Data Science & Artificial Intelligence

Summer Term 2022

L06 Classification and Regression

D. Schneider, B. Stuhr, J. Haselberger

Agenda

Theory (90 min) | Break (15 min) | Exercise (90 min)

L06.1 (20 min)
- Linear Regression

L06.2 (20 min)
- Classification
- SVM
- Kernel

L06.3 (20 min)
- Performance Evaluation

L06.4 (20 min)
- Logistic Regression

E06.1 (30 min)
- Linear Regression

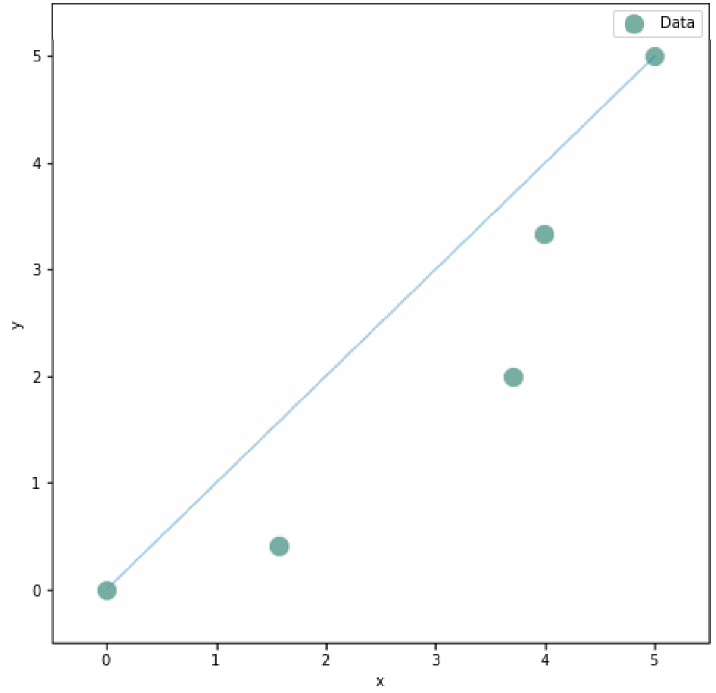
E06.2 (30 min)
- Support Vector Machine

E06.3 (30 min)
- Logistic Regression

Linear Regression

- Linear Regression helps to find the linear relationship between two continuous variables
- Variables are distinguished into independent (x) and dependent variables (y)
- Use linear model to describe the relationship between the variables

$$y = b_0 + b_1 x_1$$



- Which model is the best fitting one?

Finding the optimal line

$$\mathbf{X} = \begin{bmatrix} 0.00, 1.57, 3.71, 3.99, 5.00 \\ 0.00, 0.41, 2.00, 3.34, 5.00 \end{bmatrix}$$

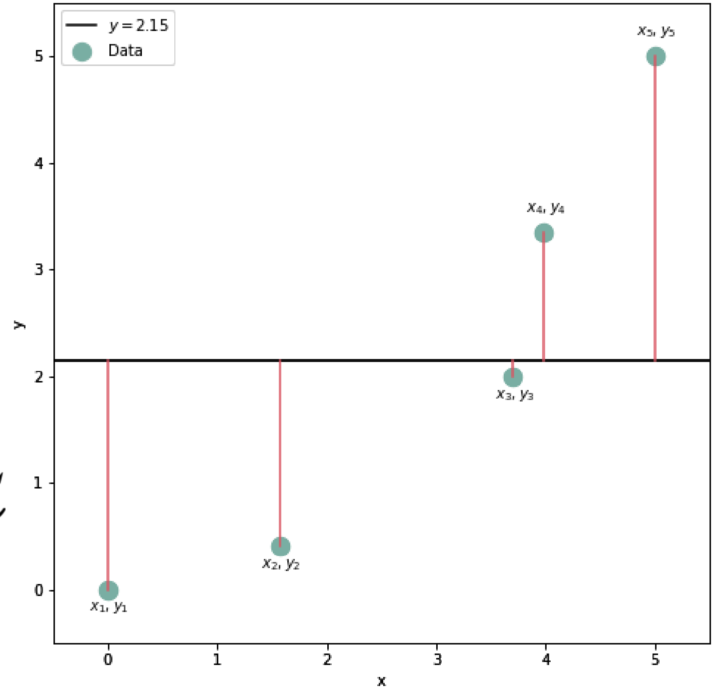
- Starting with a random guess ($y = \bar{y} = \mathrm{mean}(y) = b_0$)
- Determining the quality of the fit by means of the distance between the fit and the data point (residuals)

