Artificial Intelligence

Algorithms and Applications with Python

Chapter 6

Dr. Dominik Jung
dominik.jung42@gmail.com

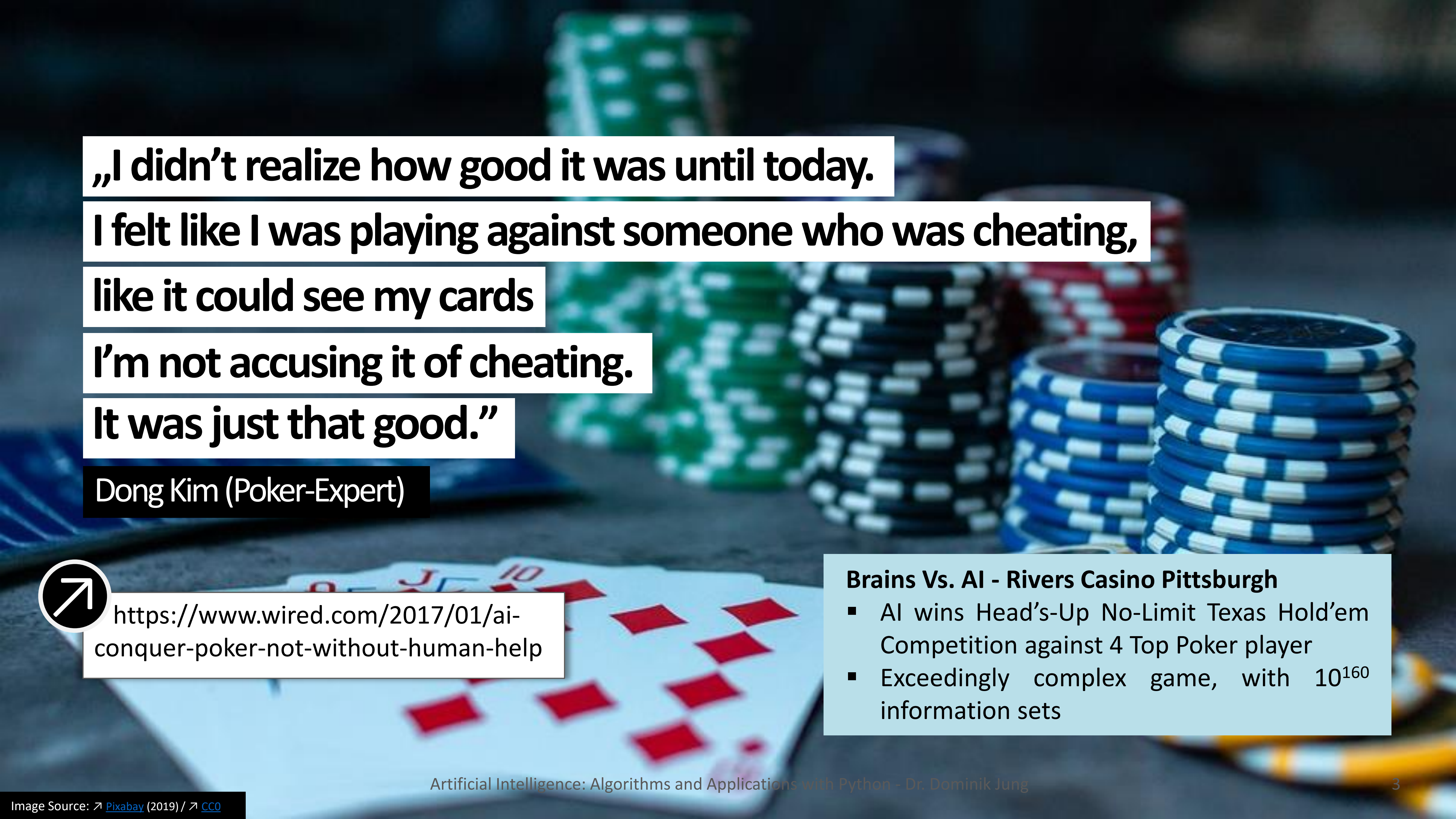
Outline

Image source: ↗ Pixabay (2019) / ↗ CC0

▶ **What we will learn:**
- General concepts of AI modelling and what types of problems match which models
- Get an intuition for which model approach fits bets for a particular learning problem
- Know general problems of machine learning and how to optimize learning models

▶ **Duration:**
- 270 min + 90 (Lectorial)

▶ **Relevant for Exam:**
- 6.1 – 6.4



„I didn't realize how good it was until today.
I felt like I was playing against someone who was cheating,
like it could see my cards
I'm not accusing it of cheating.
It was just that good."

Dong Kim (Poker-Expert)

https://www.wired.com/2017/01/ai-conquer-poker-not-without-human-help

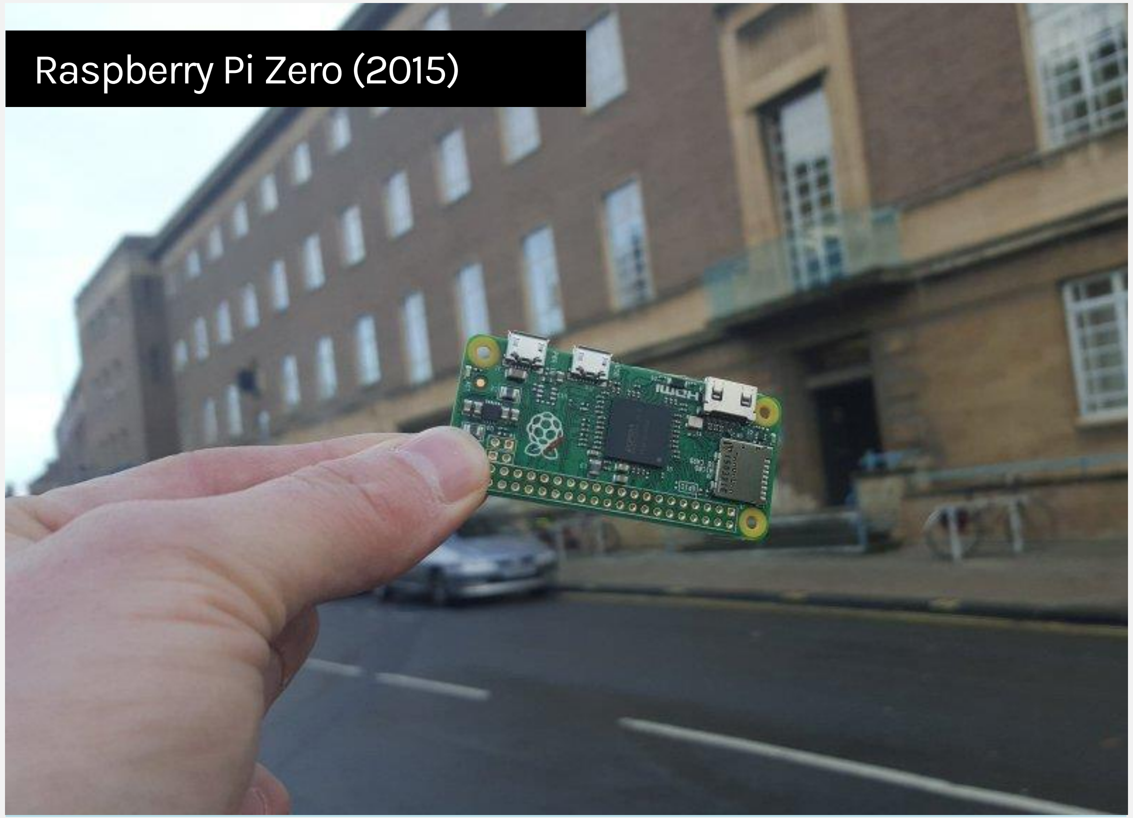
Brains Vs. AI - Rivers Casino Pittsburgh

- AI wins Head's-Up No-Limit Texas Hold'em Competition against 4 Top Poker player
- Exceedingly complex game, with 10^{160} information sets

Image Source: ↗ Pixabay (2019) / ↗ CC0

6.1 How was This Possible?

Elliott 405 (1958)



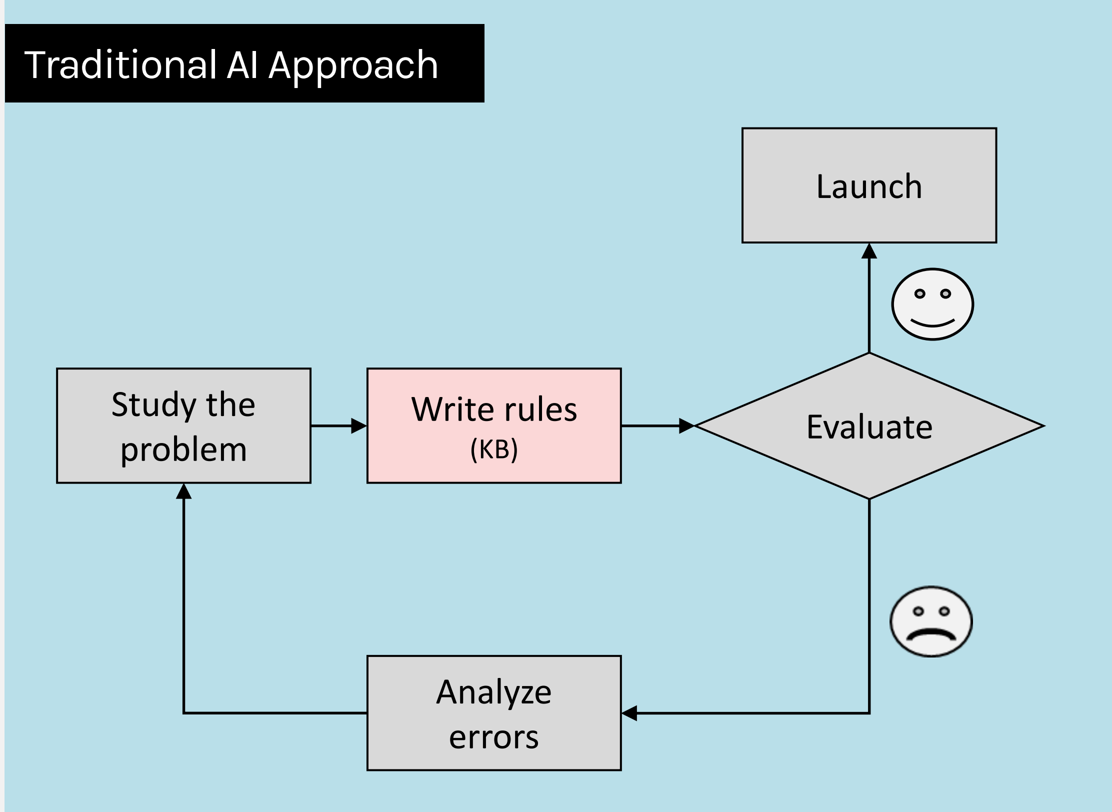
Raspberry Pi Zero (2015)

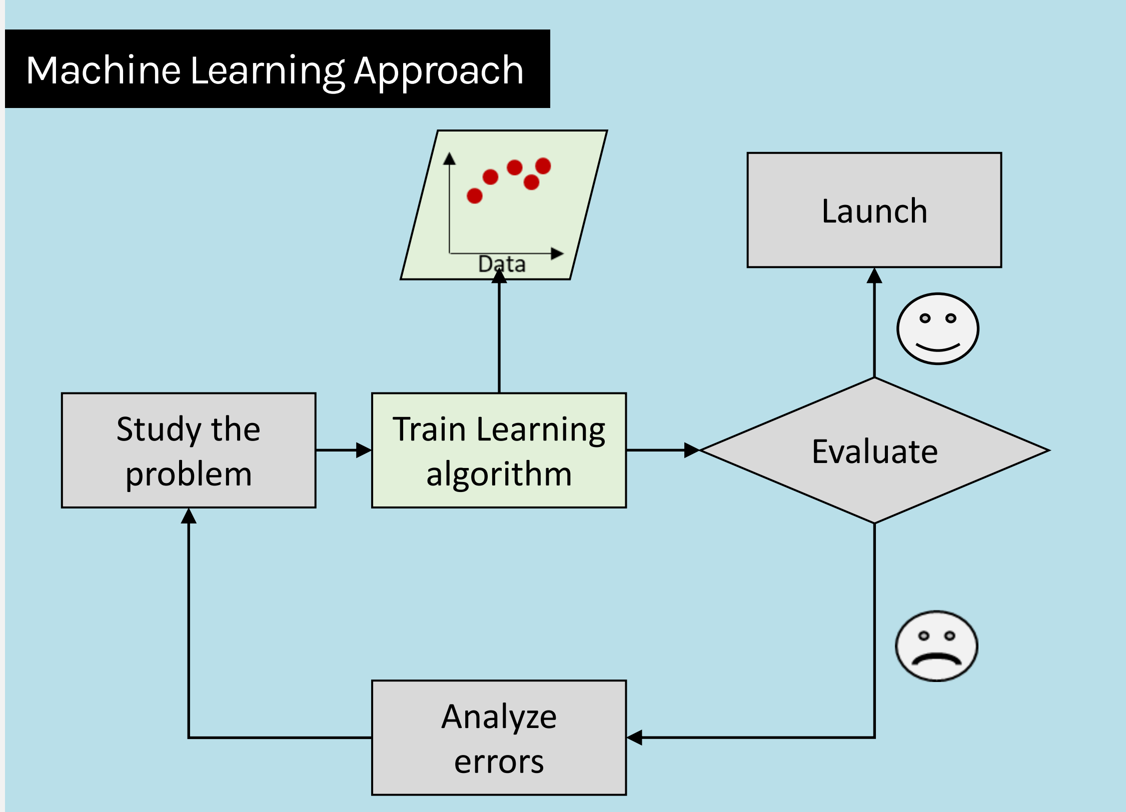
- Greater availability of data
- Increase of computational power

