poster_path": "/AEb60TGPacF0b4aM1ueyY026U.jpg", "backdrop_path": "/hwaTVcggB0D48BDjFVc1Vz7ape.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	/sQWuwvMMHCCzQR2k0w7C3c.jpg	[[{"name": "Paramount Pictures", "id": 4}, {"name": "United Artists", "id": 60}, {"name": "Metro-Goldwyn-Mayer (MGM)", "id": 9411}]]	[[{"iso_3166_1": "US", "name": "United States of America"}]]	202015		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": 9953, "name": "space"}, {"id": 11830, "name": "hot tub"}, {"id": 179431, "name": "duringcreditsstinger"}]]	[[{"cast_id": 4, "character": "Lou", "credit_id": "526a067c92514107a002d0c0", "department": "Acting", "gender": 2, "id": 1449071, "job": "First Assistant Director", "name": "Rob Corddry", "order": 0, "profile_path": "/62z3Uv1uE2uFT08uUdOdJucKz.jpg"}, {"cast_id": 5, "character": "Nick", "credit_id": "526a067c3a369478e002d0c0", "department": "Acting", "gender": 2, "id": 3227, "job": "Director", "name": "Craig Robinson", "order": 1, "profile_path": "/AvsRMuXOEVhYxImFuhWjRjzz.jpg"}, {"cast_id": 6, "character": "Jacob", "credit_id": "526a067c3a369478e002d0c0", "department": "Acting", "gender": 2, "id": 34735, "job": "Writer", "name": "Josh Heald", "order": 2, "profile_path": "/jwXJlrmDMrG7GjNkxlv1RGu.jpg"}, {"cast_id": 7, "character": "Adam Jr.", "credit_id": "526a067c3a369478e002d0c0", "department": "Acting", "gender": 2, "id": 34735, "job": "Characters", "name": "Adam Scott", "order": 3, "profile_path": "/5g955z8zd42yMAl4wv0v0f0r.jpg"}, {"cast_id": 8, "character": "Hot Tub Repairman", "credit_id": "526a067c3a369478e002d0c0", "department": "Production", "gender": 2, "id": 5802, "name": "Chevy Chase", "order": 4, "profile_path": "/5v9jyY6PwR0X9ZnOmO6hD0t.jpg"}, {"cast_id": 9, "character": "Jill", "credit_id": "526a067c3a369478e002d0c0", "department": "Production", "gender": 0, "id": 94098, "name": "Gillian Jacobs", "order": 5, "profile_path": "/t8hne5WNP4RUPh7aW0v0v0f0r.jpg"}, {"cast_id": 10, "character": "Sophie", "credit_id": "526a067c3a369478e002d0c0", "department": "Production", "gender": 2, "id": 115909, "name": "Bianca Haase", "order": 6, "profile_path": "/62z3Uv1uE2uFT08uUdOdJucKz.jpg"}]]	12314651

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	
31	3030	15031731.2122056	

< | << >> + 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 mittels ML ein Regressionsmodell
```

