

Einführung in die künstliche Intelligenz

EKI02 – ML Klassifikation

Prof. Dr. A. del Pino

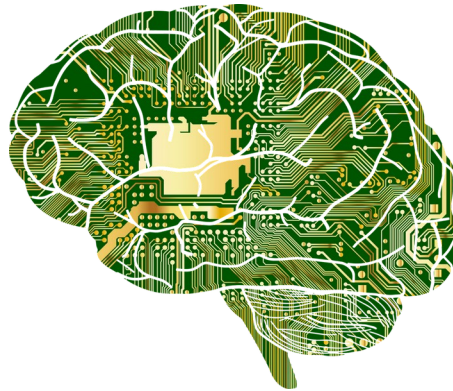
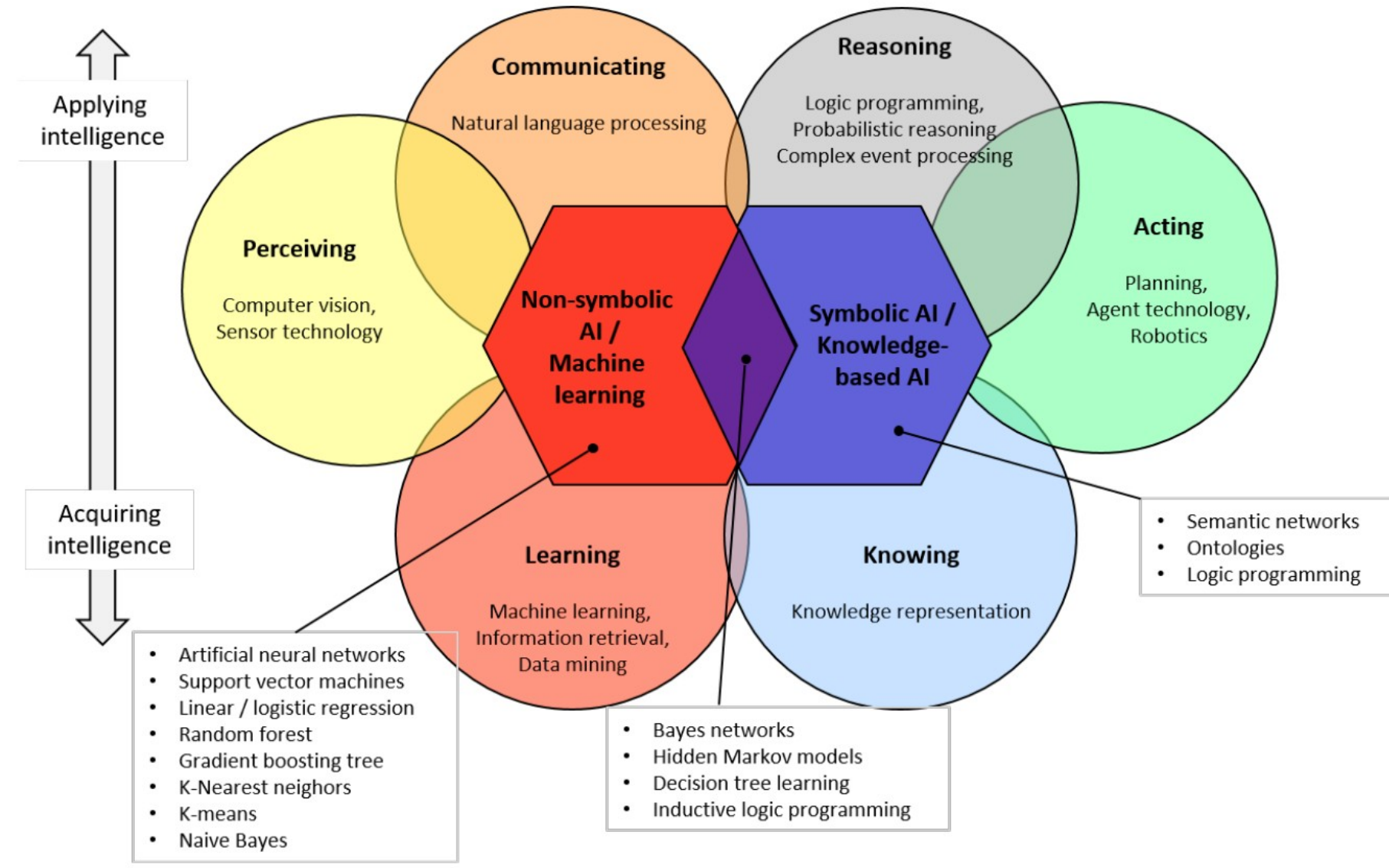


Bild: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Anatomy-1751201_1280.png

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Die KI-Landkarte



Bildquelle: B. Humm "Applied Artificial Intelligence", S. 4

Was ist Machine Learning (maschinelles Lernen) ?

Erzeugen eines Modells, basierend auf Eingaben
("Training")
und die Nutzung des Modells für Vorhersagen
("produktive Anwendung").

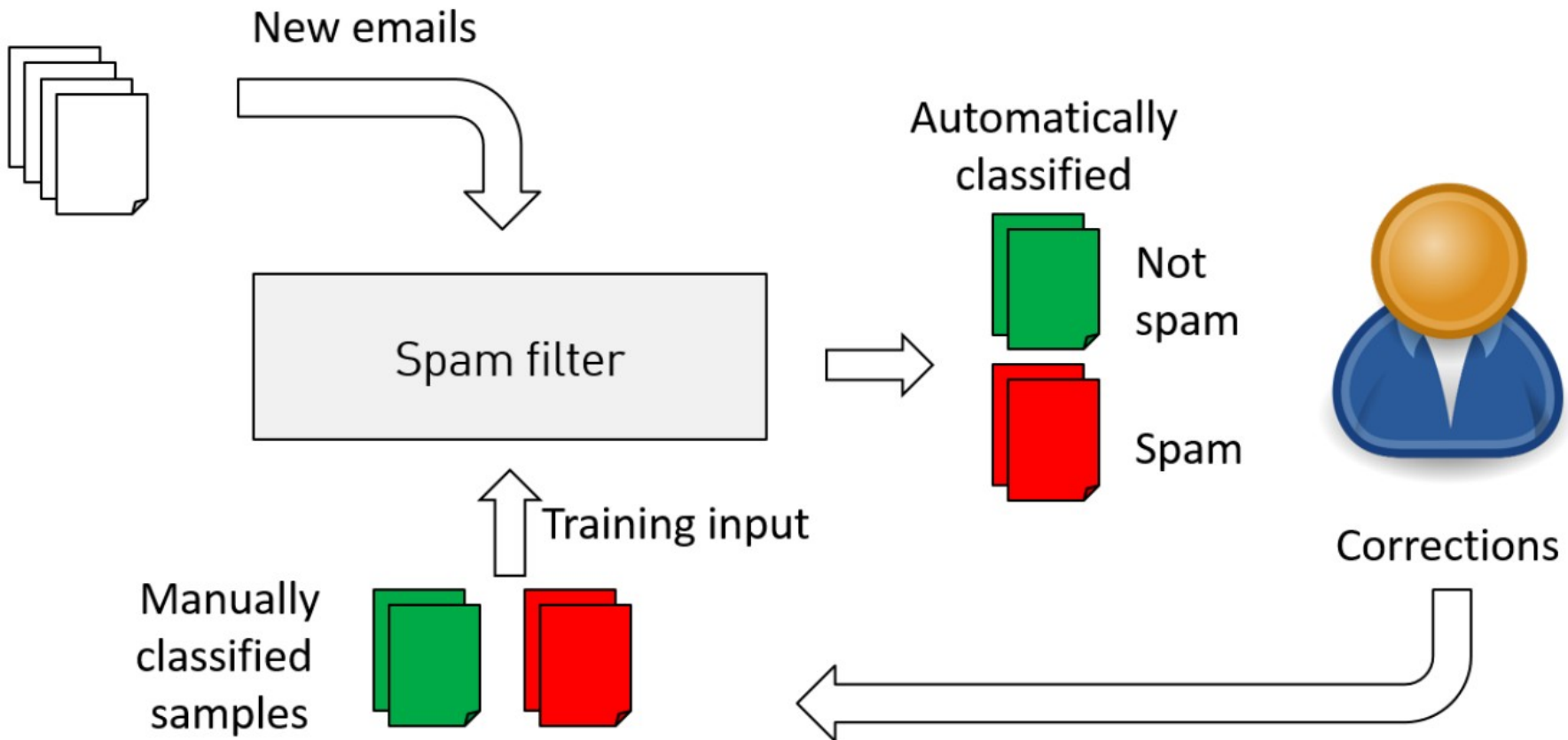
Das Modell wird also nicht explizit programmiert !!!

