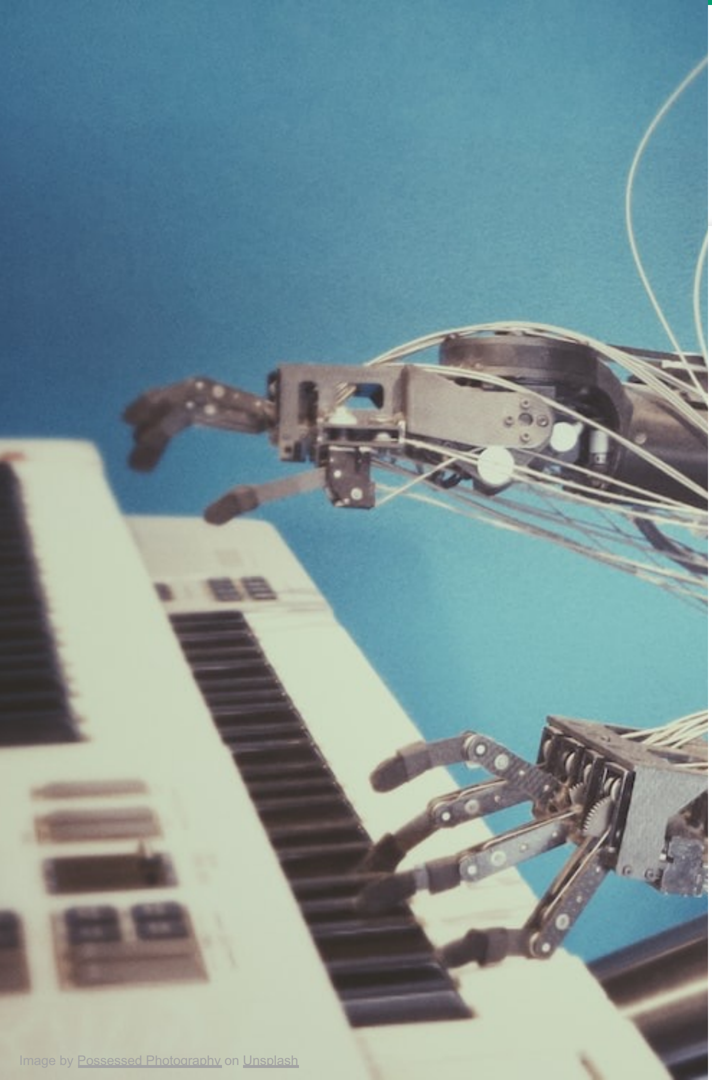


Applied Artificial Intelligence
03 - AI Lifecycle: Modeling & Evaluation

Univ.-Prof. Dr-Ing. habil. Niklas Kühl
www.niklas.xyz

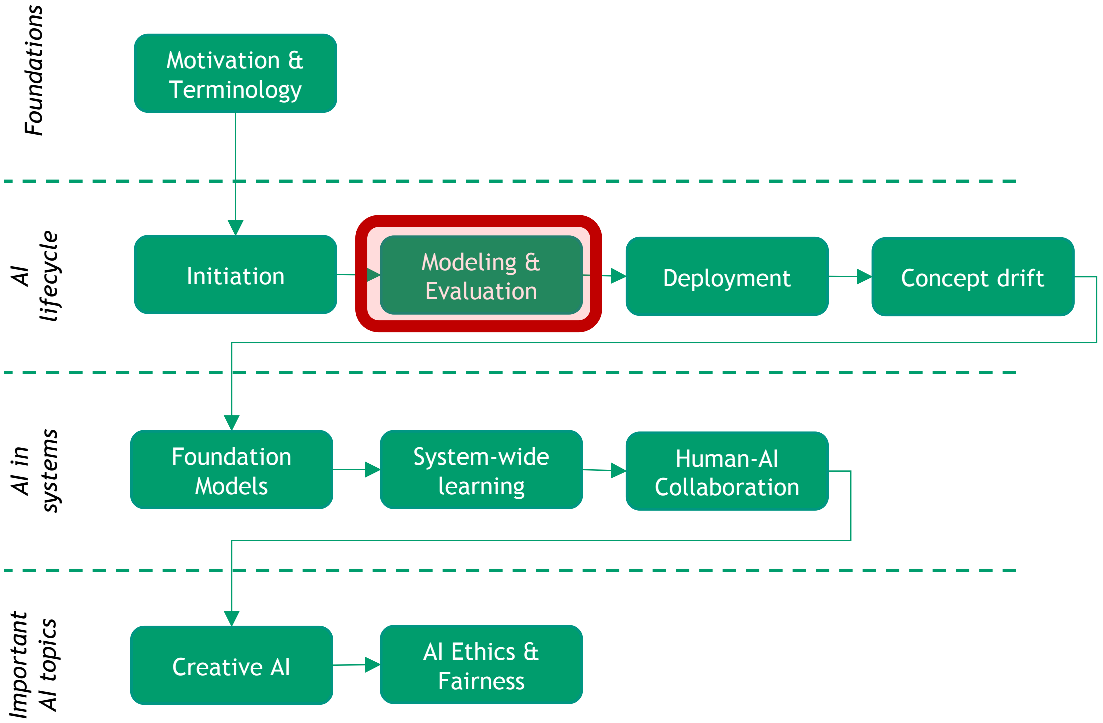
University of Bayreuth

Karlsruhe Institute of Technology

TUM School of Management

www.uni-bayreuth.de | www.kit.edu | www.tum.de | www.fim-rc.de | www.wirtschaftsinformatik.fraunhofer.de

Organizational
The story of the lecture



Repetition
Machine learning algorithms need measurable goals.

Business Problem

Objectives and business **success criteria** describe the problem. However, these descriptions are often subjective and lack measurability.

Translation

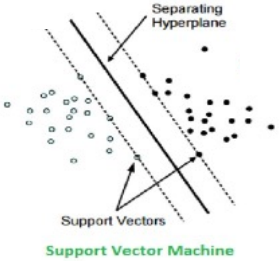
Machine Learning Algorithms

Machine learning algorithms on the other hand require a **clear, measurable goal** that can be quantified by a dominate metric.

Supervised Learning

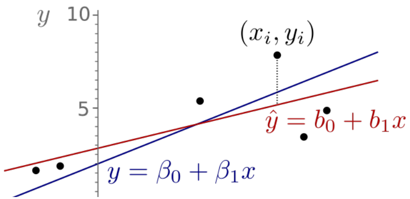
Classification | *classify instances into classes*

[1]



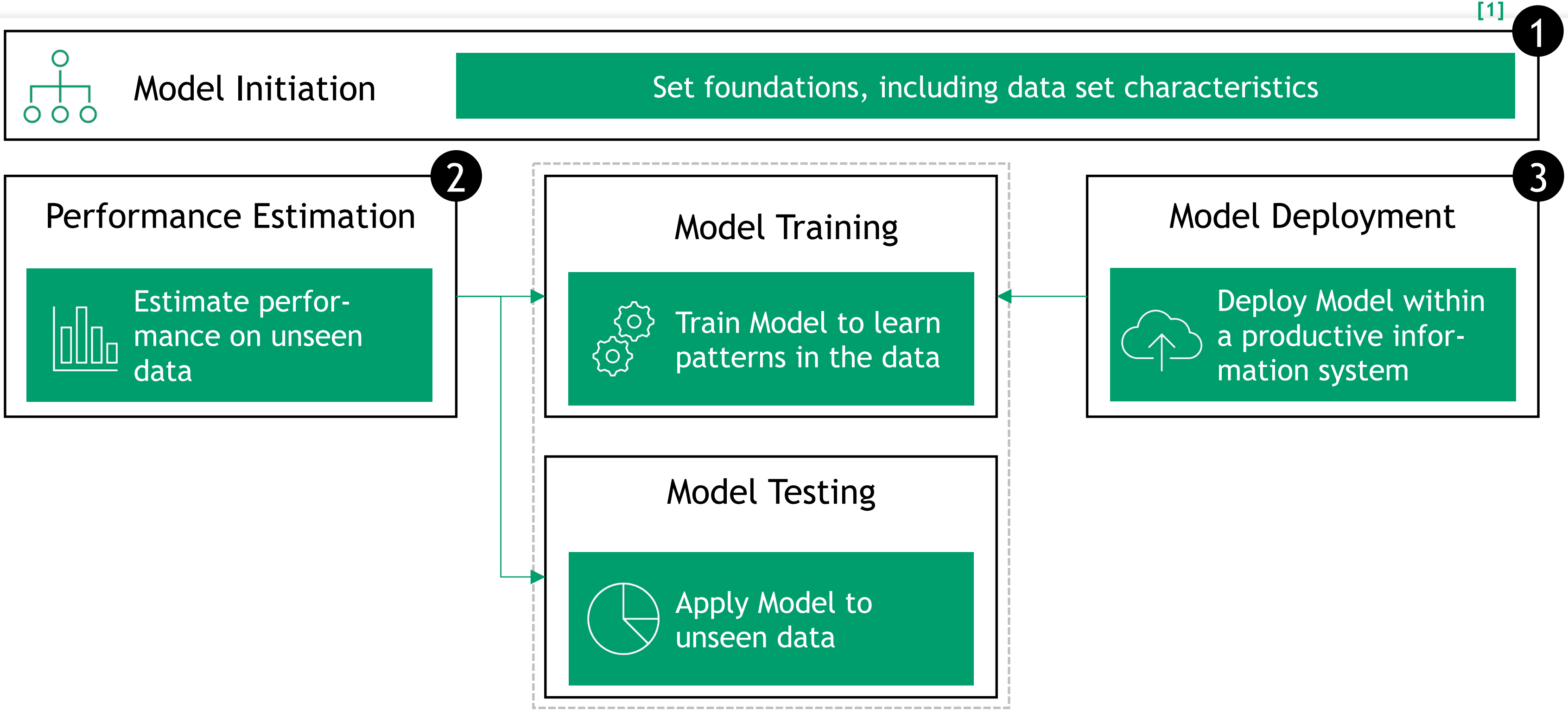
Regression | *predict numerics as close as possible*

[2]



Devi, T. & Srinivasan, A. & Sudha, S. & Narasimhan, Devadoss. (2019). Web enabled paddy disease detection using Compressed Sensing. Mathematical Biosciences and Engineering. 16. 7719-7733. 10.3934/mbe.2019387. [1]
https://dewiki.de/b/139798 [2]

The AI Lifecycle
Required steps for supervised machine learning.



