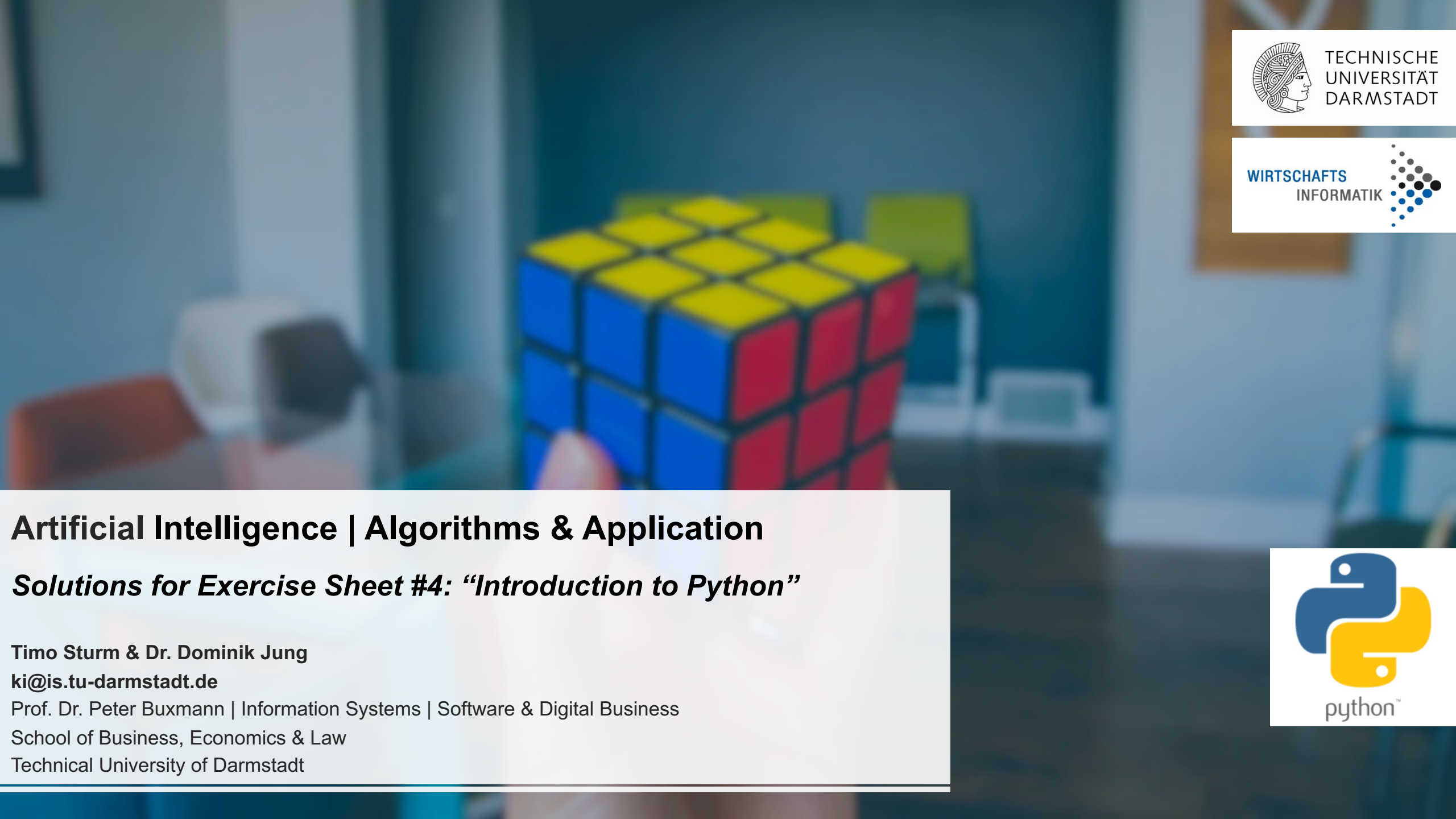


Artificial Intelligence | Algorithms & Application

Solutions for Exercise Sheet #4: “Introduction to Python”

Timo Sturm & Dr. Dominik Jung
ki@is.tu-darmstadt.de
Prof. Dr. Peter Buxmann | Information Systems | Software & Digital Business
School of Business, Economics & Law
Technical University of Darmstadt

Exercise 4.1: General Python Capabilities (1/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *1-a) Write the Pythagorian Theorem using latex code.*

Solution:

- You can simply use:

```
%%latex
$$c = \sqrt{a^2 + b^2}$$
```

Task: *1-b) What is the difference between "%magic" and "%%magic"?*

Solution:

- Magic commands with a single "%" are *called "**line magic**"* and only refer to the line in which they are inserted.
- Magic comands with "%%" are called "***cell magic***" and refer to the whole cell.