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

ML Anwendungen: Spam Filtering

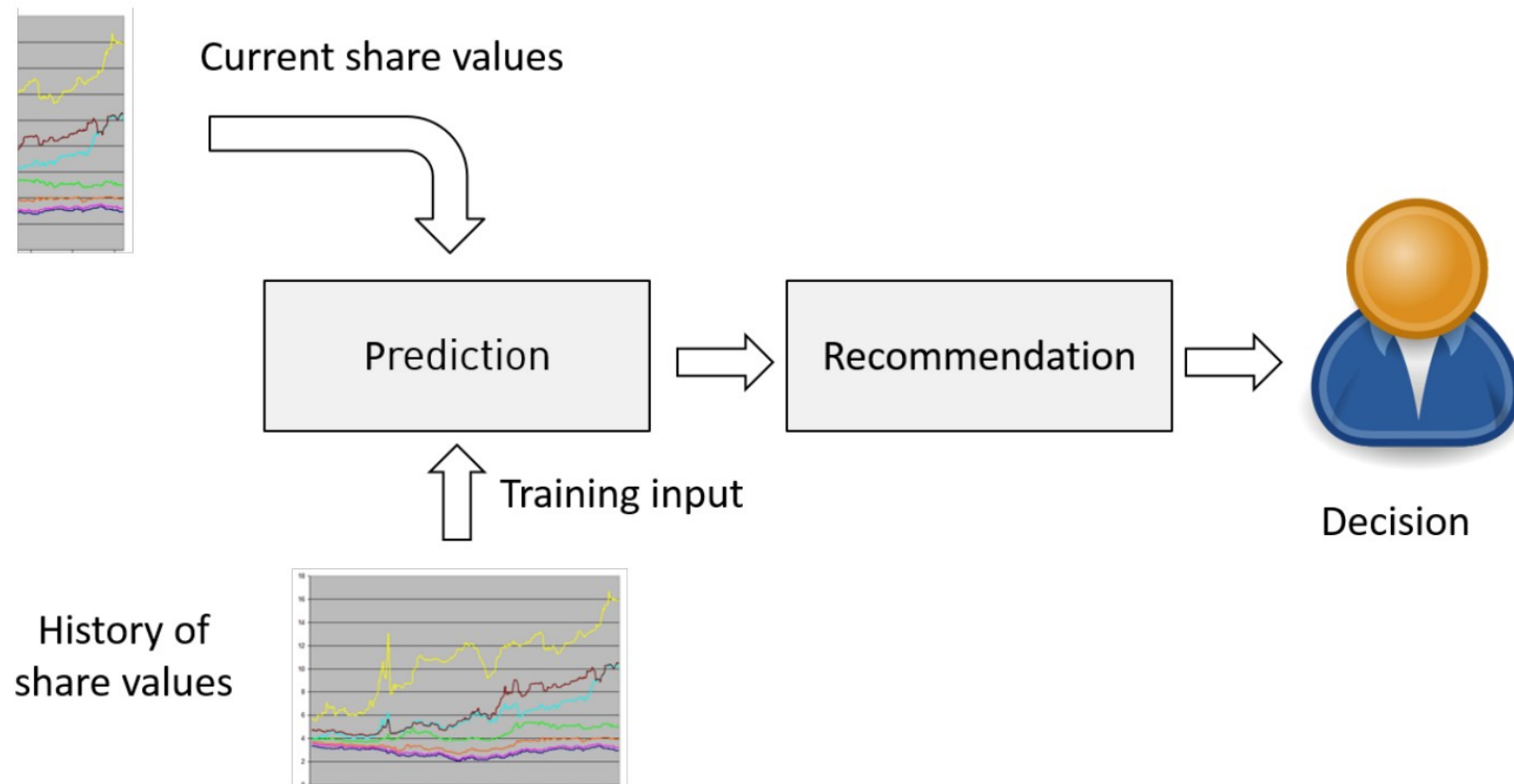
- Aufgabe: Klassifiziere neue E-Mails danach, ob sie Spam oder nicht Spam sind



Bildquelle: B. Humm "Applied Artificial Intelligence", S. 13

ML Anwendungen: Wertpapierhandel

- Aufgabe: Mache Empfehlungen, welche Aktien gekauft bzw. verkauft werden sollen

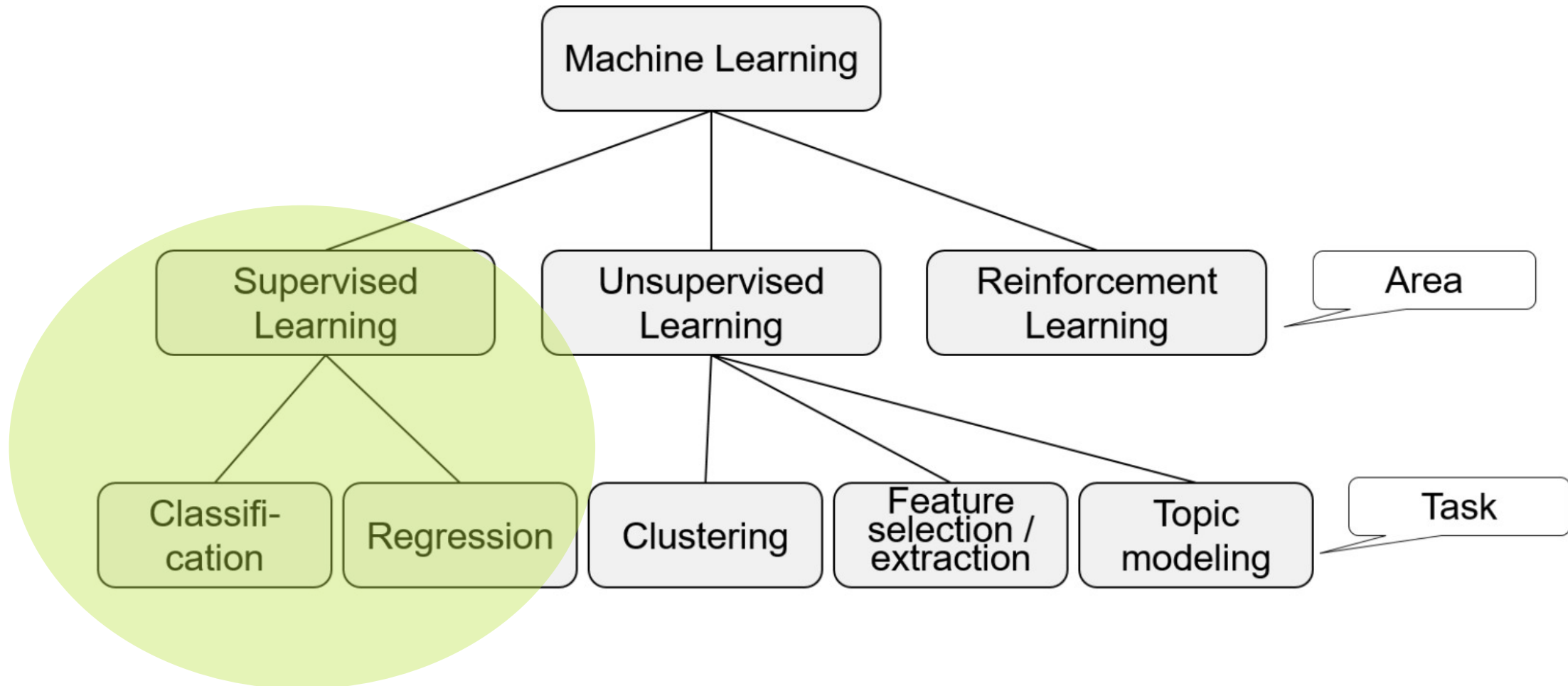


Bildquelle: B. Humm "Applied Artificial Intelligence", S. 14

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
 - Beispiel: Klassifikation der Überlebenden der Titanic
 - Kaggle
 - Analysieren von Datensätzen
 - Datensätze laden
 - Datensätze vorverarbeiten
 - ML Training
 - ML Modell zur Vorhersage anwenden
 - Mini-Test

Kategorien von ML Aufgaben



Bildquelle: B. Humm "Applied Artificial Intelligence", S. 15

Aufgaben des Supervised Learning (Überwachtes Lernen)

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,...