[1]

1 Model Initiation

Set foundations, including data set characteristics

2 Performance Estimation

Estimate performance on unseen data

Model Training

Train Model to learn patterns in the data

Model Testing

Apply Model to unseen data

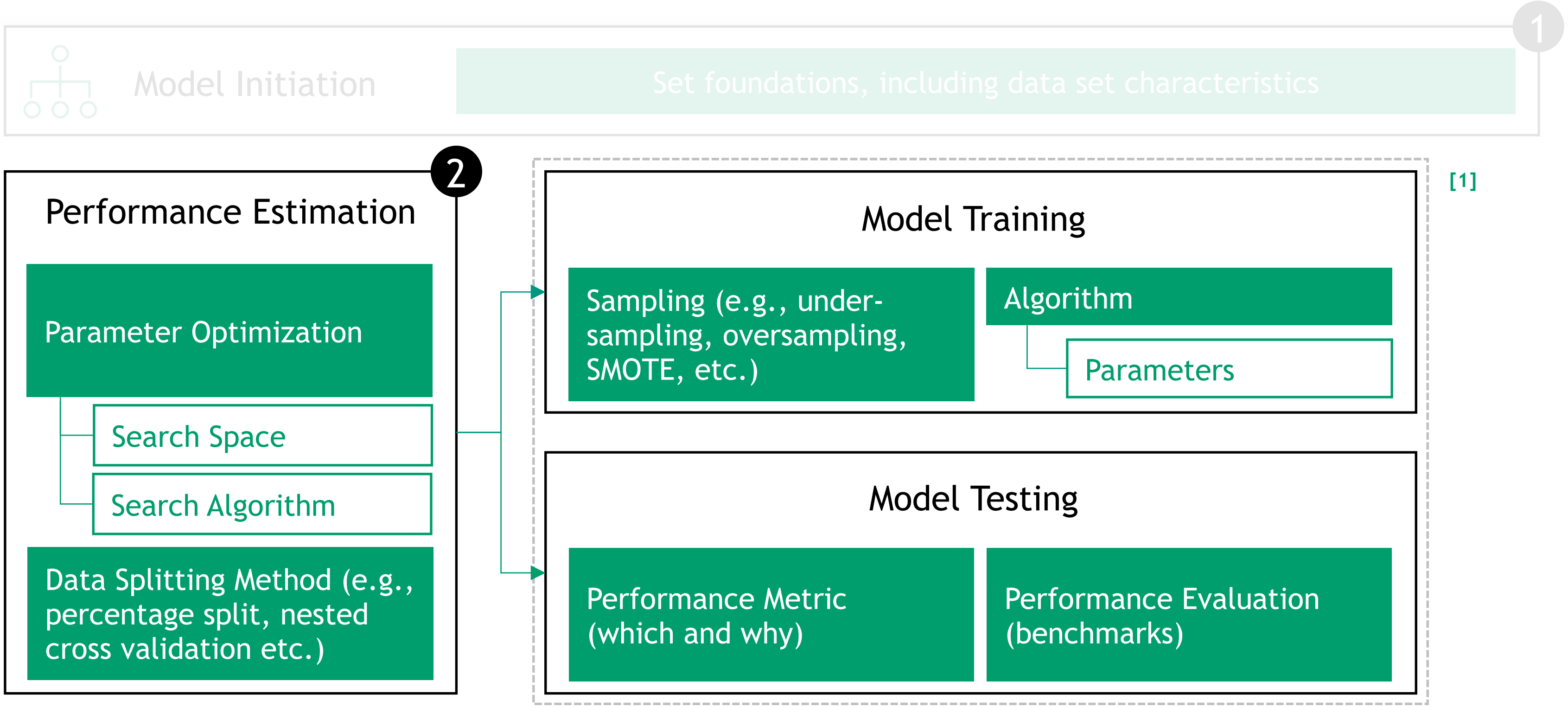
3 Model Deployment

Deploy Model within a productive information system

Kühl et al. (2021). How to Conduct Rigorous Supervised Machine Learning in Information Systems Research: The Supervised Machine Learning Report Card [1]

The AI Lifecycle

Required steps for supervised machine learning.



1

Model Initiation

Set foundations, including data set characteristics

2

Performance Estimation

Parameter Optimization

Search Space

Search Algorithm

Data Splitting Method (e.g., percentage split, nested cross validation etc.)

[1]

Model Training

Sampling (e.g., under-sampling, oversampling, SMOTE, etc.)

Algorithm

Parameters

Model Testing

Performance Metric (which and why)

Performance Evaluation (benchmarks)

Kühl et al. (2021). How to Conduct Rigorous Supervised Machine Learning in Information Systems Research: The Supervised Machine Learning Report Card [1]

Objectives

What are the learning goals of this lecture?

EXPLORE

discover the basic concepts of supervised machine learning algorithms

UNDERSTAND

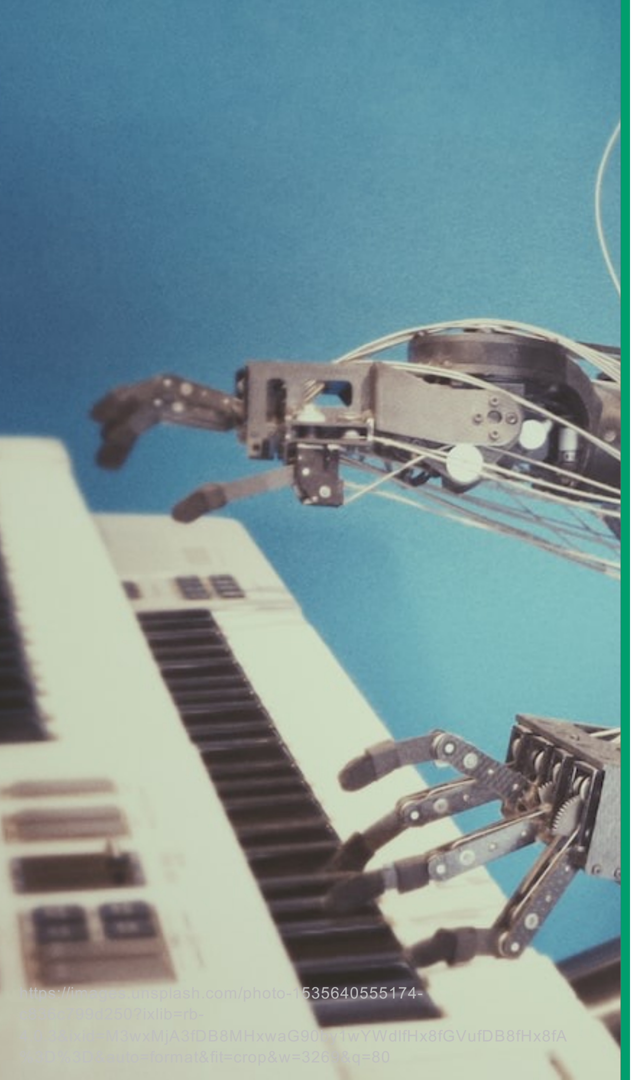
understand performance metrics for regression and classification

INTENSIFY

learn how to find optimal parameters for ML algorithms without overfitting the model

APPLY

be able to distinguish between sampling methods



https://images.unsplash.com/photo-1535640555174-c836c799d250?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=3269&q=80

1. Algorithms
   - Basic functionality
   - Selected parameters
2. Performance metrics
3. Evaluation options

https://images.unsplash.com/photo-1510511459019-5dda7724fd87?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=3270&q=80

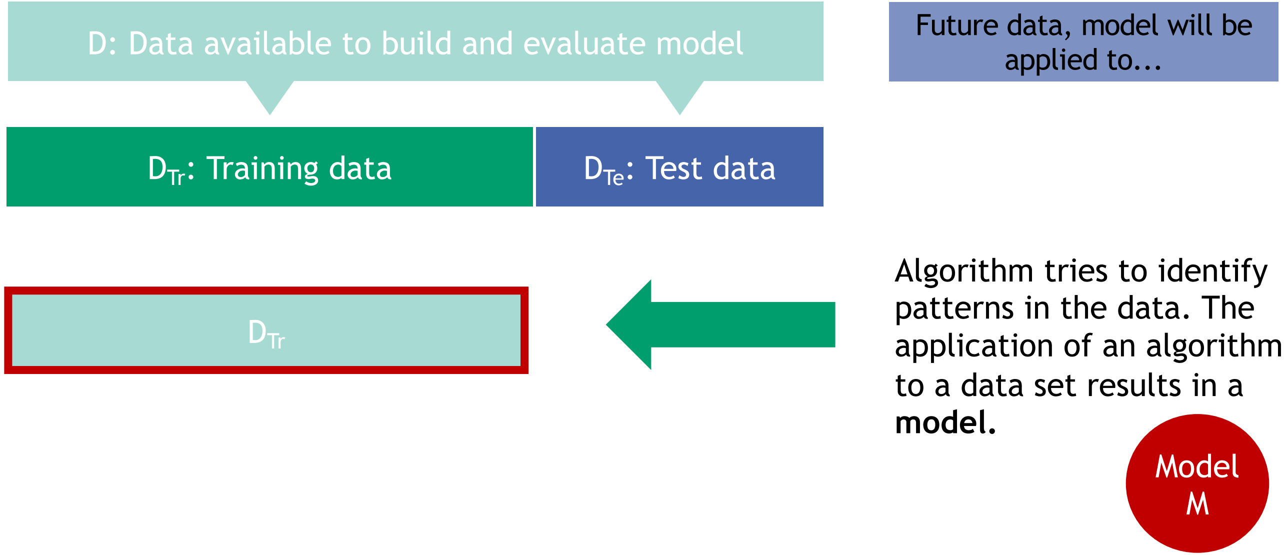
1 Algorithms

- Basic functionality
- Selected parameters

2 Performance metrics

3 Evaluation options

Algorithms
What for?


D: Data available to build and evaluate model
Future data, model will be applied to…
D_{Tr}: Training data
D_{Te}: Test data
D_{Tr}
Model M

Algorithm tries to identify patterns in the data. The application of an algorithm to a data set results in a **model.**

Algorithms
Groups and some examples

Ensemble
- Random Forest
- Bagging
- Boosting
- …

Regression
- Linear Regression
- Logistic Regression
- Multivariate adaptive Regression
- …

Decision Tree
- Random Tree
- Random Forest
- C5.0
- …

Association rules
- A priori
- Eclat
- …

Supervised Machine Learning Algorithm Groups

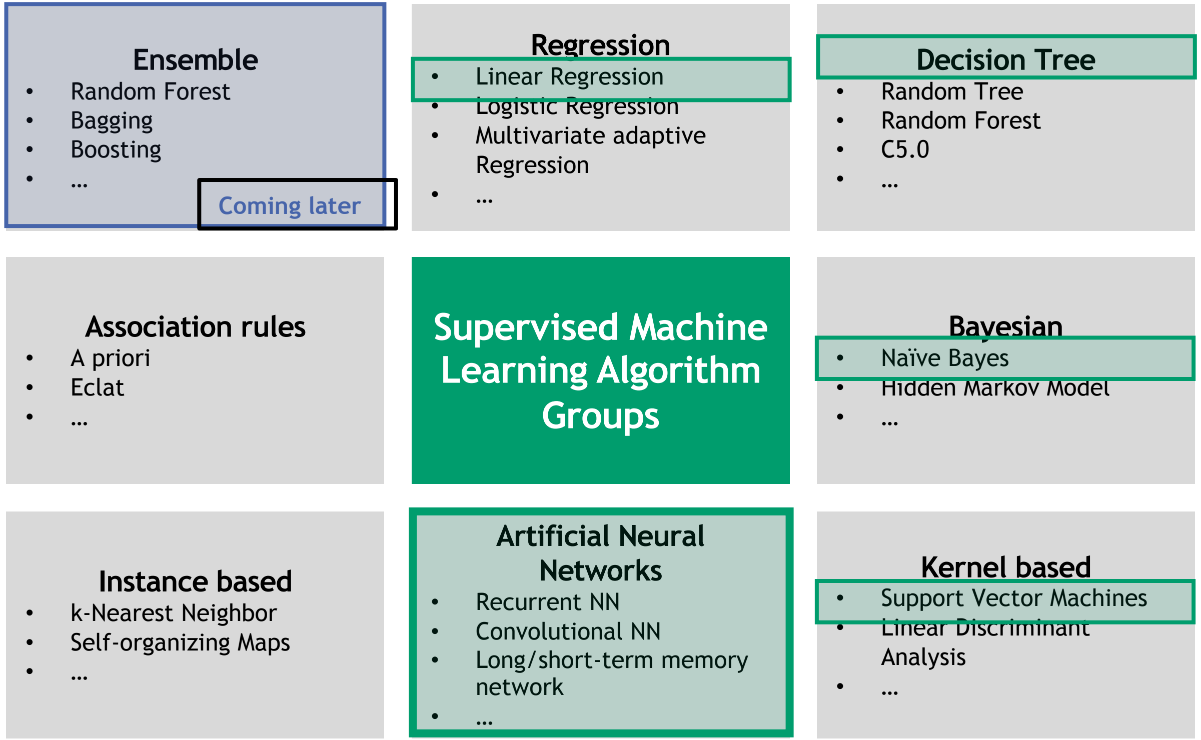
Bayesian
- Naïve Bayes
- Hidden Markov Model
- …