```
model = SVR()
```

```
# Trainiere das Regressionsmodell
```

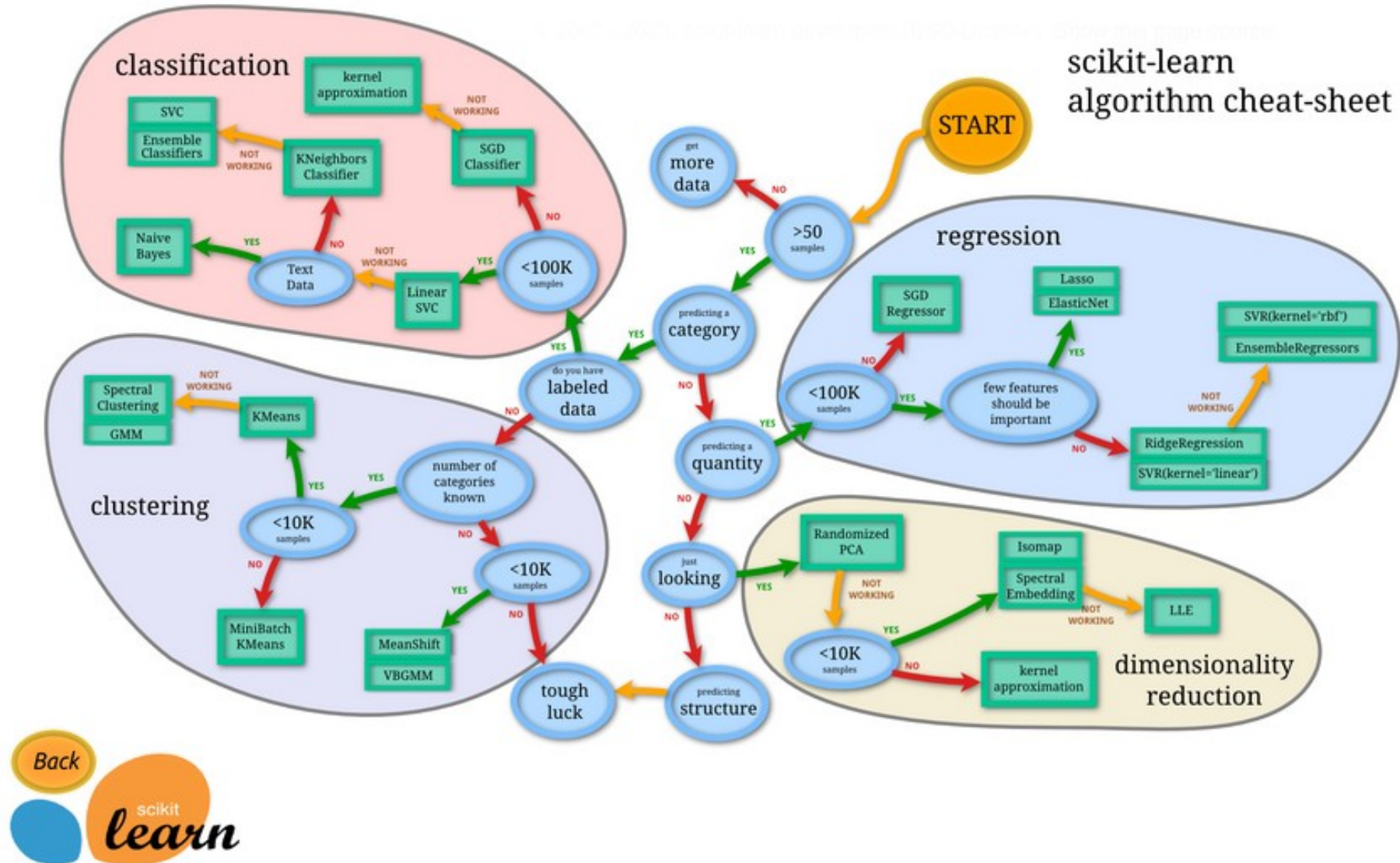
```
model.fit(X, y)
```

Gewählter ML Ansatz:
Support Vector Regression

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

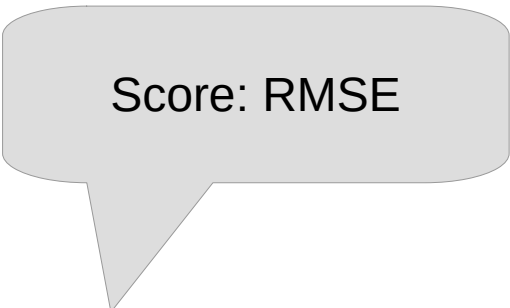
Quelle: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>

Ideen für weitere ML Ansätze: Das ML algorithm cheat sheet



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

Validierung



Score: RMSE

```
# 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)
```

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

```
# Berechne die Einreichung vor
```

```
output = pd.DataFrame({'id': test_data.id, 'revenue': predictions})
```

```
output.to_csv('data/submission.csv', index=False)
```

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

Python For Data Science Cheat Sheet

Python Basics

Learn More Python for Data Science interactively at www.datacamp.com

Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
5
```

Calculations With Variables

>>> x+2	7	Sum of two variables
>>> x-2	3	Subtraction of two variables
>>> x*2	10	Multiplication of two variables
>>> x**2	25	Exponentiation of a variable
>>> x%2	1	Remainder of a variable
>>> x/float(2)	2.5	Division of a variable

Types and Type Conversion

Function	Example	Description
str()	'5', '3.45', 'True'	Variables to strings
int()	5, 3, 1	Variables to integers
float()	5.0, 1.0	Variables to floats
bool()	True, True, True	Variables to booleans

Asking For Help

```
>>> help(str)
```

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select item at index 1
Select 3rd last item
Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
my_list[1][1:2]my_list[1][1:]
```

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