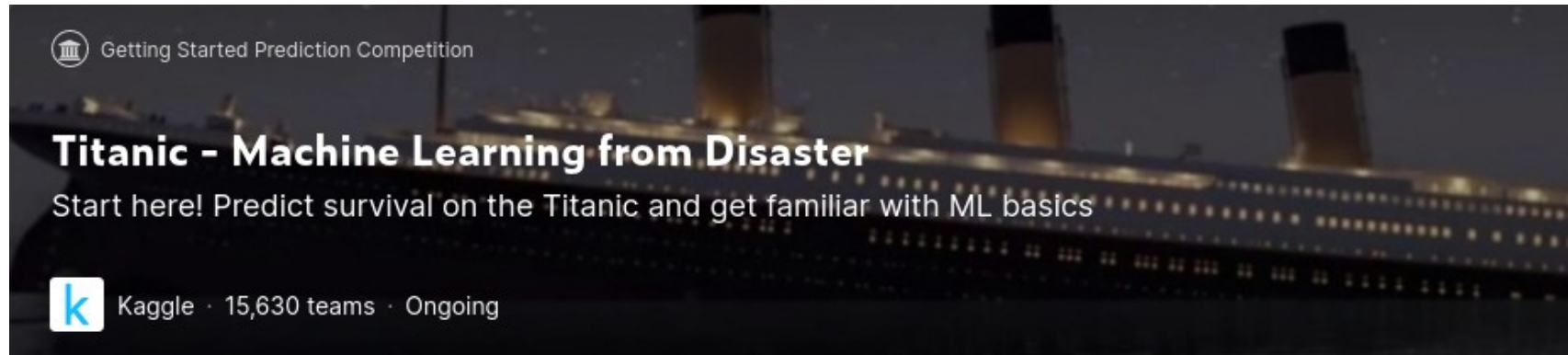
Regression

- 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 Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Beispiel: Klassifikation der Überlebenden der Titanic



Bildquelle: <https://www.kaggle.com/competitions/titanic>

- “Hello World”-Beispiel für ML
- Viele Tutorials online verfügbar, z.B. <https://www.kaggle.com/code/alexisbcook/titanic-tutorial>
- Eingabe: Informationen (“Features”) über die Passagiere, z.B. Alter, Geschlecht,...
- Ausgabe: Klassifikation nach zwei Kategorien: Überlebt bzw. Gestorben



The Challenge

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

Quelle: <https://www.kaggle.com/competitions/titanic>

Was gefällt mir nicht an dem Titanic Beispiel

- Es ist kein realistisches ML-Beispiel, aber warum ?



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

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
 - Analysieren von Datensätzen
 - Datensätze laden
 - Datensätze vorverarbeiten
 - ML Training
 - ML Modell zur Vorhersage anwenden
 - Mini-Test

Kaggle: Eine Community mit Datensätzen, Wettbewerben etc...

kaggle

Competitions

Grow your data science skills by competing in our exciting competitions. Find help in the [documentation](#) or learn about [Community Competitions](#).


[Host a Competition](#)
[Your Work](#)
 Search competitions

[Filters](#)

All Competitions 
Everything, past & present

Featured 
Premier challenges with prizes

Getting Started 
Approachable ML fundamentals

Research 
Scientific and scholarly challenges

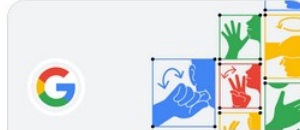


🕒 Active Competitions

Hotness  



Vesuvius Challenge - Ink Detection 
Resurrect an ancient library from the...
Featured
Code Competition · 332 Teams



Google - Isolated Sign Language Recognition 
Enhance PopSign's educational gam...
Research
Code Competition · 862 Teams



Parkinson's Freezing of Gait Prediction 
Event detection from wearable sens...
Research
Code Competition · 367 Teams

Quelle: <https://www.kaggle.com>

Daten von Kaggle herunterladen

☰

+

🔍

🏆

📊

👤

🔗

📄

🗨️

🎓

✓

Search

Overview

Data

Code

Discussion

Leaderboard

Rules

Team

Submissions

Submit Predictions

Dataset Description

Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include **gender_submission.csv**, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Files

3 files

Size

93.08 kB

Type

CSV

Data Dictionary

View more

train.csv (61.19 kB)

Download Expand Share

Detail

Compact

Column

10 of 12 columns

About this file

contains data

PassengerId	Survived	Pclass	Name
-------------	----------	--------	------

Data Explorer

93.08 kB

- gender_submission.csv
- test.csv
- train.csv

Trainingdaten
Testdata
Wettbewerbs-Einreichung

Quelle: <https://www.kaggle.com/competitions/titanic/data?select=train.csv>

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
 - Datensätze laden
 - Datensätze vorverarbeiten
 - ML Training
 - ML Modell zur Vorhersage anwenden
 - Mini-Test

Attribute des Titanic-Datensatzes

Features (ML Eingabe)

Eindeutige
Passagier-ID,
nicht benutzt
für die Klassifikation

Passagierklasse
1 = Erste Klasse
3 = Dritte Klasse

Alter
in
Jahren

Anzahl
siblings
+
spouse

Anzahl
Eltern +
Kinder

Ticket
nummer

Einstiegshafen
C=Cherbourg
S=Southampton
Q=Queenstown

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
689	690	1	1	Madill, Miss. Georgette Alexandra	female	15.0	0	1	24160	211.3375	B5	S
525	526	0	3	Farrell, Mr. James	male	40.5	0	0	367232	7.7500	NaN	Q
278	279	0	3	Rice, Master. Eric	male	7.0	4	1	382652	29.1250	NaN	Q

Label,
Ziel der
Klassifikation

0 = Gestorben
1 = Überlebt

NaN =
fehlende
Werte

Label (ML Ausgabe)

Quelle: <https://www.kaggle.com/code/gunesevitan/titanic-advanced-feature-engineering-tutorial>

Attribute des Titanic-Datensatzes (2)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
689	690	1	1	Madill, Miss. Georgette Alexandra	female	15.0	0	1	24160	211.3375	B5	S
525	526	0	3	Farrell, Mr. James	male	40.5	0	0	367232	7.7500	NaN	Q
278	279	0	3	Rice, Master. Eric	male	7.0	4	1	382652	29.1250	NaN	Q

Kategorie	Kategorisch, binär	Kategorisch, ordinal	Text	Kategorisch	Numerisch	Numerisch	Numerisch	Text	Numerisch	Text	Kategorisch
Datentyp	int	int	string	string	float	int	int	int	float	string	string
Fehlende Daten ?	nein	nein	nein	nein	ja	nein	nein	nein	ja	ja	ja

Quelle: <https://www.kaggle.com/code/gunesevitan/titanic-advanced-feature-engineering-tutorial>

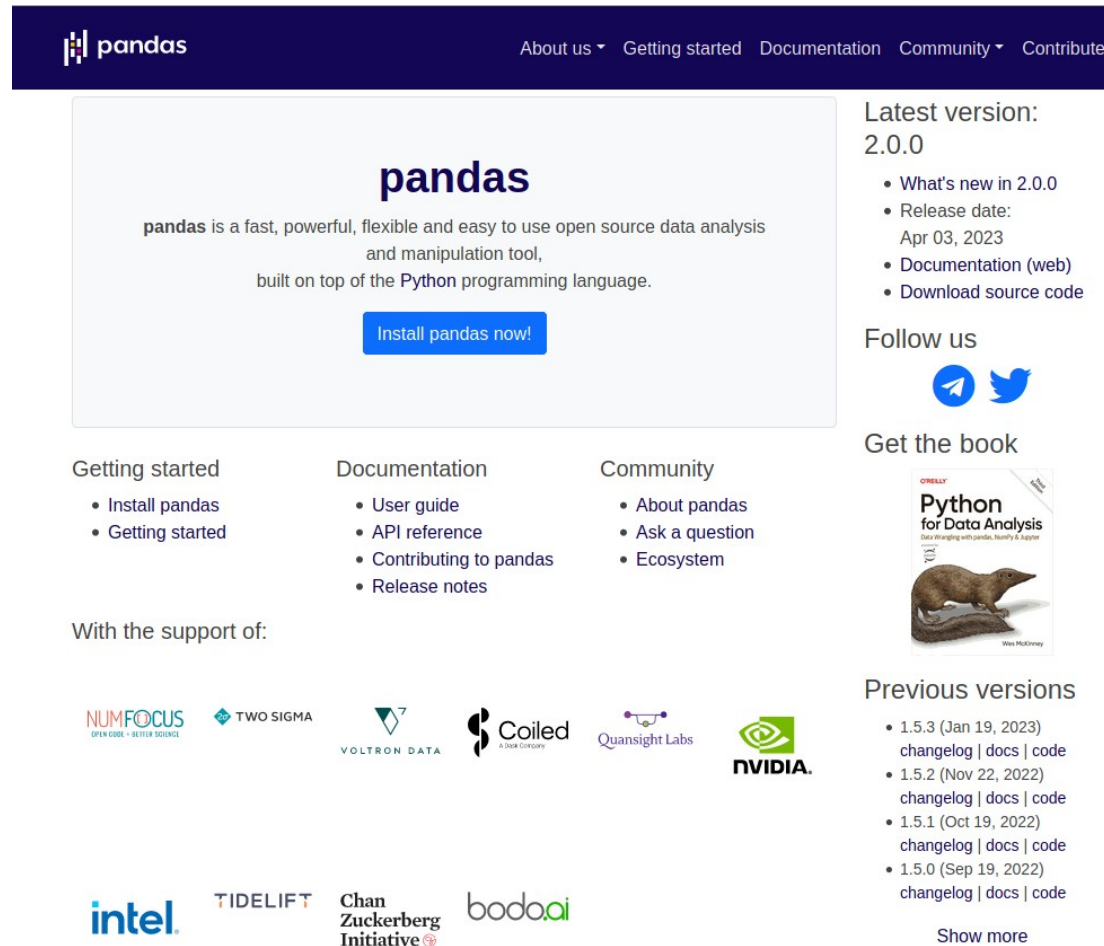
Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Das Datenanalyse-Paket pandas