Instance based
- k-Nearest Neighbor
- Self-organizing Maps
- …

Artificial Neural Networks
- Recurrent NN
- Convolutional NN
- Long/short-term memory network
- …

Kernel based
- Support Vector Machines
- Linear Discriminant Analysis
- …

Algorithms
Groups and some examples



Ensemble
- Random Forest
- Bagging
- Boosting
- …

Coming later

Regression
- Linear Regression
- Logistic Regression
- Multivariate adaptive Regression
- …

Decision Tree
- Random Tree
- Random Forest
- C5.0
- …

Association rules
- A priori
- Eclat
- …

Supervised Machine Learning Algorithm Groups

Bayesian
- Naïve Bayes
- Hidden Markov Model
- …

Instance based
- k-Nearest Neighbor
- Self-organizing Maps
- …

Artificial Neural Networks
- Recurrent NN
- Convolutional NN
- Long/short-term memory network
- …

Kernel based
- Support Vector Machines
- Linear Discriminant Analysis
- …

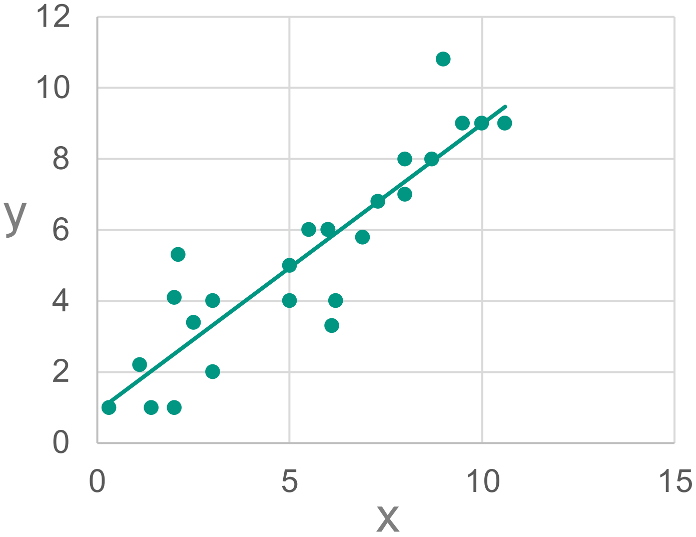
Algorithms
Linear Regression

x: observation | y: output value | b: gradient | a: interception

A linear regression represents a statistical procedure that explains an observed dependent variable (y) by one or more independent variables (x) and maps the **underlying relationship** in a regression graph.

Adding new data points influences the calculation of the linear regression line.

Regression line equation: $y_i = a + x_i \; * b$



Prof. Dr. Niklas Kühl

Algorithms
Logistic Regression

X: observation

y: output value

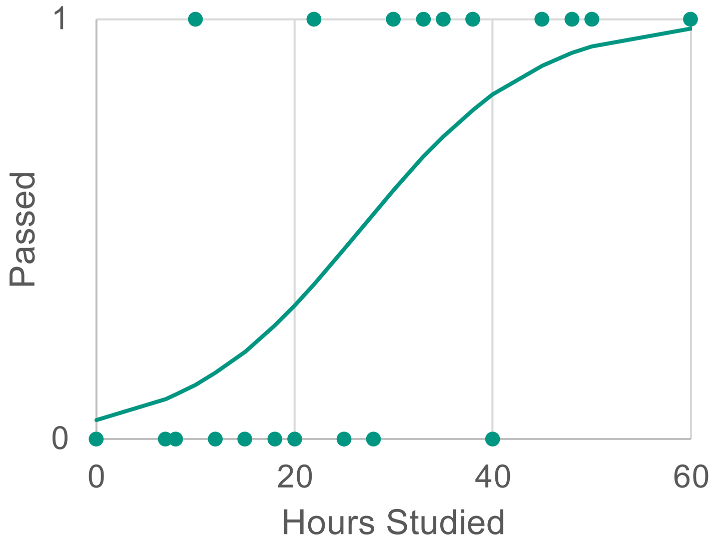
b: gradient

a: interception

Logistic regression is a statistical method for modeling the probability of a binary outcome (0 or 1) based on one or more independent variables (x). Unlike linear regression, it maps the relationship using a logistic (S-shaped) curve.

It is often used for classification problems and provides the odds ratio for the likelihood of an event occurring in response to the independent variable(s).

Regression line equation: $P(y) = \frac{1}{1+e^{bX}}$



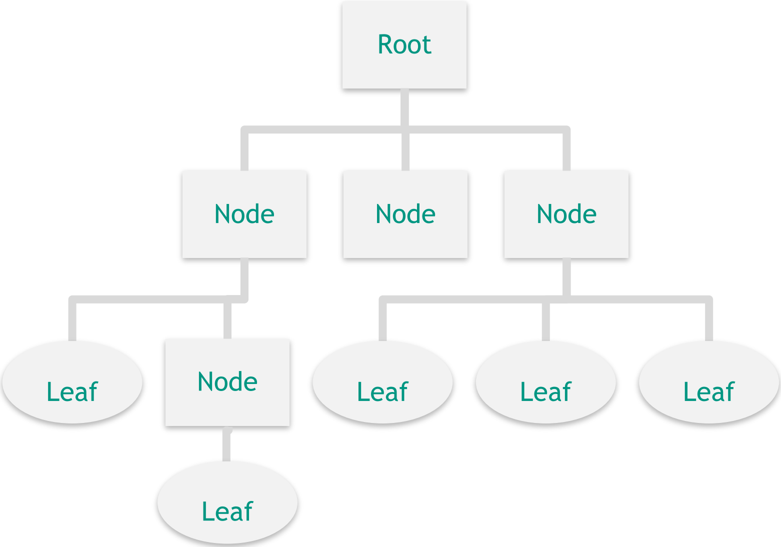
Algorithms
Decision Tree

Root node **Decision node** **Leaf**

A decision tree (DT) represents a decision procedure based on a sequence of “**questions**“. These questions are usually called **tests**. Each question is represented by a **node** in the DT.

The answer to the first question determines which question will be asked next.

Question will be asked until a **decision** is reached. The decision is the final **prediction** of the DT. The decision is represented by a **leaf** in the DT



Algorithms
Naïve Bayes (1/2)

X: observation

Y: output

Bayes classifier – is an approach for classification problems. It assigns every instance to the most probable class Y based on all given input features X.

$$P(Y|X) = \frac{P(Y) * P(X \mid Y)}{P(X)}$$

Naïve Bayes – is a special type of a Bayes classifier. It assumes conditional independence, which means that every feature X is statistically independent, and every feature contributes independently to the probability of its class Y.

$$Y = \arg\max P(Y) * \prod P(X|Y)$$

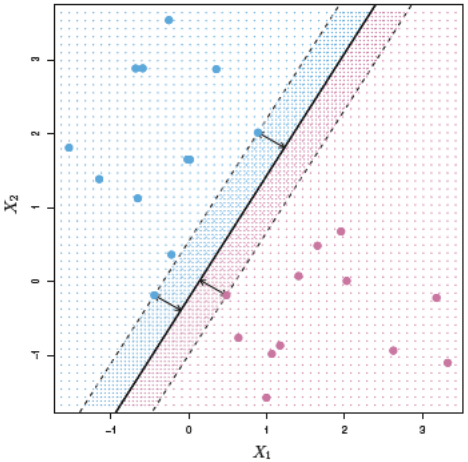
Distribution of values of each features – represented by $P(X|Y)$, is an assumption we have to make prior to applying the algorithm. Possible Naïve Bayes classifiers are *Gaussian-*, *Multinomial-* and *Bernoulli Naïve Bayes*

Algorithms
Support Vector Machine

x: observation

y: output value

Support Vector Machines – or SVMs, are based on the concept of a hyperplane. For two dimensions, a hyperplane is a line which separates the instances into two separate subsets. The closest instances to the hyperplane are called support vectors. The aim of an SVM algorithm is to maximize the distances to the support vectors.



[1]

SVMs can be used for **classification** and **regression** problems.

Multiclass problems – can be solved by SVM when using the following techniques:

One-Versus-One Classification – uses several hyperplanes and compares each pair of classes.

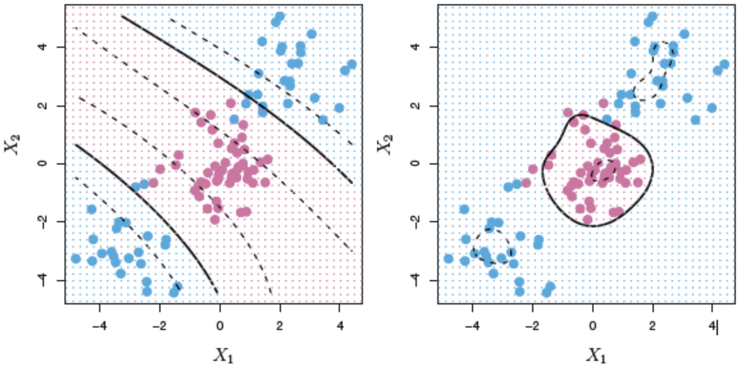
One-Versus-All Classification – compares one class with all other classes.

Mini, Tobias. (2016). Sentiment Analyse. 10.13140/RG.2.1.1957.1446.[1]

Algorithms
Support Vector Machine

Nonlinear data – can be addressed by using different kernels in the SVM algorithm. A kernel is a function which quantifies the similarity of two observations.