```
>>> my_list.index(a)
>>> my_list.count(a)
>>> my_list.append('!')
>>> my_list.remove('!')
>>> del my_list[0:1]
>>> my_list.reverse()
>>> my_list.extend('!!')
>>> my_list.pop(-1)
>>> my_list.insert(0, '!')
>>> my_list.sort()
```

Get the index of an item
Count an item
Append an item at a time
Remove an item
Remove an item
Reverse the list
Append an item
Remove an item
Insert an item
Sort the list

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
>>> Selective import
>>> from math import pi
```

pandas Data analysis
NumPy Scientific computing
matplotlib 2D plotting
Machine learning

Install Python

ANACONDA Leading open data science platform powered by Python
spyder Free IDE that is included with Anaconda
jupyter Create and share documents with live code, visualizations, text, ...

NumPy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting NumPy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

Subset 2D NumPy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

my_2darray[rows, columns]

NumPy Array Operations

```
>>> my_array > 3
array([False,  False,  True,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

NumPy Array Functions

```
>>> my_array.shape
>>> np.append(other_array)
>>> np.insert(my_array, 1, 5)
>>> np.delete(my_array, [1])
>>> np.mean(my_array)
>>> np.median(my_array)
>>> my_array.corrcoef()
>>> np.std(my_array)
```

Get the dimensions of the array
Append items to an array
Insert items in an array
Delete items in an array
Mean of the array
Median of the array
Correlation coefficient
Standard deviation

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

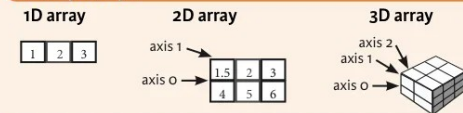
The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

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



NumPy Arrays



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))
>>> np.ones((2,3,4),dtype=np.int16)
>>> d = np.arange(10,25,5)
>>> np.linspace(0,2,9)
>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty((3,2))
```

Create an array of zeros
Create an array of ones
Create an array of evenly spaced values (step value)
Create an array of evenly spaced values (number of samples)
Create a constant array
Create a 2x2 identity matrix
Create an array with random values
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
>>> np.float32
>>> np.complex
>>> np.bool
>>> np.object
>>> np.string_
>>> np.unicode_
```

Signed 64-bit integer types
Standard double-precision floating point
Complex numbers represented by 128 floats
Boolean type storing TRUE and FALSE values
Python object type
Fixed-length string type
Fixed-length unicode type

Inspecting Your Array

```
>>> a.shape
>>> len(a)
>>> b.ndim
>>> e.size
>>> b.dtype
>>> b.dtype.name
>>> b.astype(int)
```

Array dimensions
Length of array
Number of array dimensions
Number of array elements
Data type of array elements
Name of data type
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. ]])
>>> np.subtract(a,b)
>>> b + a
>>> array([[ 2.5,  4. ,  6. ],
>>>        [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)
>>> a / b
>>> array([[ 0.66666667,  1. ,  1. ],
>>>        [ 0.25 ,  0.4 ,  0.5 ]])
>>> np.divide(a,b)
>>> a * b
>>> array([[ 1.5,  4. ,  9. ],
>>>        [ 4. ,  10. ,  18. ]])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.sqrt(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> e.dot(f)
>>> array([[ 7. ,  7. ],
>>>        [ 7. ,  7.]])
```

Subtraction

Subtraction

Addition

Addition

Division

Division

Multiplication

Multiplication

Exponentiation

Square root

Print sines of an array

Element-wise cosine

Element-wise natural logarithm

Dot product

Comparison

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

Element-wise comparison

Element-wise comparison

Array-wise comparison

Aggregate Functions

```
>>> a.sum()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
>>> a.corrcoef()
>>> np.std(b)
```

Array-wise sum

Array-wise minimum value

Maximum value of an array row

Cumulative sum of the elements

Mean

Median

Correlation coefficient

Standard deviation

Copying Arrays

```
>>> h = a.view()
>>> np.copy(a)
>>> h = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

Sorting Arrays

```
>>> a.sort()
>>> c.sort(axis=0)
```

Sort an array
Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
>>> 3
```

```
[[ 1  2  3]
 [ 4  5  6]]
```

Select the element at the 2nd index

```
>>> b[1,2]
>>> 6.0
```

```
[[ 1.5  2.   3. ]
 [ 4.   5.   6. ]]
```