pip install pandas

python3 -m pip install --user pandas



The screenshot shows the pandas website homepage. At the top is a dark blue navigation bar with the pandas logo and links for 'About us', 'Getting started', 'Documentation', 'Community', and 'Contribute'. The main content area has a light blue background with the pandas logo and a description: 'pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.' Below this is a blue button that says 'Install pandas now!'. To the right, under 'Latest version: 2.0.0', there is a list of links: 'What's new in 2.0.0', 'Release date: Apr 03, 2023', 'Documentation (web)', and 'Download source code'. Below this is a 'Follow us' section with icons for GitHub and Twitter. Further down is a 'Get the book' section featuring the cover of the book 'Python for Data Analysis' by Wes McKinney. The bottom section, 'With the support of:', displays logos for various sponsors: NUMFOCUS, TWO SIGMA, VOLTRON DATA, Coiled, Quansight Labs, NVIDIA, intel, TIDELIFT, Chan Zuckerberg Initiative, and bodai.ai.

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

Latest version:
2.0.0

- What's new in 2.0.0
- Release date: Apr 03, 2023
- Documentation (web)
- Download source code

Follow us

[GitHub](#) [Twitter](#)

Get the book

Python for Data Analysis
Data Wrangling with pandas, NumPy & matplotlib
Wes McKinney

With the support of:

NUMFOCUS OPEN CODE • BETTER SOURCE TWO SIGMA VOLTRON DATA Coiled A CODE COMPANY Quansight Labs NVIDIA intel TIDELIFT Chan Zuckerberg Initiative bodai.ai

Bildquelle: <https://pandas.pydata.org/>

Pandas Basics Cheat Sheet



Python For Data Science

Pandas Basics Cheat Sheet

Learn Pandas Basics online 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

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

Index	a	b	c	d
	3	-5	7	4

Dataframe

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

Columns	Country	Capital	Population
0	Belgium	Brussels	1190846
1	India	New Delhi	150171055
2	Brazil	Brasilia	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [1190846, 150171055, 207847528]}
>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

> Dropping

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

> Asking For Help

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

> Sort & Rank

```
>>> df.sort_index() #Sort by labels along an axis
>>> df.sort_values(by='Country') #Sort by the values along an axis
>>> df.rank() #Assign ranks to entries
```

> I/O

Read and Write to CSV

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

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Read multiple sheets from the same file
>>> xlsx = pd.ExcelFile('file.xlsx')
>>> 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()
>>> df.to_sql('mydf', engine)
```

> Selection

Also see NumPy Arrays

Getting

```
>>> s['b'] #Get one element
>>> df[1:] #Get subset of a DataFrame
>>> df['Country', 'Capital', 'Population']
1 India New Delhi 150171055
2 Brazil Brasilia 207847528
```

Selecting, Boolean Indexing & Setting

By Position

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

By Label

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

By Label/Position

```
>>> df.ix[2] #Select single row of subset of rows
Country Brazil
Capital Brasilia
Population 207847528
>>> df.ix[:, 'Capital'] #Select a single column of subset of columns
0 Brussels
1 New Delhi
2 Brasilia
>>> df.ix[1, 'Capital'] #Select rows and columns
'New Delhi'
```

Boolean indexing

```
>>> s[(s > 1)] #Series s where value is not >1
>>> s[(s < -1) | (s > 2)] #s where value is <-1 or >2
>>> df[df['Population'] > 150000000] #Use filter to adjust DataFrame
```

Setting

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

> Retrieving Series/DataFrame Information

Basic Information

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

Summary

```
>>> df.sum() #Sum of values
>>> df.cumsum() #Cumulative sum of values
>>> df.min()/df.max() #Minimum/maximum values
>>> df.idxmin()/df.idxmax() #Minimum/Maximum index value
>>> df.describe() #Summary statistics
>>> df.mean() #Mean of values
>>> df.median() #Median of values
```

> Applying Functions

```
>>> f = lambda x: x**2
>>> df.apply(f) #Apply function
>>> df.applymap(f) #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 19.0
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 19.0
b -0.0
c 5.0
d 7.0
>>> s.sub(s3, fill_value=2)
a 19.0
b NaN
c 5.0
d 7.0
>>> s.div(s3, fill_value=4)
a 19.0
b NaN
c 5.0
d 7.0
>>> s.mul(s3, fill_value=3)
```

Learn Data Skills Online at www.DataCamp.com



Bildquelle: <https://www.datacamp.com/cheat-sheet/pandas-cheat-sheet-for-data-science-in-python>

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

pd.DataFrame, die primäre pandas Datenstruktur

pandas.DataFrame

`class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)`

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

[\[source\]](#)

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Parameters: **data** : *ndarray (structured or homogeneous), Iterable, dict, or DataFrame*

Dict can contain Series, arrays, constants, dataclass or list-like objects. If data is a dict, column order follows insertion-order. If a dict contains Series which have an index defined, it is aligned by its index. This alignment also occurs if data is a Series or a DataFrame itself. Alignment is done on Series/DataFrame inputs.

If data is a list of dicts, column order follows insertion-order.

index : *Index or array-like*

Index to use for resulting frame. Will default to RangeIndex if no indexing information part of input data and no index provided.

columns : *Index or array-like*

Column labels to use for resulting frame when data does not have them, defaulting to RangeIndex(0, 1, 2, ..., n). If data contains column labels, will perform column selection instead.

dtype : *dtype, default None*

Data type to force. Only a single dtype is allowed. If None, infer.

copy : *bool or None, default None*

Copy data from inputs. For dict data, the default of None behaves like `copy=True`. For DataFrame or 2d ndarray input, the default of None behaves like `copy=False`. If data is a dict containing one or more Series (possibly of different dtypes), `copy=False` will ensure that these inputs are not copied.

Quelle: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Datenvorverarbeitung: Extrahieren der relevanten Spalten

Spalten aus einem Datensatz
extrahieren
Ein Spaltenname → Vektor
Liste von Spaltennamen → Matrix

```
# Labels extrahieren
```

```
y = train_data ["Survived"]
```

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

```
features = ["Pclass", "Sex", "SibSp", "Parch"]
```

```
X = train_data [features]
```

Namenskonventionen aus der Mathematik
Matrix: Großbuchstaben, z.B. X
Vektor: Kleinbuchstaben, z.B. y
Input: x (oder X)
Output y (oder Y)

Datenvorverarbeitung: Fehlende Werte auffüllen

```
# Fehlende Werte (n/a values) mit einem bestimmten Wert auffüllen  
data['Embarked'] = data['Embarked'].fillna('S')
```

DataFrame.fillna liefert einen modifizierten DataFrame zurück (der Original-DataFrame wird nicht verändert). Daher die Zuweisung, um ihn zu überschreiben