- **Polynomial kernel** – results in non-linear lines as separation boundaries
- **Radial kernel** – results in round separation boundaries



[1]

Parameters – for SVM typically are:

- **C** – Number of falsely assigned instances
- **ε** – defines range of acceptable error (falsely assigned instances within the range do not count for parameter C)

Slepaczuk, Robert & Zenkova, Maryna. (2019). Robustness of Support Vector Machines in Algorithmic Trading on Cryptocurrency Market. Central European Economic Journal. 5. 186-205. 10.1515/ceej-2018-0022.[1]

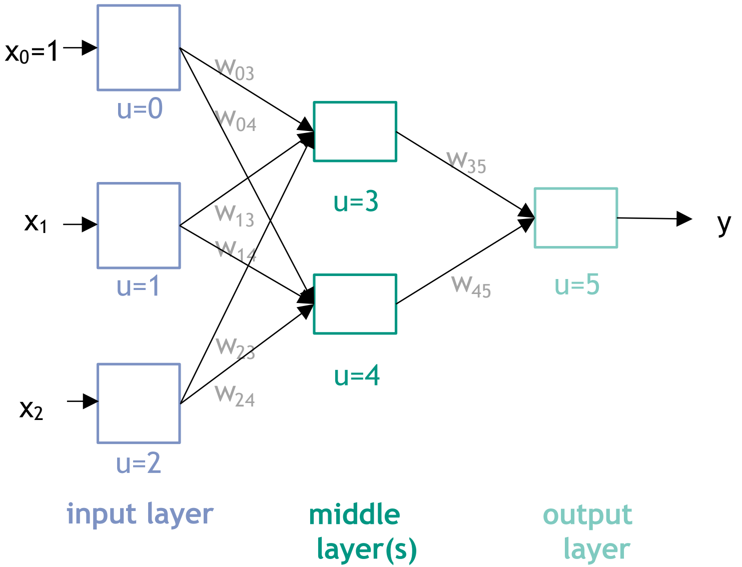
Algorithms
Artificial Neural Networks

Neurons are the fundamental building blocks of a neural network. Each neuron has input, activation and output functions that allow it to process and transmit information.

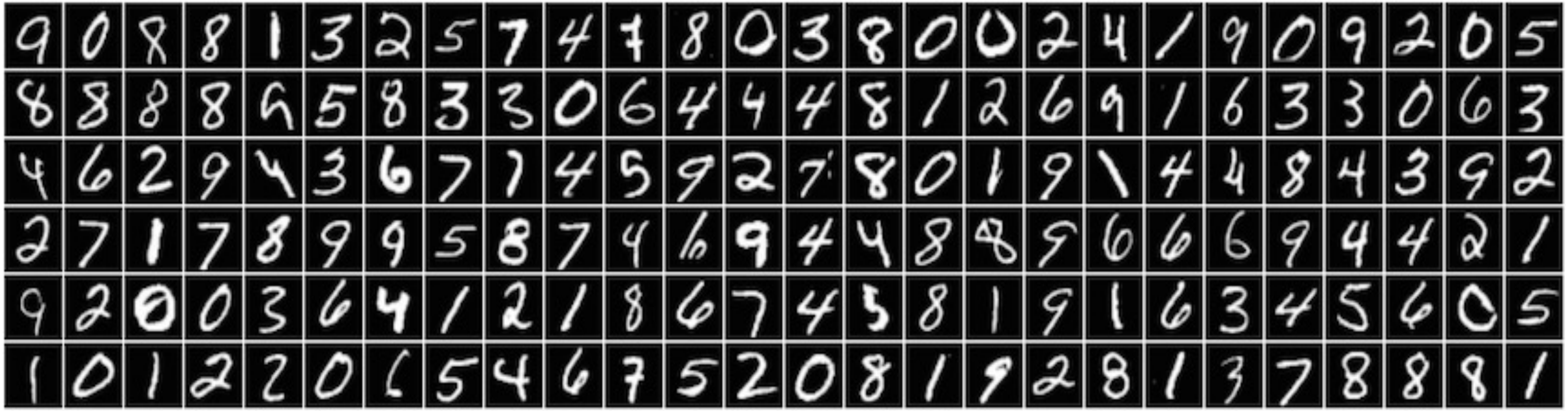
The **input layer** acts as a gateway, receiving raw data and passing it on to the network.

The **middle layer(s)**, or hidden layers, transform the raw data by applying activation functions that extract significant patterns and features that the neural network needs to understand and process.

The **output layer** synthesizes everything the network has learned and provides a refined and calculated response or prediction based on the input.



Algorithms
Neural Networks example

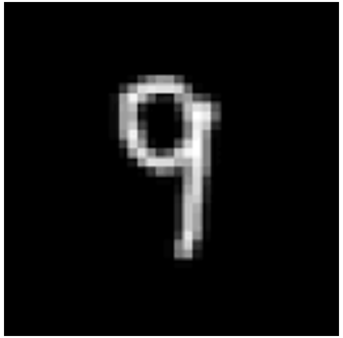


[1]

How can I recognize a handwritten number?

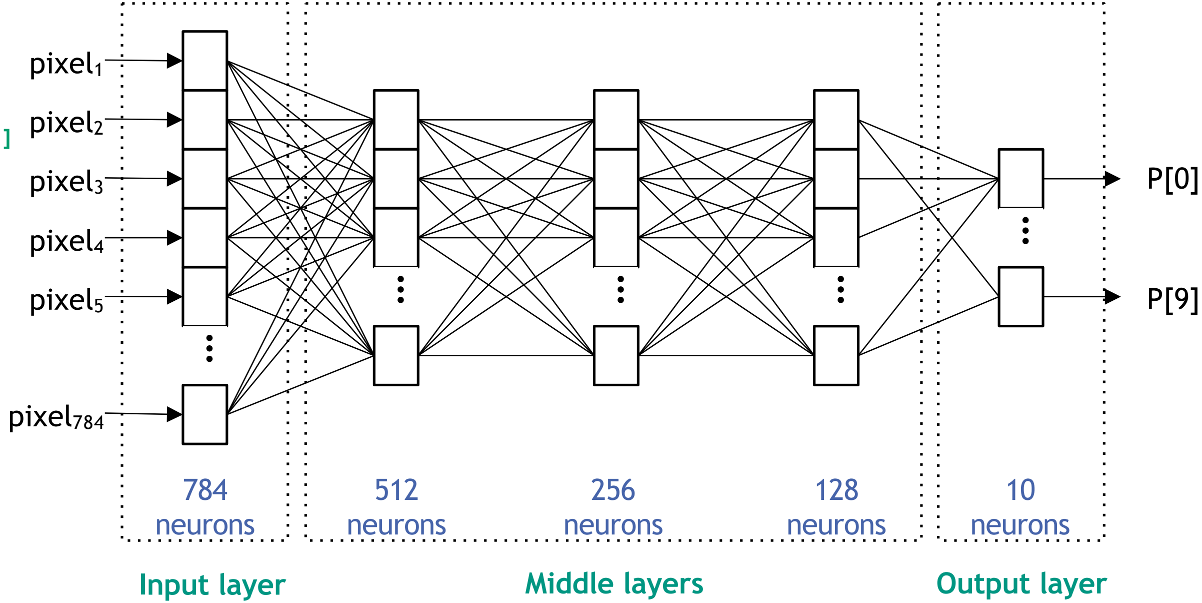
MNIST dataset [1]

Algorithms
Neural Networks example: Multi Layer Perceptron



[1]

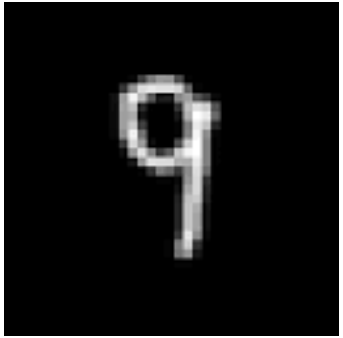
28 x 28 (=784) pixels
in different shades of grey



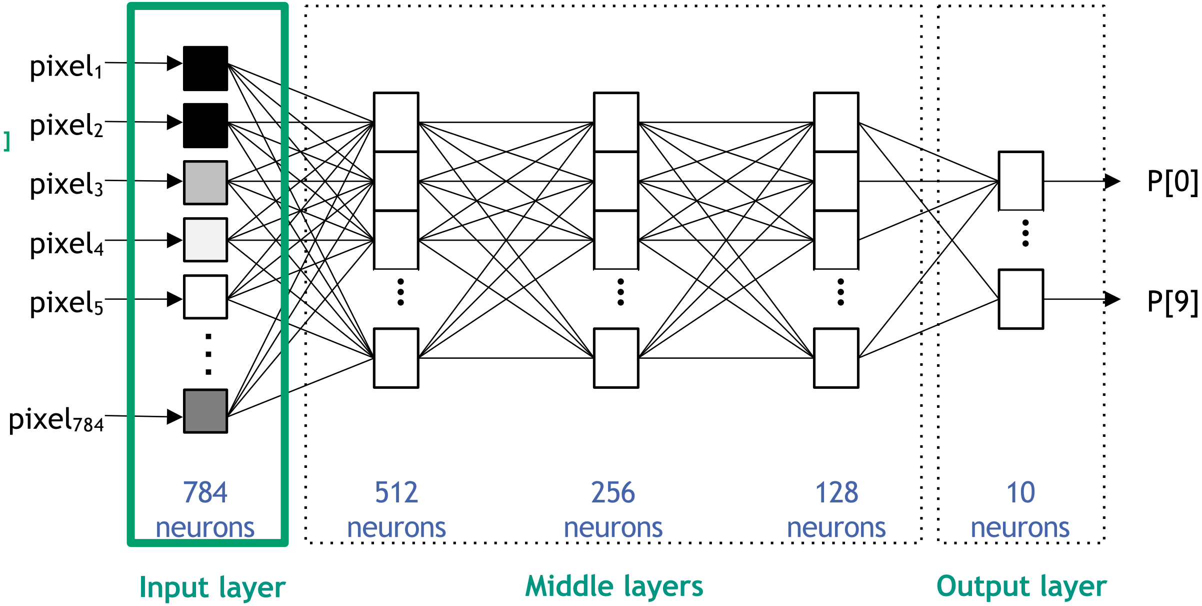
The input layer receives the grey value (0 – 255) for each individual pixel of the image

MNIST dataset [1]

Algorithms
Neural Networks example: Multi Layer Perceptron


[1]

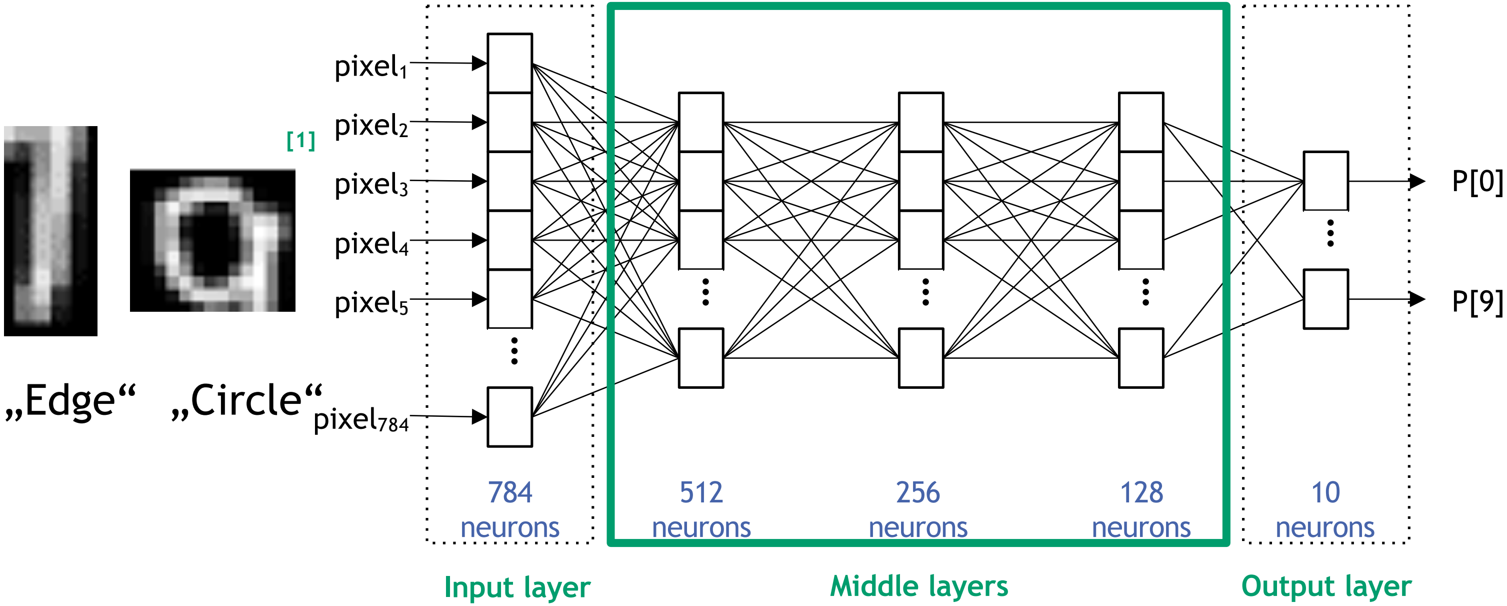
28 x 28 (=784) pixels
in different shades of grey