Exercise 4.1: General Python Capabilities (2/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *2-a) Create a python tuple with the following input: {spicy soup, spicy chicken, spicy salad, simple soup, simple chicken, raw chicken}*
→ Why is it usually not practical to use tuples?

Solution:

- `tup =("spicy soup",…)`
- The use of tuples is only reasonable if you have a set of data that you never want to change or when you want to make sure it does not get changed.

Exercise 4.1: General Python Capabilities (3/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *2-b) Transform the tuple into a pandas list. Then, transform it into a pandas dataframe with the two columns “adjective” and “noun”.*

Solution:

[Hint: To import the frameworks, they need to be installed in the python environment you are using. **]**

- ```
  import pandas as pd
  import numpy as np
  import matplotlib.pyplot as plt

  food = list(tup)
  y = list()

  for i in range(len(food)):
      x = food[i].split()
      y.append(x)
  df = pd.DataFrame(y, columns=['adjective','noun'])
  ```

Task: *2-c) Replace “spicy“ with “peppery” for all dishes.*

Solution:

- ```
  df['adjective'].replace('spicy','peppery',inplace=True)
  ```

Exercise 4.1: General Python Capabilities (4/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *3-a) Create a class "child" having the attributes name, age, height, and weight*

Solution:

```
class Child:
    def __init__(self,name, age, height, weight):
        self.name = name
        self.age = age
        self.height = height
        self.weight = weight
```

Task: *3-b) Now, create a class "adult" that inherits the attributes of "child" and has the additional attribute occupation.*

Solution:

```
class Adult(Child):
        def __init__(self,name, age, height, weight,occupation):
            super().__init__(name, age, height, weight)
            #Child.__init__(self,name, age, height, weight)  (this line is an alternative to the one above (uncomment it to use it instead))
            self.occupation = occupation
```

Exercise 4.1: General Python Capabilities (5/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *4-a) Generate a pandas dataframe with size 10,000x3 filled with random integers between 0-1,000 and the column labels “A”, “B”, and “C”.*

Solution:

- `rnd = pd.DataFrame(np.random.randint(0,1000,size=(10000,3)), columns=list('ABC'))`

Task: *4-b) Measure how long the creation of that dataframe takes and how much memory it uses*

Solution:

- Use `%time` to get the “wall time“.
- For more precice measures use `%timeit`
- The memory of some dataframe named “*rnd*“ can be accessed with `rnd.info()`

Exercise 4.1: General Python Capabilities (6/6)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *4-c) Provide some basic statistical values of the dataframe's data*

Solution:

- `rnd.describe()`

Task: *4-d) Sort the first column and visualize the data of that column using some visualization of your choice.*

Solution:

- `rnd = rnd.sort_values(by=['A'])`
- Matplotlib includes various command to visualize data. A simple method is a boxplot, which does not require any further specification:

```
plt.boxplot(rnd["A"])
```

→ You should try out further visualizations to get familiar with Python's visualization capabilities

Exercise 4.2: Titanic (1/3)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *a) First, you import the csv file into a dataframe.*

Solution:

- In case you opened a new python notebook, you need to import the frameworks of the previous tasks.
- If you store the csv file in the same folder as your notebook or python file you can import it as a pands dataframe using the following command:

```
data = pd.read_csv('titanic.csv')
```

Task: *b) You select the columns with rather high explanatory power for "survived" by using your personal logic. Then, you evaluate your logic by having a look at a correlation matrix. What does it tell you?*

Solution:

- The selection of single columns can be done with:

```
data_selected = data[['Survived','Sex','Age','Pclass','SibSp','Parch','Fare']]
```

- The correlation table can be used to get an impression of the correlations between "survived" and other values. However, you need to be careful because the range and type of data per column significantly varies.
  → E.g. "Sex" is a non-numerical value and cannot be represented and for "Age" the range of values is a lot greater than for other values.

```
data_selected.corr()
```

	Survived	Age	Pclass	SibSp	Parch	Fare
Survived	1.000000	-0.077221	-0.338481	-0.035322	0.081629	0.257307
Age	-0.077221	1.000000	-0.369226	-0.308247	-0.189119	0.096067
Pclass	-0.338481	-0.369226	1.000000	0.083081	0.018443	-0.549500
SibSp	-0.035322	-0.308247	0.083081	1.000000	0.414838	0.159651
Parch	0.081629	-0.189119	0.018443	0.414838	1.000000	0.216225
Fare	0.257307	0.096067	-0.549500	0.159651	0.216225	1.000000