```
# Fehlende Werte mit dem Mittelwert aller Werte auffüllen
```

```
import numpy as np
```

```
from sklearn.impute import SimpleImputer
```

```
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
```

```
data['Age'] = imp.fit_transform(data[['Age']])
```

SimpleImputer kennt verschiedene Strategien, um fehlende Werte aufzufüllen

Quellen: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html>
<https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>

Scientific Computing Paket NumPy



The fundamental package for scientific computing with Python

LATEST RELEASE:
NUMPY 1.24.2. VIEW
ALL RELEASES.

Meet the new NumPy docs team leads

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

OPEN SOURCE

Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

Quelle: <https://numpy.org/>

ML Paket: scikit-learn

`pip install scikit-learn`

`python3 -m pip install --user scikit-learn`

The screenshot shows the scikit-learn website with the following content:

- Navigation:** Install, User Guide, API, Examples, Community, More.
- Header:** scikit-learn Machine Learning in Python. Buttons for Getting Started, Release Highlights for 1.2, and GitHub.
- Key Features:**
 - Simple and efficient tools for predictive data analysis
 - Accessible to everybody, and reusable in various contexts
 - Built on NumPy, SciPy, and matplotlib
 - Open source, commercially usable - BSD license
- Classification:** Identifying which category an object belongs to. Applications: Spam detection, image recognition. Algorithms: SVM, nearest neighbors, random forest, and more...
- Regression:** Predicting a continuous-valued attribute associated with an object. Applications: Drug response, Stock prices. Algorithms: SVR, nearest neighbors, random forest, and more...
- Clustering:** Automatic grouping of similar objects into sets. Applications: Customer segmentation, Grouping experiment outcomes. Algorithms: k-Means, spectral clustering, mean-shift, and more...
- Dimensionality reduction:** Reducing the number of random variables to consider. Applications: Visualization, Increased efficiency. Algorithms: PCA, feature selection, non-negative matrix factorization, and more...
- Model selection:** Comparing, validating and choosing parameters and models. Applications: Improved accuracy via parameter tuning. Algorithms: grid search, cross validation, metrics, and more...
- Preprocessing:** Feature extraction and normalization. Applications: Transforming input data such as text for use with machine learning algorithms. Algorithms: preprocessing, feature extraction, and more...

Bildquelle: <https://scikit-learn.org/stable/>

scikit-learn cheat sheet



Python For Data Science Scikit-Learn Cheat Sheet

Learn Scikit-Learn online 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.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :4], 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 DataFrames, are also acceptable.

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

Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
>>>                                                    y,
>>>                                                    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 algo
```

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
>>> scalar = Normalizer().fit(X_train)
>>> normalized_X = scalar.transform(X_train)
>>> normalized_X_test = scalar.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.8).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)
```

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

```
>>> knn.score(X_test, y_test) #Estimator score method
>>> from sklearn.metrics import accuracy_score #Metric scoring functions
>>> accuracy_score(y_test, y_pred)
```

Classification Report

```
>>> from sklearn.metrics import classification_report #Precision, recall, f1-score and support
>>> 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)
```

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(1,5),
>>>           '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=5, n_iter=5, random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```



Learn Data Skills Online at www.DataCamp.com

Quelle: <https://www.datacamp.com/cheat-sheet/scikit-learn-cheat-sheet-python-machine-learning>

Weitere Ideen für die Vorverarbeitung der Daten ?



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

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

ML Training

```
# ML Modell erzeugen
```

```
model = DecisionTreeClassifier ()
```

Ausgewählter Ansatz zum ML:
DecisionTreeClassifier