Trend away from rule-based and manually specified models to probabilistic data-driven modes

Adapted from Géron, A. (2017) | Image source: The Norwich Computer ↗ [Norfolk Record Office online catalogue](#) (1957); Unknown source, Raspberry Pi Zero (2015)

6.1 How is This Possible?



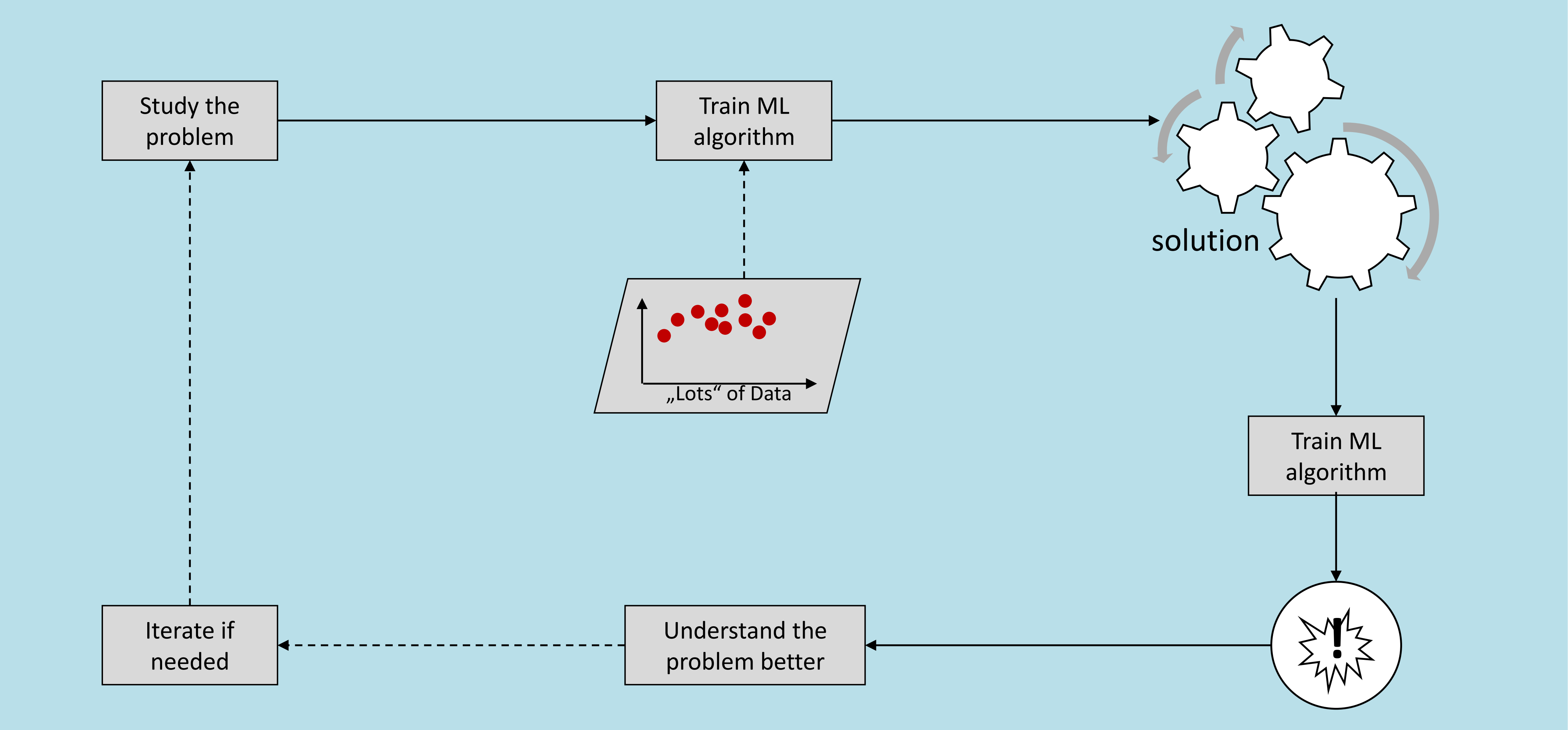


- Greater availability of data
- Increase of computational power

Trend away from rule-based and manually specified models to probabilistic data-driven modes

Adapted from Géron, A. (2017) | Image source: Géron, A. (2017)

6.1 Many New Applications: Use Machine Learning for Data Mining



Adapted from Géron, A. (2017) | Image source: Géron, A. (2017)

„Learning is any process by which a system improves performance from experience“

Herbert Simon

Image source : ↗ Chess Programming Wiki (2019)

6.1 Formalization of Machine Learning

Input	Model	Output
X	$f: X \rightarrow Y$	$H = \{h \mid h: X \rightarrow Y\}$
Set of possible instances	*Unknown target function*	*Set of function hpyotheses*

Goal: Find hypothesis $h \in H$ that best approximates target function f

so that $f(X) = \arg\max_{y} p(Y \mid X)$

X

Attention! Do not get confused by the different meanings of *hypotheses* in science, statistics and machine learning!

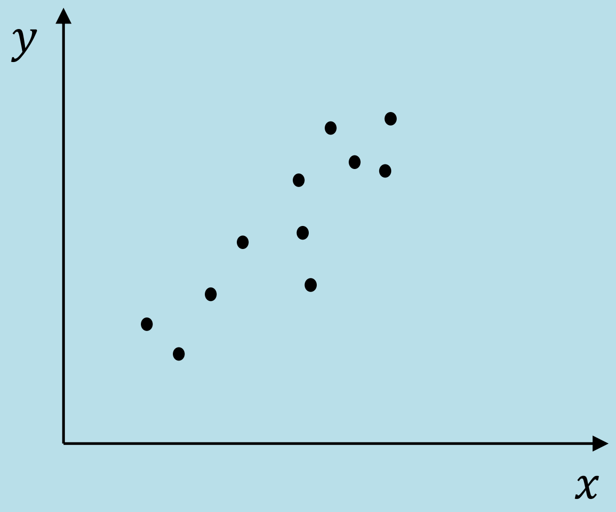
D

Machine Learning

A computer program is said to learn from **experience 'E'**, with respect to some class of **tasks 'T'** and **performance measure 'P'** if its performance at tasks in 'T' as measured by 'P' improves with experience 'E'. (Mitchell, 1997)

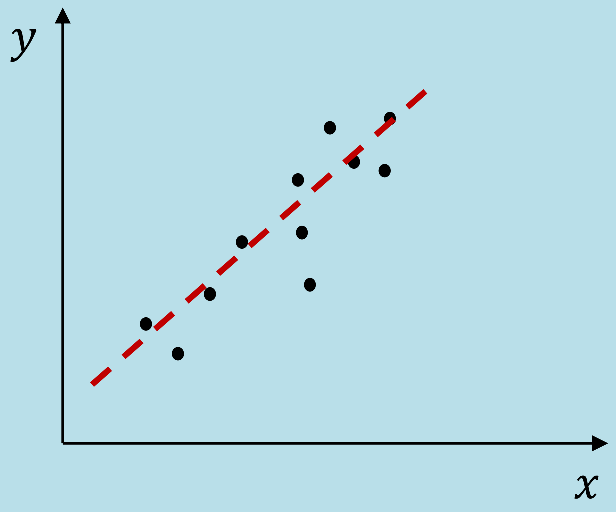
6.1 Geometric Interpretation of Machine Learning

1



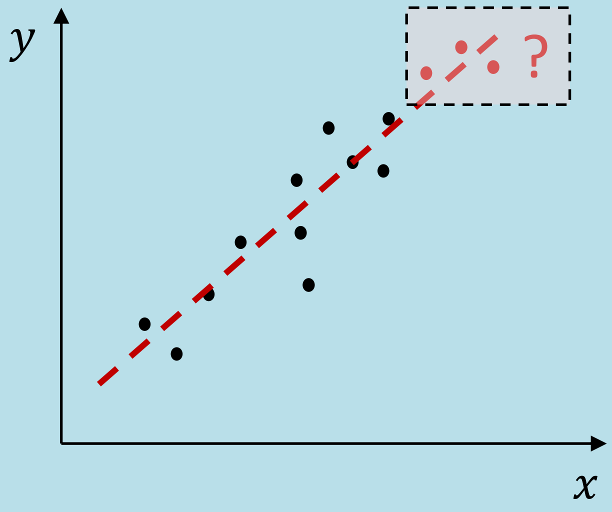
- You have given data X and want to build a model describes the relation between X and Y. Usually, you want to predict the target variable.

2



- You can model this problems as a regression problem. Hence you decide to use a linear regression model to fit your data

3



- You can use your model to predict future values of Y

6.1 Recap: Pocker AI

- **Task**: Play Poker
- **Performance Measure**: Percentage of games/money won?
- **Experience**: Previous games

D

Machine Learning
A computer program is said to learn from **experience 'E'**, with respect to some class of **tasks 'T'** and **performance measure 'P'** if its performance at tasks in 'T' as measured by 'P' improves with experience 'E'. (Mitchell, 1997)

6.1 Taxonomy of Machine Learning Types (Simplification)

D

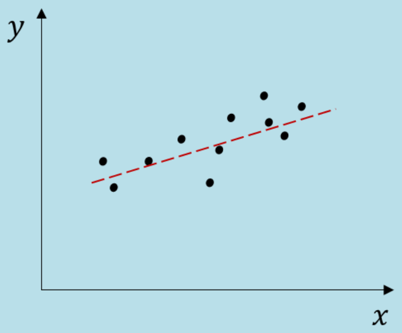
This taxonomy is just a very simplified representation of the topic machine learning. We will expand this taxonomy step by step in this lecture.

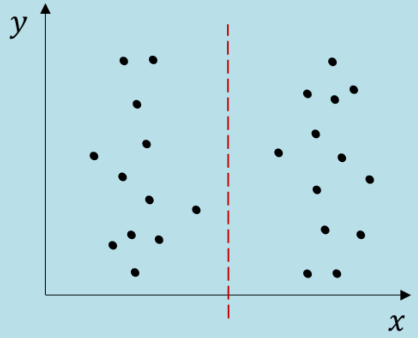
Supervised Learning

An algorithm uses human-prepared training data to learn the relationship between given inputs and a given outcome.

Supervised

Regression | Classification





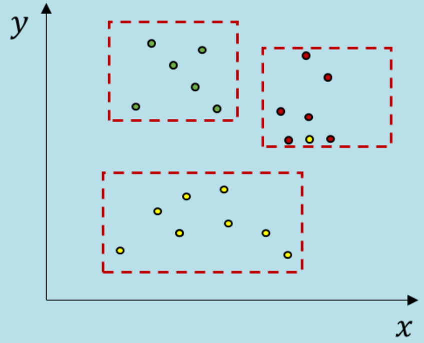
- Predict or explain differences between **continuous** variables
- Predict or explain differences between **categorical** variables

Unsupervised Learning

An algorithm examines input data without knowing about attributes and possible results.

Unsupervised

Clustering

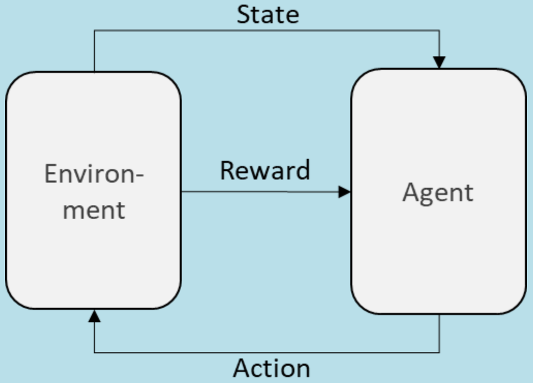


- Cluster observations into (distinct/different) groups

Reinforcement Learning

An algorithm learns to perform a task by trying to maximize the rewards it receives for its actions.