$$(y - y_1) + (y - y_2) + (y - y_3) + \ldots$$

$$\underbrace{(2.15 - 0.00)}_{2.15} + \underbrace{(2.15 - 0.41)}_{1.74} + \underbrace{(2.15 - 2.0)}_{0.15} + \underbrace{(2.15 - 3.34)}_{-1.19} - \underbrace{(2.15 - 5.00)}_{-2.85} = \underbrace{\Delta}_{0}$$

$$(y - y_1)^2 + (y - y_2)^2 + (y - y_3)^2 + \ldots$$

$$\underbrace{(2.15 - 0.00)^2}_{4.62} + \underbrace{(2.15 - 0.41)^2}_{3.03} + \underbrace{(2.15 - 2.0)^2}_{0.02} + \underbrace{(2.15 - 3.34)^2}_{1.42} - \underbrace{(2.15 - 5.00)^2}_{8.12} = \underbrace{RSS}_{17.21}$$



- Sum of squared residuals (RSS, Residual Sum of Squares)

Finding the optimal line

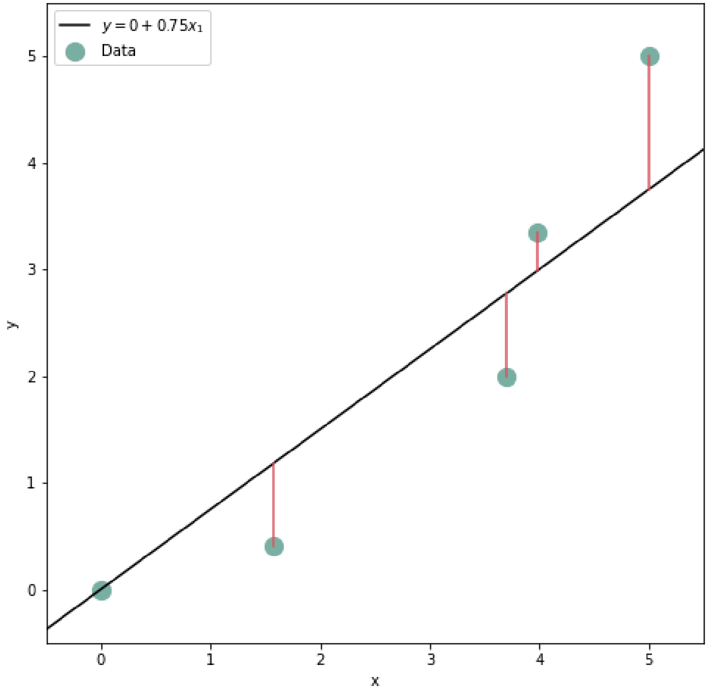
$$\mathbf{X} = \begin{bmatrix} 0.00, 1.57, 3.71, 3.99, 5.00 \\ 0.00, 0.41, 2.00, 3.34, 5.00 \end{bmatrix}$$

- If we rotate the line, we come back to our initial equation $y = b_0 + b_1 x_1$
- We get for a specific slope ($b_1 = 0.75$) a prediction of the value $\bar{y}$ on which we can apply the RSS

$$(f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + (f(x_2) - y_3)^2 + \ldots$$

$$f(x_2) = b_0 + b_1 x_2 = 0 + 0.75\ 1.57 = 1.1775$$

$$\underbrace{(f(x_1) - 0.00)^2}_{0.00} + \underbrace{(f(x_2) - 0.41)^2}_{0.60} + \underbrace{(f(x_3) - 2.0)^2}_{0.61} + \underbrace{(f(x_4) - 3.34)^2}_{0.12} - \underbrace{(f(x_5) - 5.00)^2}_{1.56} = \underbrace{RSS}_{2.89}$$



- Using objective parameters to express the fitting quality

Residual Sum of Squares (RSS):

$$RSS = \sum_{i=0}^{n-1} (f(x_i) - y_i)^2$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} (f(x_i) - y_i)^2$$

