ATB
Leibniz-Institut für
Agrartechnik und Bioökonomie

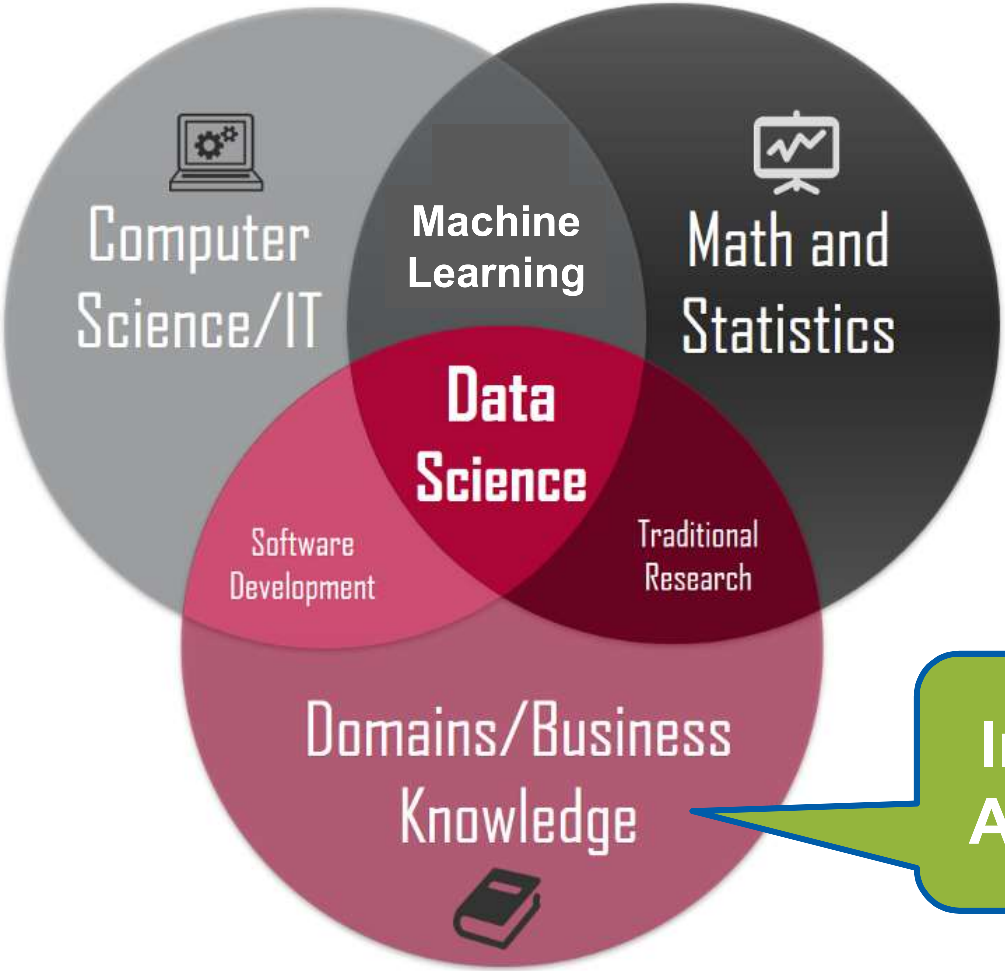
Data Science & Machine Learning in Agriculture

Introduction

Dr. Julian Adolphs
Department Data Science

Leibniz
Association

What is Data Science and Machine Learning ?



In our case: Agriculture !

Picture from https://towardsdatascience.com

Machine Learning Algorithms

Three categories of Machine Learning (ML):

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine Learning Algorithms

Three categories of Machine Learning (ML):

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Classification

Learn and predict group memberships

Regression

Learn and predict **continuous** values

Machine Learning Algorithms

Three categories of Machine Learning (ML):

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Classification

Learn and predict group memberships

Regression

Learn and predict **continuous** values

$$\boxed{y = 1.3 \cdot 10^6}$$

for instance house prices

Machine Learning Algorithms

Three categories of Machine Learning (ML):

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Classification

Learn and predict
group memberships

Regression

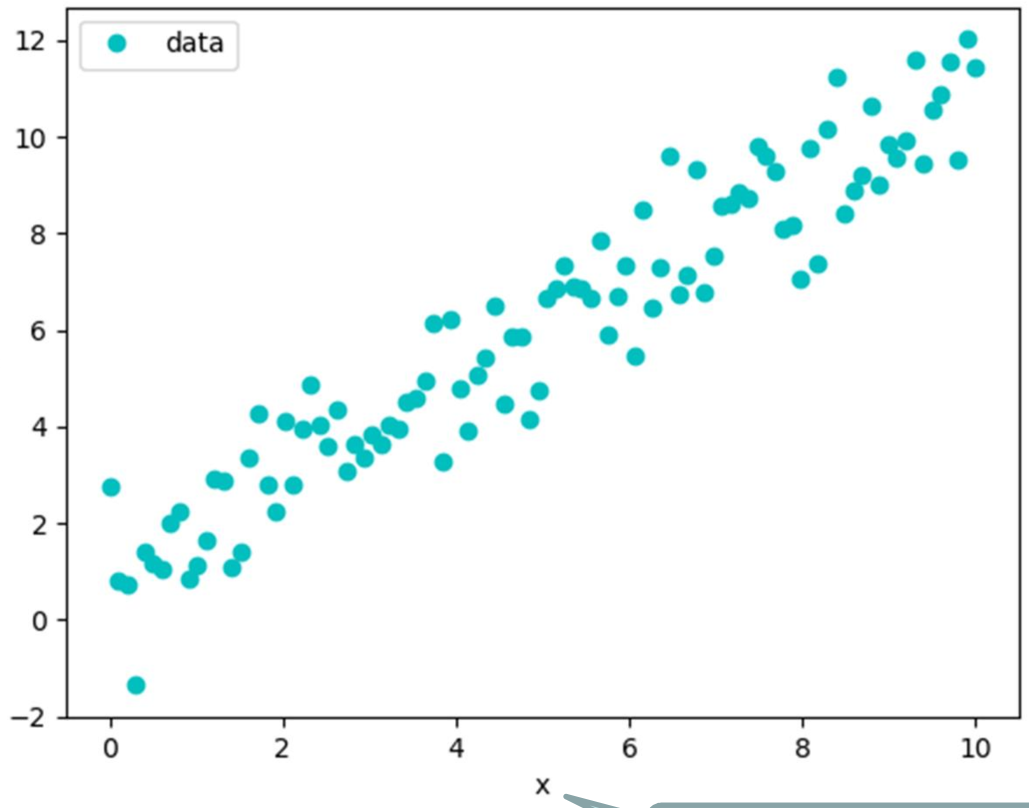
Learn and predict
continuous values

$y = 1.3 \cdot 10^6$

for instance
house prices

Supervised ML – Regression (Continuous Values)