```
# ML Modell trainieren
```

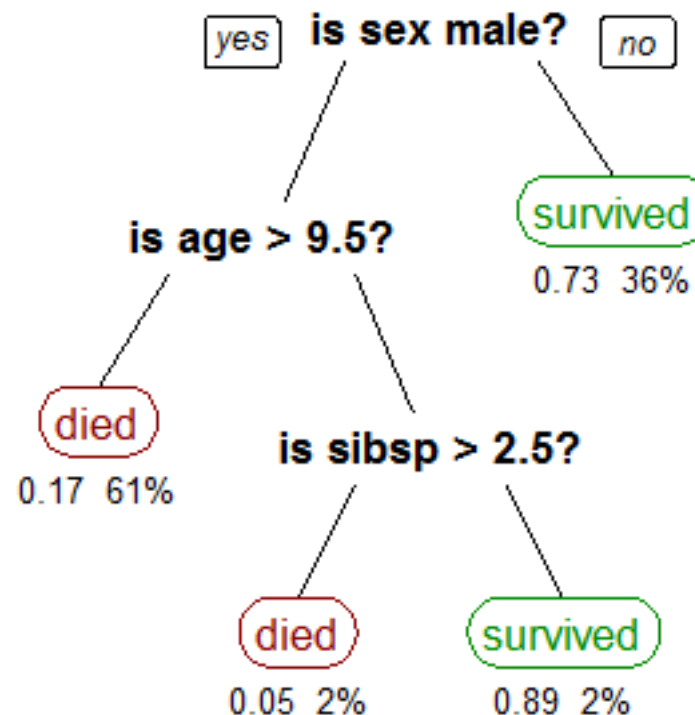
```
model.fit (X, y)
```

Trainieren (“fitting”) des
Modells mit den Features (X)
und den Labels (y)

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

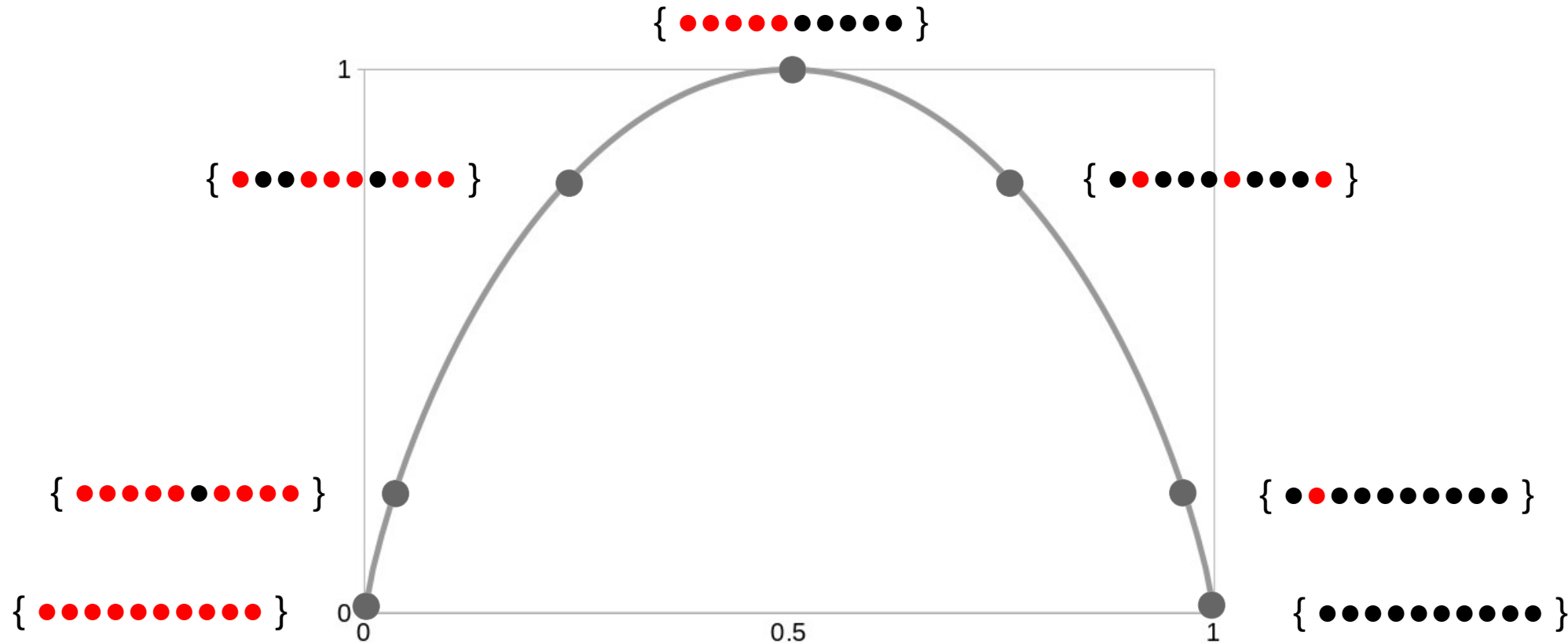
Lernen mit einem Entscheidungsbaum

- Entscheidungsbaum: Ein Baum mit den Klassen als Blättern (hier: überlebt, gestorben). Die Entscheidungen sind in den inneren Knoten (z.B. männlich?) und die Werte an den Kanten des Baums (z.B. Ja, Nein).
- Der Lernvorgang: Automatisch einen Entscheidungsbaum aus den Trainingsdaten (features und Labels) aufbauen.
- Beispiel: Entscheidungsbaum für die Klassifikation der Überlebenden der Titanic



Bildquelle: https://commons.wikimedia.org/wiki/File:CART_tree_titanic_survivors.png

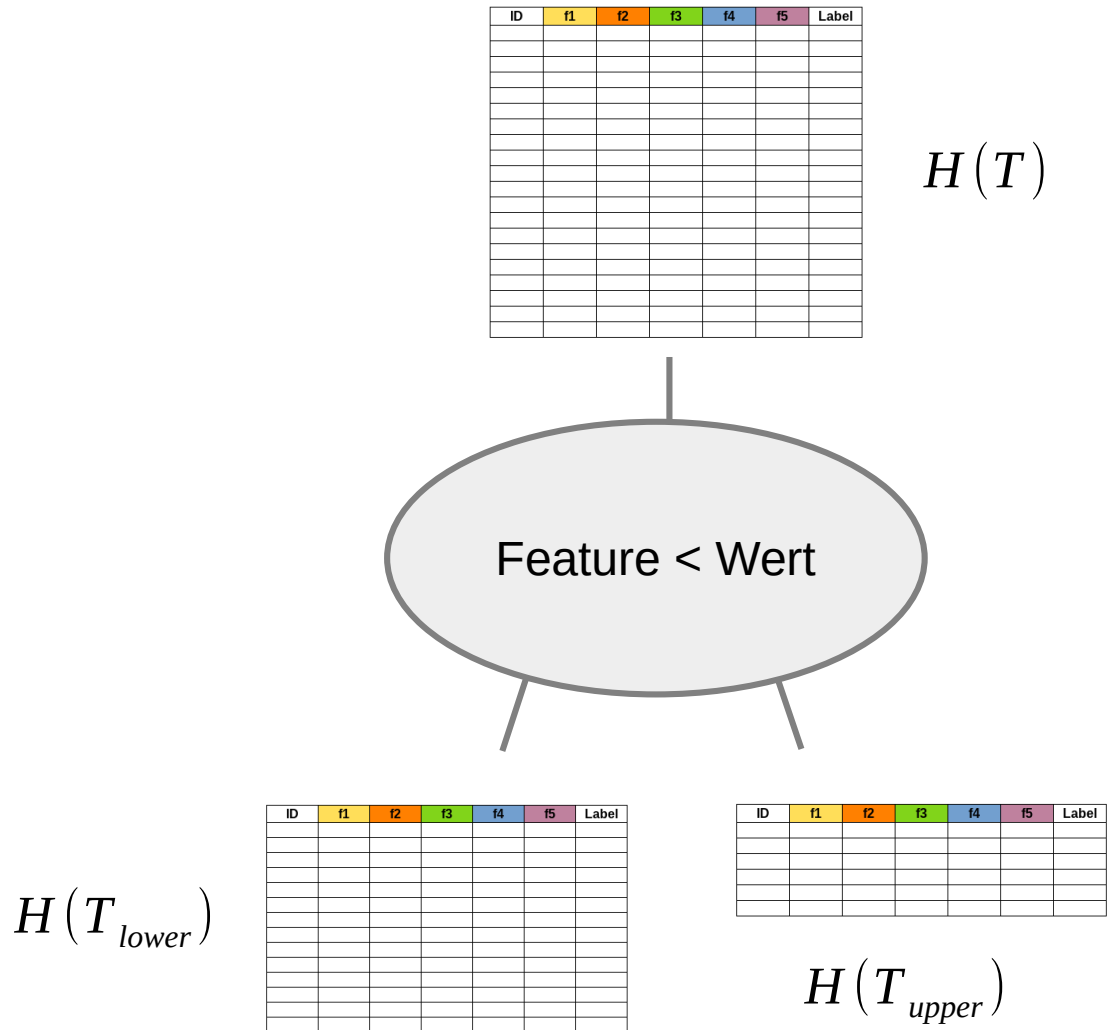
Aufbauen eines Entscheidungsbaums - Entropie



$$H(M) = \sum_{i=1}^n -c_i * \log_2(c_i) \quad \text{mit} \quad \sum_{i=1}^n c_i = 1$$

$$H(\{ \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \}) = -3/10 * \log_2(3/10) - 7/10 * \log_2(7/10) = 0.881$$

Aufbauen eines Entscheidungsbaums – Entropieverlust



Wähle Feature und Wert so, dass ein möglichst großer Entropieverlust entsteht.

$$H(T) - \frac{|T_{lower}|}{|T|} * H(T_{lower}) - \frac{|T_{upper}|}{|T|} * H(T_{upper})$$

Aufbauen eines Entscheidungsbaums – Beispiel

ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

$$H(T) = 0.992$$

f1	Label
122.5	B
153.3	A
211.2	A
275.1	A
277.5	A
354.1	B
357.6	A
366.4	A
396.6	A
399.6	A
----- ✂ -----	
409.0	B
481.7	A
482.9	B
498.2	B
510.8	B
521.8	A
683.6	B
697.4	B
710.0	A
761.8	B

$$H(T_{lower}) = 0.721$$

$$f1 < 404.3$$

$$H(T_{upper}) = 0.881$$

$$H(T) - \frac{10}{20} * H(T_{lower}) - \frac{10}{20} * H(T_{upper}) = 0.191$$

Aufbauen eines Entscheidungsbaums – Beispiel

ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

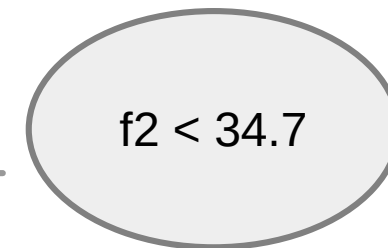
$$H(T) = 0.992$$



f2	Label
1.6	A
2.4	B
4.1	A
7.1	B
7.5	B
11.9	B
14.8	A
15.0	B
16.2	A
22.1	B
25.6	B
28.0	A
28.9	A
31.8	B
33.3	B

36.1	A
36.2	A
37.2	A
38.4	A
38.7	A

$$H(T_{lower}) = 0.97$$



$$H(T_{upper}) = 0.0$$

$$H(T) - \frac{15}{20} * H(T_{lower}) - \frac{5}{20} * H(T_{upper}) = 0.264$$

Aufbauen eines Entscheidungsbaums – Beispiel

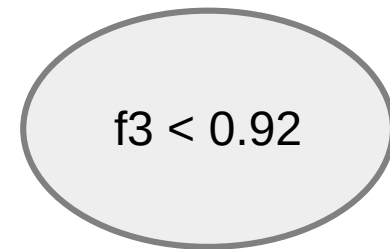
ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

$$H(T) = 0.992$$

f3	Label
0.04	A
0.05	B
0.12	A
0.20	B
0.28	A
0.31	B
0.52	A
0.56	B
0.58	A
0.62	B
0.72	A
0.73	B
0.81	B
0.83	B
0.83	A
0.87	A
0.91	B

0.94	A
0.95	A
0.99	A

$$H(T_{lower}) = 0.997$$



$$H(T_{upper}) = 0.0$$

$$H(T) - \frac{17}{20} * H(T_{lower}) - \frac{3}{20} * H(T_{upper}) = 0.144$$

Aufbauen eines Entscheidungsbaums – Beispiel

ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

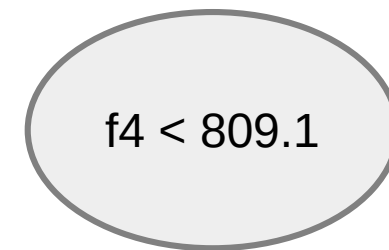
$$H(T) = 0.992$$



f4	Label
9.8	B
111.5	B
124.2	A
125.7	B
144.1	B
153.7	A
159.9	B
343.9	A
502.2	A
608.2	B
658.3	B
663.1	B
756.6	A
802.3	B

815.9	A
841.7	A
861.4	A
886.1	A
894.0	A
935.4	A

$$H(T_{lower}) = 0.94$$



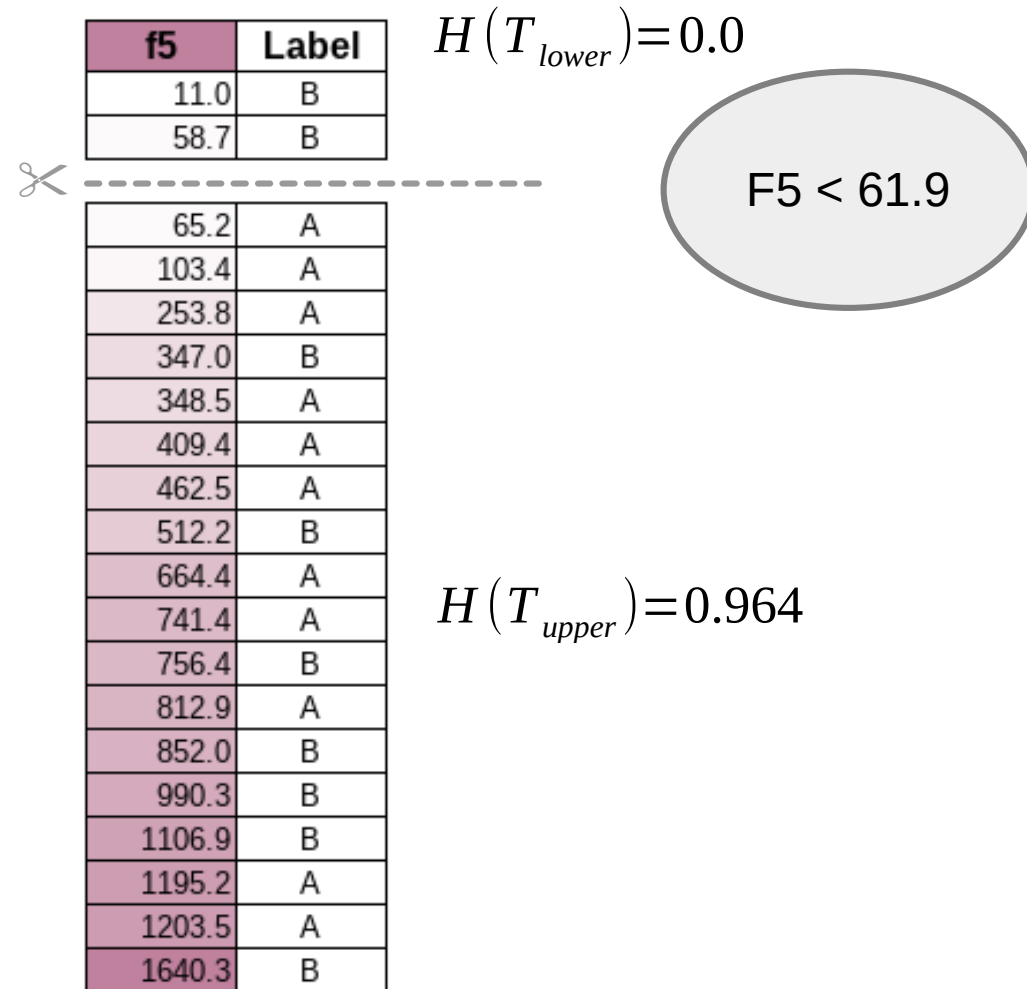
$$H(T_{upper}) = 0.0$$

$$H(T) - \frac{14}{20} * H(T_{lower}) - \frac{6}{20} * H(T_{upper}) = 0.334$$

Aufbauen eines Entscheidungsbaums – Beispiel

ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

$$H(T) = 0.992$$

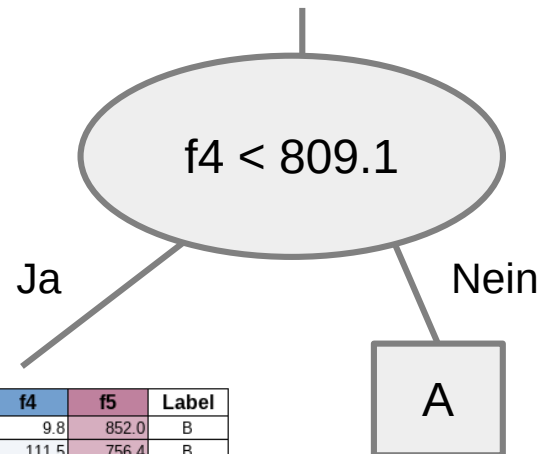


$$H(T) - \frac{2}{20} * H(t_{lower}) - \frac{18}{20} * H(T_{upper}) = 0.125$$

Aufbauen eines Entscheidungsbaums – Zwischenergebnis

ID	f1	f2	f3	f4	f5	Label
1	354.1	15.0	0.73	802.3	347.0	B
2	366.4	28.9	0.99	153.7	348.5	A
3	697.4	31.8	0.83	9.8	852.0	B
4	521.8	36.1	0.87	343.9	1195.2	A
5	482.9	22.1	0.31	159.9	1106.9	B
6	510.8	7.1	0.05	663.1	58.7	B
7	710.0	38.4	0.52	861.4	812.9	A
8	275.1	36.2	0.12	841.7	409.4	A
9	122.5	7.5	0.91	111.5	756.4	B
10	357.6	16.2	0.28	815.9	664.4	A
11	761.8	2.4	0.62	658.3	11.0	B
12	498.2	33.3	0.81	125.7	512.2	B
13	683.6	11.9	0.56	608.2	1640.3	B
14	481.7	4.1	0.95	756.6	253.8	A
15	153.3	1.6	0.58	502.2	65.2	A
16	399.6	37.2	0.94	894.0	741.4	A
17	396.6	38.7	0.72	886.1	1203.5	A
18	211.2	28.0	0.04	124.2	462.5	A
19	409.0	25.6	0.20	144.1	990.3	B
20	277.5	14.8	0.83	935.4	103.4	A

$$H(T) = 0.992$$



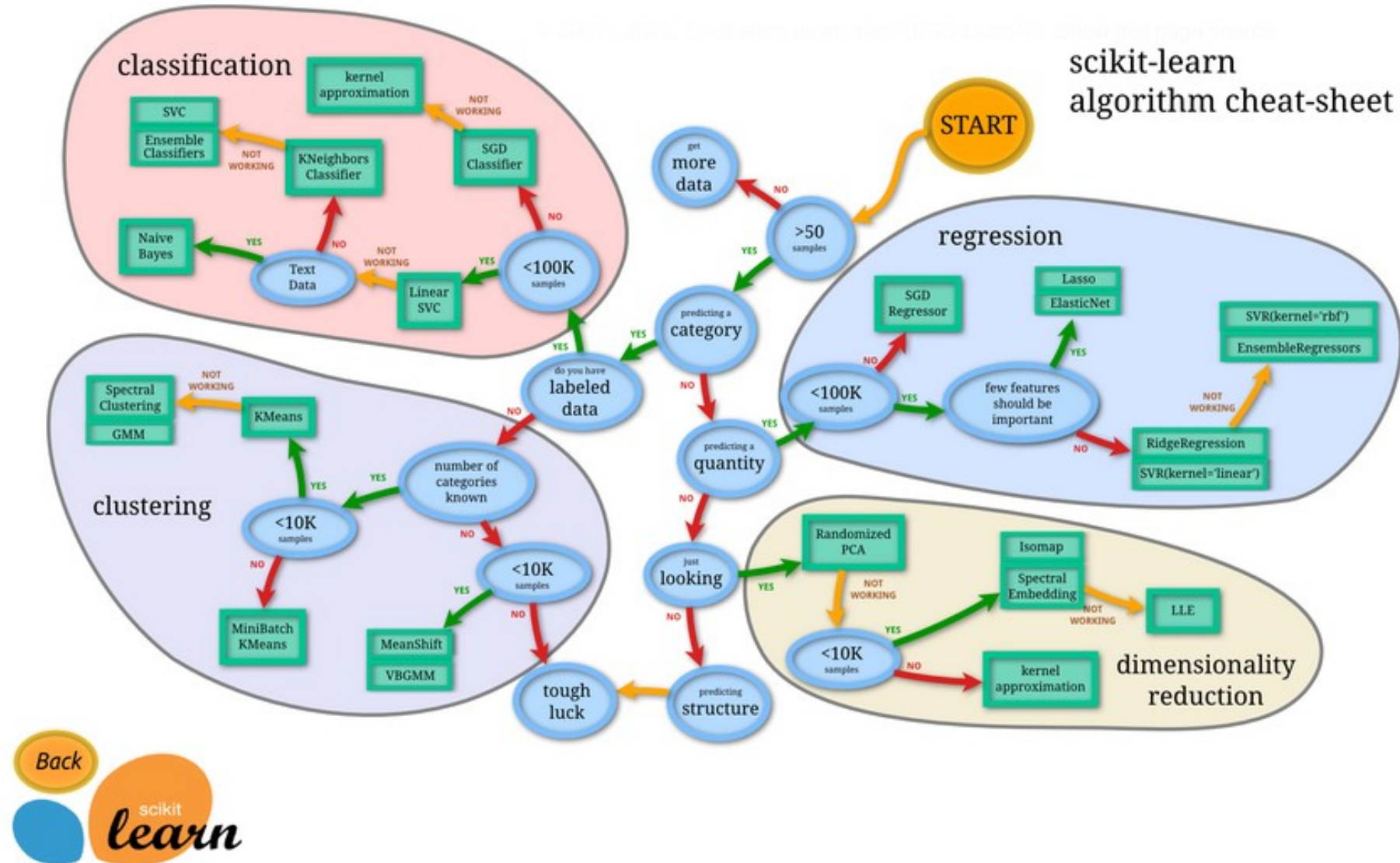
$$H(T_{lower}) = 0.94$$

ID	f1	f2	f3	f4	f5	Label
3	697.4	31.8	0.83	9.8	852.0	B
9	122.5	7.5	0.91	111.5	756.4	B
18	211.2	28.0	0.04	124.2	462.5	A
12	498.2	33.3	0.81	125.7	512.2	B
19	409.0	25.6	0.20	144.1	990.3	B
2	366.4	28.9	0.99	153.7	348.5	A
5	482.9	22.1	0.31	159.9	1106.9	B
4	521.8	36.1	0.87	343.9	1195.2	A
15	153.3	1.6	0.58	502.2	65.2	A
13	683.6	11.9	0.56	608.2	1640.3	B
11	761.8	2.4	0.62	658.3	11.0	B
6	510.8	7.1	0.05	663.1	58.7	B
14	481.7	4.1	0.95	756.6	253.8	A
1	354.1	15.0	0.73	802.3	347.0	B

ID	f1	f2	f3	f4	f5	Label
10	357.6	16.2	0.28	815.9	664.4	A
8	275.1	36.2	0.12	841.7	409.4	A
7	710.0	38.4	0.52	861.4	812.9	A
17	396.6	38.7	0.72	886.1	1203.5	A
16	399.6	37.2	0.94	894.0	741.4	A
20	277.5	14.8	0.83	935.4	103.4	A

$$H(T_{upper}) = 0$$

Ideen für weitere ML Ansätze: Das ML algorithm cheat sheet



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

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Das ML Modell zur Vorhersage anwenden