Reinforcement



- An AI system learns how to behave in a specific environment

Adapted from Géron, A. (2017)

6.1 Other Machine Learning Taxonomy Criterions

- **Batch and online learning**: whether or not the system can learn incrementally from a stream of incoming data

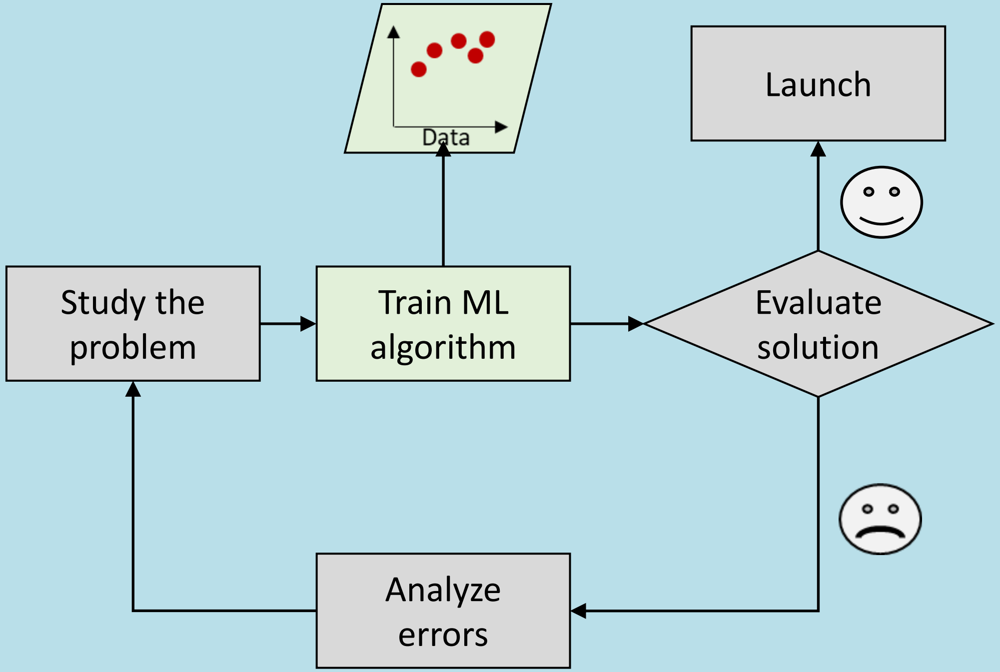
- **Instance-based vs. model-based learning**: whether the system generalizes to new cases by comparing them to the learned examples or to build a model to make predictions

- **Discriminantive vs. Generative**: Model the decision boundary between the classes or estimate the actual distribution of each class

Adapted from Géron, A. (2017)

6.1 How to Improve Learning Performance

Machine Learning Approach

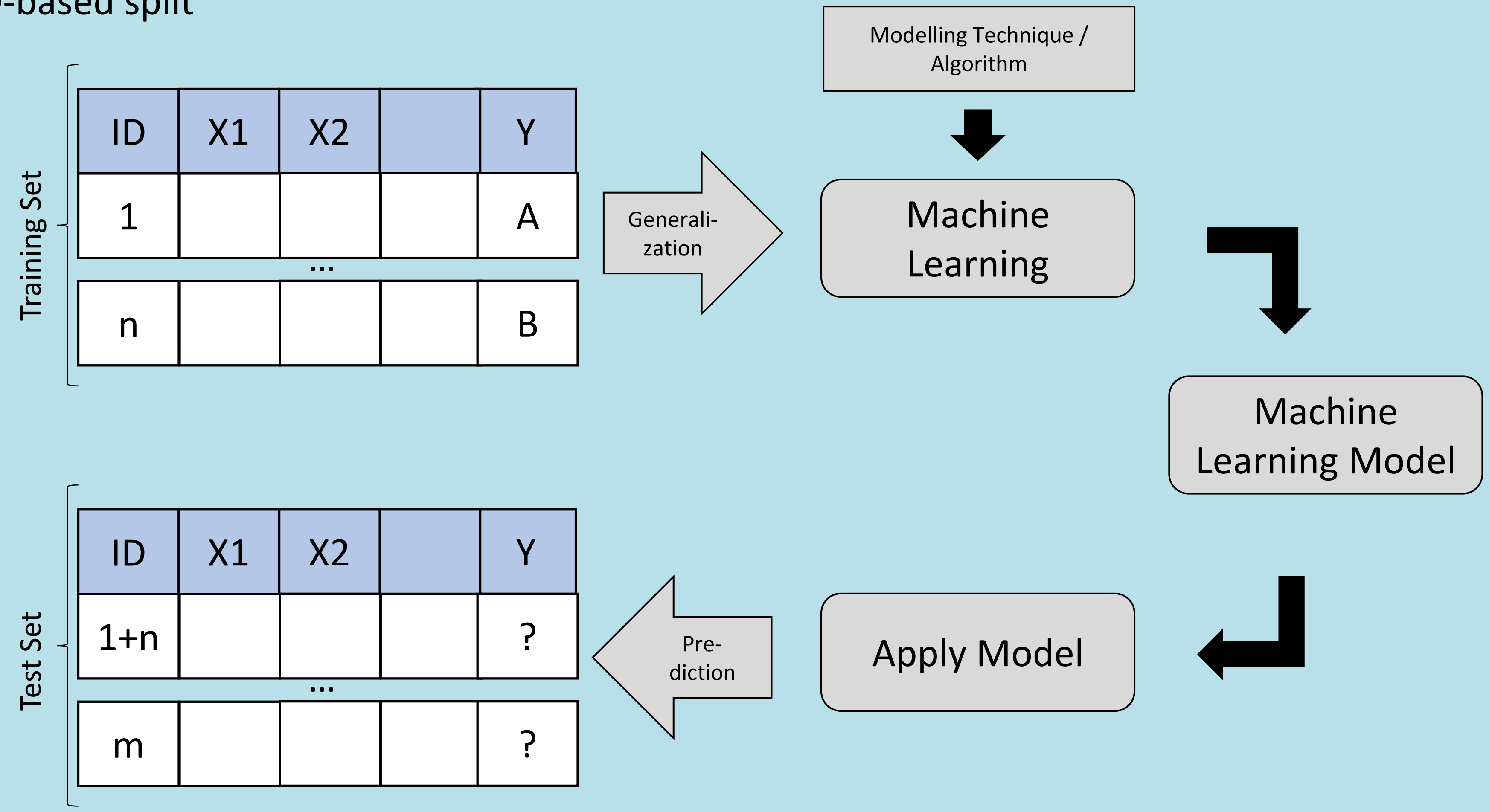


Adapted from Géron, A. (2017)

- How can we know how good our agent is learning the current problem?
- What is the correct way to compute our performance measure?
- **Naive appproach**: Learning a machine learning model and testing it on the same data
- **Problem**: We measure how good the model is to predict the classes of the test data and not how good it generalizes (the model learnt to solve such problems)

6.1 Split Data into Test and Train to Measure Performance

Simple ID-based split



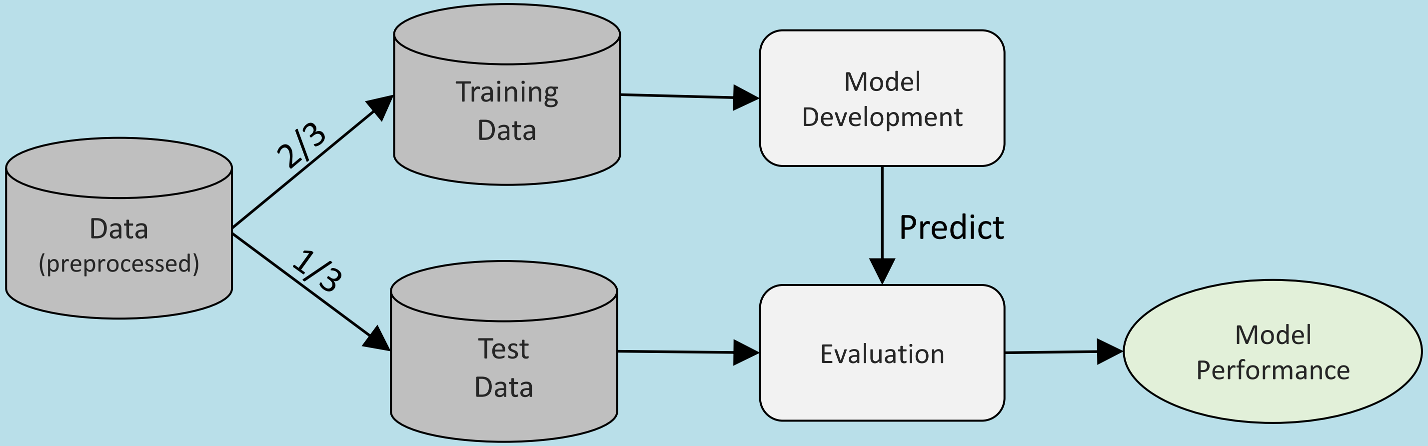
Adapted from Scitkit-learn (www.scikit-learn.org); Géron, A. (2017)

6.1 Holdout-Method

Procedure

- Split your labelled data into train and test set, Build your model based on train set, and measure the performance based on the test set
- In problems where we have a sparse dataset we may not be able to afford the “luxury” of setting aside a portion of the dataset for testing
- Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an “unfortunate” split

Visualization

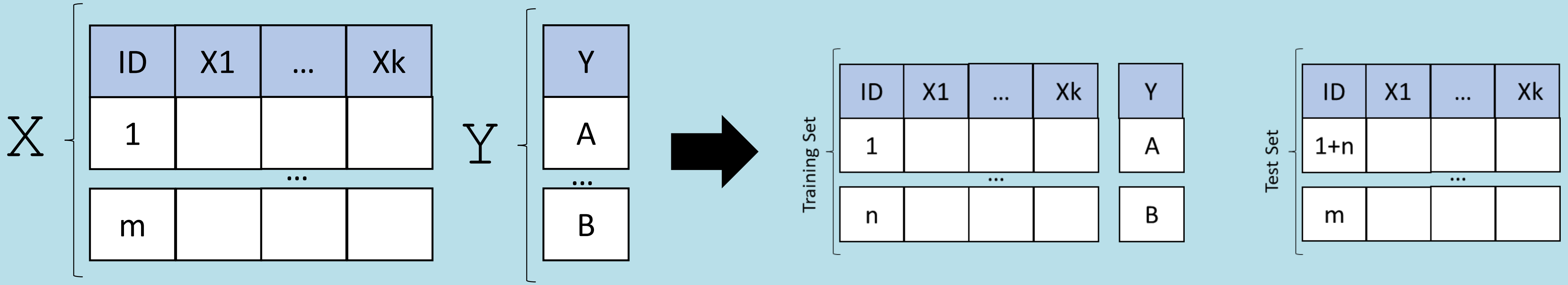


Adapted from Kim, J.-H. (2009); Géron, A. (2017)

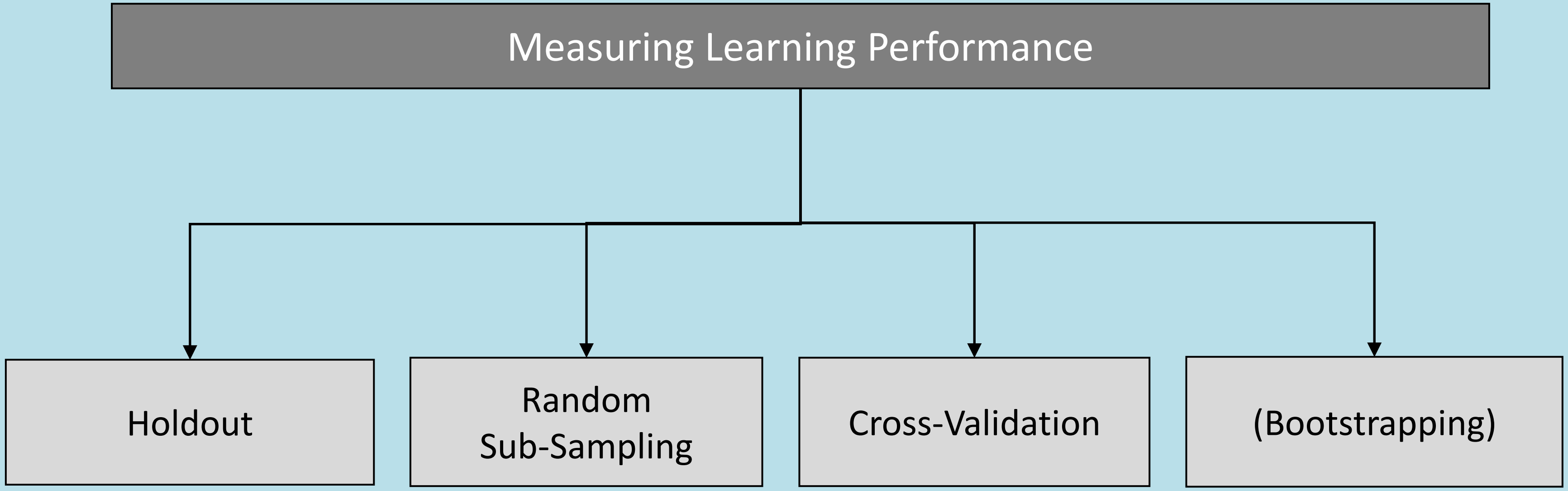
6.1 Split your Data into `train` and `test` Set with Python

- In scikit-learn a random split into training and test sets can be quickly computed with the `train_test_split` helper function.

```
import numpy as np
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=0)
```