Experimental Data

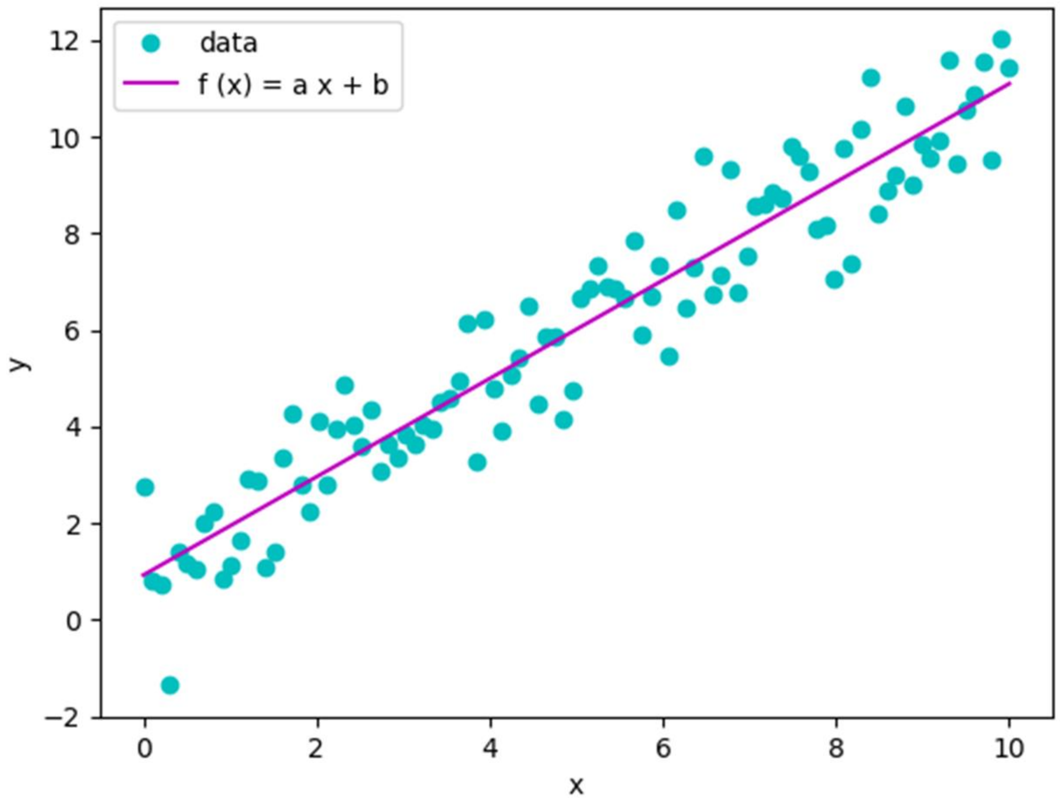


for instance: emissions

for instance: time

Linear Regression

Fit of the Data with linear Function: $f(x) = a\,x + b$



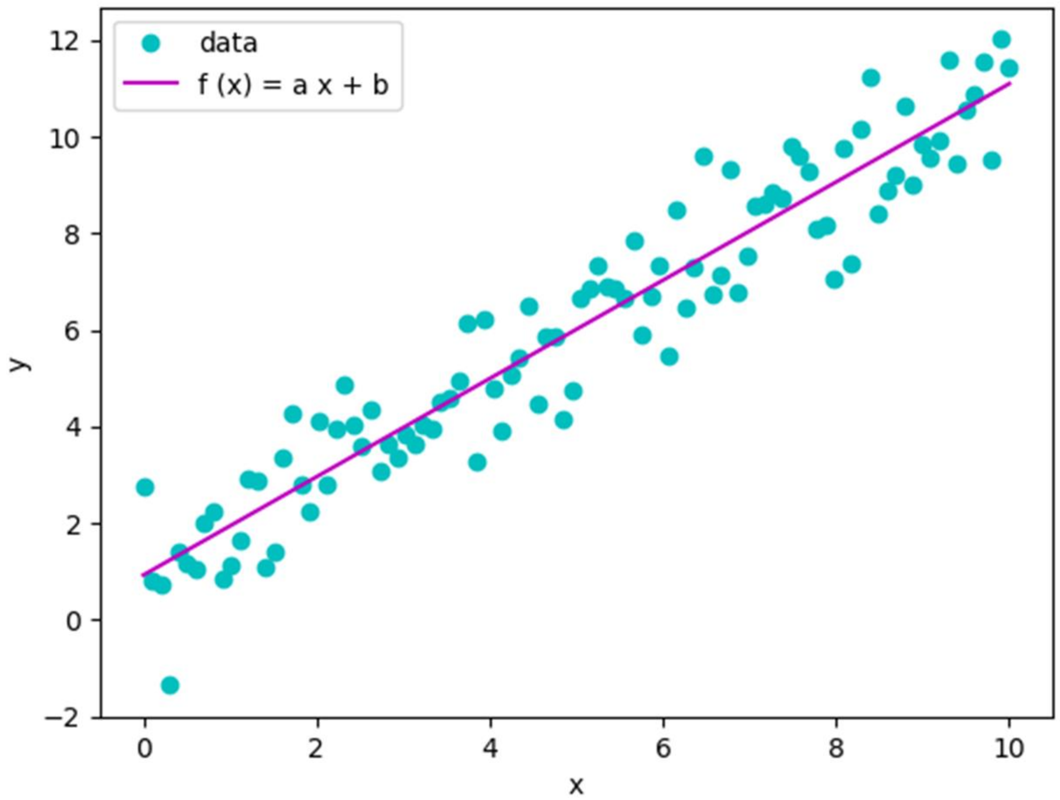
$a = 1.0167$

$b = 0.9280$

Linear Regression

Fit of the Data with linear Function: $\boxed{f(x) = a\,x + b}$

generalised Model



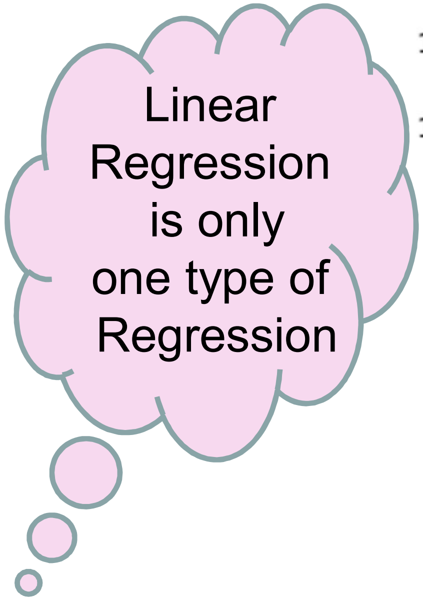
$a = 1.0167$

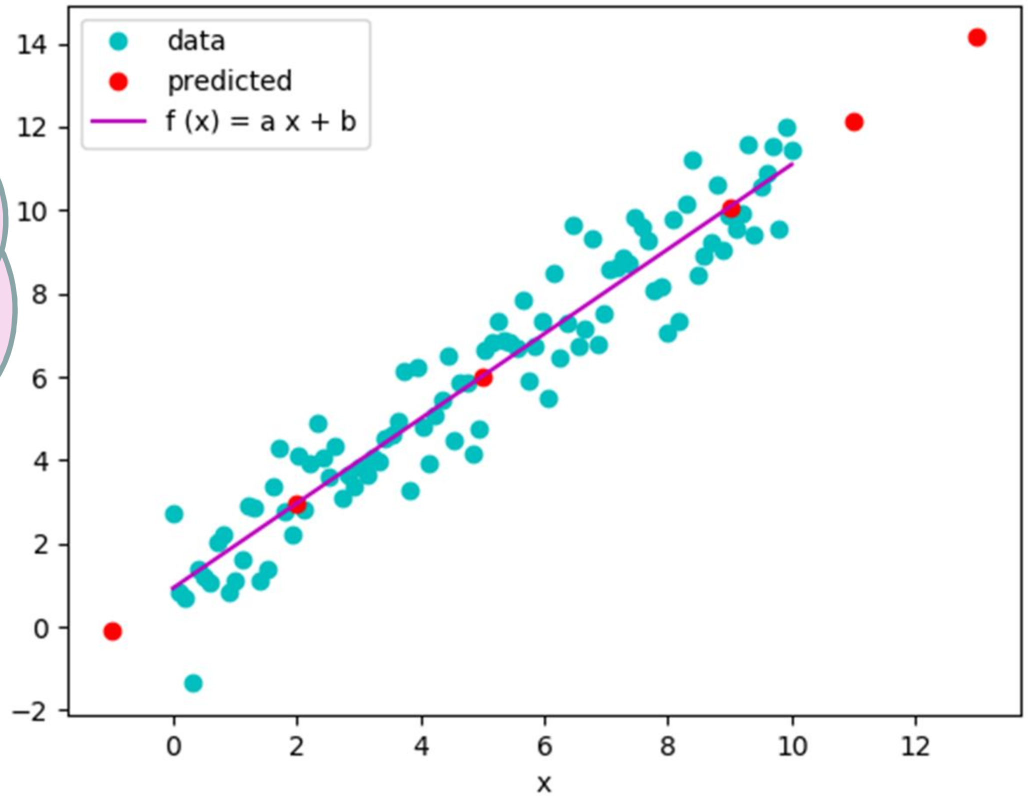
$b = 0.9280$

Linear Regression

Prediction:

Extrapolate unknown y-values for new x-values using $f(x) = a\,x + b$



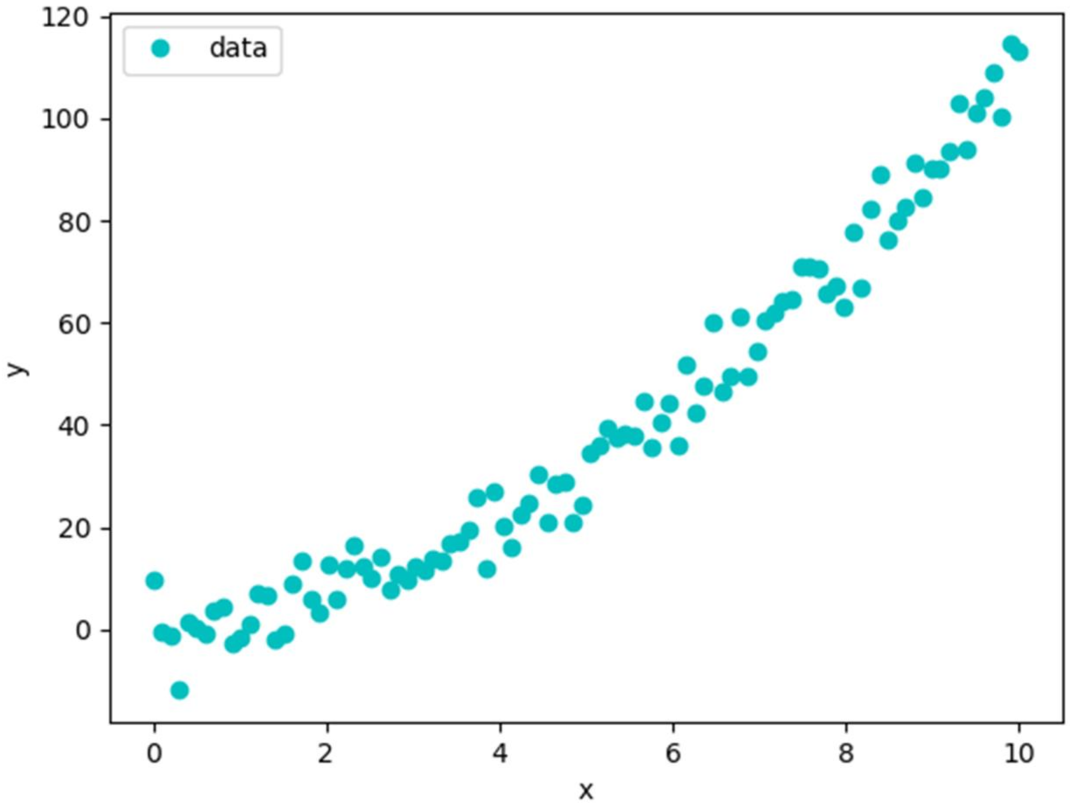


$a = 1.0167$

$b = 0.9280$

Linear Regression

Non-Linear Experimental Data

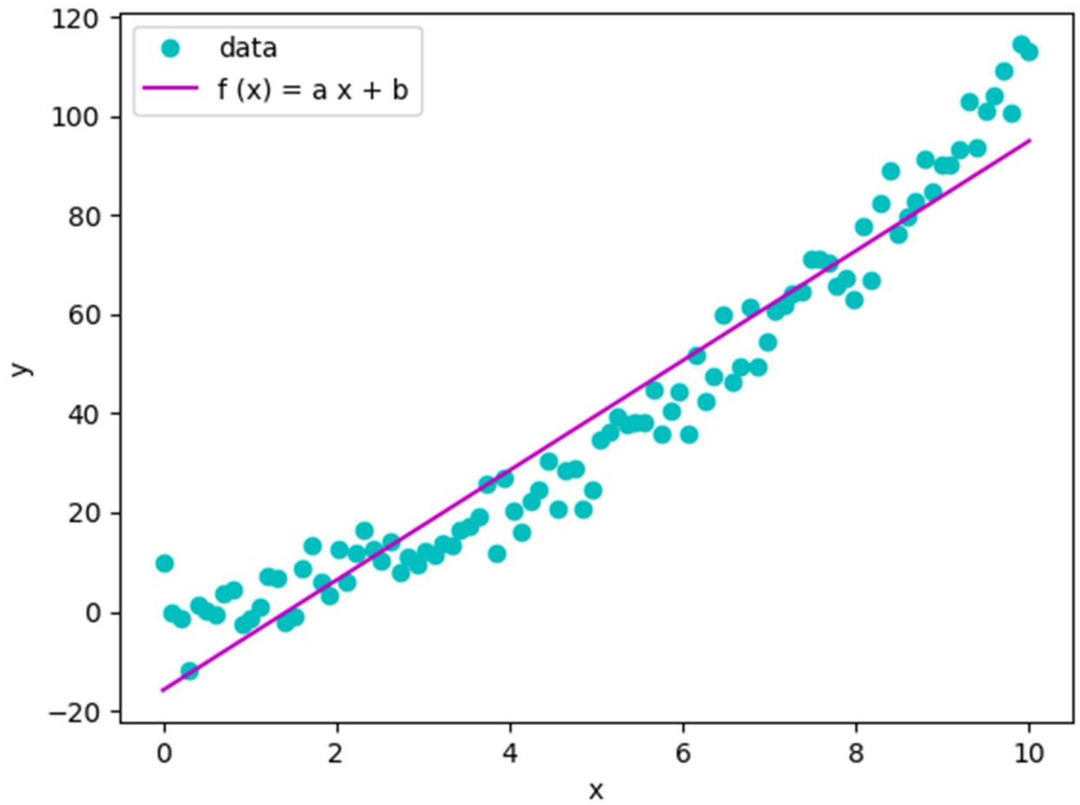


$a = 1.0167$

$b = 0.9280$

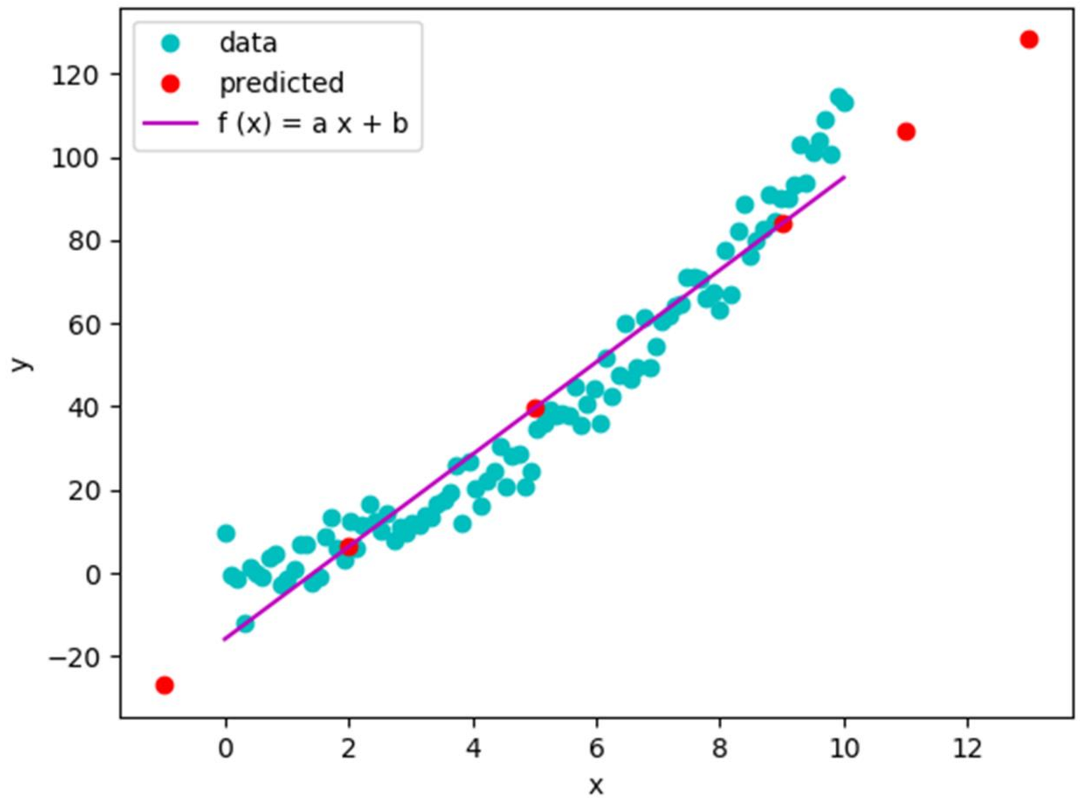
Linear Regression

Fit of the Data with linear Function: $f(x) = a\,x + b$



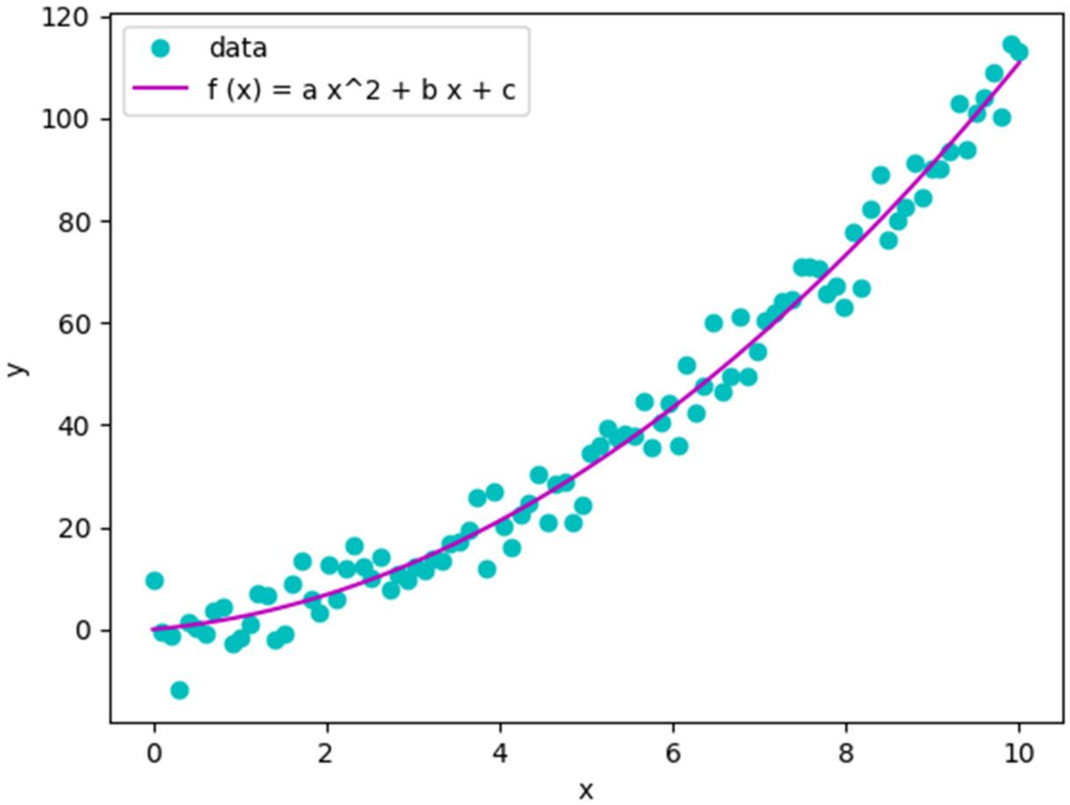
Linear Regression

Extrapolate unknown y-values for new x-values using $f(x) = a\,x + b$



Linear Regression with Polynomial

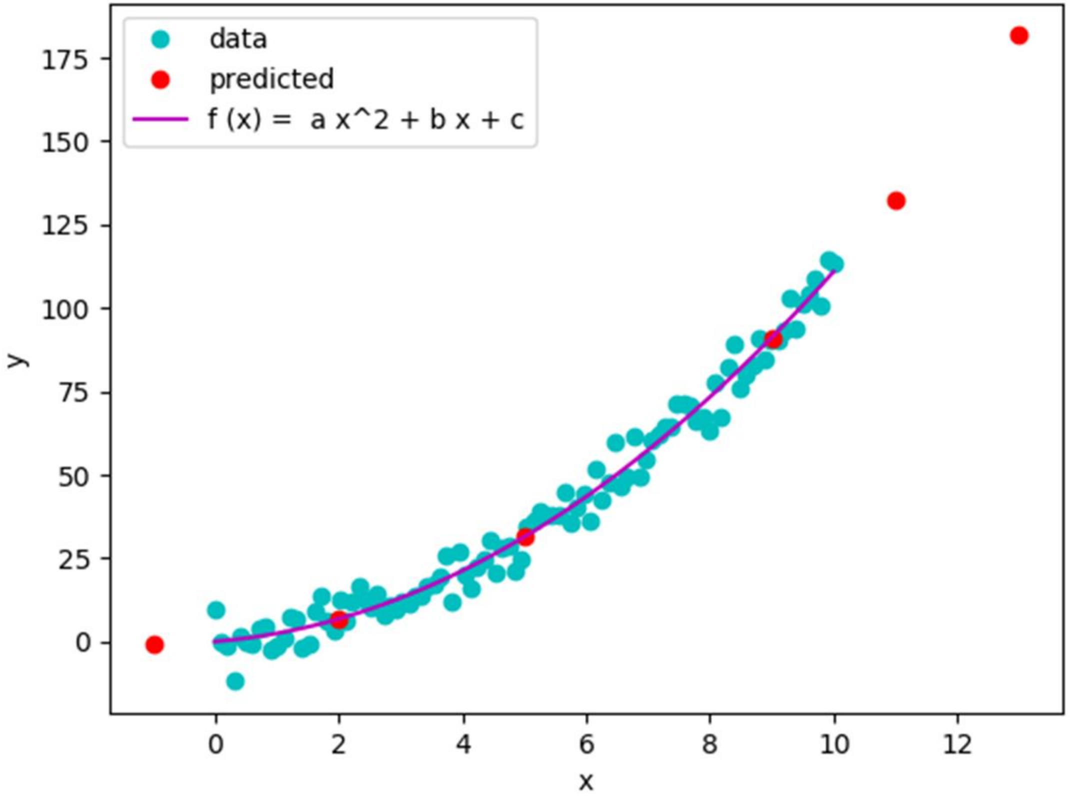
Fit the Data with Polynom (degree = 2): $f(x) = a\,x^2 + b\,x + c$



Linear Regression with Polynomial

Extrapolate unknown y-values for new x-values using $f(x) = a\,x^2 + b\,x + c$

Predict



Model Fit

How are the solutions computed, that were shown in the last slides?

We need a **quantitative measure** that is optimized to find the optimal model.

The idea is to **minimize** a **loss-function** (or cost-function).

Mean Absolute Error (MAE):

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|$$

Root Mean Squared Error (RMSE):

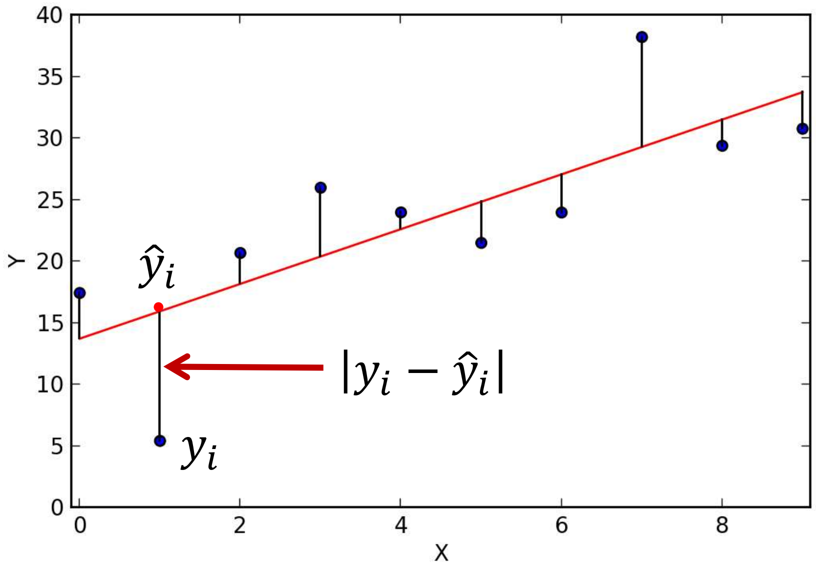
$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2}$$

n number of samples

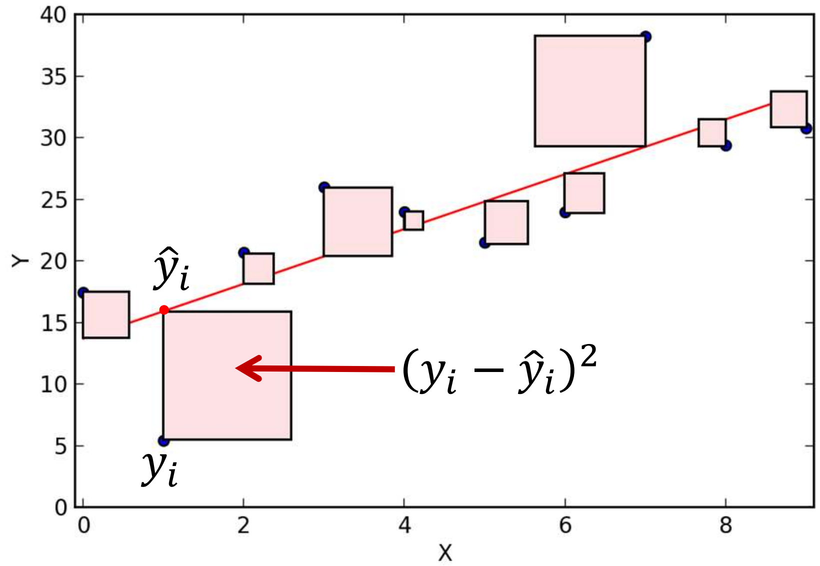
MAE is linear and in many cases more robust. RMSE overrates outliers.