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

Slicing

```
>>> a[0:2]
>>> array([1, 2])
```

```
[[ 1  2  3]
 [ 4  5  6]]
```

Select items at index 0 and 1

```
>>> b[0:2,1]
>>> array([ 2.,  5.])
```

```
[[ 1.5  2.   3. ]
 [ 4.   5.   6. ]]
```

Select items at rows 0 and 1 in column 1

```
>>> b[:1]
>>> array([[1.5, 2., 3.]])
```

```
[[ 1.5  2.   3. ]
 [ 4.   5.   6. ]]
```

Select all items at row 0 (equivalent to b[0:1, :])

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

```
[[ 1.5  2.   3. ]
 [ 4.   5.   6. ]]
```

Same as [1, :, :]

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

```
[[ 1  2  3]
 [ 4  5  6]]
```

Reversed array a

Boolean Indexing

```
>>> a[a<2]
>>> array([1])
```

```
[[ 1  2  3]
 [ 4  5  6]]
```

Select elements from a less than 2

Fancy Indexing

```
>>> b[[1, 0, 1, 0], [0, 1, 2, 0]]
>>> array([ 4. ,  2. ,  6. ,  1.5])
>>> b[[1, 0, 1, 0]][:, [0,1,2,0]]
>>> array([[ 4. ,  5. ,  6. ,  4. ],
>>>        [ 1.5,  2. ,  3. ,  1.5]])
```

```
[[ 1.5  2.   3. ]
 [ 4.   5.   6. ]]
```

Select elements (1,0), (0,1), (1,2) and (0,0)
Select a subset of the matrix's rows and columns

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```

Permute array dimensions
Permute array dimensions

Changing Array Shape

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Flatten the array
Reshape, but don't change data

Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```

Return a new array with shape (2,6)
Append items to an array
Insert items in an array
Delete items from an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
>>> array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))
>>> array([[ 1. ,  2. ,  3. ],
>>>        [ 1.5,  2. ,  3. ],
>>>        [ 4. ,  5. ,  6. ]])
```

Concatenate arrays
Stack arrays vertically (row-wise)

```
>>> np.r_[e,f]
>>> np.hstack((e,f))
>>> array([[ 7. ,  7. ,  1. ,  0. ],
>>>        [ 7. ,  7. ,  0. ,  1.]])
```

Stack arrays vertically (row-wise)
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]
>>> array([[ 1, 10],
>>>        [ 2, 15],
>>>        [ 3, 20]])
```

Create stacked column-wise arrays

Splitting Arrays

```
>>> np.hsplit(a,3)
>>> [array([1]),array([2]),array([3])]
>>> np.vsplit(c,2)
>>> [array([[ 1.5,  2. ,  1. ],
>>>        [ 4. ,  5. ,  6. ]]),
>>> array([[ 3. ,  2. ,  3. ],
>>>        [ 4. ,  5. ,  6. ]])]
>>> array([[ 3. ,  2. ,  3. ],
>>>        [ 4. ,  5. ,  6. ]])
```

Split the array horizontally at the 3rd index
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](https://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.

pandas

Use the following import convention:

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

Pandas Data Structures

Series

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

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

DataFrame

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)]
>>> df[df['Population'] > 1200000000]
```

Series s where value is not > 1
s where value is < -1 or > 2
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.idxmin() / df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values
Cumulative 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)
```

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

Data Wrangling
with pandas
Cheat Sheet
<http://pandas.pydata.org>

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

	a	b	c
n			
v			
d	1	4	7
e	2	5	8
	2	6	9
			12

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
        [('d', 1), ('d', 2), ('e', 2)],
        names=['n', 'v']))
```

Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
      .rename(columns={
          'variable': 'var',
          'value': 'val'})
      .query('val >= 200'))
```

Tidy Data – A foundation for wrangling in pandas

In a tidy data set:
Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A
F

Reshaping Data – Change the layout of a data set

pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1, df2])
Append rows of DataFrames

pd.concat([df1, df2], axis=1)
Append columns of DataFrames

```
df.sort_values('mpg')
Order rows by values of a column (low to high).

df.sort_values('mpg', ascending=False)
Order rows by values of a column (high to low).

df.rename(columns={'y': 'year'})
Rename the columns of a DataFrame

df.sort_index()
Sort the index of a DataFrame

df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(['Length', 'Height'], axis=1)
Drop columns from DataFrame
```

Subset Observations (Rows)

df[df.Length > 7]
Extract rows that meet logical criteria.

```
df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.
```

```
df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.
```