6.1 Splitting Paradigms for Measuring Learning Performance



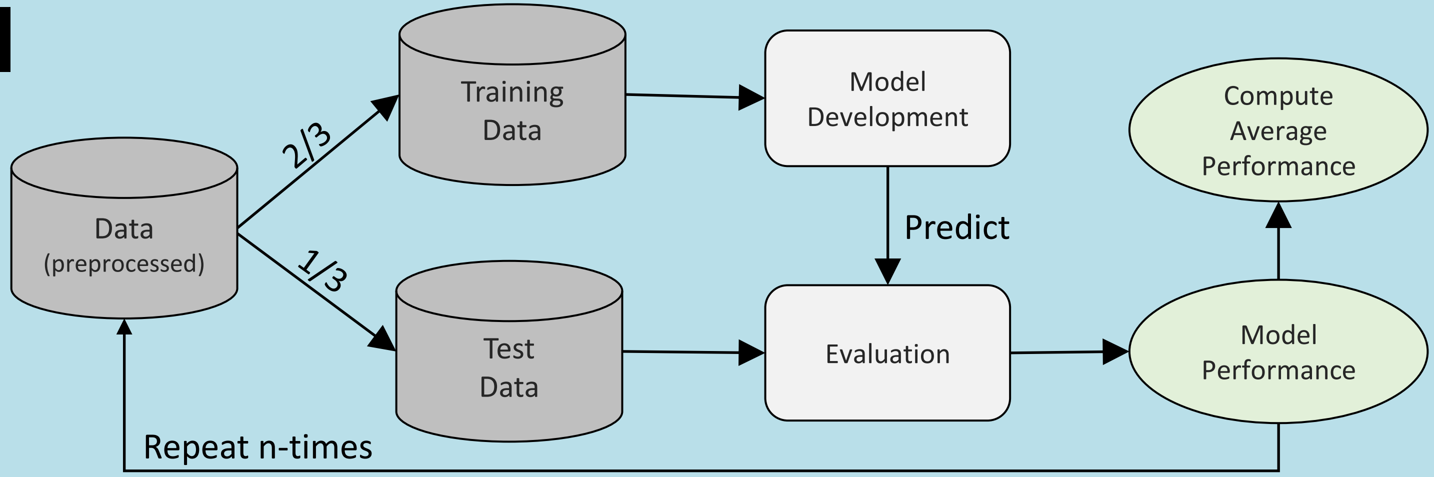
Adapted from Géron, A. (2017)

6.1 Random Sub-Sampling

Procedure

- Repeated holdout with different samples, but measure the average performance
- Multiple models, where you can choose from
- Same disadvantages then simple holdout
- No control how often an observation is used for model building

Visualization



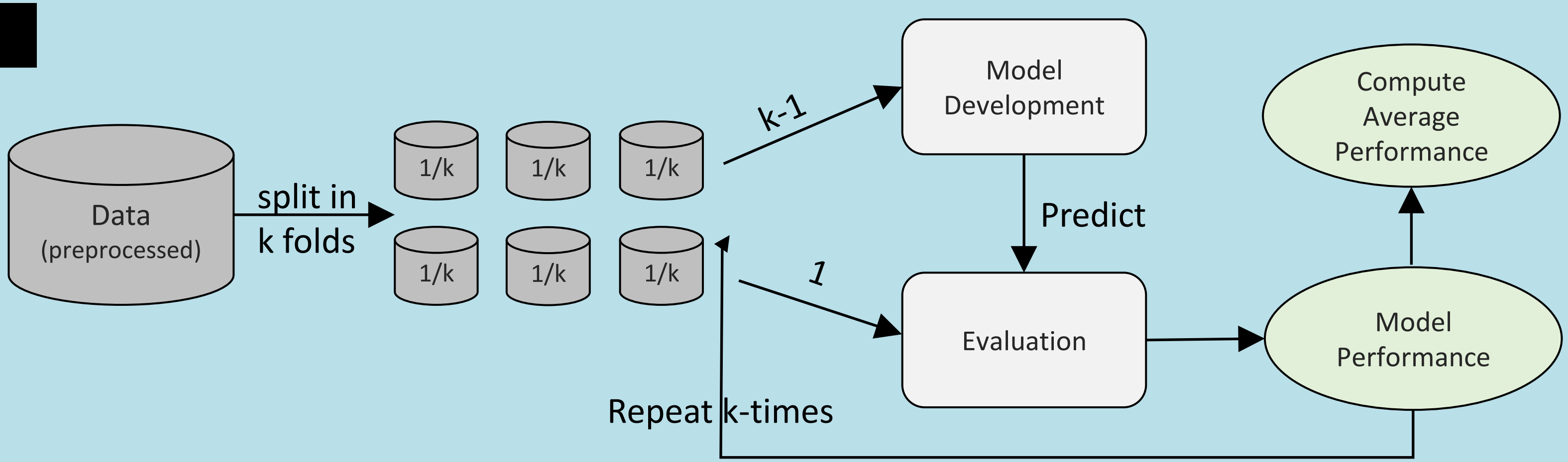
Adapted from Kohavi, R. (1995); Géron, A. (2017)

6.1 K-fold Crossvalidation

Procedure

- Split data into k same-sized samples, use k-1 samples for training, and 1 sample set for testing. Each observation is used the same time for training
- Benefit is that it uses as many model building examples as possible and test sets disjunct
- High complexity of the process (k-runs), reliability of the performance statement is weakened, since these statements are derived from only one example.

Visualization



Adapted from Kim, J.-H. (2009); Géron, A. (2017)

6.1 K-fold Crossvalidation with Python

Crossvalidation()

```
cross_validate(estimator, X, y, cv)
```

Parameters

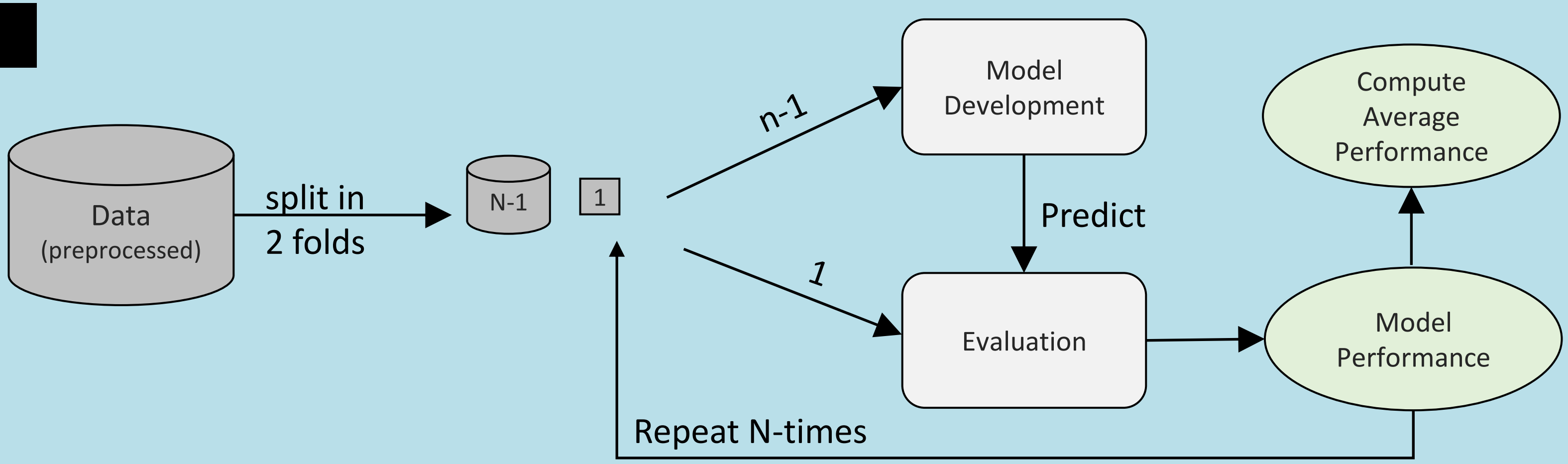
`estimator`	The object to use to fit the data.
`X`	The data to fit. Can be for example a list, or an array.
`y`	The target variable to try to predict in the case of supervised learning.
`cv`	Determines the cross-validation splitting strategy. Possible inputs for cv are: ▪ None, to use the default 5-fold cross validation, ▪ int, to specify the number of folds in a (Stratified)KFold, ▪ CV splitter, ▪ An iterable yielding (train, test) splits as arrays of indices.

6.1 Leave-one Out Cross-Validation

Procedure

- K-fold cross validation taken to its logical extreme, with K equal to the number of data points in the set (N)
- the model is trained on all the data except for one point and a prediction is made for that point
- As before the average error is computed and used to evaluate the model

Visualization



Adapted from Kim, J.-H. (2009); Géron, A. (2017)

6.1 Leave-one Out Cross-Validation with Python

- In scikit-learn we can use the `LeaveOneOut` helper function to perform leave-one-out crossvalidation.

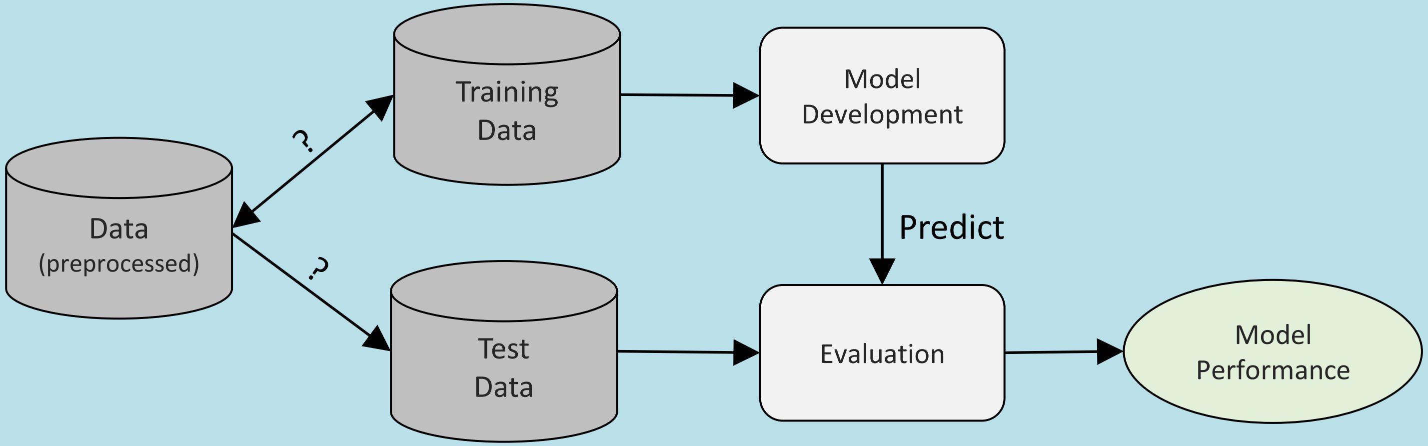
```
>>> from sklearn.model_selection import LeaveOneOut
>>> X = [1, 2, 3, 4]
>>> loo = LeaveOneOut()
>>> for train, test in loo.split(X):
        ("%s %s" % (train, test))

[1 2 3] [0]
[0 2 3] [1]
[0 1 3] [2]
[0 1 2] [3]
```

6.1 Bootstrapping

Procedure

- In the previous procedures, an example was considered several times as a training example (in the same cycle). Here a training set is generated by random selection from the entire set of classified examples (Sampling with replacement).
- Perform sampling with replacement on your original dataset, but use the data points that have not been choses as the test dataset. Repeat this procedure several times and compute the average score as estimation of your model performance.

Visualization



Adapted from Kohavi, R. (1995); Géron, A. (2017)

Your turn!

Task

Identify the *task*, *performance measure* and *experience* of the following machine learning problems. Please discuss your results with your neighbor!