Visualization of MAE and RMSE

Residues for MAE-Calculation
(Mean Absolute Error)



Residues for RMSE-Calculation
(Root Mean Squared Error)

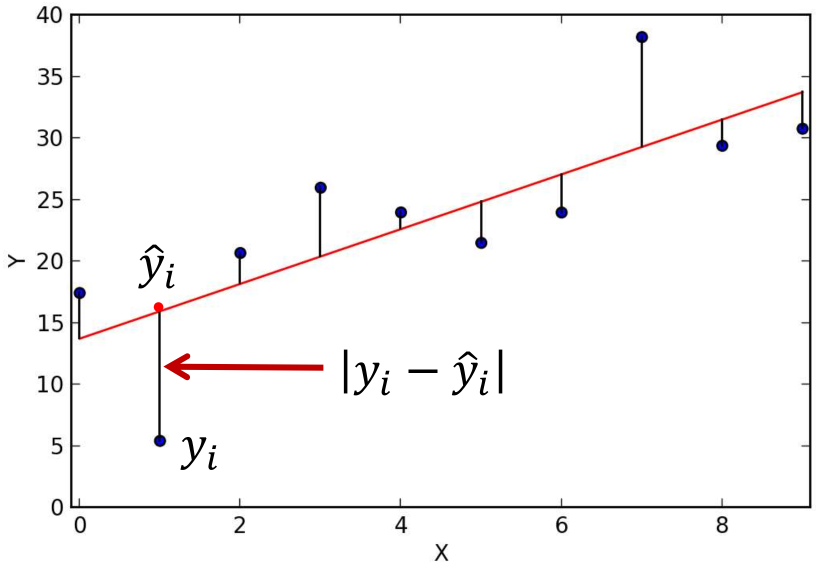


$$MAE = \frac{1}{n}\sum_{1}^{n} |y_i - \hat{y}_i|$$

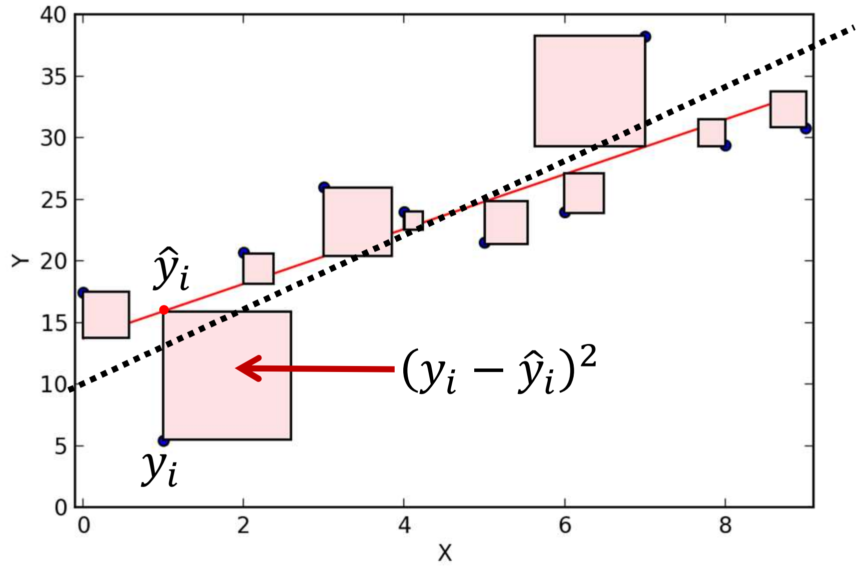
$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n} (y_i - \hat{y}_i)^2}$$

Visualization of MAE and RMSE

Residues for MAE-Calculation
(Mean Absolute Error)



Residues for RMSE-Calculation
(Root Mean Squared Error)



$$MAE = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2}$$

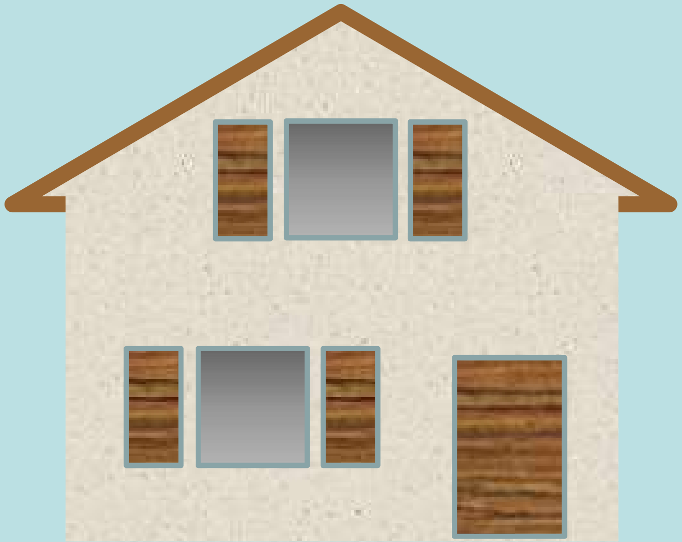
Regression: N-dimensions, non-linear relation
(for instance House-Prices)

We have a **data** base with **many houses** and **their prices**.

We want to **predict** (unknown) prices of houses.

A certain sample of the data base could look like that:

Features (of the house):
- 100 sqm
- 4 rooms
- 2 bathrooms
- renovated
- 20 years old
- 180 sqm garden
- 1 km to U-station
- 15 sqm garage
- …



Target:
Price of the house
900.000 $

1 sample of the data set

Regression: N-dimensions, non-linear relation
(for instance House-Prices)

Train the Model

Input-data

Features

(size, #rooms, age, …)

Training / Learning

(known) Output-data

Labels / Targets

(House price)

more complex than

$f(x) = ax + b$!

- # bathro
- condition
- age / years
- garden / sqm
- U-station / km
- garage / sqm
- …

Targets:

Prices of the houses / $

ℓ samples

Regression: N-dimensions, non-linear relation
(for instance House-Prices)

Train the Model

Input-data
Features
(size, #rooms, age, …)

Training / Learning

(known)
Output-data
Labels / Targets
(House price)

Features (of the houses):

- size / sqm
- # rooms
- # bathrooms
- condition
- age / years
- garden / sqm
- U-station / km
- garage / sqm
- …

number of features: d

Targets:

Prices of the houses / $

ℓ samples

Regression: N-dimensions, non-linear relation
(for instance House-Prices)

Train the Model

Input-data

Features

(size, #rooms, age, …)

Training / Learning

(known) Output-data

Labels / Targets

(House price)

i-th feature vector

$$x^i = (\overbrace{x_1{}^i, \quad \cdots, \quad x_d{}^i}^{d \text{ features}})$$

i-th sample

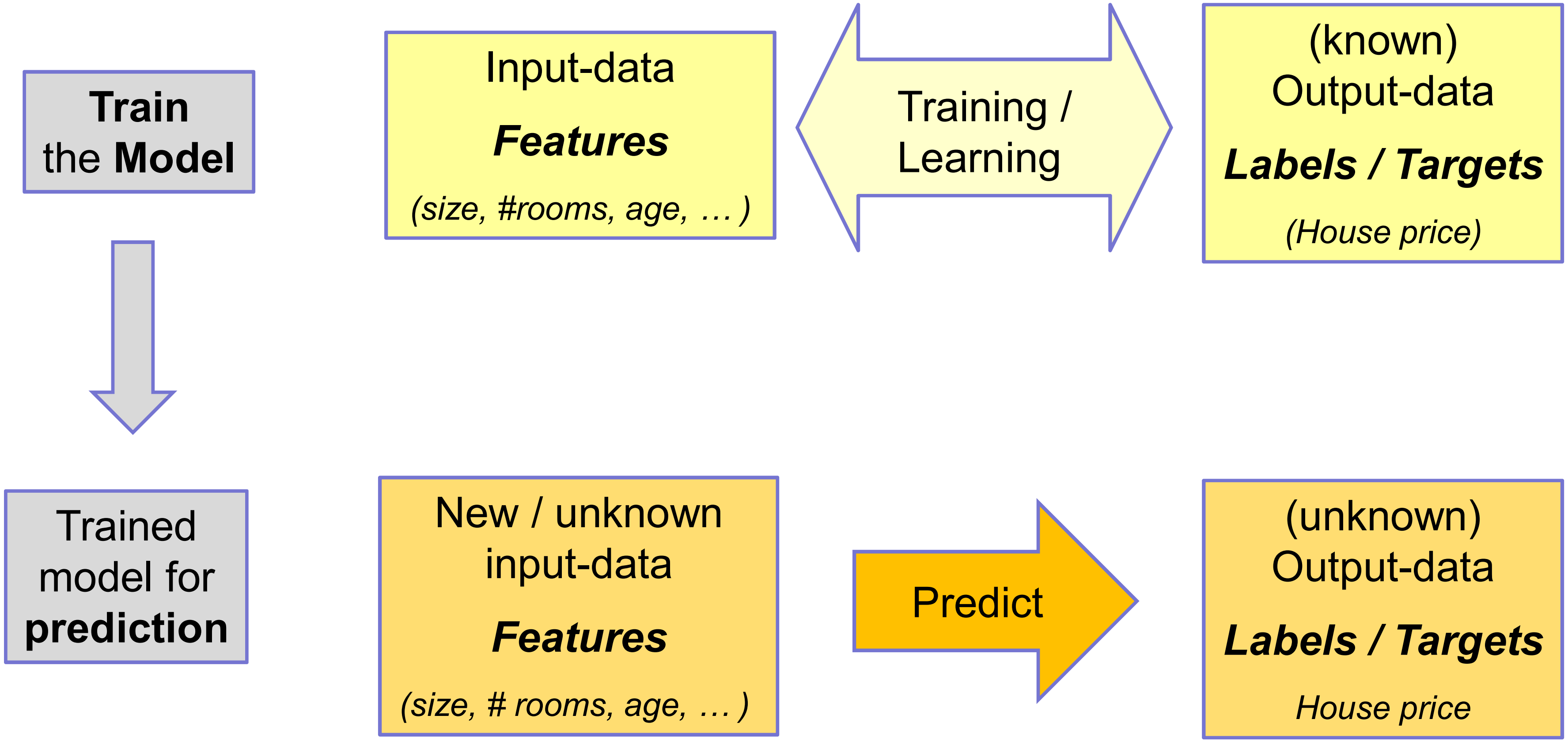
matrix of feature vectors

$$X = \begin{bmatrix} x^1 \\ \vdots \\ x^l \end{bmatrix} = \left.\begin{pmatrix} x_1{}^1 & \cdots & x_d{}^l \\ \vdots & \ddots & \vdots \\ x_1{}^l & \cdots & x_d{}^l \end{pmatrix}\right\} \ell \text{ samples}$$

target / label vector

$$y = \begin{bmatrix} y^1 \\ \vdots \\ y^l \end{bmatrix}$$

Regression: N-dimensions, non-linear relation
(for instance House-Prices)



Model Evaluation

In the **1-D**imensional example: obvious which model was better.