```
# Labels für den Testdatensatz vorhersagen  
predictions = model.predict(X_test)
```

Vorhersage aller Fälle in
dem Testdatensatz

Wichtig: Der Testdatensatz
muss genau so wie der
Trainingsdatensatz
vorverarbeitet worden sein

Von Kaggle benutztes Format für die Einsendung der Vorhersagen

	A	B	C
1	PassengerId	Survived	
2	892	0	
3	893	1	
4	894	0	
5	895	0	
6	896	1	
7	897	0	
8	898	1	
9	899	0	
10	900	1	
11	901	0	
12	902	0	
13	903	0	
14	904	1	
15	905	0	
16	906	1	
17	907	1	
18	908	0	
19	909	0	

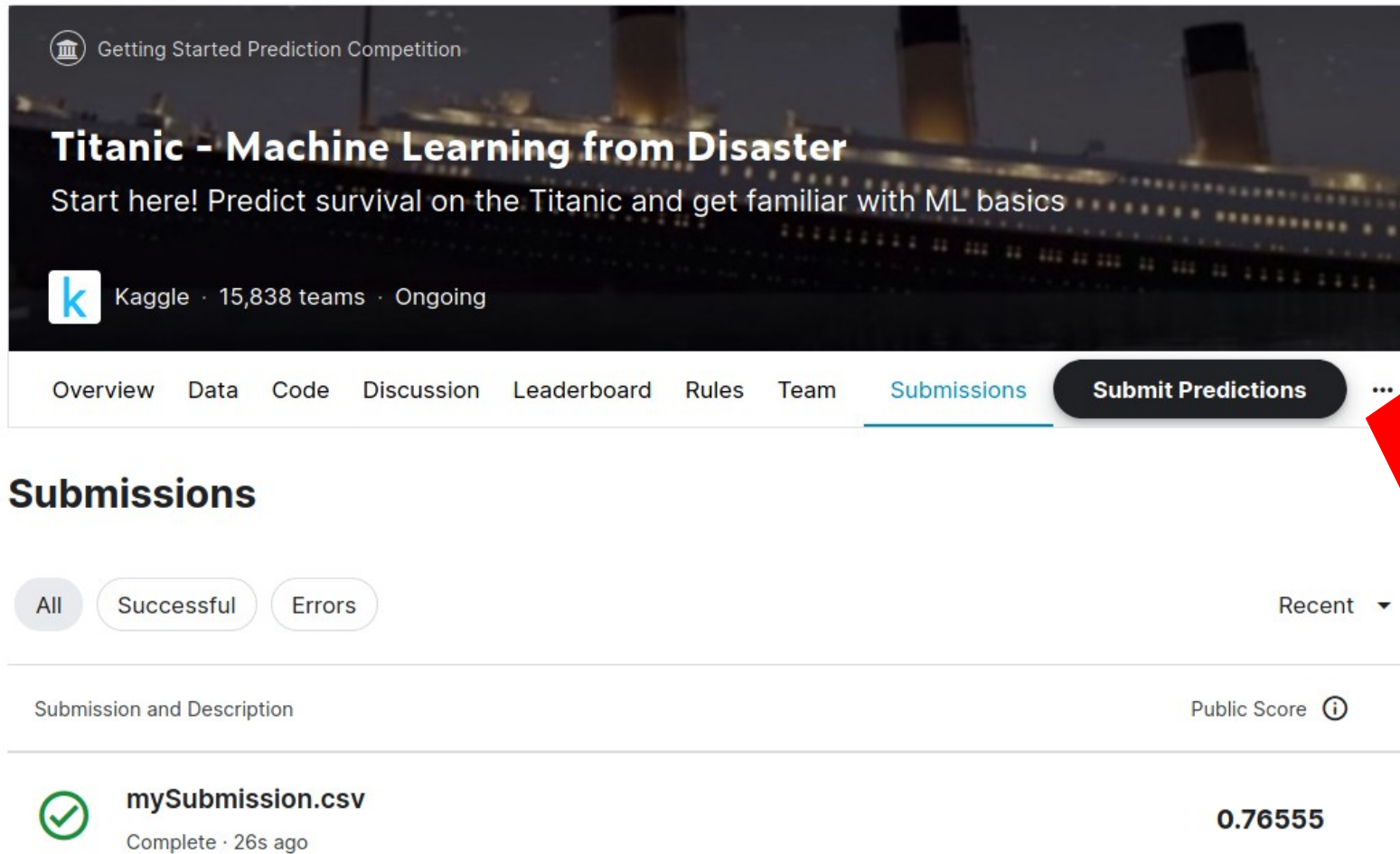
Erzeugen der CSV-Datei für die Vorhersagen