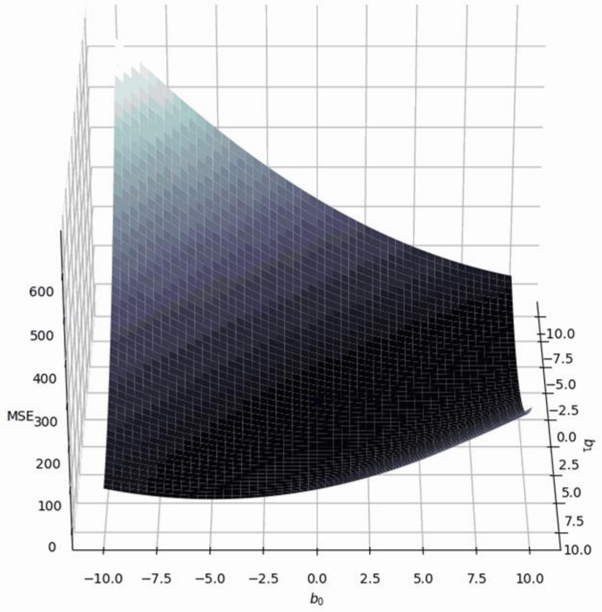
Gradient descent

Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} (f(x_i) - y_i)^2 = J$$

- The method is used to find the minimum of a given cost function J which corresponds to the best fitting line ($\min_{b_0} \min_{b_1} J(\cdot)$)

- The overall idea of that approach is:
  1. Picking a random point of the data set
  2. Finding the slope of the point at this position
  3. Move the point towards (with step-size α) the minimum
  4. Repeat until convergence



Gradient Descent:

$$x_n = x_{n-1} - \alpha \nabla J(x_{n-1})$$

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial b_o} \\ \frac{\partial J}{\partial b_1} \end{bmatrix}$$

$$\frac{\partial J}{\partial b_o} = -2\left(y_i - (b_0 + b_1 x_i)\right)$$

$$\frac{\partial J}{\partial b_1} = -2x_i\left(y_i - (b_0 + b_1 x_i)\right)$$

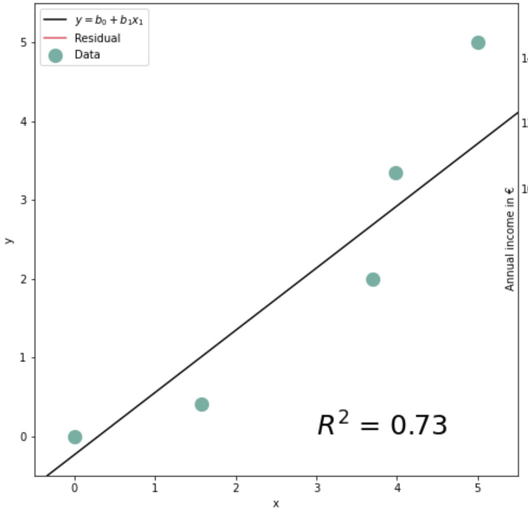
How representative is our fitted model?

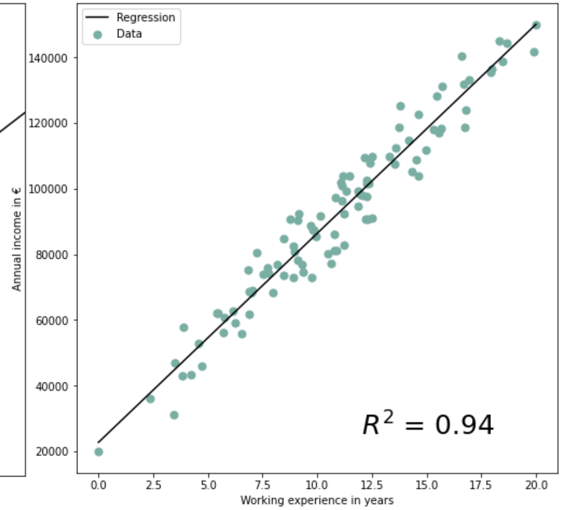
- If we want to evaluate how good/well measured values fit our model, we use the so-called R^2 value (Bestimmtheitsmaß or Coefficient of determination)

- Here, we evaluate the proportion of the variation in the dependent variable that is predictable from the independent variable

Coefficient of Determination:

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=0}^{n-1}(y_i - f(x_i))^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}$$





Classification in general

- Is the process to assign a specified class to an extracted/segmented object
- Classifier is trained by using labeled data set
- For training and validation, the data set is divided into:
  1. Training data set
  2. Validation data set