For higher dimensional problems: **quantitative measures**!

Mean Absolute Error (MAE):

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|$$

Root Mean Squared Error (RMSE):

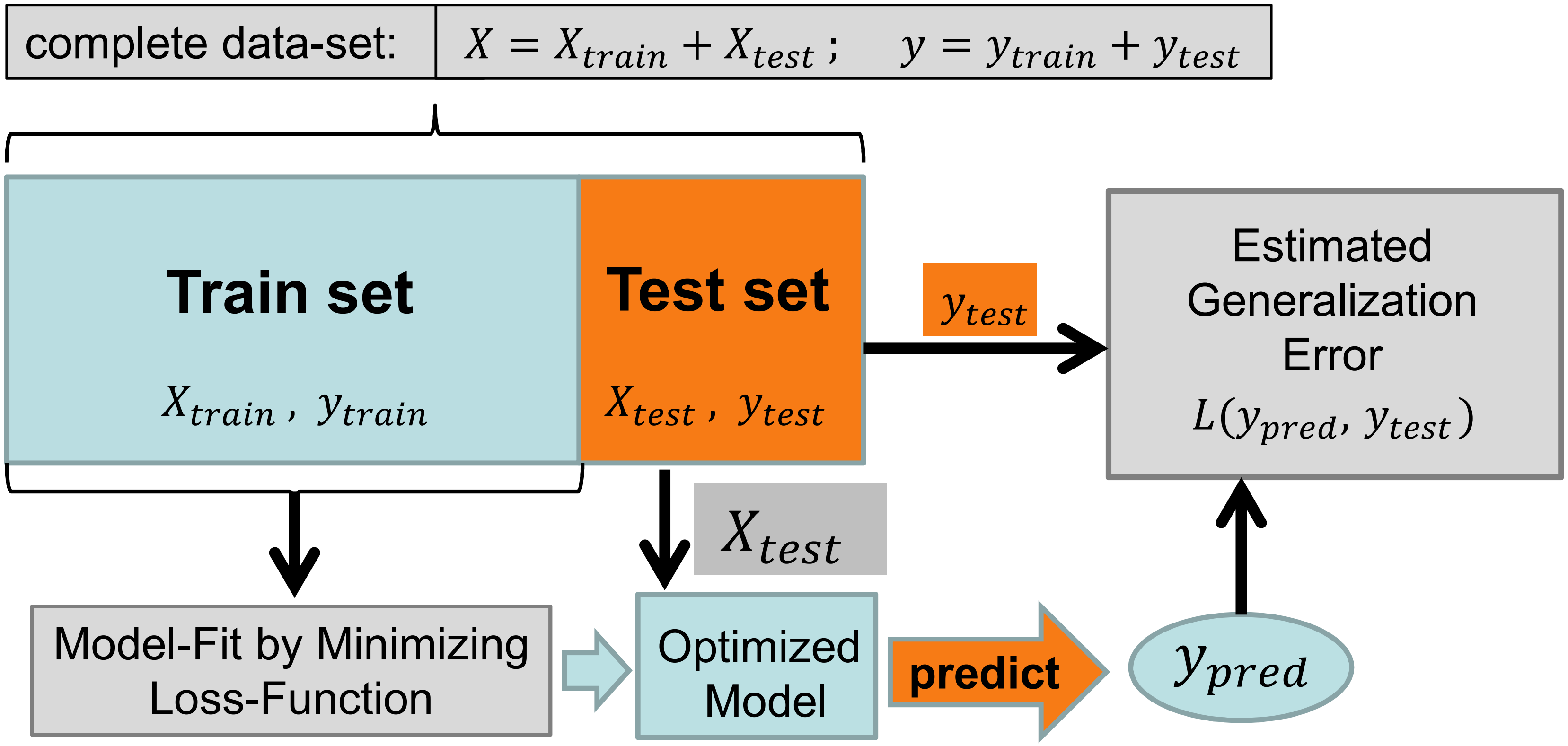
$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2}$$

n number of samples

Model Validation – Test Set Method

An estimate of the **generalization error** of the model is needed.

Solution: split the data into a **train set** and a **test set !**

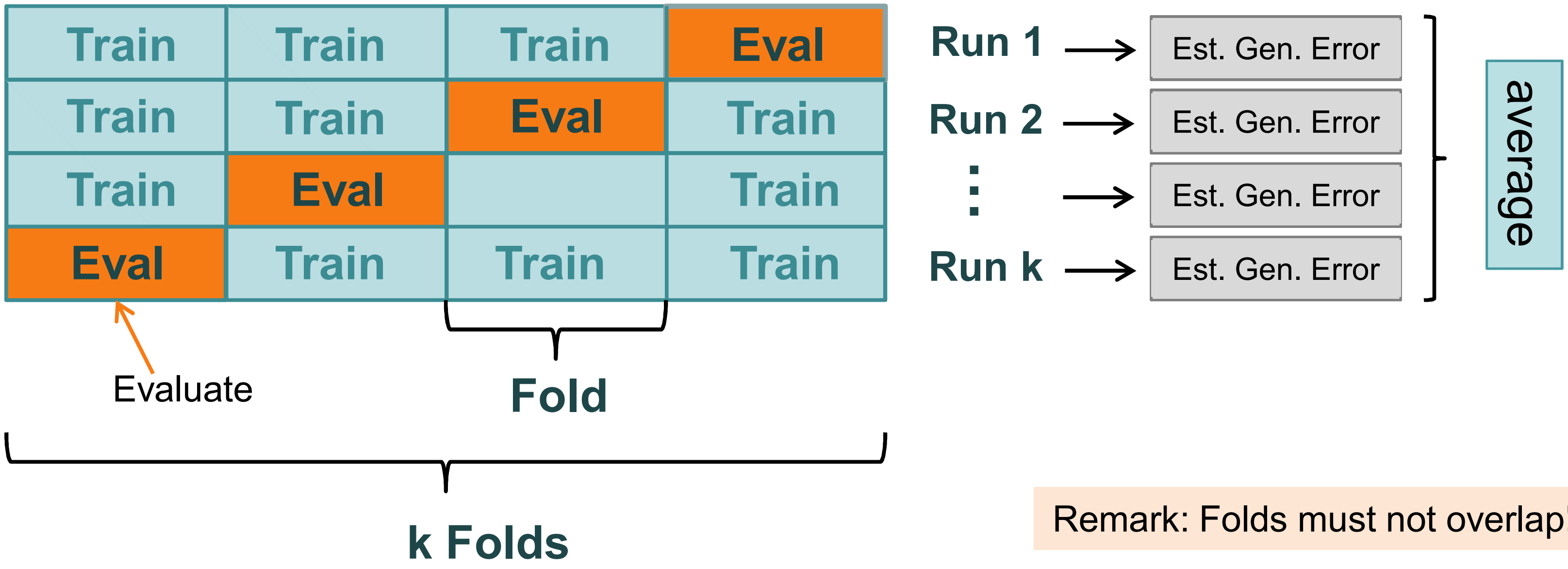


K-Fold Cross Validation

The *Holdout Method* is a bit **wasteful** use of data. **K-Fold** is more efficient:

On each run the procedure of the former slide is performed.

The Estimated Generalization Error is the average of k runs.



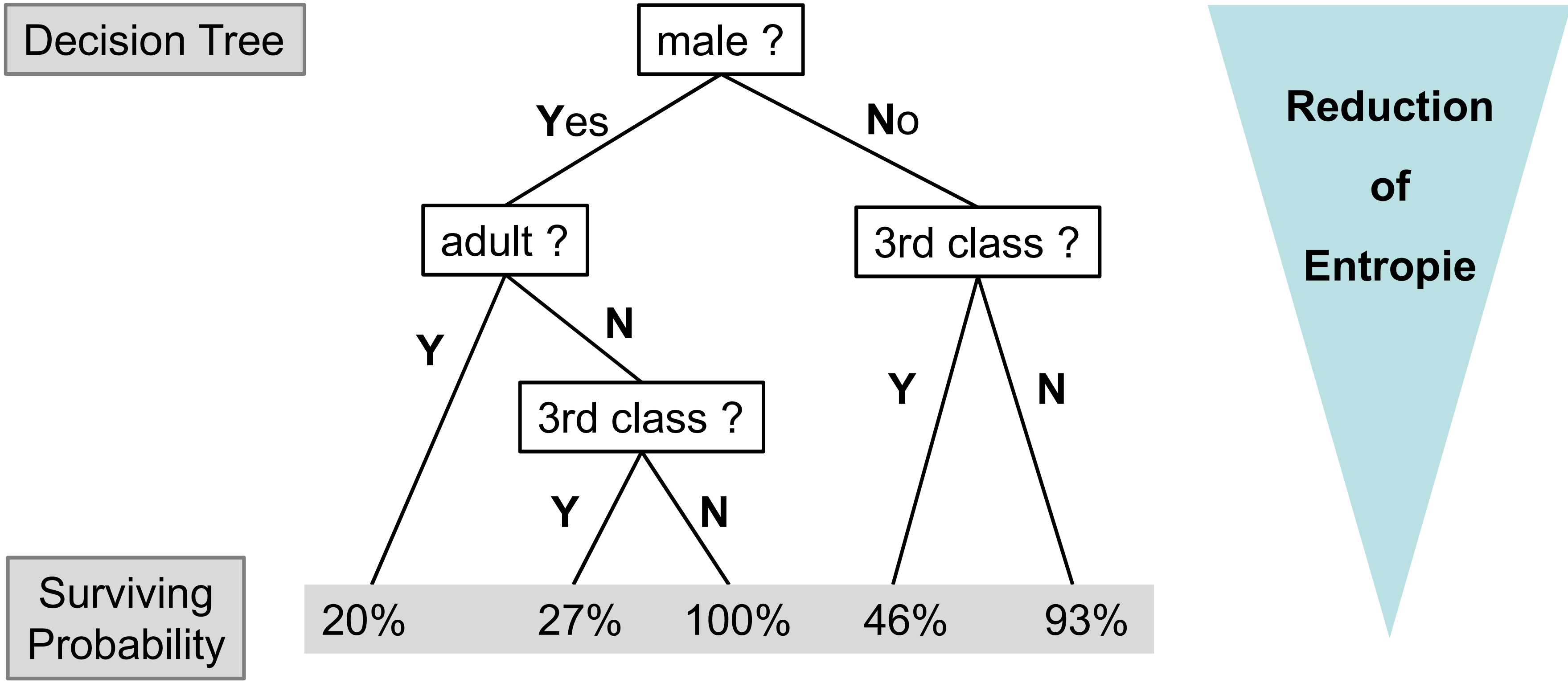
Remark: Folds must not overlap!