The input layer receives the grey value (0 – 255) for each individual pixel of the image

MNIST dataset [1]

Algorithms
Neural Networks example: Multi Layer Perceptron

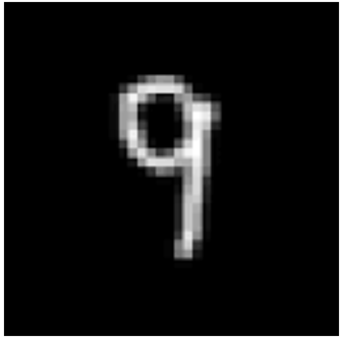


It can be assumed that abstract features such as edges are initially recognized in the intermediate layers, while shapes are derived in the following layers

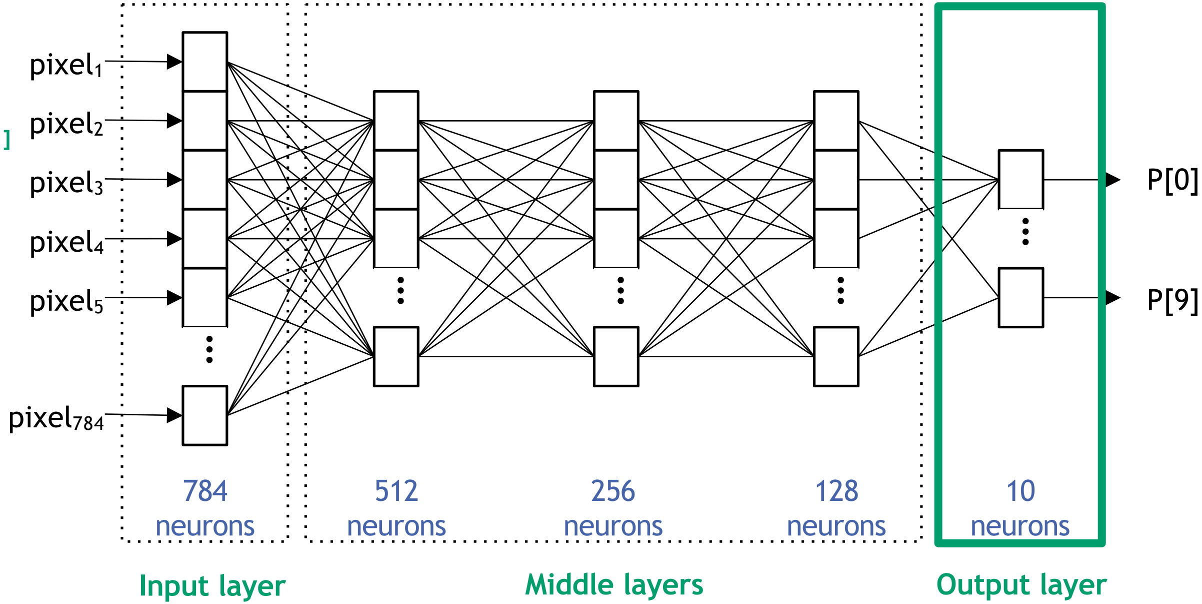
MNIST dataset [1]

Algorithms

Neural Networks example: Multi Layer Perceptron



28 x 28 (=784) pixels
in different shades of grey



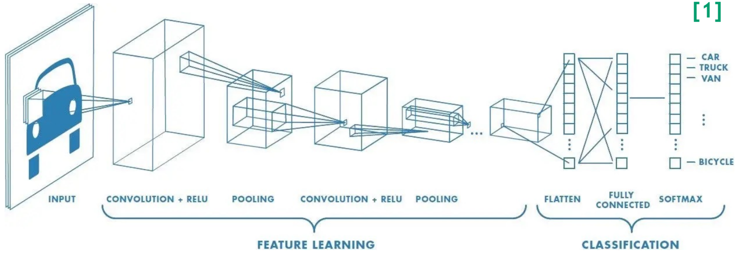
In our example, the output layer consists of the 10 possible digits. The output layer finally reduces to these possibilities.

MNIST dataset [1]

Algorithms
Convolutional neural networks (CNN)

Convolutional neural networks – is a special type of ANN with multiple hidden layers for special proposes. The different layers are often stacked and occur on multiple positions in the CNN.

- **Convolutional layers** – reduce the number of required parameters by convolving input data.
- **Pooling layers** – reduce unimportant information by pooling, e.g. pass only the strongest signal of multiple input signals
- **Fully connected layers** – connect their output with every neuron of the next layer, similar to a hidden layer in a standard ANN.



[1]

Use cases – for CNNs are visual data. CNNs enables to reduce the high amount of input data (pixels) and allows a learning with slight preprocessing.

Parameters – for CNNs are:

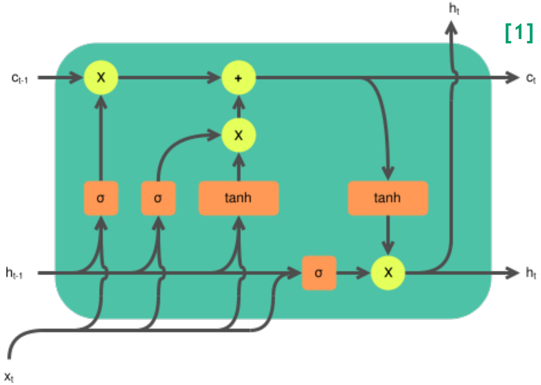
Order of layers – is key for learning with CNNs

Size – for convolving or pooling define the tradeoff between information loss and complexity reduction

https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/ [1]

Algorithms
Long short term memory network (LSTM)

Long short term memory networks – or LSTMs, are a subset of RNNs with the focus on learning long-term dependencies. The recurrent element by using previous outputs is still valid but is complemented by the concept of a cell state.



[1]

Every neuron in the LSTM is extended to a cell with the capability to store an internal state. The state is regulated by:

Input gate – controls which values of the cell state get updated

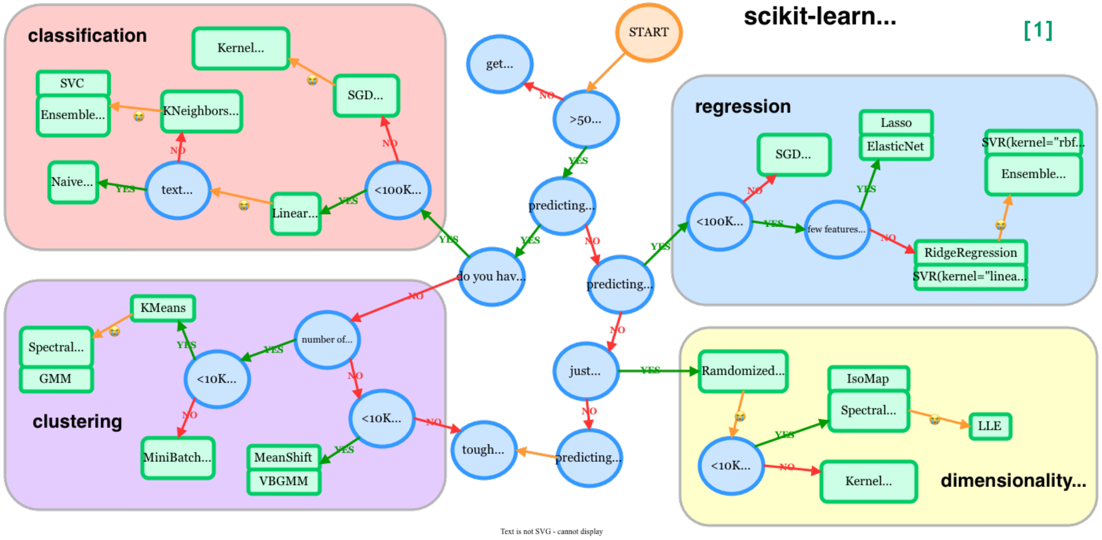
Output gate – defines which part of the cell state will be used for the calculation of the output of the whole cell

Forget gate – decides which information is removed from the cell state

https://commons.wikimedia.org/wiki/File:LSTM_cell.svg [1]

Algorithms
How to choose the right algorithm?

- ”Ground Rules”, e.g. # of observations
- Logic
- Explainability & Transparency
- Experience
- Intuition



https://scikit-learn.org/stable/machine_learning_map.html [1]

https://images.unsplash.com/photo-1518186233392-c232efbf2373?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=3174&q=80

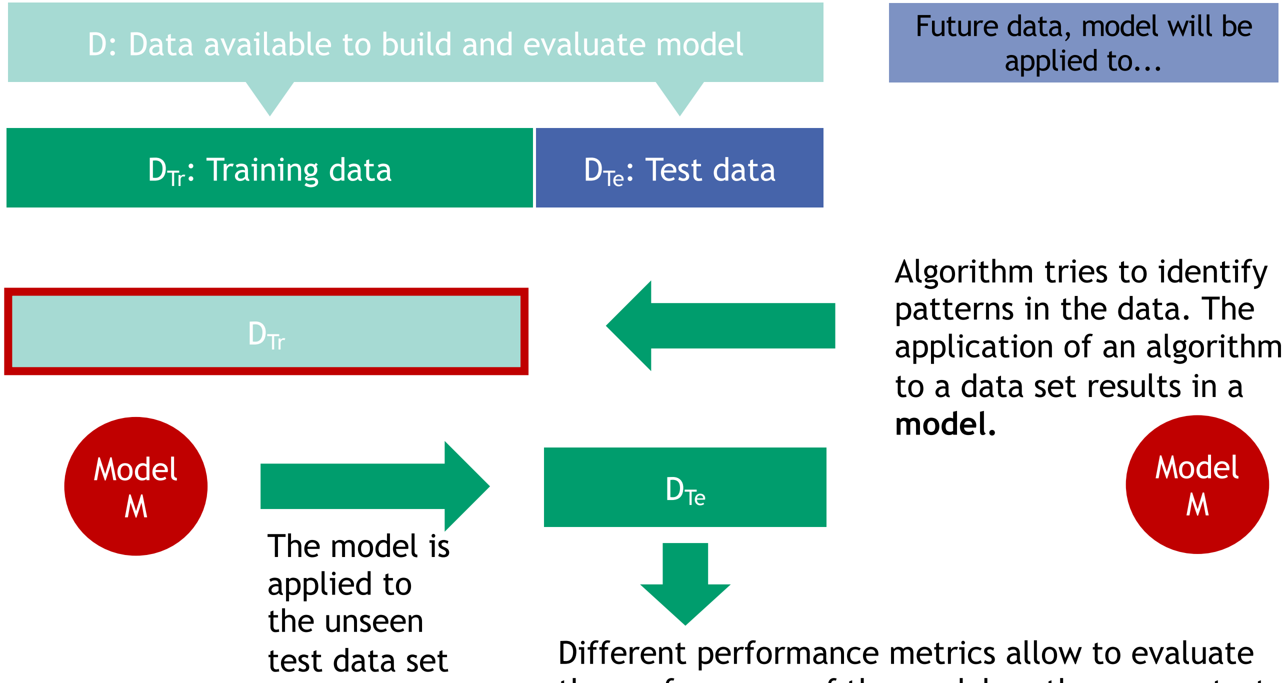
1 Algorithms

- Basic functionality
- Selected parameters

2 Performance metrics

3 Evaluation options

Performance metrics
what for?