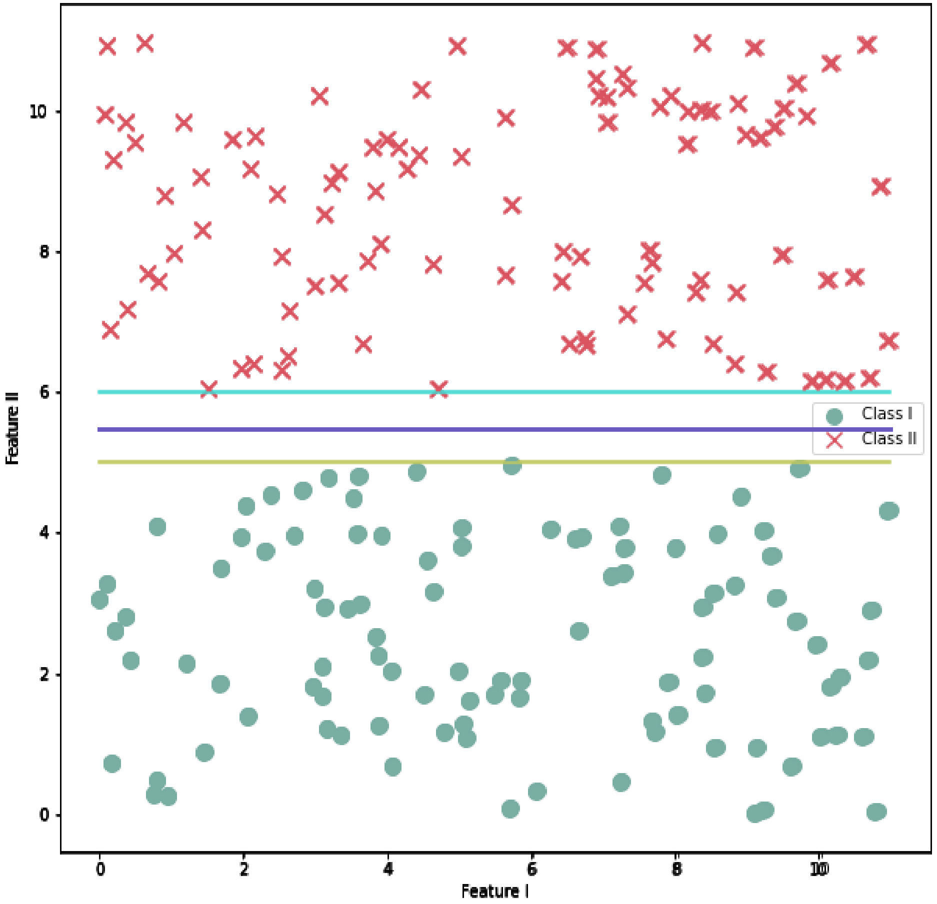
- Several classifier available
  - Decision Trees/Forests
  - Support-Vector-Machines
  - Logistic Regression
  - Deep-Neural-Networks
  - …

```
import torchvision
import torchvision.datasets as datasets
mnist_trainset = datasets.MNIST(root='./data', train=True,
download=True, transform=None)  # Training
mnist_testset = datasets.MNIST(root='./data', train=False,
download=True, transform=None)  # Testing
```

Support Vector Machine (SVM)

- A SVM builds an n-dimensional hyperplane from a training dataset with the help of support vectors
- This hyperplane separates the n-dimensional feature space from each other
- Hyperplane reduces to a line for 2D-reprenstation, but scales for n-dimensions

- Mathematical idea:
  - Maximize the distance (Margin-Of-Separation) between the support vectors



Support Vector Machine (SVM)

- To ensure maximum discriminativity between the two classes it is necessary to maximize this separation width D
- By using primal-dual optimization, we can express the maximization as minimization of a dual-equation
- $y_l \in \{-1, 1\}$ denotes the class assignment of the pattern r

- Finally, the classification by SVM is given by:

$$d_{w,b}(r) = \mathrm{sgn}(\mathbf{w}^T \mathbf{r} + b)$$

$$\mathrm{sgn}(u) = \begin{cases} 1 & u \leq 0 \\ -1 & u < 0 \end{cases}$$

- Generalization is indicated by the amount of support vectors

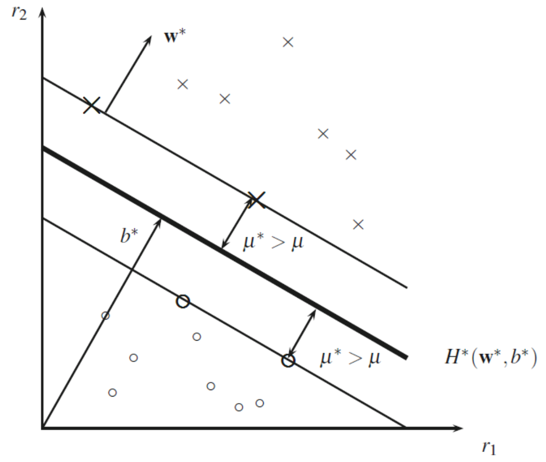


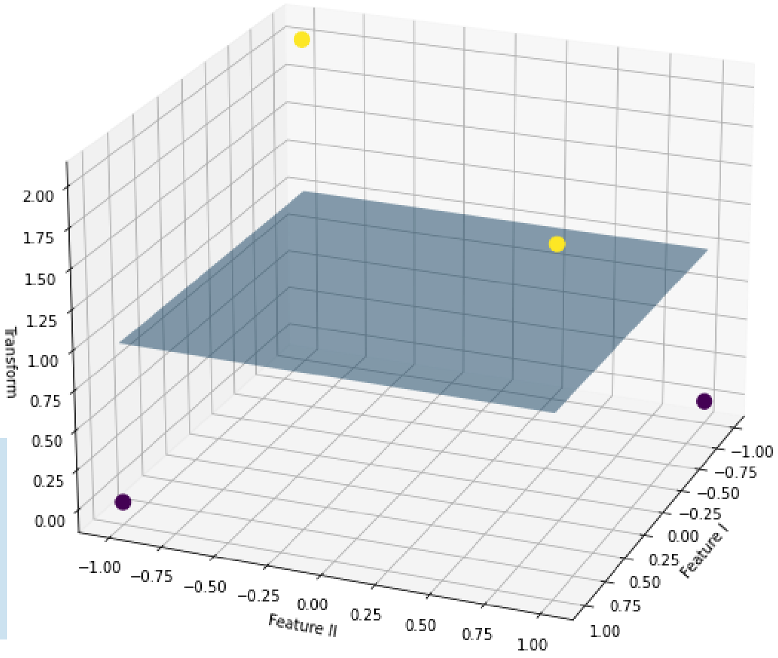
Fig. 1: Linear SVM from [KrRiSchu11]

Kernel

Self-defined Kernel:

$$K(x_1, x_2) = \sqrt{(x_1 + x_2)^2}$$

- To separate non-linear feature spaces, the so-called kernel trick is used
- The idea behind that is to transform a non-linear problem into a linear-problem by means of different kernels
  - RBF
  - Sigmoid (tanh)
  - Bessel
  - …
- A common example is the XOR data set
- We can solve this issue by using the RBF-kernel (Radial Basis Function)



RBF-kernel:

$$K(x_1, x_2) = \exp\left(-\frac{||x_1 - x_2||^2}{2\sigma^2}\right)$$

Sigmoid-Kernel:

$$K(x_1, x_2) = \tanh(\alpha x_1^T x_2 + c)$$

Performance of a binary classifier

- To determine the performance of a (binary) classifier, we use the so-called confusion-matrix
- Therefore, we assign four types to the results:

Binary Confusion Matrix:

		Ground Truth	
		Positive	Negative
Output	Positive	**TP** (hit)	**FP**
	Negative	**FN** (miss)	**TN**

1. True positive (TP)
2. True negative (TN)
3. False positive (FP)
4. False negative (FN)

TP: The woman is pregnant, and the test showed this correctly
TN: The woman is not pregnant, and the test showed this correctly
FP: The woman is not pregnant, and the test showed this falsely
FN: The woman is pregnant, and the test turned out negative

- To check the results, we need a-priori knowledge:
  - Condition positive (CP) → Number of real positive cases within the data set
  - Condition negative (CN) → Number of real negative cases within the data set

Performance Indicators

True-positive rate (TPR): recall, hit rate, sensitivity

$$\mathrm{TPR} = \frac{TP}{CP} = \frac{TP}{TP+FN} = \mathrm{RE}$$

! The probability of correct classification of a positive element is described by the Recall (RE). It is a measure of how well the classifier can recognize positive elements [RM15]

Precision (PR): Positive predictive value

$$\mathrm{PR} = \frac{TP}{FP+TP}$$

! The Precision (PR) indicates the ratio of how many elements that were classified as positive are actually positive [RM15]

Accuracy (ACC):

$$\mathrm{ACC} = \frac{TP+TN}{CP+CN} = \frac{TP+TN}{TP+FN+FP+TN}$$

! The probability of correct classification of an element is described by the Accuracy (ACC) [RM15]