Goto House-Price Example in Jupyter

ML-Algorithms: Random Forest

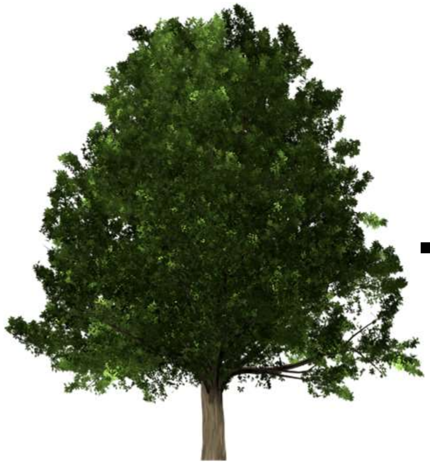
(Ensemble of Decision Trees)

Example: **Titanic**, **70%** of the 2200 passengers died, **30%** survived.



ML-Algorithms: **Random Forest**

A **Random Forest** is an Ensemble of **Decision Trees** calculated bei **Entropy Minimization**



+

Random

Decision Tree
calculated by
Entropy Minimization

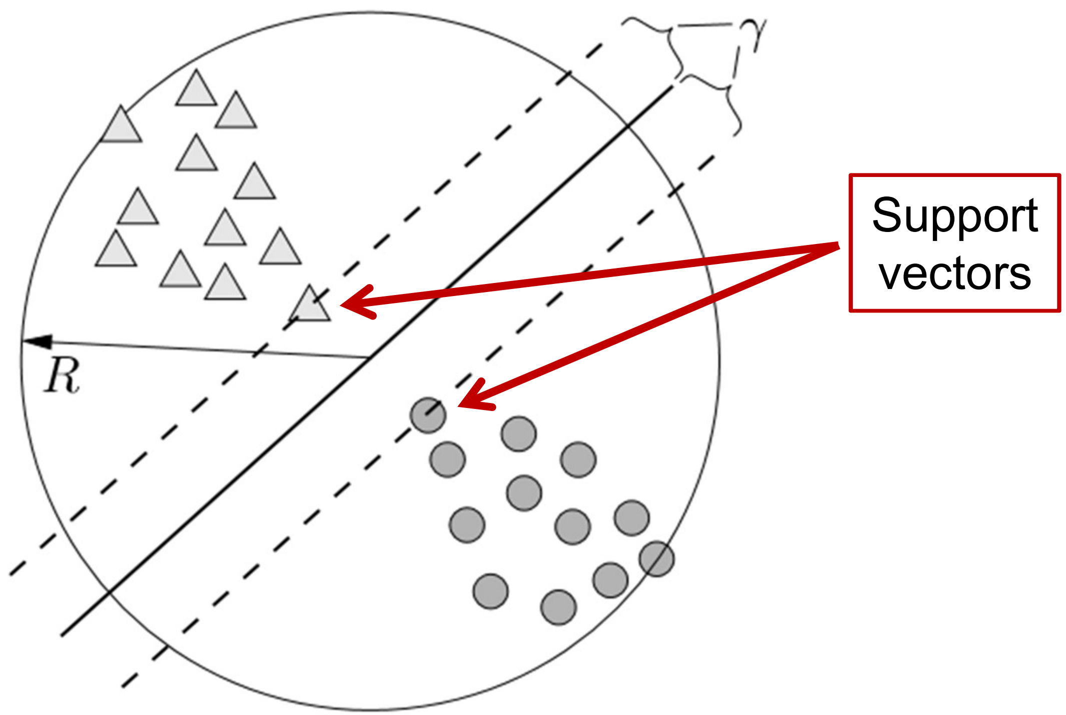
Random Forest

ML-Algorithms: **Support Vector Machine (SVM)**

Classification Task:

seperate

△ from ●

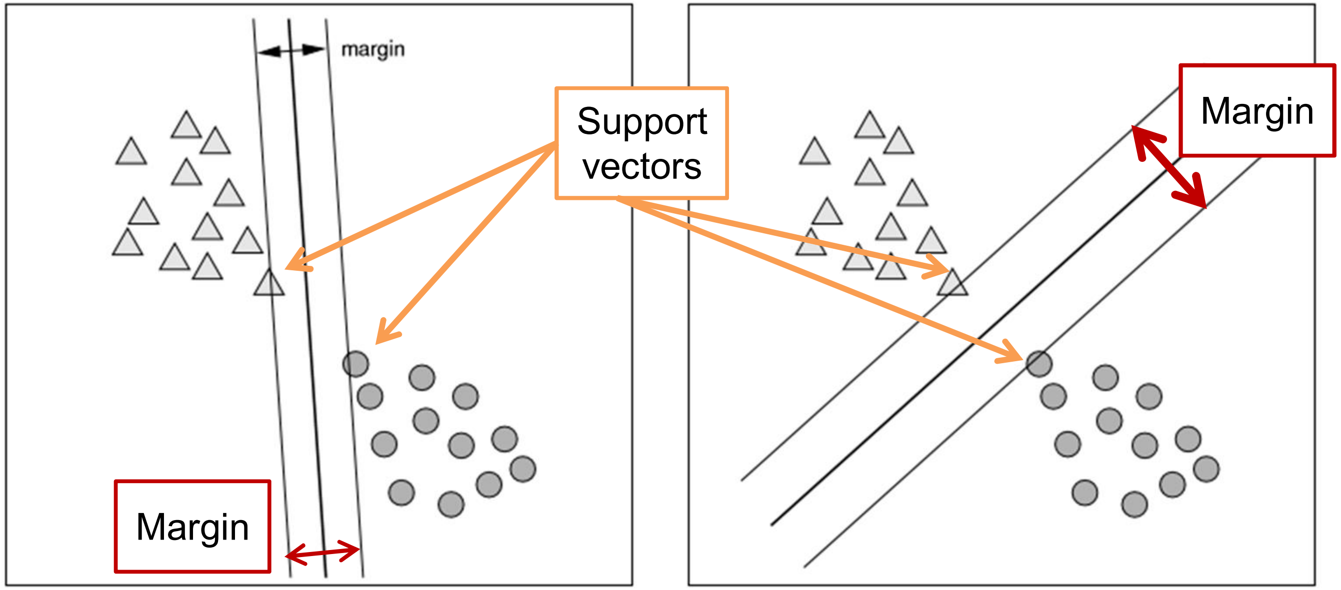


<u>Remark</u>:

works also for Regression!

ML-Algorithms: **Support Vector Machine (SVM)**

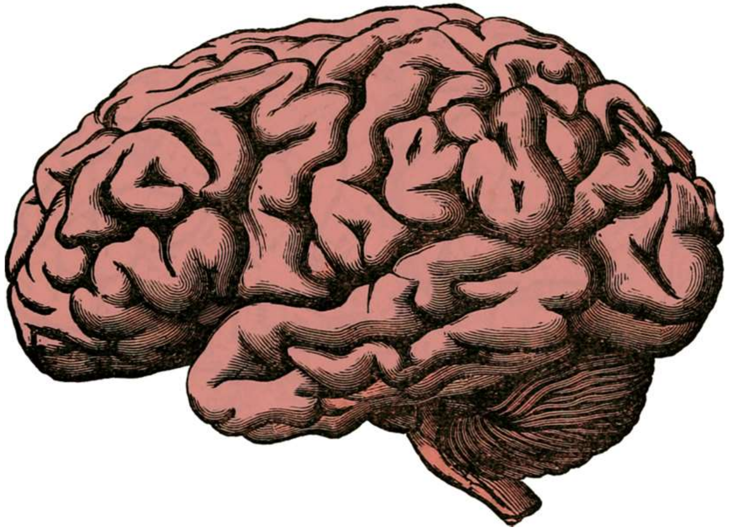
Which of all possible lines between and is the best?

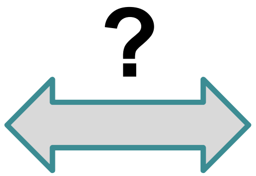


The line with the **maximum margin** is the best !

Remark: SVMs are also called *Maximum Margin Classifier*

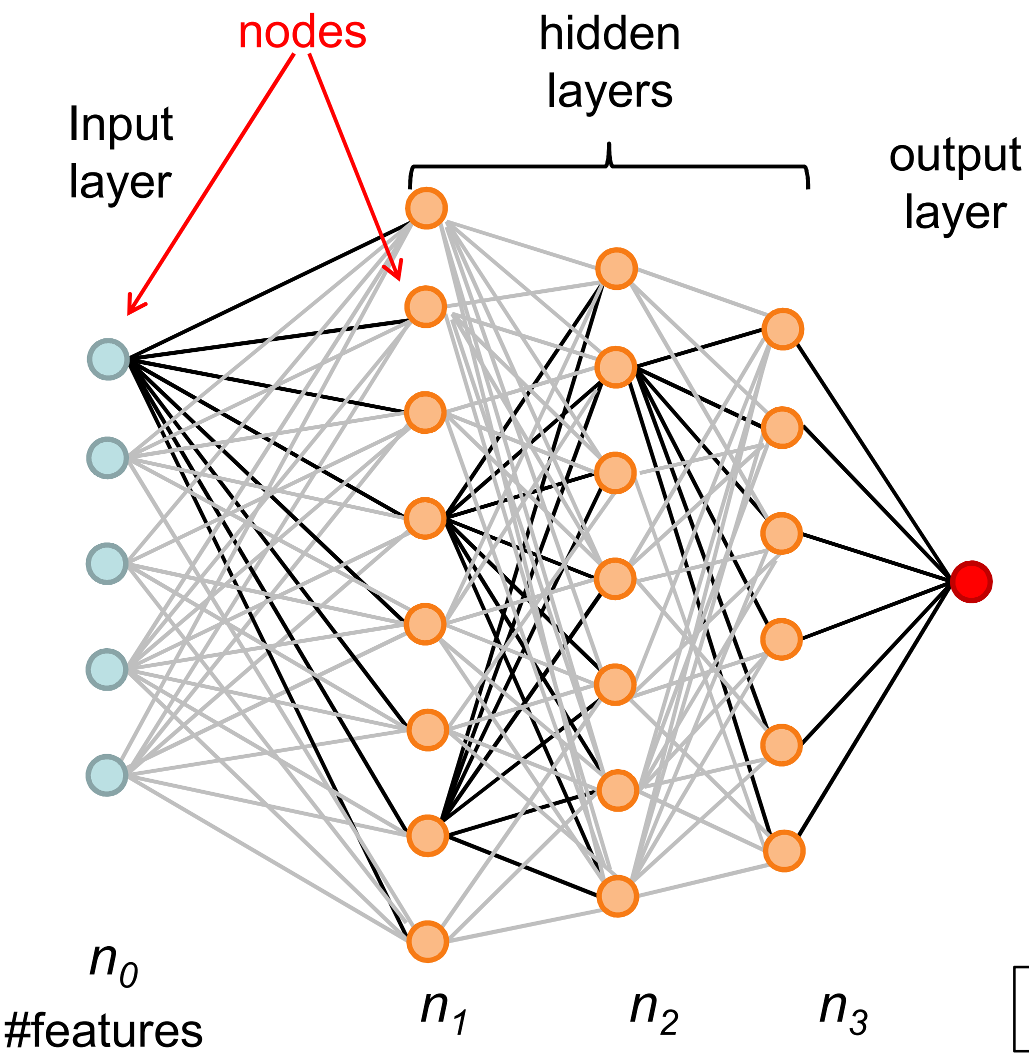
ML-Algorithms: **Artificial Neural Network**





How similar are the human brain and artificial neural networks ?