- Build a chess-bot to win an online chess competition
- Build an information system that recognizes hand written addresses
- Build an automated spam-mail system for your email server

Outline

Image source: ↗ Pixabay (2019) / ↗ CC0

▶ **What we will learn:**

- General concepts of AI modelling and what types of problems match which models
- Get an intuition for which model approach fits bets for a particular learning problem
- Know general problems of machine learning and how to optimize learning models

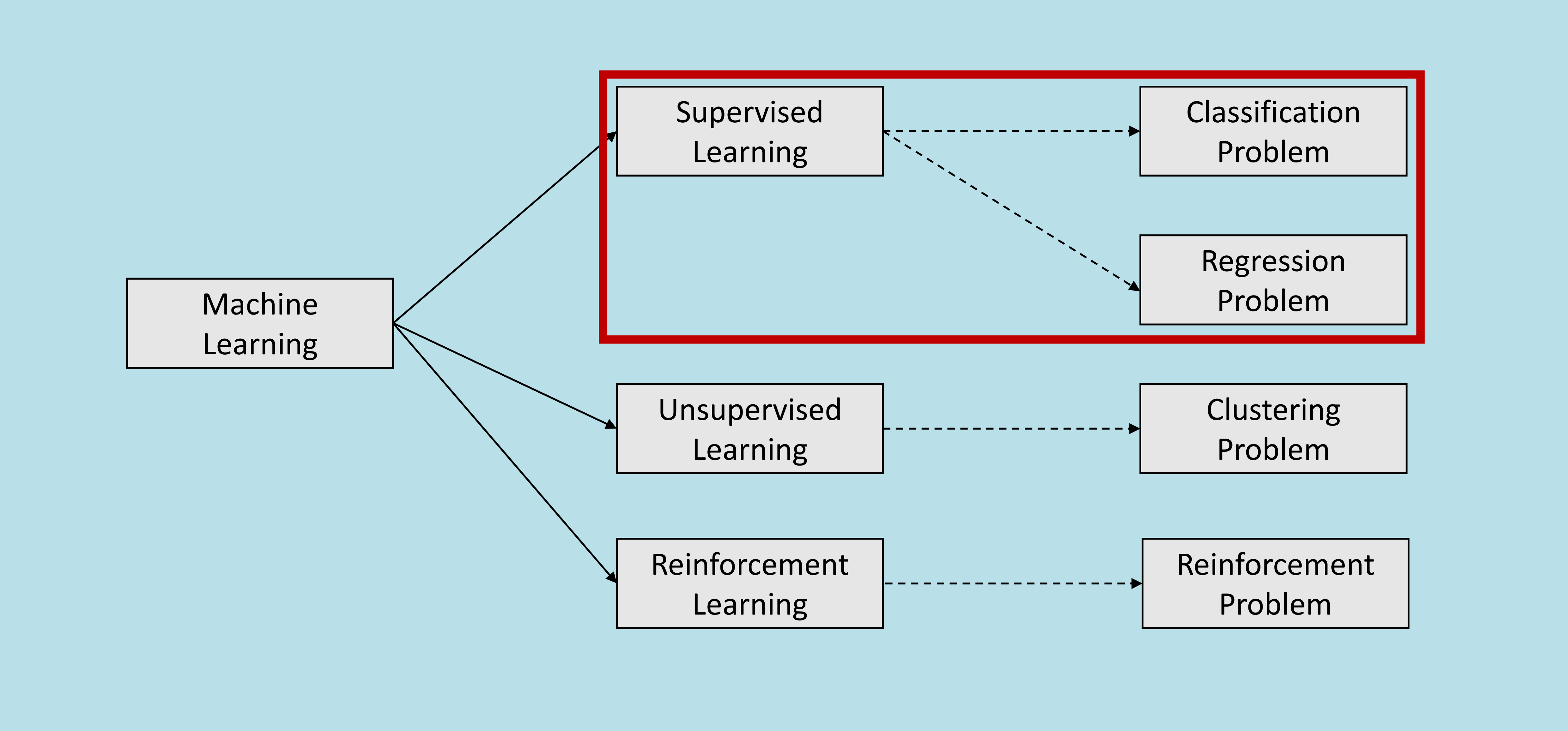
▶ **Duration:**

- 270 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 6.1 – 6.4

6.2 Problem Types in Machine Learning (High-Level)



6.2 Classification Problem and Category Learning

ID	X1	…	Xn	Y
1				A
…				
n				B

$$f: X \rightarrow Y$$

Assign to one of a given set of finite classes

Common real-life tasks:

- Fraud detection: Credit card applications or transactions
- Spam filtering in email
- Recommend news articles books, movies, music etc.
- Handwritten letters
- Astronomical images

6.2 Formalization of Category Learning

- **Task**: Predict if customer will buy a car based on configuration
- **Instance**: consists of different attributes, e.g. <price, maintenance, number_doors>
- Feature vs. attribute vs. value

ID	price	maintenance	number_doors	class
1	small	cheap	3	buy
2	high	expensive	5 or more	no buy
3	small	cheap	4	buy
…	…	…	…	…
n	medium	medium	2	no buy

- price $\in$ {small, medium, high}
- maintenance $\in$ {cheap, medium, expensive}
- number_doors $\in$ {2,3,4,5 more}
- **class** $\in$ {positive, negative}

Adapted from Hunt et al. (1966); Rusell, S., & Norvig, P. (2016)

- Possible hypotheses are consistent with the training data set

ID	price	maintenance	number_doors	class
1	small	cheap	3	buy
2	high	expensive	5 or more	no buy
3	small	cheap	4	buy
…	…	…	…	…
n	medium	medium	2	no buy

1	$price = small$
2	$maintenance = cheap \cup price = small$
3	$number\ of\ doors = 3 \cup number\ of\ doors = 2$
4	$price = small \cap maintenance = cheap$

? Do you have any ideas of possible hypotheses based on this training data?

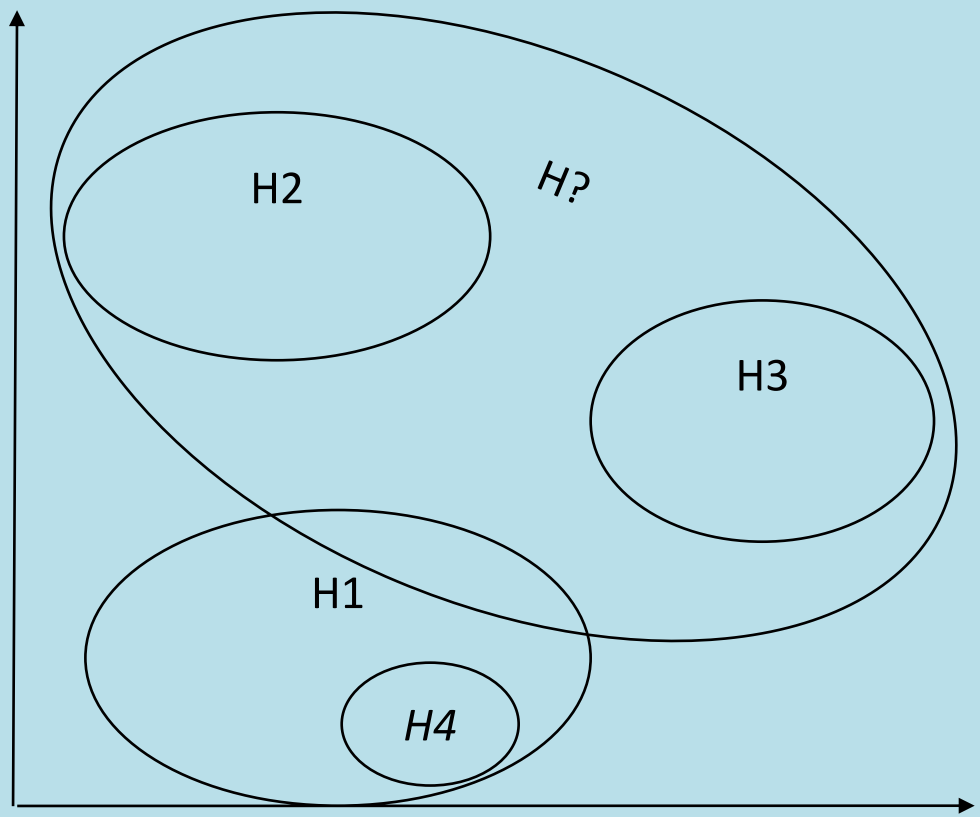
$$H = \{h\ | h : X \rightarrow Y\}$$

! Some older, psychological papers term these kind of hypotheses “concepts”. See e.g. Bruner, J. S., & Austin, G. A. (1986)

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Hypothesis Generation and Hypothesis Space II

$H = \{h \mid h: X \rightarrow Y\}$



- For learning concepts on instances described by *n* discrete-valued features, consider the space of conjunctive hypotheses represented by a vector of *n* constraints

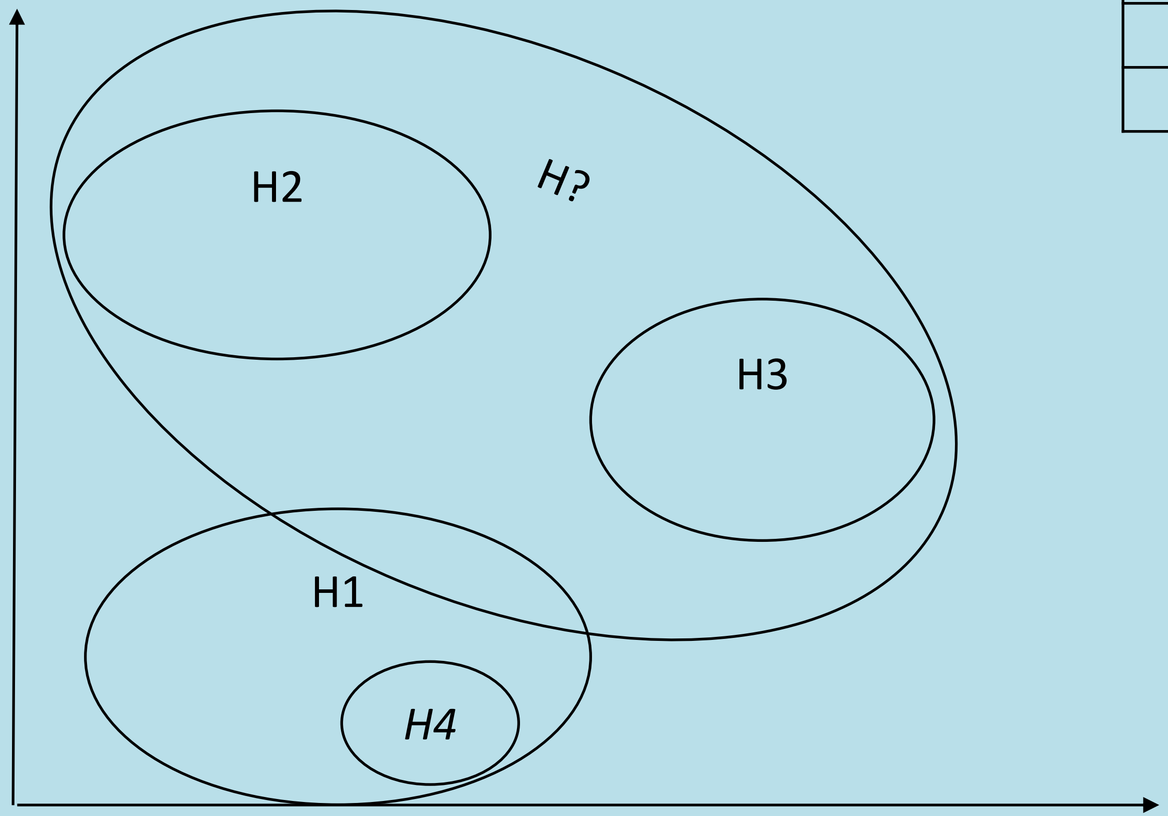
$< c_1, c_2, \ldots\ c_n >$ where each c_i is either:
- ?, a wild card indicating no constraint on the ith feature
- A specific value from the domain of the ith feature
- Ø indicating no value is acceptable

Sample conjunctive hypotheses:
- $< price = small, doors = 3, ? >$
- $<?\,,?\,,?>$ (most general hypothesis)
- $<\ \emptyset, \emptyset, \emptyset\ >$ (most specific hypothesis)

Adapted from Mitchell, T. M. (1997); Rusell, S., & Norvig, P. (2016)

6.2 Hypothesis Generation and Hypothesis Space III

$$H = \{h \mid h: X \rightarrow Y\}$$



1	$price = small$
2	$maintenance = cheap \cup price = small$
3	$number\ of\ doors = 3 \cup number\ of\ doors = 2$
4	$price = small \cap maintenance = cheap$

Adapted from Mitchell, T. M. (1997); Rusell, S., & Norvig, P. (2016)

6.2 Challenge: Learning and Generalization

- There are many possible hypotheses. Our purpose is to find the hypothesis that is able to predict the training data correctly AND new data correctly

ID	price	maintenance	number_doors	class
1	small	cheap	3	buy
42	small	expensive	4	no buy

X

Our current best hypothesis "$price = small$" can not handle it

- Hypotheses must generalize to correctly classify instances not in the training data

- Simply memorizing training examples is a consistent hypothesis, but it does not generalize (see "learning by enumeration")

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 How to Generalize: Inductive Learning Assumption

- But how to achieve generalization?

- **Assumption**: If any function approximates the target concept well on a sufficiently large set of training examples, it also will approximate the target function well on unobserved examples

- We assume that the training and test examples are drawn independently from the same underlying distribution

Adapted from Mitchell, T. M. (1997); Rusell, S., & Norvig, P. (2016)

6.2 Learning by Enumeration

- **Other idea**: Category learning model as search problem (see lecture 2)

Algorithm: Learning by Enumeration

persistent:
H, hypothesisspace with h hypotheses
D, training data

For each *h in H :*
    If *h is consistent with D*
        return *h*

- This algorithm is guaranteed to terminate with a consistent hypothesis if one exists; however, it is obviously computationally intractable for almost any practical problem.

Adapted from Mitchell, T. M. (1997); Rusell, S., & Norvig, P. (2016)

6.2 More Efficient Learning?

- Is there a way to learn conjunctive concepts more efficient to solve our classification problems?

- How do humans learn hypotheses about the world?