```
# Erzeugen der CSV-Datei mit den Vorhersagen des Modells  
output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})  
output.to_csv('data/submission.csv', index=False)
```

DataFrame mit den zwei Spalten PassengerId und Survived

Schreiben des DataFrame in eine CSV-Datei

Einreichen der Lösung in den Kaggle-Wettbewerb



Getting Started Prediction Competition

Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Kaggle · 15,838 teams · Ongoing

Overview Data Code Discussion Leaderboard Rules Team Submissions Submit Predictions ...

Submissions

All Successful Errors Recent ▾

Submission and Description	Public Score ⓘ
 mySubmission.csv Complete · 26s ago	0.76555

Bildquelle: <https://www.kaggle.com/competitions/titanic/submissions>

Überprüfen der eigenen Position im Kaggle Leaderboard

Overview	Data	Code	Discussion	Leaderboard	Rules	Team	Submissions	Submit Predictions	...
12376	Muhammad Zohaib #4						0.76555	2	36m
12377	matzeseeger						0.76555	1	24m
12378	Alexander del Pino						0.76555	1	8m
 Your Best Entry! Your most recent submission scored 0.76555, which is the same as your previous score. Keep trying!									
12379	Nikita Kuzlyaev						0.76315	1	2mo
12380	Jatin Kumar #4						0.76315	1	2mo
12381	Nimo Xie						0.76315	1	2mo
12382	Hikaru20						0.76315	1	2mo
12383	Gormelin						0.76315	1	2mo
12384	Antonio #11						0.76315	4	2mo
12385	Didac Sheruzum						0.76315	1	2mo
12386	chandra #3						0.76315	1	2mo
12387	Tiiino						0.76315	2	2mo
12388	Ashish Goyal #3						0.76315	2	2mo
12389	Ekaterina Afonina						0.76315	4	2mo

Bildquelle: https://www.kaggle.com/competitions/titanic/leaderboard#user_row

Aufbau

- Überblick
- ML Anwendungen
- Kategorien von ML Aufgaben
- Beispiel: Klassifikation der Überlebenden der Titanic
- Kaggle
- Analysieren von Datensätzen
- Datensätze laden
- Datensätze vorverarbeiten
- ML Training
- ML Modell zur Vorhersage anwenden
- Mini-Test

Mini-Test “ML Klassifikation”

- Was ist ML?
- Wann soll man ML benutzen, wann nicht?
- Nennen Sie ML Anwendungen
- Erklären Sie die ML Kategorie “Supervised Learning”
- Erklären Sie die ML Aufgabe Klassifikation
- Was ist Kaggle?
- Warum ist es wichtig, einen Datensatz zu analysieren?
- Warum ist die Vorverarbeitung wichtig? Nennen Sie hierzu Beispiele
- Was ist ML Training?
- Wie baut man einen Entscheidungsbaum auf?
- Wie sagt man neue Fälle vor?