Artificial Neural Network



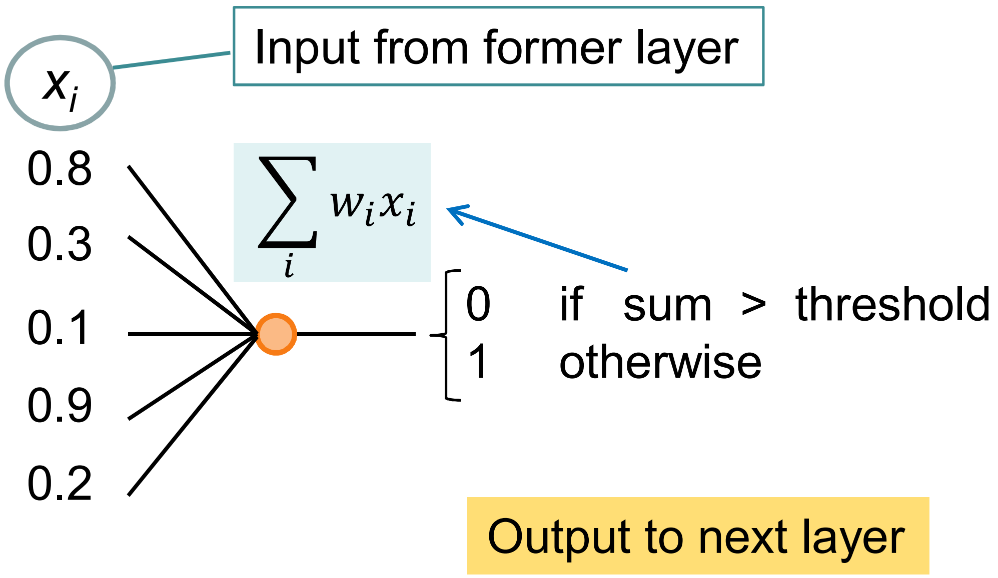
Simplest architecture: all nodes of the hidden layers h_i are connected to all nodes of layers h_{i-1} and h_{i+1}

In case of regression: Output is a single real number

n_i #nodes

ANN – Nature and Function of the Nodes

Each **node** has n_{i-1} input values from the former layer.
The output of each **node** is simply **0 or 1**, depending on the input.
All nodes contribute with different **weights** w_i.

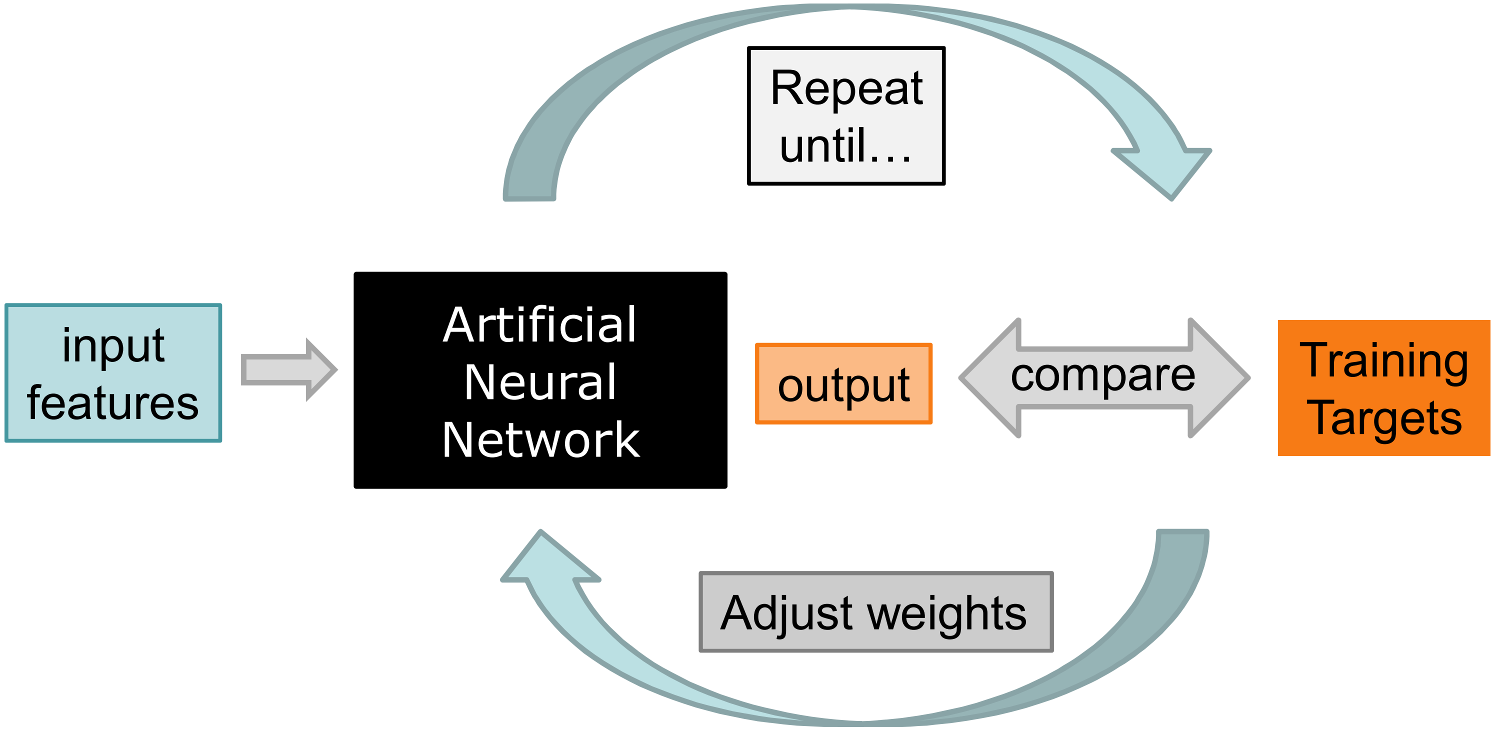


The nodes are also called:
perceptron
or
artificial neuron

ANN – Training

In the training process, the **weights** are optimized to describe the training target.

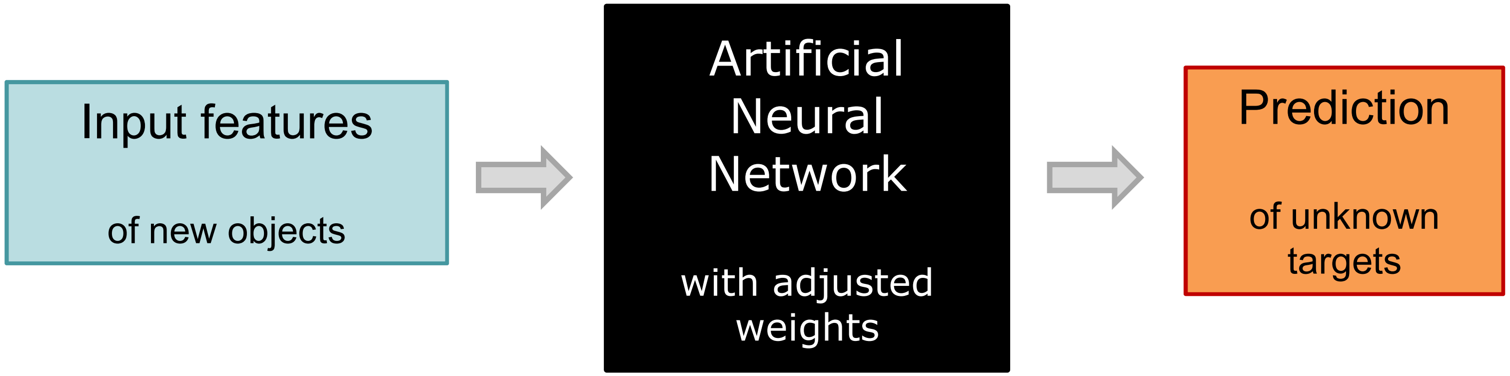
The ANN is a highly non-linear function, that can „*learn*“ complex non-linear problems.



ANN – Prediction

The prediction is performed with the optimized **weights.**

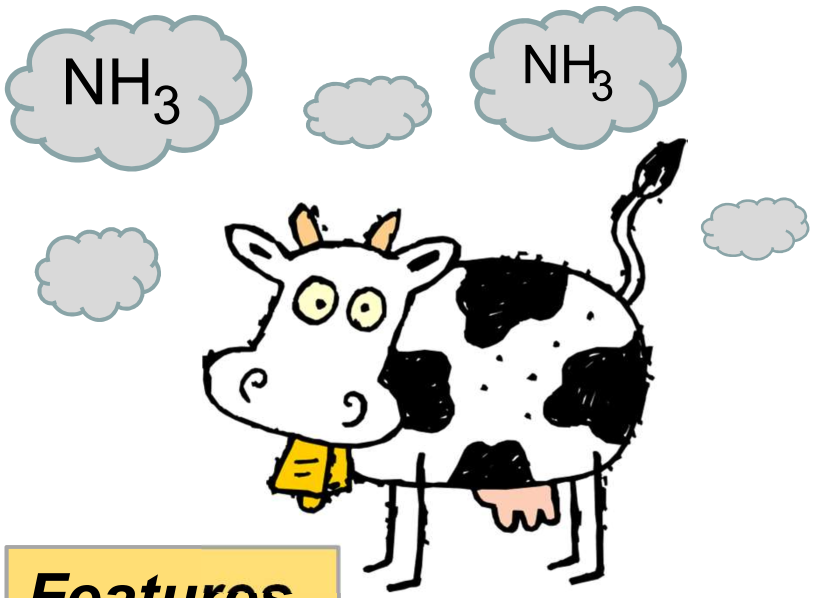
The training process can be very time-consuming, but the **prediction** ist very **fast**.



Emission Data Dummerstorf

Messuring period: November 1st, 2016 to August 30th, 2017 (**10 months**), with a short break of 3 weeks (from April 4th to April 26th, 2017).