- Concept Learning System (CLS), Hunt et al. (1966): find distinguishing features between two categories, adjust concept by divide-and-conquer

- Computer Science: (Decision) Tree

Adapted from Mitchell, T. M. (1997); Hunt et al. (1966); Rusell, S., & Norvig, P. (2016)

6.2 Most Popular Supervised Learning Algorithms

Supervised Learning Algorithms

Classification Algorithms

- Decision Tree
- KNN
- SVM

Hybrid

- Logistic Regression
- Regression Trees
- Model Trees

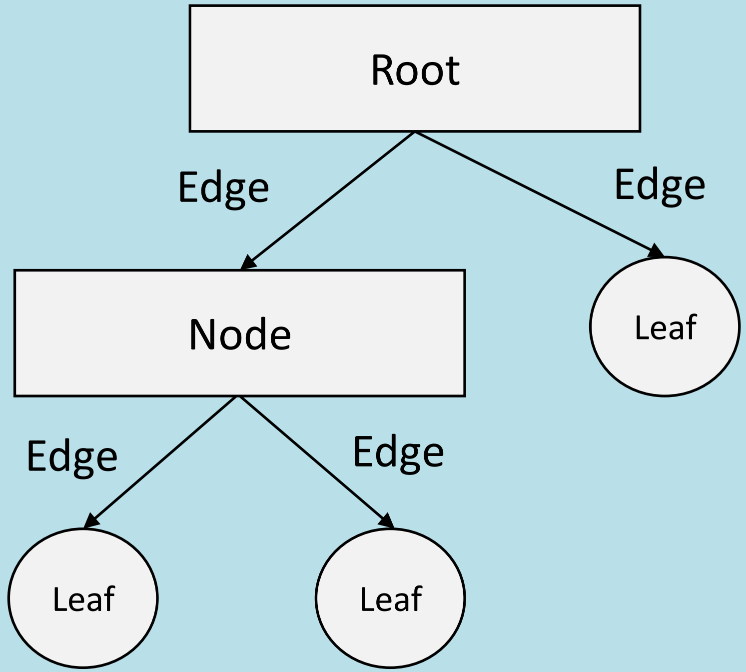
Regression Algorithms

- Linear Regression
- Polynominal Regression
- Generalized Linear Models etc.

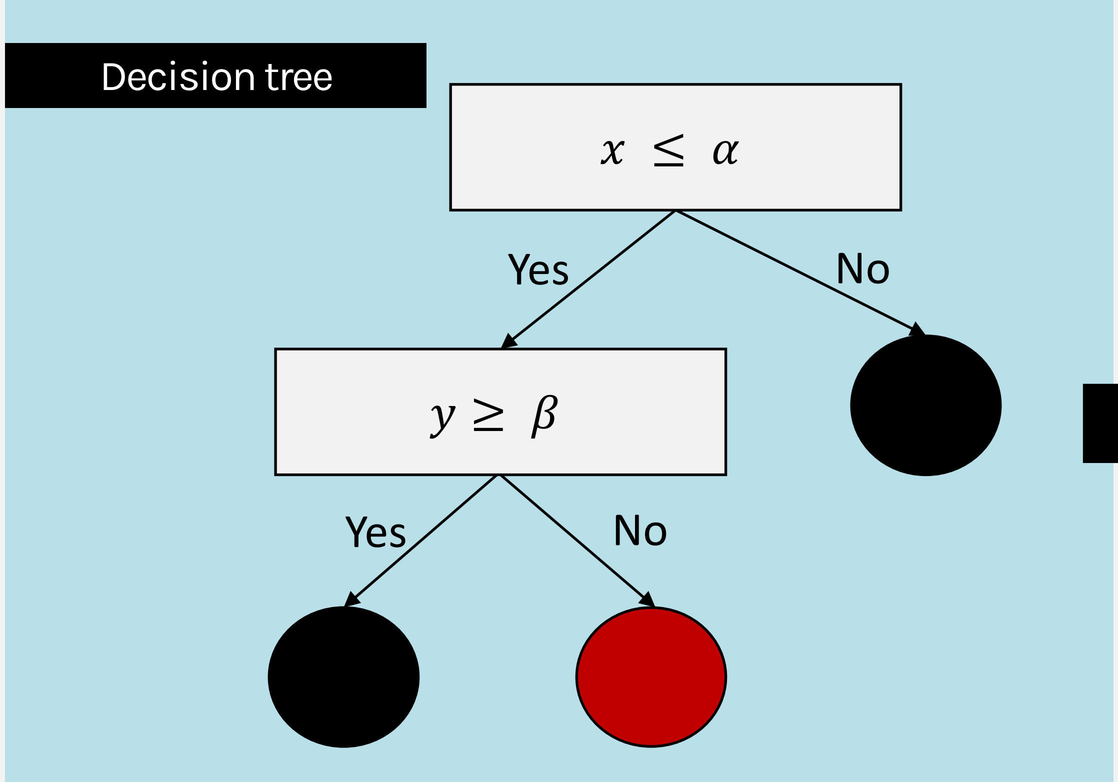
Adapted from Géron, A. (2017)

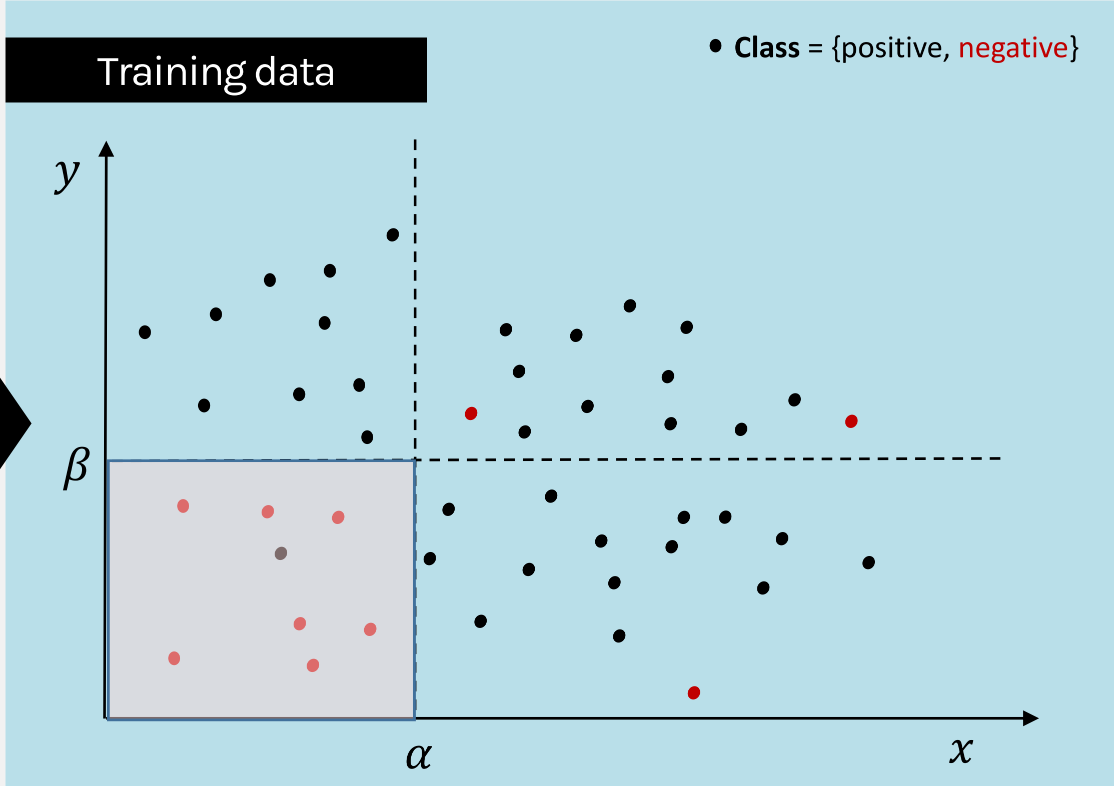
6.2 Decision Trees

- A decision tree is made of nodes (representing attributes), edges (representing possible values), and leaves (representing categories or classes)
- Each path in a decision tree from the root to a leafe can rewritten as a set of rules, i.e. disjunctive normal form (DNF)
- Can represent arbitrary conjunction and disjunction



6.2 Geometric Interpretation: Decision Tree

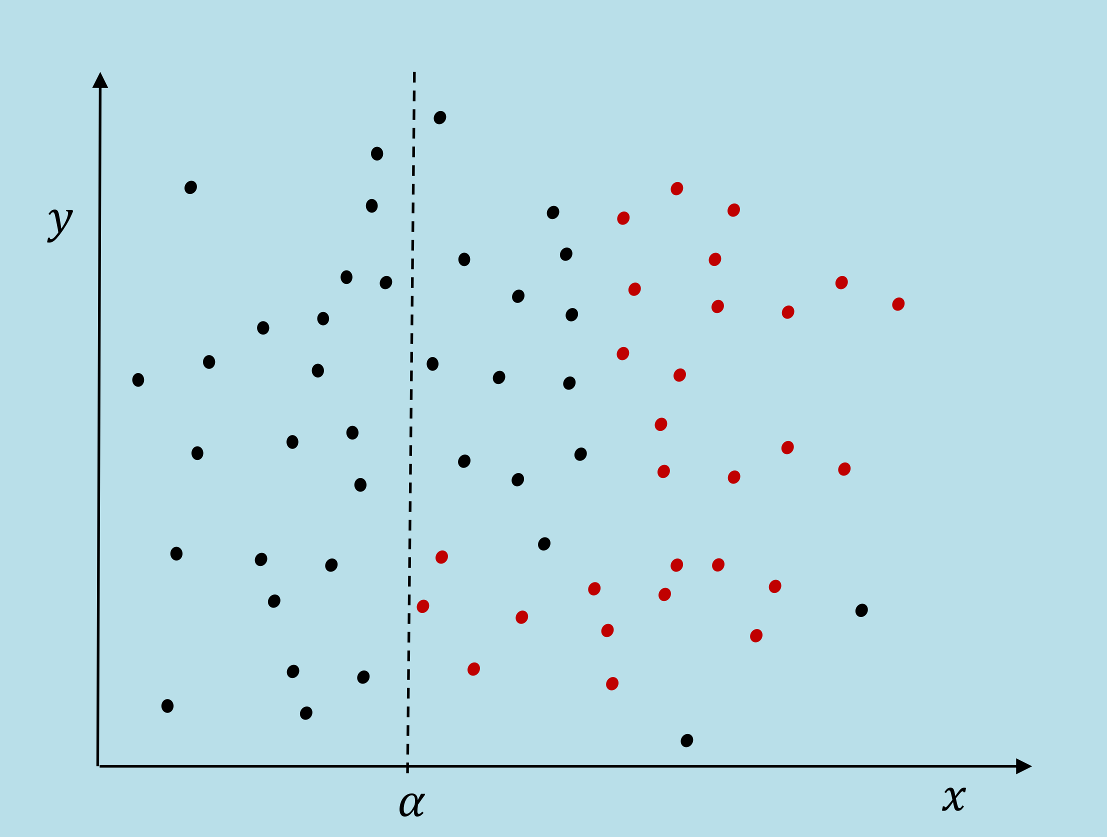


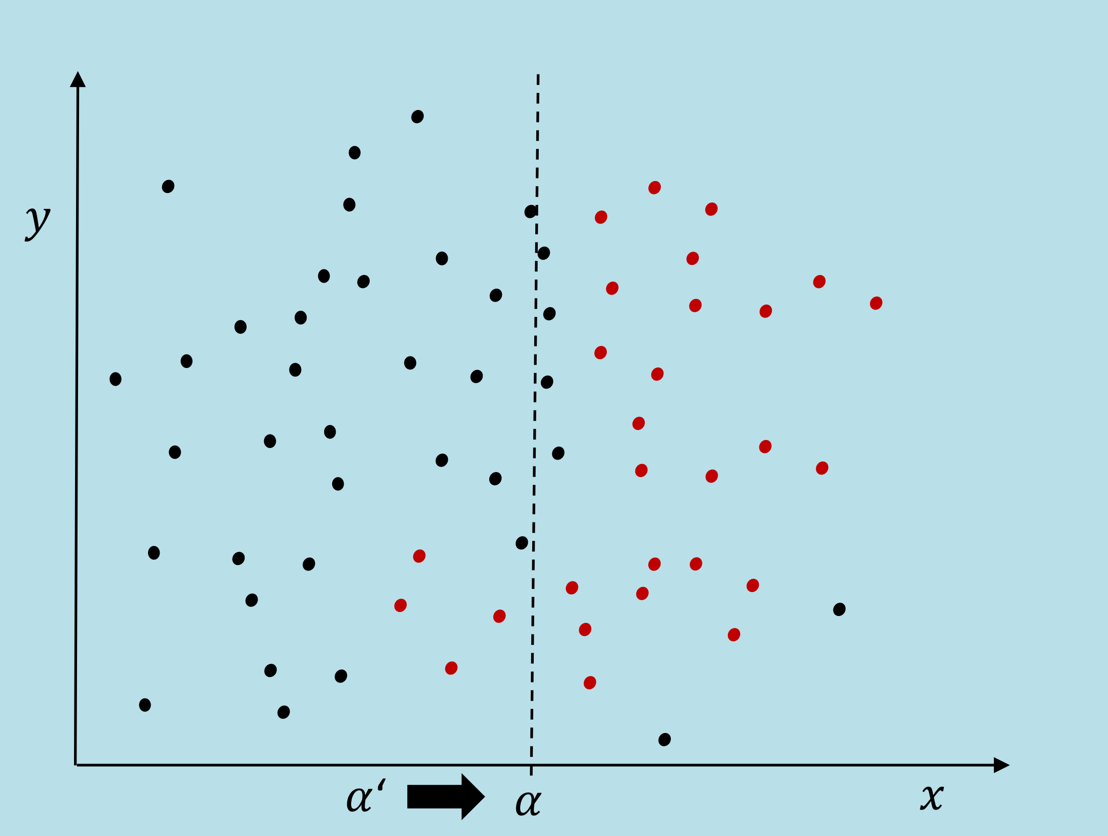


Top-Down Decision Tree Induction: Decision trees divide the decision space into subspaces. Recursively build the decision tree top-down by divide and conquer. Objects falling into a certain space are classified as one respective group.