F1-Score (F1):

$$\mathrm{F1} = \frac{2\ PR\ RE}{PR+RE}$$

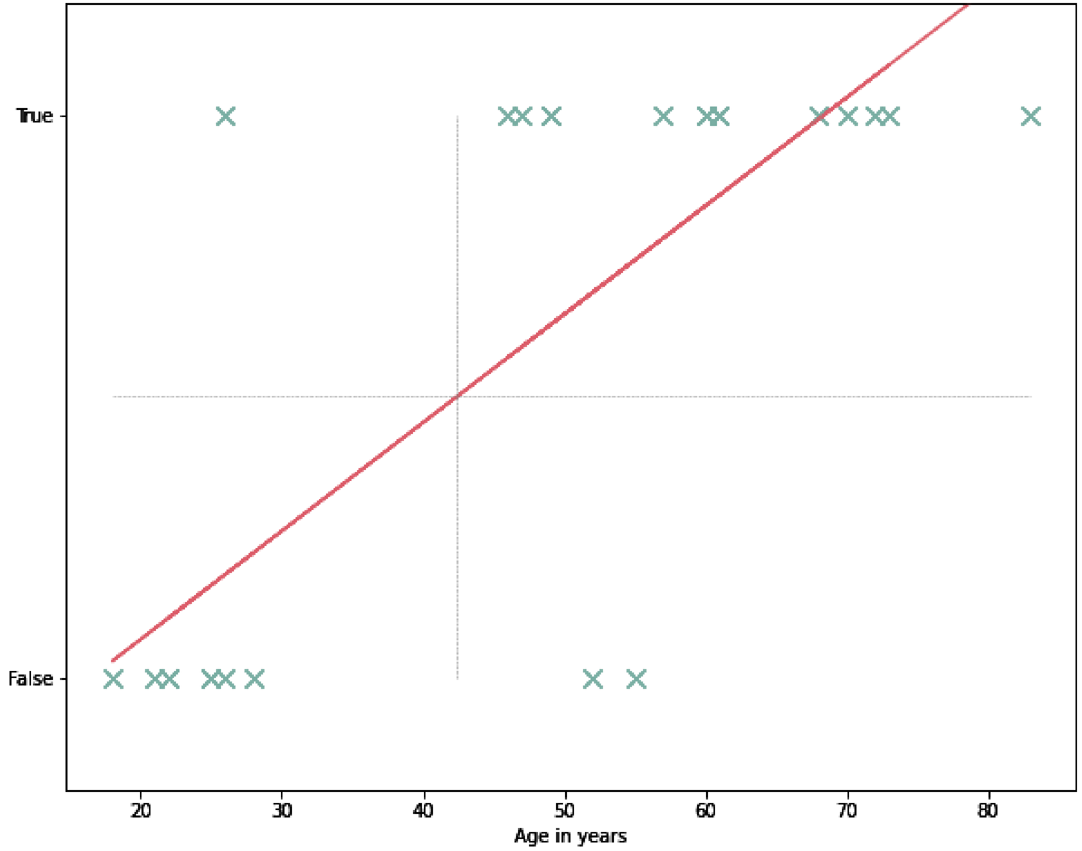
! Usually, there is a competing relationship between RE and PR. To obtain a single characteristic value for the assessment, the F1 score (F1) is defined. It combines RE and PR by forming the harmonic mean [RM15]

Logistic Regression vs. Linear Regression

- Considering discrete data instead of continuous data
- Obtain Boolean decision instead of value prediction
- Both Regressions are capable of multi-class problems

- Example: Data set with information about the ownership of a car about age

1. Visualize the data set as scatter plot
2. Fitting a linear model (linear regression) to it
3. Decision rule based on probability