D: Data available to build and evaluate model

Future data, model will be applied to…

D_{Tr}: Training data

D_{Te}: Test data

D_{Tr}

Algorithm tries to identify patterns in the data. The application of an algorithm to a data set results in a **model.**

Model M

The model is applied to the unseen test data set

D_{Te}

Model M

Different performance metrics allow to evaluate the performance of the model on the unseen test set

Performance metrics
Classification: Selection

		Prediction	
		Positive	*Negative*
Ground truth	*Positive*	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$	**Accuracy** describes the relation between correct classifications and the total number of input samples and gives a first insight on an ML application's overall performance.
Precision $= \frac{TP}{FP+TP}$	**Precision** is the proportion of correctly predicted positive cases regarding all positive predicted cases.
Recall $= \frac{TP}{FN+TP}$	**Sensitivity/ Recall** is the proportion of corrected predicted positive cases regarding all positive cases.
F1-Score $= 2\ x\ \frac{Precision\ x\ Recall}{Precison+Recall}$	**F1-score** is the harmonic mean between precision and recall.

Performance metrics
Classification: The F-metric

Fß -Score : = $\frac{(1+\beta^2)\,TP}{(1+\beta^2)\,TP+\,\beta^2 * FN+FP}$

$$= (1+\beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall}$$

What is the performance of the model, weighting precision and recall by ß?

How good is the performance emphasizing…

F1 -Score : = $2 * \frac{precision * recall}{precision + recall}$

What is the harmonic average of precision and recall?

…recall and precision equally?

F0.5 -Score : = $1.25 * \frac{precision * recall}{0.25 * precision + recall}$

What is the ratio when weighting precision twice?

…precision over recall?

F2 -Score : = $5 * \frac{precision * recall}{4 * precision + recall}$

What is the F-Score when weighting recall twice?

…recall over precision?

Performance metrics
Regression: Selection

$$\text{MAE} = \frac{1}{N} * \sum_1^N |y_i - \hat{y}_\imath|$$

Mean absolute error defines the average distance between single data points and the arithmetic mean.

$$\text{MSE} = \frac{1}{N} * \sum_1^N (y_i - \hat{y}_\imath)^2$$

Mean squared error squares the difference between actual values and predicted values, making MSE helpful when outliers occur as the difference is more penalized.

$$\text{RMSE} = \sqrt{\frac{1}{N} * \sum_1^N (y_i - \hat{y}_\imath)^2}$$

Root mean squared error is the standard deviation of prediction errors in the model. It states the concentration of data points around the regression line.

$$\mathbf{R^2} = \frac{\sum_1^N (\bar{y}_i - \hat{y}_\imath)^2}{\sum_1^N (y_i - \hat{y}_\imath)^2} = \frac{\textit{explained var}}{\textit{total var}}$$

$\mathbf{R^2}$ compares the regression model with a random baseline model and outlines how much variation of a dependent variable is explained by the independent variables.

$\hat{y}$ = arithmetic mean of the estimation, y = (real) single data points, $\bar{y}$ = predicted value for a single data point

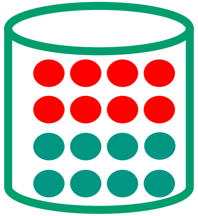
Performance metrics
How to choose the right performance metric

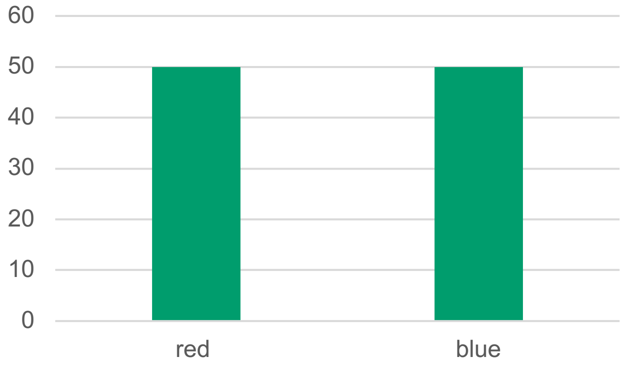
Model initiation	What is the underlying **business problem** and which metric precisely addresses this problem?
	Consider the **distribution of data** when choosing performance metric(s)
Examples	**Customer satisfaction prediction:** The costs for winning new customers are at least five times higher than turning unsatisfied customers to a satisfied state (Hart, Heskett and Sasser, 1990)
	Rare disease detection: What do you think?

Performance metrics
Classification: Addressing class imbalance

Simple use case: We want to predict if a ball is red with a supervised classification model

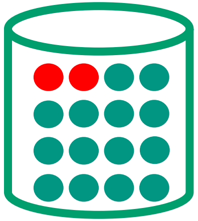
red	Not red
50	50

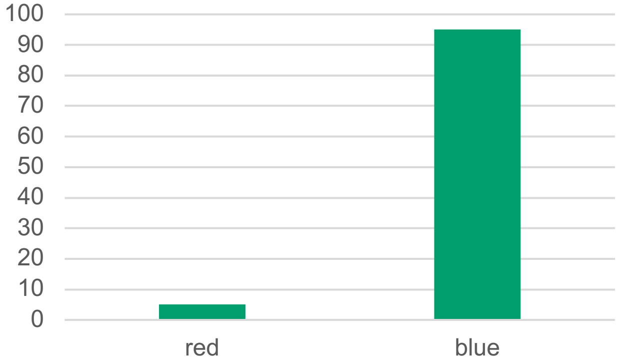




No class imbalance

red	Not red
5	95





Class imbalance

Performance metrics
Classification: Addressing class imbalance

- In many use cases detection of the minority class is the crucial task
    - Cancer detection
    - Failure detection
    - Credit card fraud detection

- Broadly speaking, algorithms tend to “ignore” the minority class since they optimize an overall measure of goodness like “Prediction Accuracy”.

Performance metrics
Classification: Addressing class imbalance

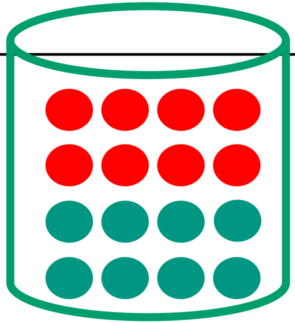
No class imbalance

		Prediction		
		Red (=p)	Not Red (=n)	Marginal frequency ground truth
Ground truth	Positive	25	25	50
	Negative	25	25	50
	Marginal frequency ML classifier	50	50	

Prediction Accuracy: $\frac{25+25}{(24+25+25+25)} = 50\%$

Recall: $\frac{25}{(25+25)} = 50\%$

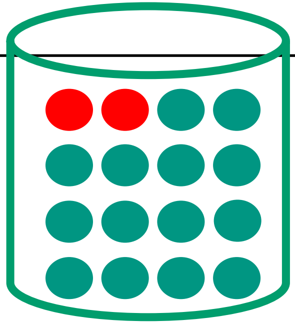
Specificity: $\frac{25}{(50)} = 50\%$



Performance metrics
Classification: Addressing class imbalance

Class imbalance

Prediction				
Ground truth		Red (=p)	Not Red (=n)	Marginal frequency ground truth
	Positive	5	5	10
	Negative	5	85	90
	Marginal frequency ML classifier	10	90	



Prediction Accuracy: $\frac{5+85}{(5+45+5+45)} = 90\%$ ⚡ !

Sensitivity/Recall: $\frac{5}{(5+5)} = 50\%$

Specificity: $\frac{85}{(5+85)} = 94.444\%$ ⚡ !

→The Model did not learn the minority class as good

https://images.unsplash.com/photo-1454165804606-c3d57bc86b40?ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=3270&q=80

1 Algorithms

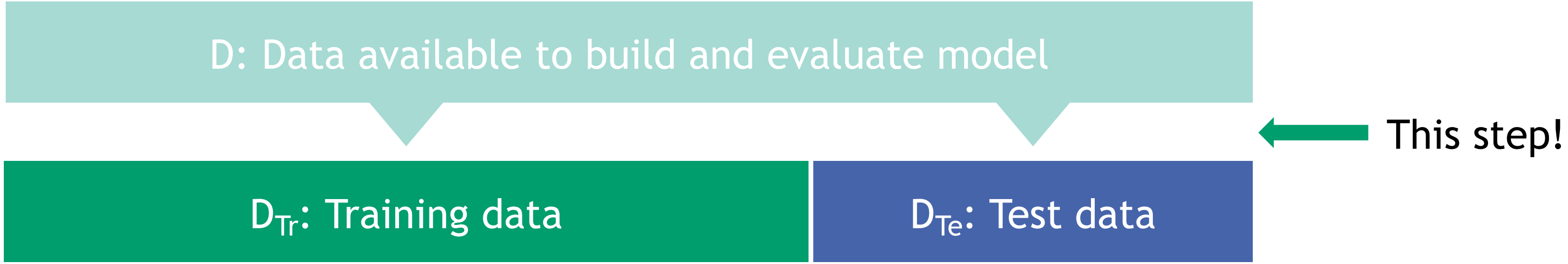
- Basic functionality
- Selected parameters