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Geometric Interpretation: Linear Classification Problem



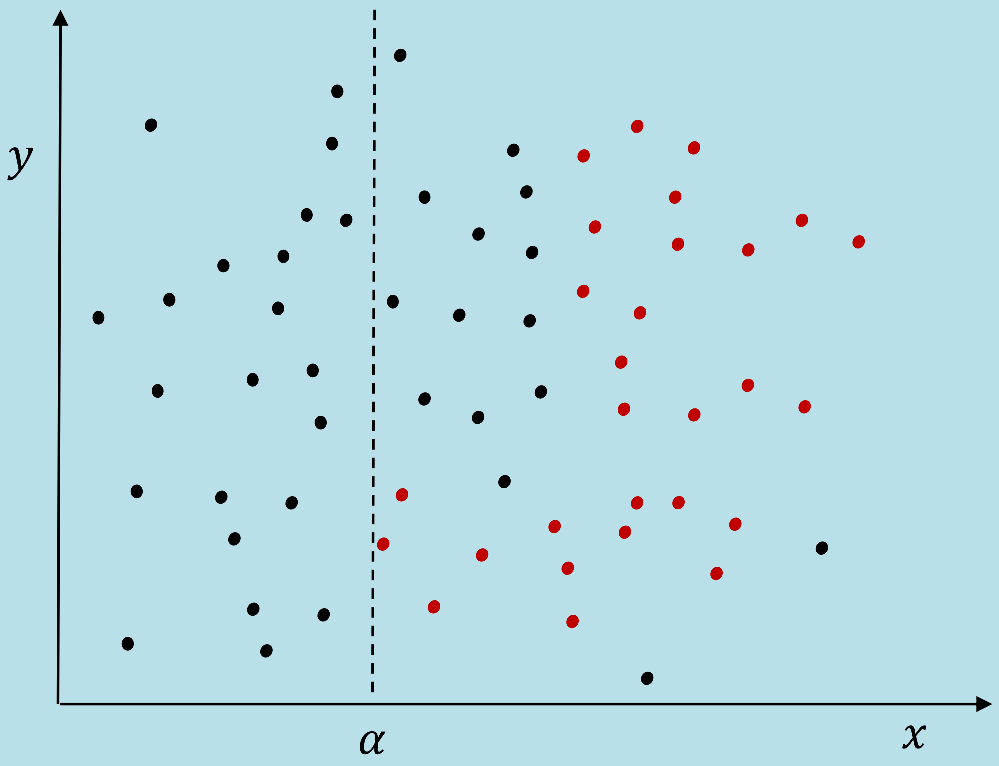


?

Which one of these would be better?

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Entropy as a Measure of Impurity



- We need a function that describes the *"impurity"* of a subset of data

- **Solution**: *Entropy*

$$H(S) = -\sum_{i=1}^{n} p_i \cdot log_2\, p_i$$

Here: $H(S) = -p_1 log_2(p_1) - p_2 log_2(p_2)$

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

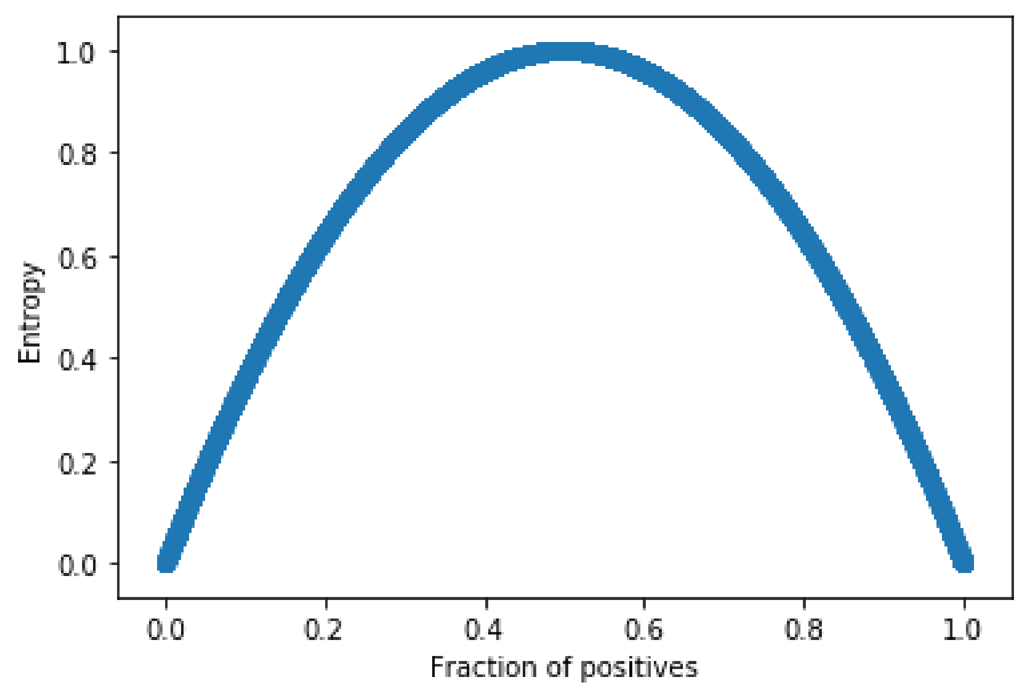
6.2 Entropy as a Measure of Impurity

- Entropy (disorder, impurity) of a set of examples S relative to a binary classification is:

$$Entropy(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

where p_1 is the fraction of positive examples in S and p_0 is the fraction of negatives

- If all examples are in one category, entropy is zero (we define $0\log(0) = 0$)
- If examples are equally mixed ($p_1 = p_0 = 0.5$), entropy is a maximum of 1



Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Multi-Class Entropy

- Entropy can be viewed as the number of bits required on average to encode the class of an example in S where data compression (e.g. Huffman coding) is used to give shorter codes to more likely cases.

- For multi-class problems with c categories, entropy generalizes to:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \, log_2(p_i)$$

6.2 Picking a Good Split Feature

- Goal is to have the resulting tree be as small as possible (remember our generalization discussion about learning each single instance)
- Finding a minimal decision tree (nodes, leaves, or depth) is an NP-hard optimization problem
- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest
- Want to pick a feature that creates subsets of examples that are relatively “pure” in a single class so they are “closer” to being leaf nodes

6.2 Information Gain and Gini Impurity

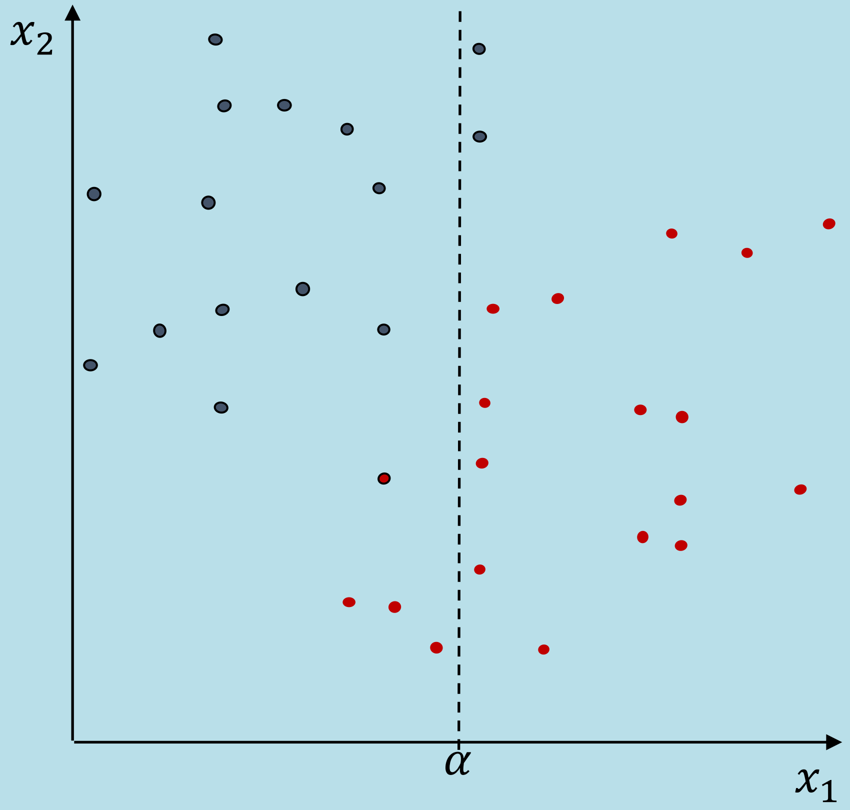
- Besides there are a variety of metrics that can be used as heuristic for picking a good split, like *information gain* of class label S and attribute A or *gini impurity*

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

$$Gain\ (S, A) = Entropy\ (S) - Entropy\ (S, A)$$

6.2 Entropy and Information Gain

Entire population (34 instances)



Entire population (34 instances) "Parent entropy":

$$-\left(\frac{15}{34}\log_2\frac{15}{34}\right)-\left(\frac{19}{34}\log_2\frac{19}{34}\right)=0.989$$

"Child entropy" of left side (17 instances):

$$-\left(\frac{13}{17}\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\log_2\frac{4}{17}\right)=0.787$$

"Child entropy" of right side (17 instances):

$$-\left(\frac{2}{17}\log_2\frac{2}{17}\right)-\left(\frac{15}{17}\log_2\frac{15}{17}\right)=0.523$$

Entropy of children:

$$\left(\frac{17}{34}0.787\right)+\left(\frac{17}{34}0.523\right)=0.655$$

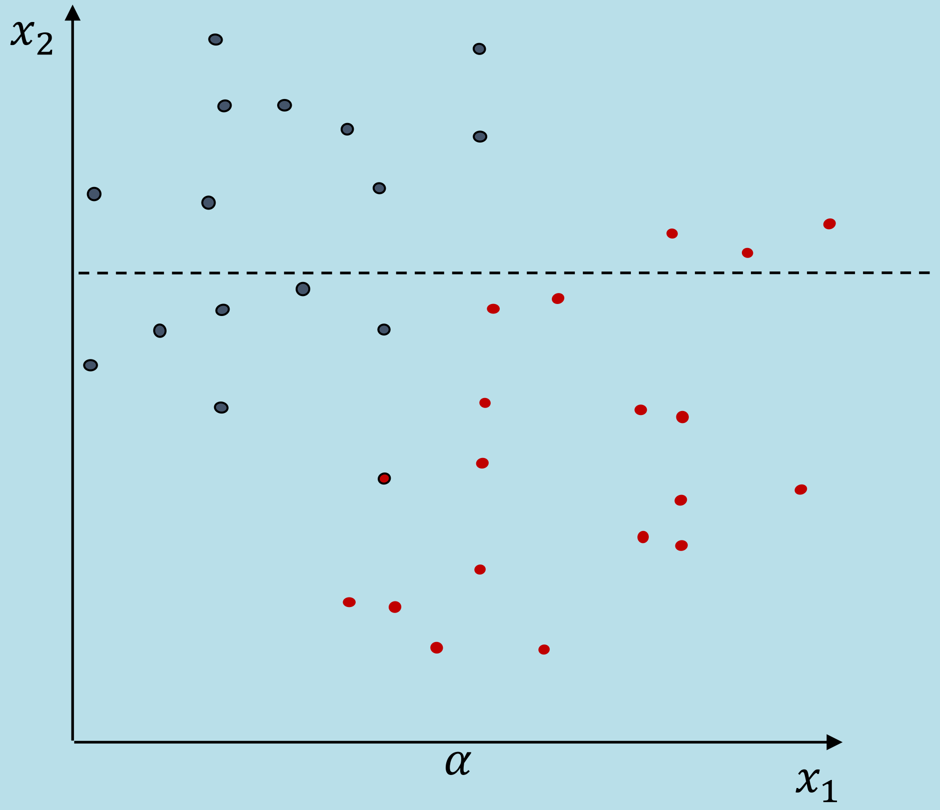
Information gain:

$$0.989-0.655=0.334$$

Adapted from Rusell, S., & Norvig, P. (2016)

6.2 Entropy and Information Gain

Entire population (34 instances)



Entire population (34 instances):

$$-\left(\frac{15}{34}\log_2\frac{15}{34}\right)-\left(\frac{19}{34}\log_2\frac{19}{34}\right)=0.989$$

$$\left(\frac{12}{34}0.811\right)+\left(\frac{22}{34}0.845\right)=0.833$$

Entropy above (12 instances):

$$-\left(\frac{9}{12}\log_2\frac{9}{12}\right)-\left(\frac{3}{12}\log_2\frac{3}{12}\right)$$
$$=0.811$$