Subset Variables (Columns)

```
df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] or df.width
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression regex.
```

regex (Regular Expressions) Examples

regex	Matches
'.'	Matches strings containing a period '.'
'Length\$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]\$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species)\$'	Matches strings except the string 'Species'

```
df.loc[:, 'x2': 'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:, [1, 2, 5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns.
```

	Logic in Python (and pandas)
<	Less than
>	Greater than
==	Equals
<=	Less than or equals
>=	Greater than or equals
!=	Not equal to
df.column.isin(values)	Group membership
pd.isnull(obj)	Is NaN
pd.notnull(obj)	Is not NaN
&, , ~, ^, df.any(), df.all()	Logical and, or, not, xor, any, all

<http://pandas.pydata.org> This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (<https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf>) Written by Irv Luvitz, Princeton Consultants

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

Summarize Data

df['w'].value_counts()
Count number of rows with each unique value of variable

len(df)
of rows in DataFrame.

df['w'].nunique()
of distinct values in a column.

df.describe()
Basic descriptive statistics for each column (or GroupBy)



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

sum() Sum values of each object.	min() Minimum value in each object.
count() Count non-NA/null values of each object.	max() Maximum value in each object.
median() Median value of each object.	mean() Mean value of each object.
quantile([0.25, 0.75]) Quantiles of each object.	var() Variance of each object.
apply(function) Apply function to each object.	std() Standard deviation of each object.

Group Data



df.groupby(by="col")
Return a GroupBy object, grouped by values in column named "col".

df.groupby(level="ind")
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

size()
Size of each group.

agg(function)
Aggregate group using function.

Windows

df.expanding()
Return an Expanding object allowing summary functions to be applied cumulatively.

df.rolling(n)
Return a Rolling object allowing summary functions to be applied to windows of length n.

Handling Missing Data

df.dropna()
Drop rows with any column having NA/null data.

df.fillna(value)
Replace all NA/null data with value.

Make New Columns



df.assign(Area=lambda df: df.Length*df.Height)
Compute and append one or more new columns.

df['Volume'] = df.Length*df.Height*df.Depth
Add single column.

pd.qcut(df.col, n, labels=False)
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

max(axis=1) Element-wise max.	min(axis=1) Element-wise min.
clip(lower=-10, upper=10) Trim values at input thresholds	abs() Absolute value.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

shift(1) Copy with values shifted by 1.	shift(-1) Copy with values lagged by 1.
rank(method='dense') Ranks with no gaps.	cumsum() Cumulative sum.
rank(method='min') Ranks. Ties get min rank.	cummax() Cumulative max.
rank(pct=True) Ranks rescaled to interval [0, 1].	cummin() Cumulative min.
rank(method='first') Ranks. Ties go to first value.	cumprod() Cumulative product.

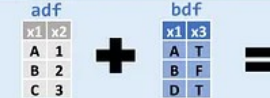
Plotting

df.plot.hist()
Histogram for each column

df.plot.scatter(x='w', y='h')
Scatter chart using pairs of points



Combine Data Sets



Standard Joins

pd.merge(adf, bdf, how='left', on='x1')
Join matching rows from bdf to adf.

pd.merge(adf, bdf, how='right', on='x1')
Join matching rows from adf to bdf.

pd.merge(adf, bdf, how='inner', on='x1')
Join data. Retain only rows in both sets.

pd.merge(adf, bdf, how='outer', on='x1')
Join data. Retain all values, all rows.

Filtering Joins

adf[adf.x1.isin(bdf.x1)]
All rows in adf that have a match in bdf.

adf[~adf.x1.isin(bdf.x1)]
All rows in adf that do not have a match in bdf.



Set-like Operations

pd.merge(ydf, zdf)
Rows that appear in both ydf and zdf (Intersection).

pd.merge(ydf, zdf, how='outer')
Rows that appear in either or both ydf and zdf (Union).

pd.merge(ydf, zdf, how='outer', indicator=True)
.query('_merge == "left_only"')
.drop(['_merge'], axis=1)
Rows that appear in ydf but not zdf (Setdiff).