2 Performance metrics

3 Evaluation options

Evaluation options
Evaluation methods

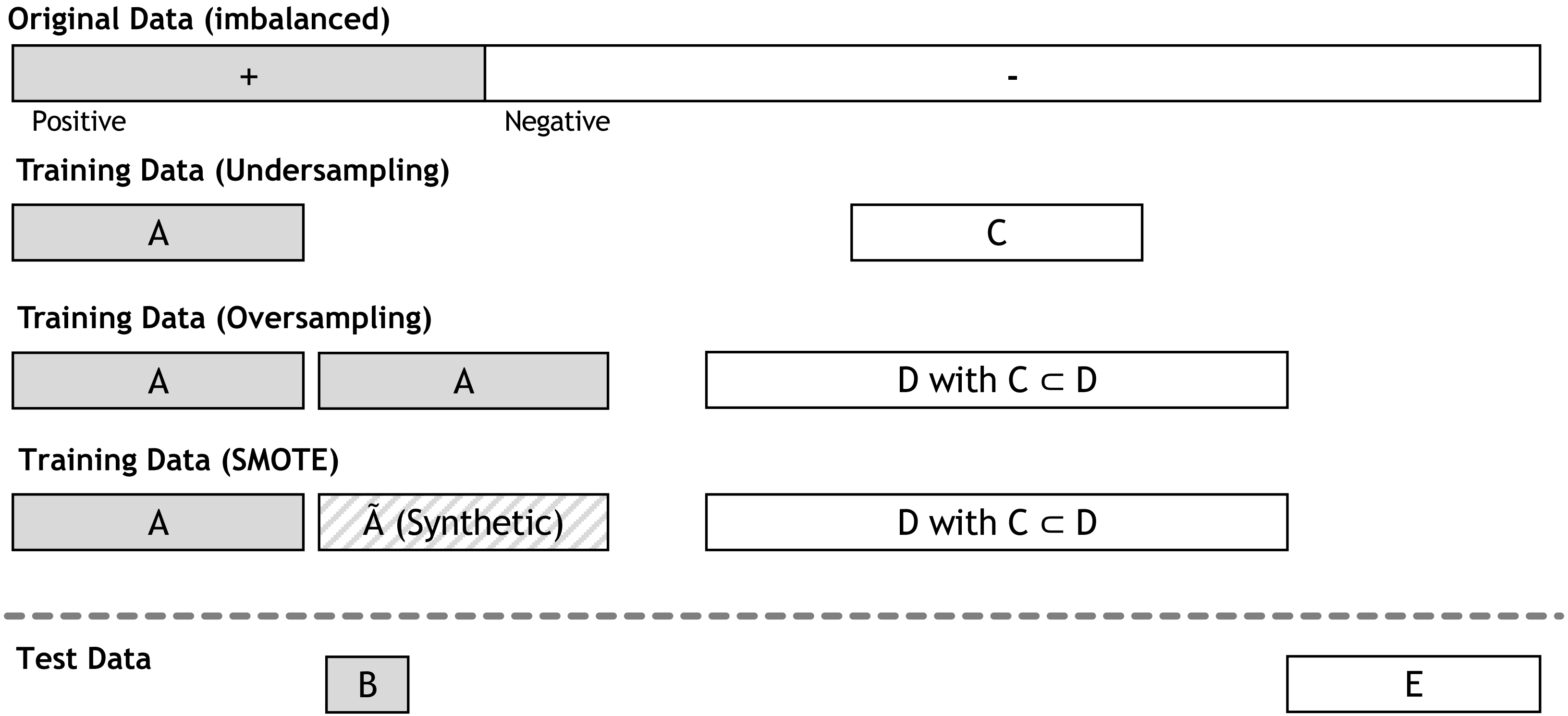
- Different possibilities on how to split and evaluate a model



- Without parameter optimization
  - Percentage Split (with or without sampling)
  - Cross Validation (with or without sampling)
- With parameter optimization
  - Global hold-out set
  - Nested Cross Validation (with or without sampling)

Evaluation options

Percentage Split & Sampling – exemplary for binary classification



Original Data (imbalanced)

+

-

Positive

Negative

Training Data (Undersampling)

A

C

Training Data (Oversampling)

A

A

D with C ⊂ D

Training Data (SMOTE)

A

Ã (Synthetic)

D with C ⊂ D

Test Data

B

E

Evaluation options

Percentage Split & Sampling: Example from Research

- Whole-genome sequencing (WGS) is a medical technique to determine a DNA sequence completely and at a single time. WGS can be applied to the human genome and gives insights about its composition. [1]

[2]

- Machine Learning is used to find disease-associated genetic variations. To train such algorithms, the dataset is imbalanced: Positive examples of a disease accumulate to several hundred instances, whereas negative examples go into the millions.

[3]

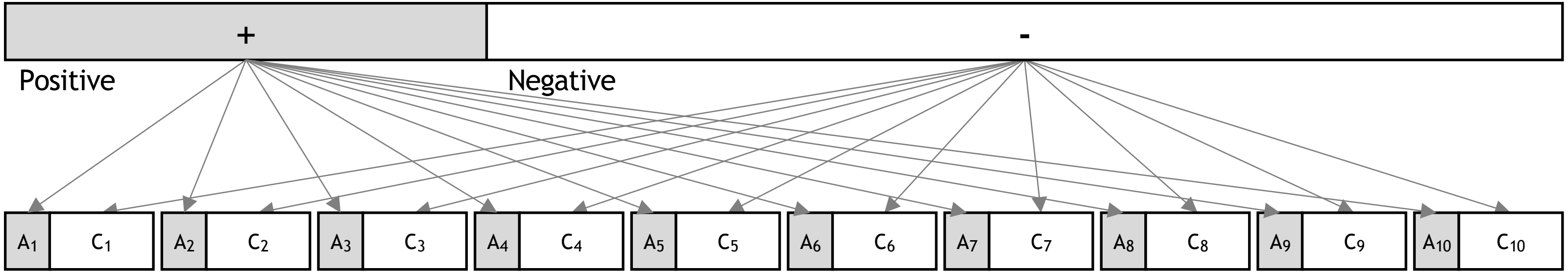
- By plotting a bar chart of the dataset one would quickly recognize the imbalance.

Solution → Oversample the positive (and undersample the negative examples) to get a balanced set of training data.

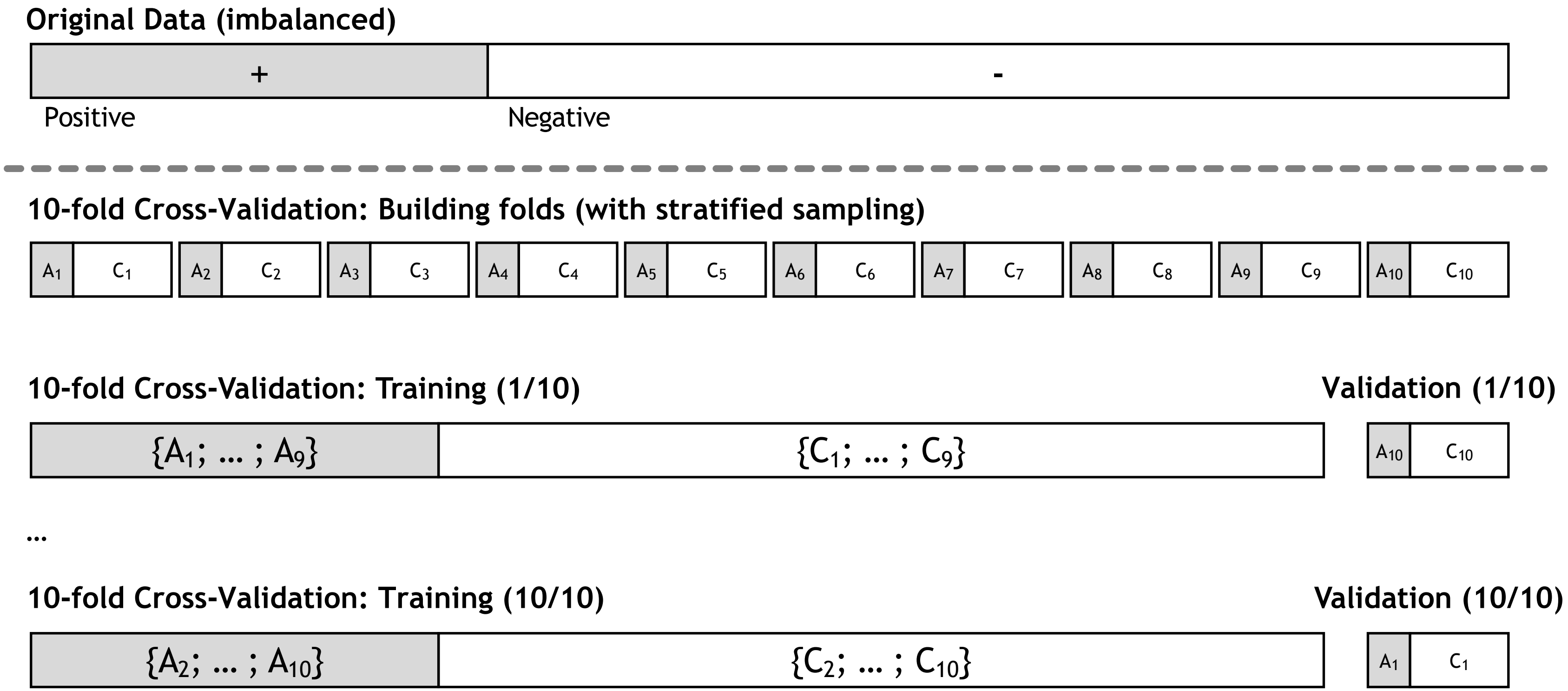
Schubach et al. (2017), Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants [1]
https://www.flaticon.es/autores/smashicons [2]
https://www.flaticon.com/de/kostenlose-icons/dna [3]

Evaluation options
Cross Validation

Original Data (imbalanced)



Evaluation options
Cross Validation



Original Data (imbalanced)

+ | -

Positive | Negative

10-fold Cross-Validation: Building folds (with stratified sampling)

A_1 C_1 | A_2 C_2 | A_3 C_3 | A_4 C_4 | A_5 C_5 | A_6 C_6 | A_7 C_7 | A_8 C_8 | A_9 C_9 | A_{10} C_{10}

10-fold Cross-Validation: Training (1/10) **Validation (1/10)**

$\{A_1; \ldots ; A_9\}$ $\{C_1; \ldots ; C_9\}$ | A_{10} C_{10}

…

10-fold Cross-Validation: Training (10/10) **Validation (10/10)**

$\{A_2; \ldots ; A_{10}\}$ $\{C_2; \ldots ; C_{10}\}$ | A_1 C_1

Evaluation options
“Too good to be true” performance - Data Leakage

Definition Data Leakage

[1]

The use of information in the model training process which would not be expected to be available at prediction time, causing the predictive scores (metrics) to overestimate the model's utility when run in a production environment.