Exercise 4.2: Titanic (2/3)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *c) Now, you are trying to comprehend the quality of the data again. How many missing attribute values are there? You decide to fill missing values with the respective column average values.*

Solution:

- ```
  data_selected = data[['Survived','Sex', 'Age','Pclass','SibSp','Parch','Fare']]
  print(data_selected.isnull().sum())

  data2 = data_selected.fillna(data_selected.mean())
  ```

Task: *d) You create a pivot chart indicating the means of “Age” and “Sex” depending on survival.*

Solution:

- First, “Sex“ has to be converted to numerical values. Then a simple pivot table can be used.

- ```
  data2['Sex'].replace(['male','female'],[1,0],inplace=True)
  data2.pivot_table(index='Survived',values=['Age','Sex'],aggfunc='mean')
  ```

Exercise 4.2: Titanic (3/3)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *e) Lastly, you perform binning for "Age" again by defining the intervals manually (use the bin sizes of Exercise 3.3). Add the binned values as a new column to your dataset. Compute the correlation with "survived" and compare the binned age with the numeric age.*

Solution:

- ```
  intervals = [(0,19),(19,45),(45,65),(65, 200]]
  bins = pd.IntervalIndex.from_tuples(intervals)
  age_cat = pd.cut(data2["Age"], bins)
  data2["age_cat"] = age_cat
  data2['age_category'] = data2.age_cat.cat.codes
  # data2['age_cat'].replace(['(0, 19]',`(19, 45]',`(45, 65]`, `(65,200]`],[1,2,3,4],inplace=True)
  ```
  (the last line is an alternative to code above (uncomment it to use it instead))
  ```

Exercise 4.3: Wine Quality (1/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *a) You import the data and start exploring to gather a better understanding of it by checking, e.g., data types, number of columns and rows, number of missing values, etc.*

Solution:

- Import required frameworks and the CSV file:

```
import pandas as pd
import matplotlib.pyplot as plt

wine = pd.read_csv("winequality.csv")
```

- You can use `wine.info()` to get an overview of contained columns, number of rows, respective data types, and which columns hold any missing values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null float64
volatile acidity        1599 non-null float64
citric acid             1599 non-null float64
residual sugar          1599 non-null float64
chlorides               1599 non-null float64
free sulfur dioxide     1599 non-null float64
total sulfur dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

- You can use `wine.isnull().sum()` to get the number of missing values per column. In this case, you could have already seen the number of missing values by using `info()` as there are no missing values contained ☺

```
Out[4]: fixed acidity           0
        volatile acidity        0
        citric acid             0
        residual sugar          0
        chlorides               0
        free sulfur dioxide     0
        total sulfur dioxide    0
        density                 0
        pH                      0
        sulphates               0
        alcohol                 0
        quality                 0
        dtype: int64
```

Exercise 4.3: Wine Quality (2/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *b) Then, you decide to bin the "quality" column into two categories "mediocre" and "excellent". However, before you do, you have to get an overview of the quality values and their distribution by visualizing them in a fitting plot. How many data points did you place in each category?*

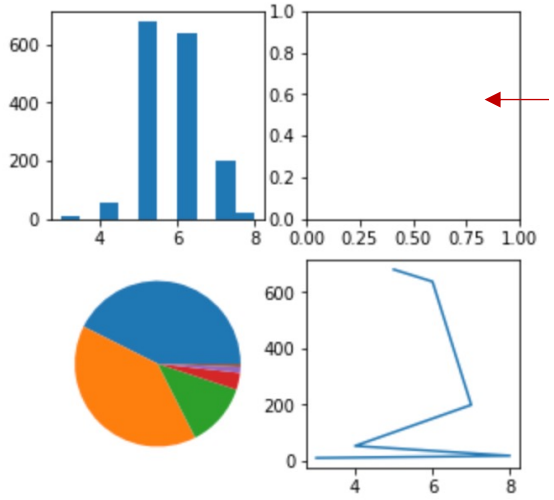
Solution:

- Exploring the data:

```
count = wine["quality"].value_counts()
quality = pd.Series(wine["quality"])
quality = quality.sort_values(ascending=True)
print(count)
fig, chart = plt.subplots(2, 2, figsize=(5,5))
chart[0,0].hist(quality)
chart[1,0].pie(count)
chart[1,1].plot(count)
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64

[<matplotlib.lines.Line2D at 0x121bcbd90>]
```



You can add a fourth visualization here

Choose visualizations wisely as, e.g., this line chart does not make it easy to comprehend the desired information

- Binning:

```
bins=(2,6.9,8)
groups=['mediocre','excellent']
wine['quality'] = pd.cut(wine['quality'],bins=bins,labels=groups)
wine['quality'].unique()
```

- Number of data points in each bin:

```
print(wine['quality'].value_counts())
```

```
mediocre     1382
excellent     217
```

Exercise 4.3: Wine Quality (3/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *c) Afterwards, you set the “quality” column as the label.*

Solution:

- ```
  import sklearn.preprocessing as spp
  LE=spp.LabelEncoder()

  wine['quality‘]= LE.fit_transform(wine['quality‘])
  ```

NOTE: Please note that you do not have to do it like this by using the encoder. This simply encodes the values of our target variable to 0 and 1 (which some algorithms require) which you could also do using some other function.
  ```

Exercise 4.3: Wine Quality (4/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *d) Then, you decide to scale the data using the scikit-learn function "StandardScaler". What does it do?*

Solution:

- ```
  import sklearn.preprocessing as spp
  scaler=spp.StandardScaler()

  X = wine.drop('quality',axis=1)
  X = scaler.fit_transform(X)
  ```

Resulting scaled data:

```
array([[-0.52835961,  0.96187667, -1.39147228, ...,  1.28864292,
        -0.57920652, -0.96024611],
       [-0.29854743,  1.96744245, -1.39147228, ..., -0.7199333 ,
         0.1289504 , -0.58477711],
       [-0.29854743,  1.29706527, -1.18607043, ..., -0.33117661,
        -0.04808883, -0.58477711],
       ...,
       [-1.1603431 , -0.09955388, -0.72391627, ...,  0.70550789,
         0.54204194,  0.54162988],
       [-1.39015528,  0.65462046, -0.77526673, ...,  1.6773996 ,
         0.30598963, -0.20930812],
       [-1.33270223, -1.21684919,  1.02199944, ...,  0.51112954,
         0.01092425,  0.54162988]])
```

- When to use StandardScaler:

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample `x` is calculated as:

$$z = (x - u) / s$$

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the **`transform`** method.

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

For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger that others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Excerpt from: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
```

Exercise 4.3: Wine Quality (5/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *e) To finalize your preprocessing, you split the data into a training and test set.*

Solution:

- ```
  import sklearn.model_selection as model_selection

  X = wine.drop('quality',axis=1)
  X = scaler.fit_transform(X)
  y = wine['quality‘]

  X_train, X_test, y_train, y_test = model_selection.train_test_split(X,y, test_size = 0.33, random_state = 42)
  ```
  ```

Exercise 4.3: Wine Quality (6/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *f) You decide to train several models using the following algorithms:*
Random Forest, Linear Regression, Support Vector Machine, Decision Tree, and Neural Network.

Solution:*

- Required Imports:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn import svm
from sklearn import tree
from sklearn.neural_network import MLPClassifier
```

- Example: Training and Testing a Random Forest (with 150 trees):

```
forest = RandomForestClassifier(n_estimators=150)
forest.fit(X_train, y_train)                    # Train the classifier
forest_predict = forest.predict(X_test)         # Apply the trained classifier to the test set
```

- The other algorithms can be used similarly

* **NOTE #1:** Don‘t worry if you do not (yet) understand the algorithms listed here. We ask you to use some more complex algorithms in this task to show you that even such complex algorithms can be integratet easily with python libraries today.

* **NOTE #2:** See solution python file for a more detailed solution.

Exercise 4.3: Wine Quality (7/7)

WIRTSCHAFTS INFORMATIK

TECHNISCHE UNIVERSITÄT DARMSTADT

Task: *g) To evaluate the models, you make use of the classification report and the respective confusion matrices. How do you interpret the results?*

Solution:*

- Required Import:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

- Example: Analyzing the performance metrics of the trained Random Forest (with 150 trees):

```
print(classification_report(y_test, forest_predict))
print(confusion_matrix(y_test, forest_predict))
print("\n")
print(accuracy_score(y_test, forest_predict))
```

```
              precision    recall  f1-score   support

           0       0.61      0.44      0.51        77
           1       0.91      0.95      0.93       451

    accuracy                           0.88       528
   macro avg       0.76      0.70      0.72       528
weighted avg       0.86      0.88      0.87       528

[[ 34  43]
 [ 22 429]]


0.8768939393939394
```

- Please be aware that you have to treat the linear regression differently as it performs a regression and no classification
  E.g., you can calculate its root mean square error instead:

```
print(sqrt(mean_squared_error(y_test, LR_predict)))
```

*** NOTE:** See solution python file for a more detailed solution.