<https://pandas.pydata.org/> This cheat sheet inspired by RStudio Data Wrangling Cheatsheet (<https://www.rstudio.com/wp-content/uploads/2015/07/data-wrangling-cheatsheet.pdf>) Written by Irv Lustin, Princeton Consultants

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](https://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'])
>>> 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)
```

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)
```

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

Prediction

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

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score	Estimator score method
<pre>>>> knn.score(X_test, y_test) >>> from sklearn.metrics import accuracy_score >>> accuracy_score(y_test, y_pred)</pre>	Metric scoring functions
Classification Report	Precision, recall, f1-score and support
<pre>>>> from sklearn.metrics import classification_report >>> print(classification_report(y_test, y_pred))</pre>	
Confusion Matrix	
<pre>>>> from sklearn.metrics import confusion_matrix >>> print(confusion_matrix(y_test, y_pred))</pre>	

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)
```

R² 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

Python For Data Science Cheat Sheet

Matplotlib

Learn Python interactively at [www.DataCamp.com](https://www.datacamp.com)



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



1 Prepare The Data

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 * X**2 + Y
>>> V = 1 * X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

Figure

```
>>> import matplotlib.pyplot as plt
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row=col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> fig, ax = plt.subplots()
>>> lines = ax.plot(x, y)
>>> ax.scatter(x, y)
>>> axes[0,0].bar([1,2,3], [3,4,5])
>>> axes[1,0].barh([0.5,1.2,5], [0,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.65)
>>> ax.fill(x,y,color='blue')
>>> ax.fill_between(x,y,color='yellow')
```

Draw points with lines or markers connecting them
Draw unconnected points, scaled or colored
Plot vertical rectangles (constant width)
Plot horizontal rectangles (constant height)
Draw a horizontal line across axes
Draw a vertical line across axes
Draw filled polygons
Fill between y-values and 0

2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
>>>                  cmap='gist_earth',
>>>                  interpolation='nearest',
>>>                  vmin=-2,
>>>                  vmax=2)
```

Colormapped or RGB arrays

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes
Plot a 2D field of arrows
Plot a 2D field of arrows

Data Distributions

```
>>> ax1.hist(y)
>>> ax3.boxplot(y)
>>> ax3.violinplot(z)
```

Plot a histogram
Make a box and whisker plot
Make a violin plot

Pseudocolor plot of 2D array

```
>>> axes2[0].pcolor(data2)
>>> axes2[0].pcolormesh(data)
>>> CS = plt.contour(Y,X,U)
>>> axes2[2].contourf(data1)
>>> axes2[2] = ax.clabel(CS)
```

Pseudocolor plot of 2D array
Pseudocolor plot of 2D array
Plot contours
Plot filled contours
Label a contour plot

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
>>>                  cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x, y, marker="s")
>>> ax.plot(x, y, marker="o")
```

Linestyles

```
>>> plt.plot(x, y, linewidth=4.0)
>>> plt.plot(x, y, ls='solid')
>>> plt.plot(x, y, ls='--')
>>> plt.plot(x, y, '--', x**2, y**2, '-.')
>>> plt.setp(lines, color='r', linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
>>>         -2.1,
>>>         'Example Graph',
>>>         style='italic')
>>> ax.annotate("Sine",
>>>              xy=(8, 0),
>>>              xycoords='data',
>>>              xytext=(10.5, 0),
>>>              textcoords='data',
>>>              arrowprops=dict(arrowstyle="->",
>>>                              connectionstyle="arc3"),)
```

Mathtext

```
>>> plt.title(r'$\sigma_i=155$', fontsize=20)
```

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0, y=0.1)
>>> ax.axis('equal')
>>> ax.set(xlim=(0,10.5), ylim=(-1.5,1.5))
>>> ax.set_xlim(0,10.5)
```

Add padding to a plot
Set the aspect ratio of the plot to 1
Set limits for x-and y-axis
Set limits for x-axis

Legends

```
>>> ax.set(title='An Example Axes',
>>>         ylabel='Y-Axis',
>>>         xlabel='X-Axis')
>>> ax.legend(loc='best')
```

Set a title and x-and y-axis labels
No overlapping plot elements

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
>>>               ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',
>>>                 direction='inout',
>>>                 length=10)
```

Manually set x-ticks
Make y-ticks longer and go in and out

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,
>>>                       hspace=0.3,
>>>                       left=0.125,
>>>                       right=0.9,
>>>                       top=0.9,
>>>                       bottom=0.1)
```

Adjust the spacing between subplots

Axis Spines

```
>>> fig.tight_layout()
>>> ax1.spines['top'].set_visible(False)
>>> ax1.spines['bottom'].set_position(('outward', 10))
```

Fit subplot(s) in to the figure area
Make the top axis line for a plot invisible
Move the bottom axis line outward

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```

Clear an axis
Clear the entire figure
Close a window

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?