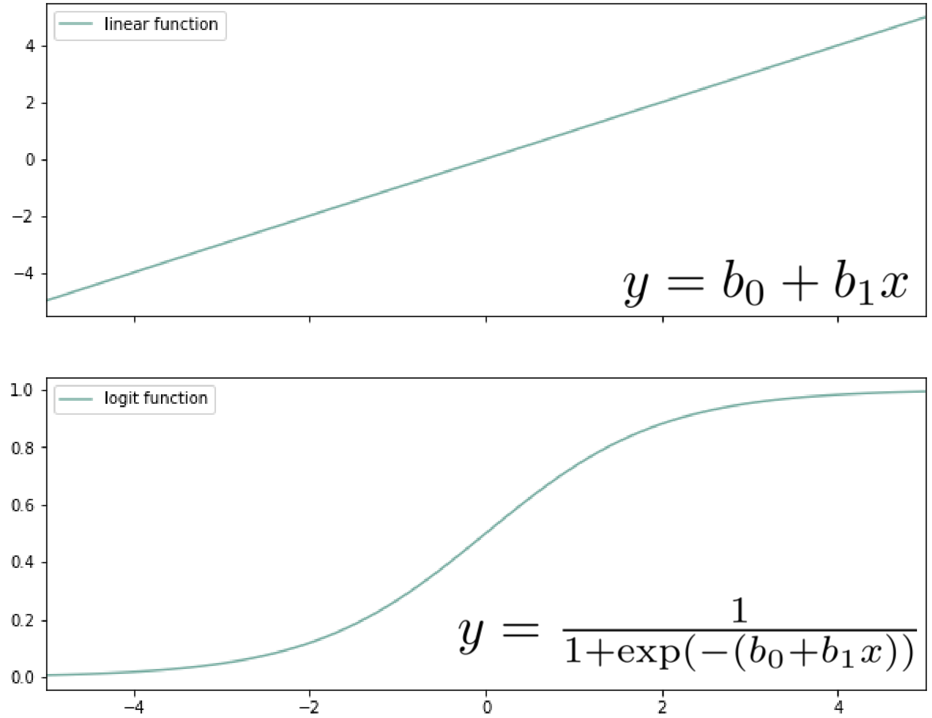


Logit Function

- An S-shaped function would capture the data set more accurately
- We use the so-called Logit-function (or Sigmoid) to solve that problem

Logit-Function:

$$h_\theta(x) = \frac{1}{1+\exp(-x)}$$



- The Sigmoid function scales all input values to

$$h_\theta(x) \rightarrow]0;1[$$

```python
def logit(x):
    return 1 / (1 + np.exp(-x))
```

Model training

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X[['Age']],
X.Class, test_size=0.1)

model = LogisticRegression()  # Object on the LogisticRegression
class
model.fit(X_train, y_train)  # As usual in sklearn: .fit trains
the model by means of the parsed data
model.predict(X_test)  # Apply the model in our test data set.
The result is the classification of the X_test data
```

- We use the linear regression as initial point for model training

$$h_\theta(x) = b_0 + b_1 x$$

- We modify the linear regression to match the S-shape in the latter

$$h_\theta(x) = \frac{1}{1+\exp(-(b_0+b_1 x))}$$

- Cost function of the logistic regression is given as

$$J(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{else} \end{cases}$$

- and can be reduced to

$$J(\theta) = -\frac{1}{n} \sum \left[y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i)) \right]$$

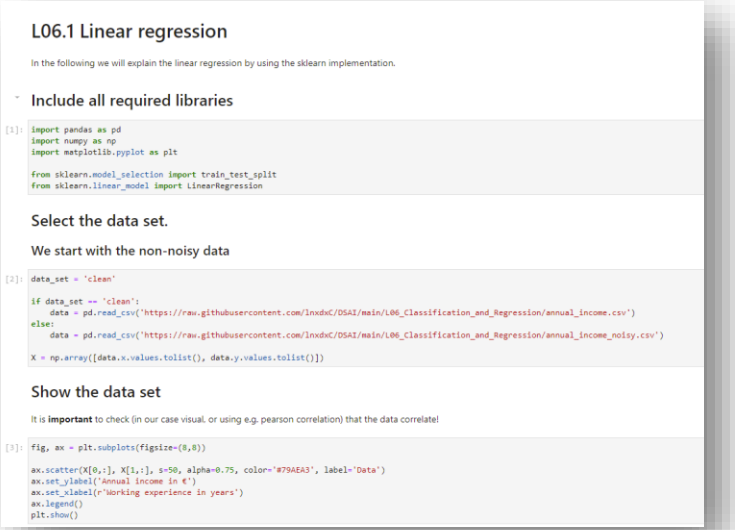
- for the final gradient descent approach

<u>Gradient Descent:</u>

$$x_n = x_{n-1} - \alpha \nabla J(x_{n-1})$$

Break

E06.X - Hands on



L06.1 Linear regression

In the following we will explain the linear regression by using the sklearn implementation.

Include all required libraries

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
```

Select the data set.

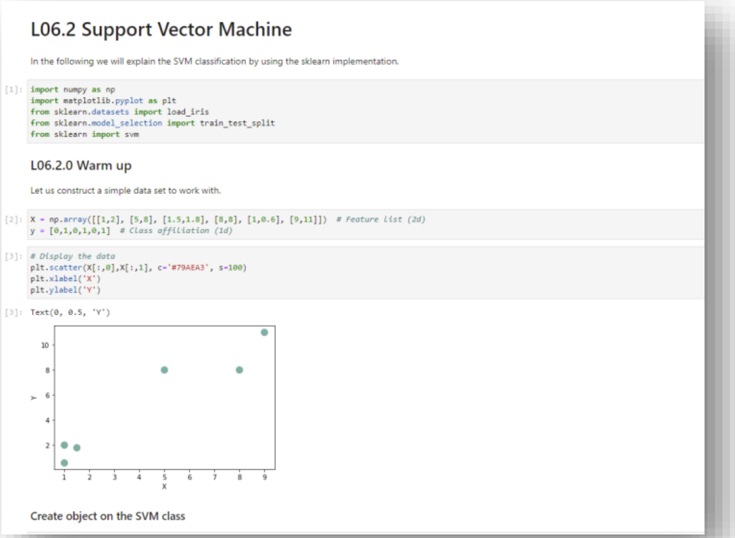
We start with the non-noisy data

```
[2]: data_set = 'clean'

     if data_set == 'clean':
         data = pd.read_csv('https://raw.githubusercontent.com/lnxdxC/DSAI/main/L06_Classification_and_Regression/annual_income.csv')
     else:
         data = pd.read_csv('https://raw.githubusercontent.com/lnxdxC/DSAI/main/L06_Classification_and_Regression/annual_income_noisy.csv')

     X = np.array([data.x.values.tolist(), data.y.values.tolist()])
```