- **Feature leakage:** a proxy for the label or the label itself
- **Time leakage:** randomly splitting a time-series dataset
- **Group leakage:** not including a grouping split column

Employee ID	Age	MonthlySalary	Year	Job	YearlySalary
1	30	5000	2023	Programmer	60000
2	25	4500	2023	Data Analyst	54000
3	35	5500	2023	Manager	66000
4	28	4800	2023	Programmer	57600
5	40	6000	2023	Data Analyst	72000
1	31	5100	2024	Programmer	61200
2	26	4600	2024	Data Analyst	55200
					target

Shachar Kaufman; Saharon Rosset; Claudia Perlich (January 2011). Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. Vol. 6. pp. 556–563 [1]

Evaluation options
Be careful when preprocessing data - Data Leakage

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing

X, y = fetch_california_housing()
scaler = StandardScaler()

scaler.fit(X)
X_scaled = scaler.transform(X)

X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y)
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing

X, y = fetch_california_housing()
scaler = StandardScaler()

X_train, X_test, y_train, y_test =
train_test_split(X, y)

scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Example [1]
If you normalize or standardize your entire dataset, then estimate the performance of your model using cross validation …

Problem:
knowledge of the full distribution of data

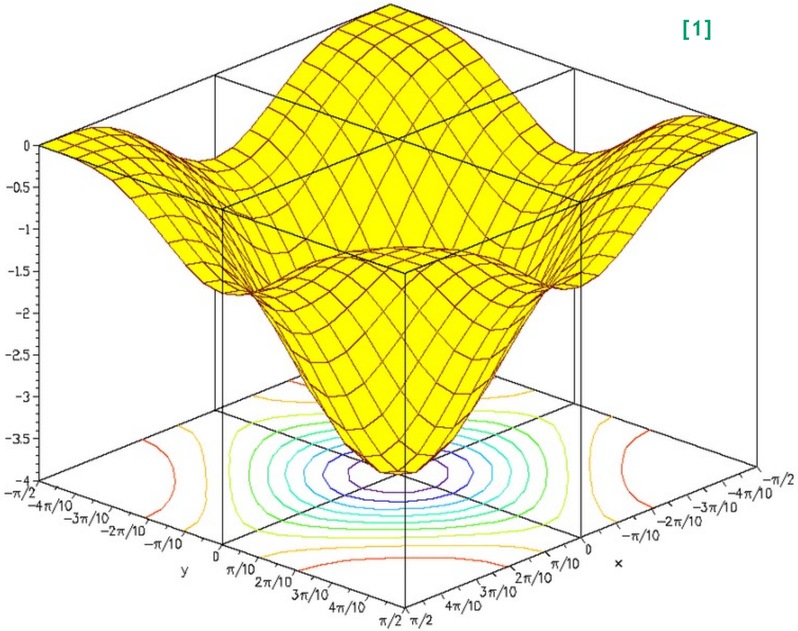
Solution:
calculate the parameters for rescaling data within each fold of the cross validation

Leakage Happens [2]
Even if you don't explicitly understand and game the leakage, your model will do it for you.

https://machinelearningmastery.com/data-leakage-machine-learning/ [1]
Schutt, Rachel, and Cathy O'Neil. Doing data science: Straight talk from the frontline. O'Reilly, 2014. [2]

Evaluation options
Optimization

- How do we know the parameters we chose are “the best”?



[1]

https://commons.wikimedia.org/wiki/File:Crosshatch_3D_plot_with_projected_level_curves.PNG [1]

Evaluation options
Optimization: Grid Search

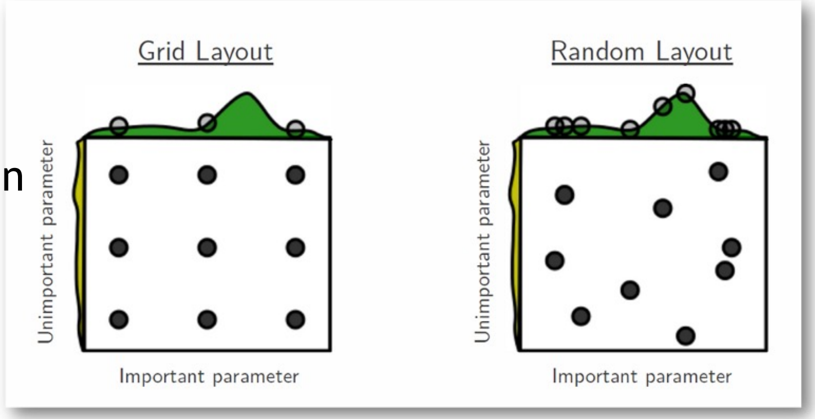
- Grid Search – is a simple way to find good parameters by brute forcing discrete values. [1]
  - Define possible values or a range with an increment (="search space") – for every parameter. Every combination of every possible value will be tested in a grid search.
  - Select a performance metric – to evaluate and rank the result of a parameter combination.
  - Choose test data - to apply the performance metric on after training and validating the model. Test data can be a global holdout set or a nested cross validation.

- All defined combinations are tested during a grid search. The result is the value for every parameter which was part of the combination with the highest performance.

Bergstra and Bengio (2012), Random search for hyper-parameter optimization [1]

Evaluation options
Optimization: Random Search

- Every possible combination of possible pre-defined parameter values is tried when performing a grid search. Since brute forcing is costly, one limits the number of combination by e.g. increasing the increment for ranges. This leads to roughly searching the defined search space equally instead of focusing on promising areas.



[2]

- Random Search – is a way to find parameters [1] by testing predefined samples of possible parameters combinations:
  - Set the distribution – of the parameters additionally to the defined possible values or range.
  - Set a budget – on how often a combination of parameters is sampled. The sampling only includes possible values and takes the distribution into account.

Bergstra and Bengio (2012), Random search for hyper-parameter optimization [1]
Tahmassebi, Amirhessam & Ehtemami, Anahid & Mohebali, Behshad & Gandomi, Amir & Pinker, Katja & Meyer-Base, Anke. (2019). Big data analytics in medical imaging using deep learning. 13. 10.1117/12.2516014. [2]

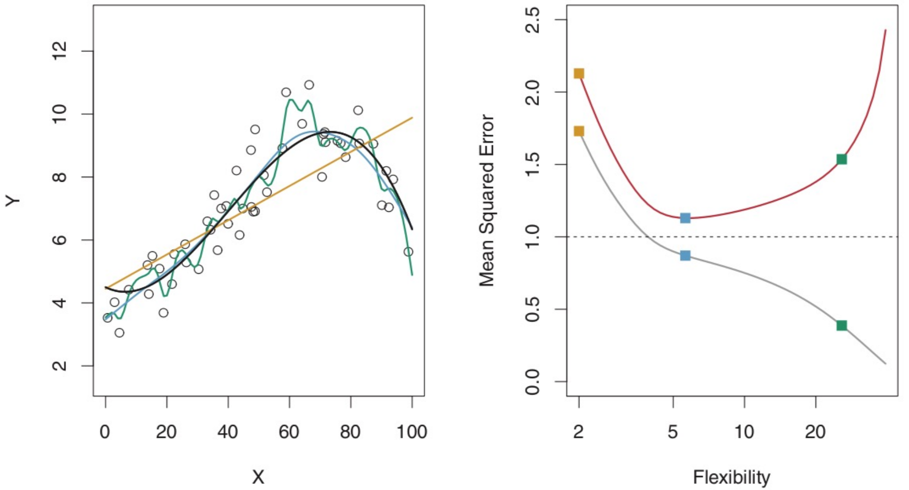
Evaluation options
Optimization

- Although grid search as well as random search supports the user to select good parameters, it is almost “impossible” to find the perfect parameters
  - Performance metrics – usually favor different parameter combinations. One can find the perfect parameter combination for recall by forcing the algorithm to select every instance, however, the precision will be very bad.
  - Limiting the search space in advance – is necessary for computational reasons but has the risk of excluding the best areas.
  - Searching within the search space – cannot be performed perfectly. There is no option to find all local optima within the search space. One can only use (randomized) heuristics which try to find good parameter combinations.

Evaluation options

Overfitting

- A model needs to be complex enough to explain existing relation, but too much complexity leads to overfitting.
- Yellow: underfitted model
- Green: overfitted model
- Blue: balance between variance and bias



[1]

FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

Gareth et al. (2013), An introduction to statistical learning [1]

Evaluation options
Global Holdout Set

- Detecting overfitting – is only possible when testing the performance of the model on test data. Test data is a subset of original data which the model has never seen, neither in the training nor in the validate phase. [1]
- Global holdout set – is a set of data which is extracted from the original data before the model training. Instances, which are part of the global holdout set, must not be used in training or validation.
  - Testing – the model is performed by applying the trained model to the global holdout set. The value of the performance metric determines the test result for the model.

Gareth et al. (2013), An introduction to statistical learning [1]

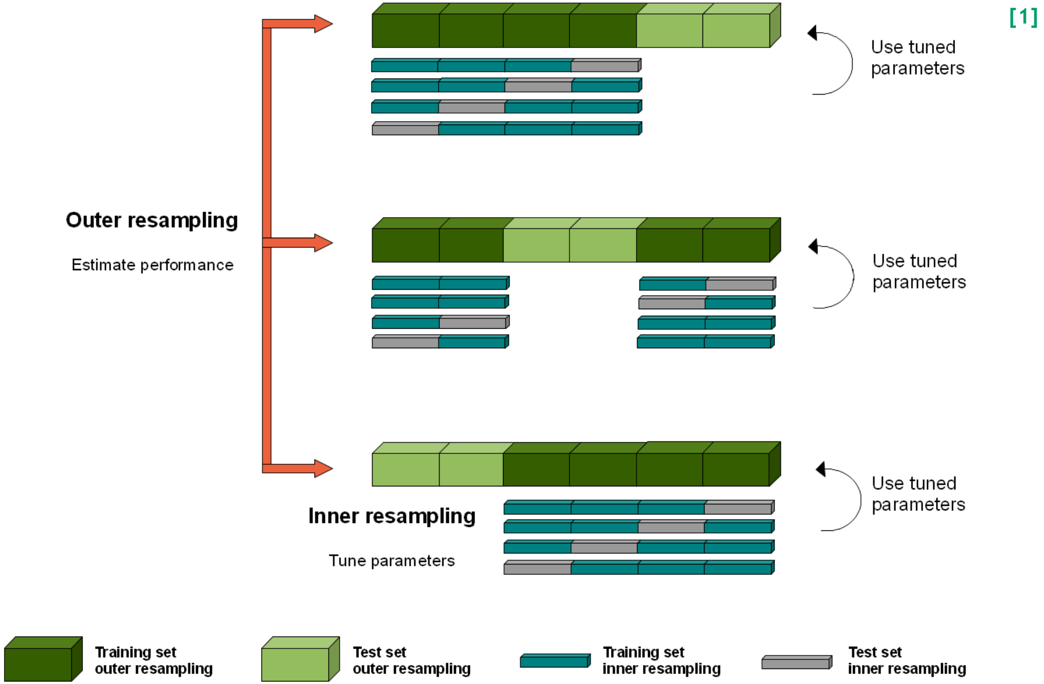
Evaluation options
Nested Cross Validation

- Nested cross validation – is another way to test the performance of a model by conducting multiple cross validations. [1]
  - Inner cross validation – corresponds to the cross validation already presented in this lecture: Splitting data into folds and use them multiple times as training and validation set. Every inner cross validation selects its own (best) parameter combination.
  - Outer cross validation – repeats inner cross validations several times. Before fitting the model by inner cross validation, it extracts another fold of the data as global holdout set and uses it to test the resulting model.

- Overfitting – is detected by comparing
  - different, optimal parameters of every inner cross validation.
  - performance results within the inner cross validation.
  - The higher the variance, the more likely the model is to overfit.

Cawley and Talbot (2010), On over-fitting in model selection and subsequent selection bias in performance evaluation ; Weina (2018), Nested cross validation explained [1]

Evaluation options
Nested Cross Validation visualized



Mehta, Ritesh & Pramov, Aleksandar & Verma, Shashank. (2024). Machine Learning for ALSFRS-R Score Prediction: Making Sense of the Sensor Data. 10.48550/arXiv.2407.08003. [1]

Summary

Multiple concepts for supervised machine learning

- The performance of each algorithm is determined by the structure of the data—and the algorithms capabilities to capture patterns in that very data.

Algorithm parameters

- Algorithms can be optimized by tuning their required parameters.

Performance metric

- The selection of the performance metric sets the aim for the model.

AI lifecycle

- It is important to understand the interplay of problem statement, data (distribution), algorithm and performance metric.