Messuring frequency: every 60 minutes



Measured values:

time [date/ day of the year]

day time [hour]

temperature (inside) [C]

wind direction [degree]

wind speed [m/s]

NH3 emission [kg/year per animal place]

Features

(input-values)

Target

(output-values)

Features / Input-variables: **Time**

Both time components are **periodic variables**!

minutes of the day:

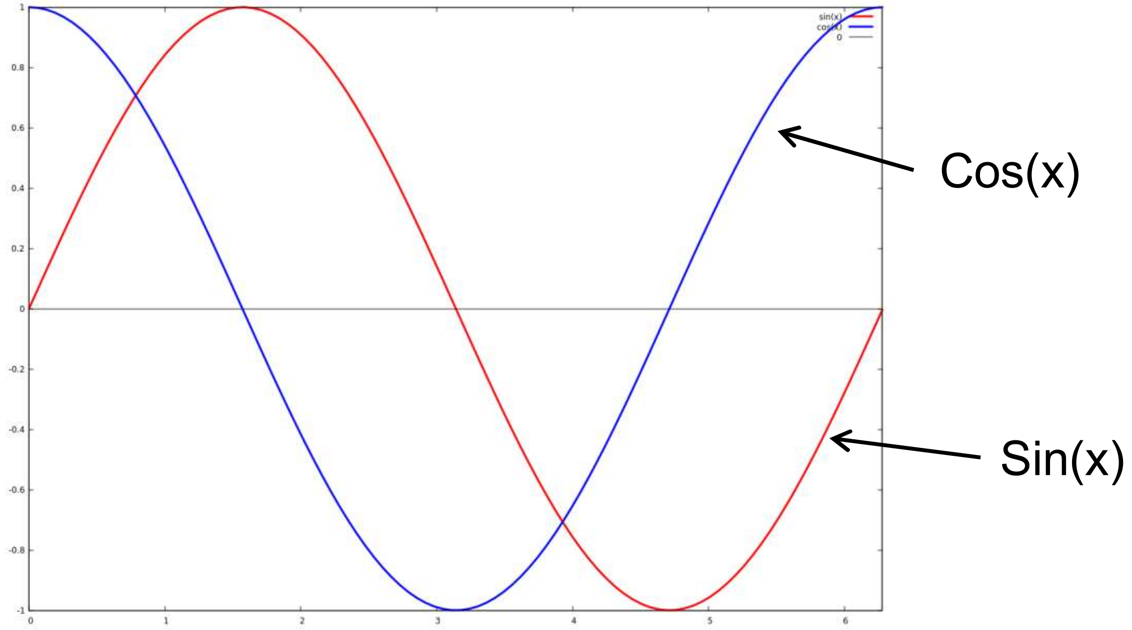
hour_1 = Sin (time * 2π / 24)

hour_2 = Cos (time * 2π / 24)

days of the year:

days_1 = Sin (time * 2π / 365.25)

days_2 = Cos (time * 2π / 365.25)

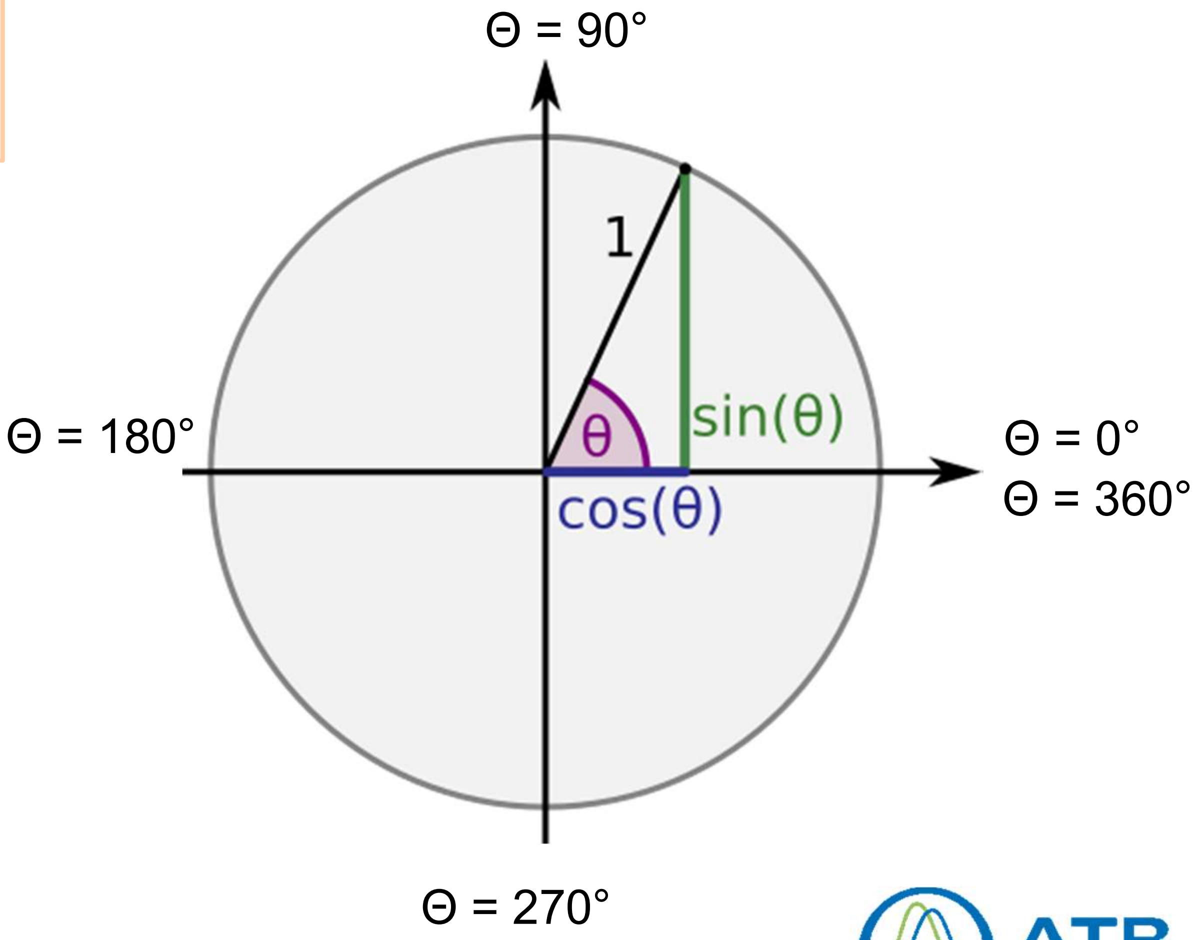


Features / Input-variables: **Wind Direction**

The wind direction is **periodic** with respect to the angle Θ

wdir_1 = Sin (2π Θ / 360)

wdir_2 = Cos (2π Θ / 360)



Features / Input-variables

The remaining variables can be used directly as input variables !

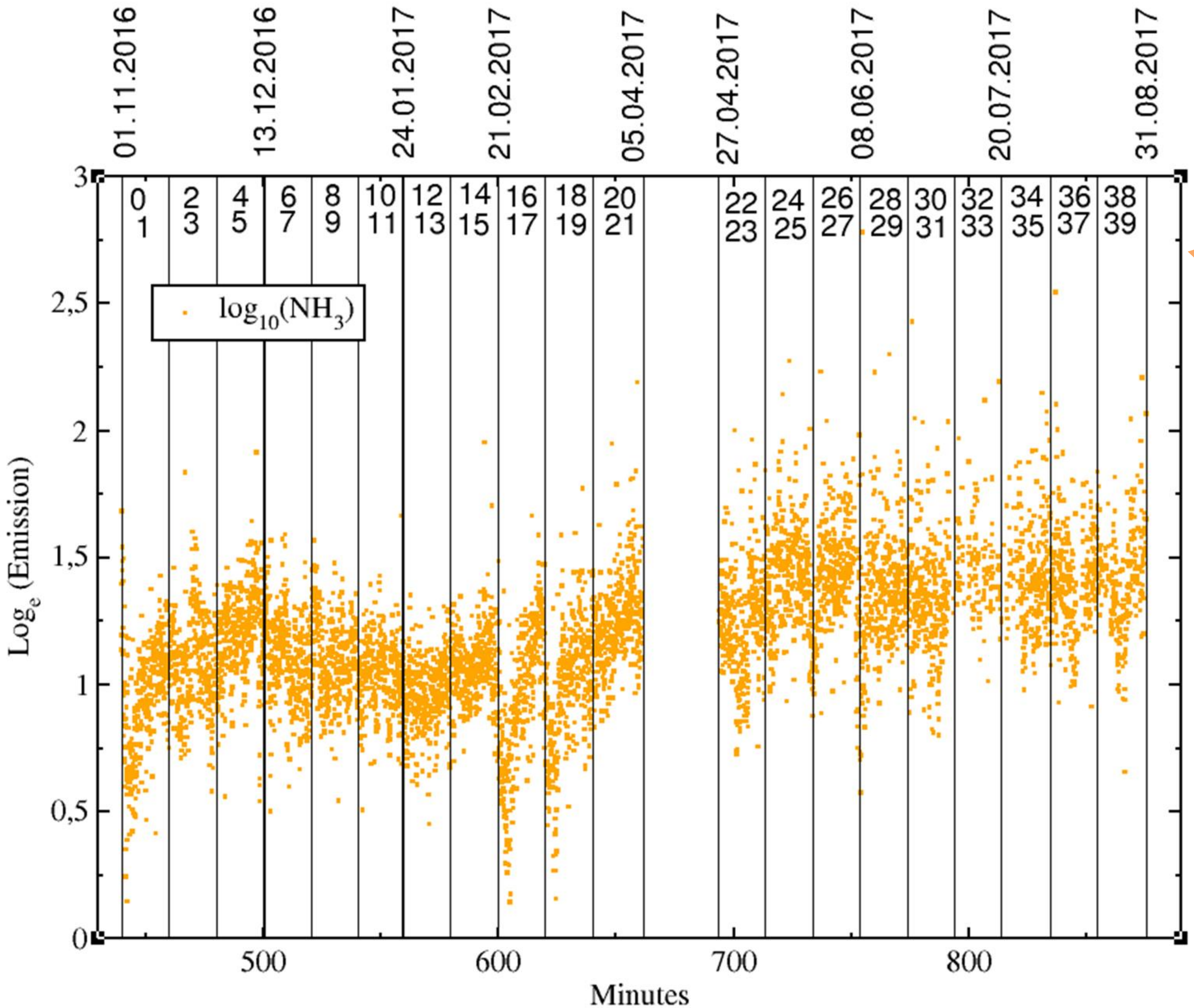
temperature (inside) [C]

windspeed (10 m height) [m/s]

Remark:

For numerically reasons, it is useful to **normalize** the variables to fulfil a standard-normal distribution.

Dummerstorf Data

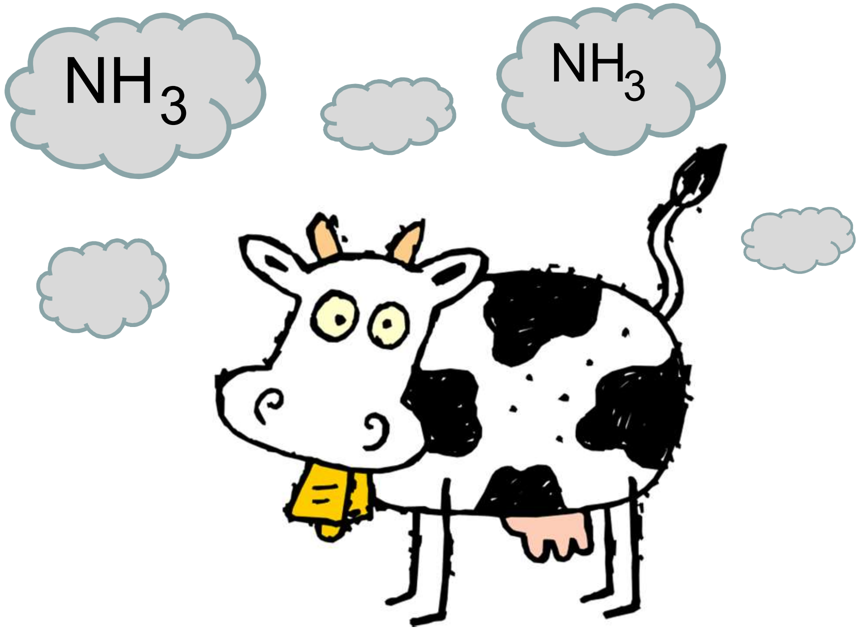


20 Intervals
of 2 weeks each

Remark:

On the y-axis the Logarithm of the measured emission is shown!

Goto Jupyter Emission Notebooks !



jupyter

python™