Show the data set

It is **important** to check (in our case visual. or using e.g. pearson correlation) that the data correlate!

```
[3]: fig, ax = plt.subplots(figsize=(8,8))

     ax.scatter(X[0,:], X[1,:], s=50, alpha=0.75, color='#79AEA3', label='Data')
     ax.set_ylabel('Annual income in €')
     ax.set_xlabel(r'Working experience in years')
     ax.legend()
     plt.show()
```



L06.2 Support Vector Machine

In the following we will explain the SVM classification by using the sklearn implementation.

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.datasets import load_iris
     from sklearn.model_selection import train_test_split
     from sklearn import svm
```

L06.2.0 Warm up

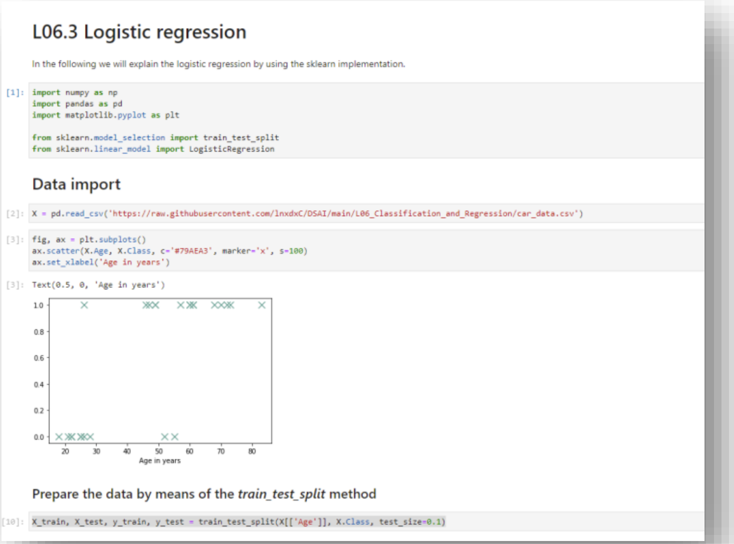
Let us construct a simple data set to work with.

```
[2]: X = np.array([[1,2], [5,8], [1.5,1.8], [8,8], [1,0.6], [9,11]])  # Feature list (2d)
     y = [0,1,0,1,0,1]  # Class affiliation (1d)
```

```
[3]: # Display the data
     plt.scatter(X[:,0],X[:,1], c='#79AEA3', s=100)
     plt.xlabel('X')
     plt.ylabel('Y')
```

```
[3]: Text(0, 0.5, 'Y')
```


Create object on the SVM class



L06.3 Logistic regression

In the following we will explain the logistic regression by using the sklearn implementation.

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
```

Data import

```
[2]: X = pd.read_csv('https://raw.githubusercontent.com/lnxdxC/DSAI/main/L06_Classification_and_Regression/car_data.csv')
```

```
[3]: fig, ax = plt.subplots()
     ax.scatter(X.Age, X.Class, c='#79AEA3', marker='x', s=100)
     ax.set_xlabel('Age in years')
```

```
[3]: Text(0.5, 0, 'Age in years')
```


Prepare the data by means of the *train_test_split* method

```
[10]: X_train, X_test, y_train, y_test = train_test_split(X[['Age']], X.Class, test_size=0.1)
```

E06.1_Linear_Reg.iypnb

E06.2_SVM.iypnb

E06.3_Logistic_Reg.iypnb

HA06.1 - Hands on

HA06.1_LR.iypnb

IFM

Institut für Fahrerassistenz und vernetzte Mobilität

Hochschule Kempten

University of Applied Sciences

www.hs-kempten.de/ifm