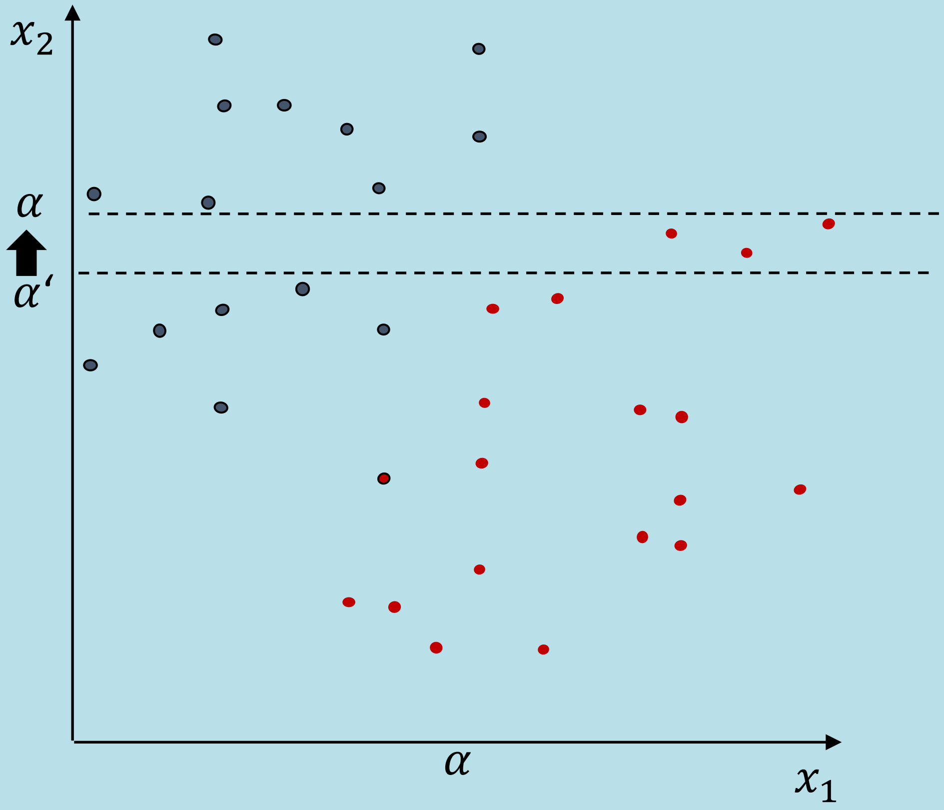
Entropy below (22 instances):

$$-\left(\frac{6}{22}\log_2\frac{6}{22}\right)-\left(\frac{16}{22}\log_2\frac{16}{22}\right)$$
$$=0.845$$

Adapted from Rusell, S., & Norvig, P. (2016)

6.2 Entropy and Information Gain

Entire population (34 instances)



Entire population (34 instances):

$$-\left(\frac{15}{34}\log_2\frac{15}{34}\right)-\left(\frac{19}{34}\log_2\frac{19}{34}\right)=0.989$$

$$\left(\frac{12}{34}0.811\right)+\left(\frac{22}{34}0.845\right)=0.833$$

Entropy above (9 instances):

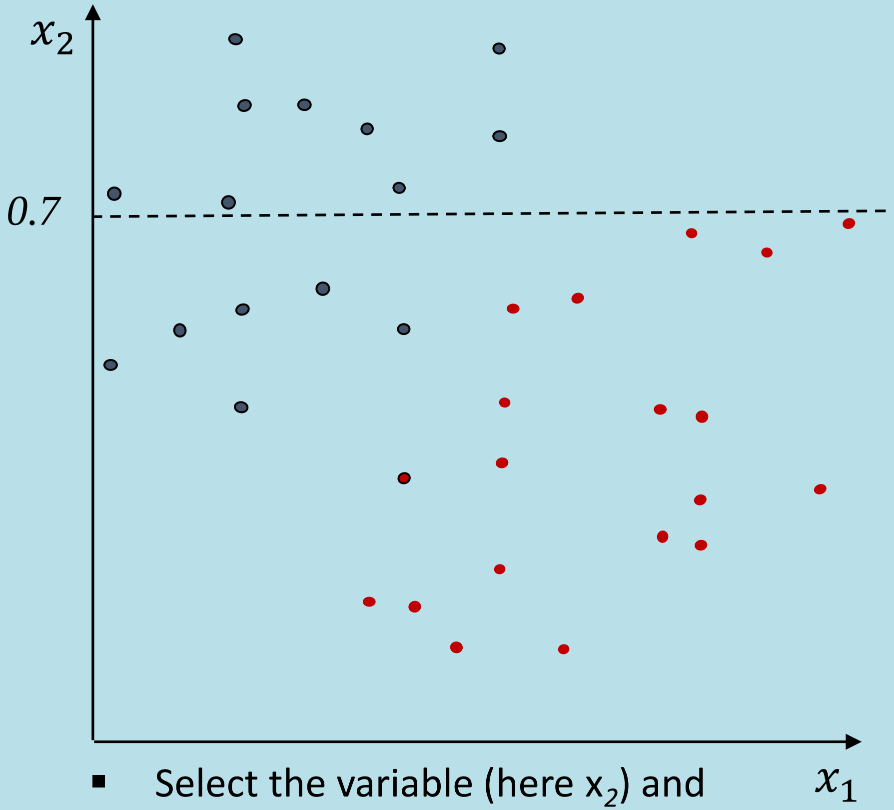
$$-\left(\frac{9}{9}\log_2\frac{9}{9}\right)-\left(\frac{0}{9}\log_2\frac{0}{9}\right)$$
$$=0$$

Entropy below (25 instances):

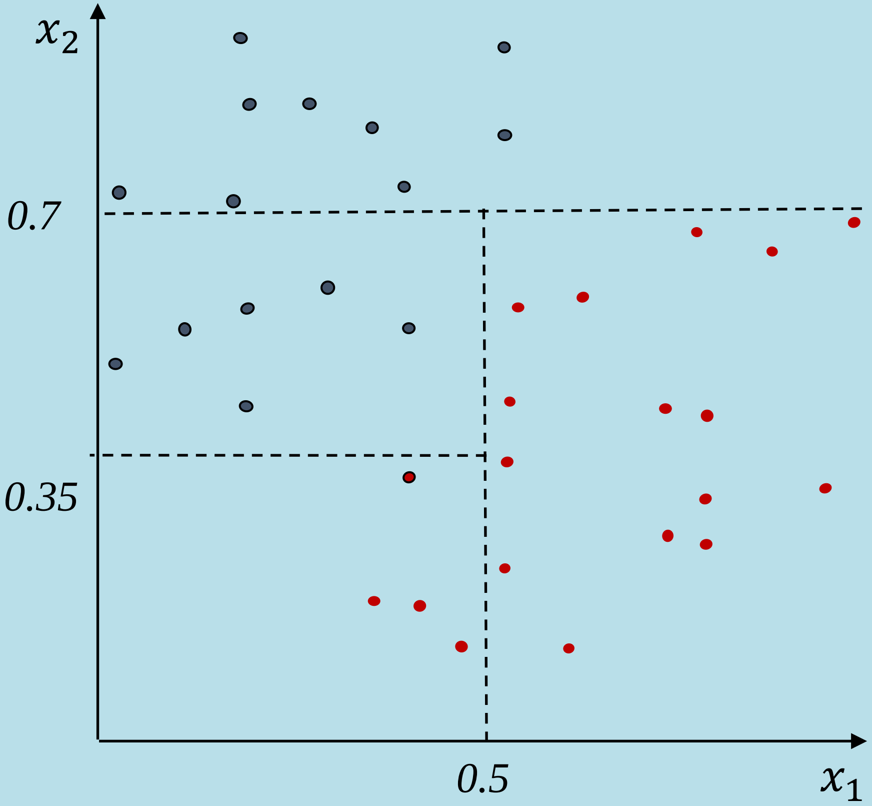
$$-\left(\frac{6}{25}\log_2\frac{6}{25}\right)-\left(\frac{19}{25}\log_2\frac{19}{25}\right)$$
$$=0.795$$

Adapted from Rusell, S., & Norvig, P. (2016)

6.2 Entropy and Information Gain



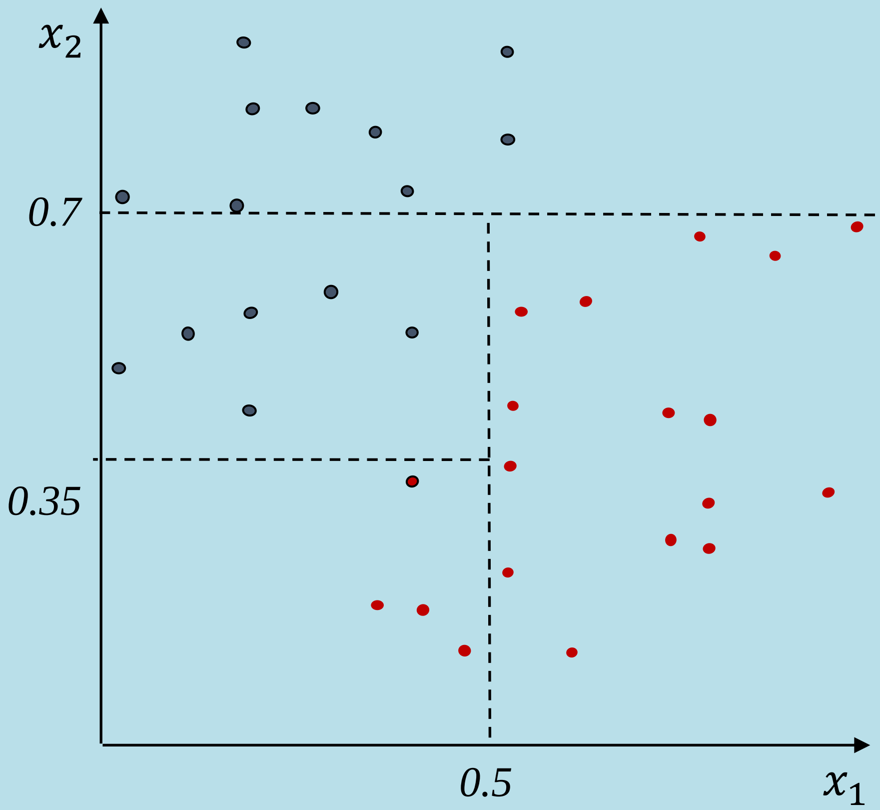
- Select the variable (here x_2) and split (here at $\alpha = 0.7$) that yields the highest information gain (best discriminator)



- Do this iteratively until no more impurity exists (the set of points is fully classified)

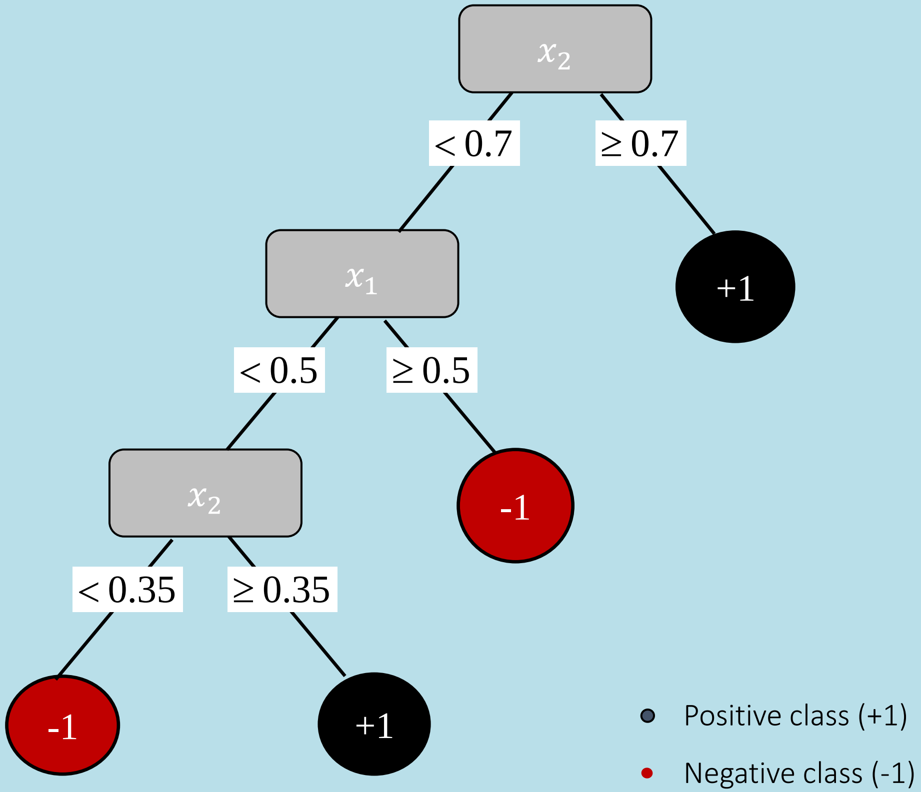
Adapted from Rusell, S., & Norvig, P. (2016)

6.2 Final Decision Tree



- Do this iteratively until no more impurity exists (the set of points is fully classified)

- This iterative process of splitting can be expressed as a "*decision tree*":



Adapted from Rusell, S., & Norvig, P. (2016)

6.2 `DecisionTreeClassifier` in Python

`DecisionTreeClassifier()`

```
DecisionTreeClassifier(criterion, max_depth)
```

Parameters

`criterion`	The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
`max_depth`	You can set an individual type of index for the row labels, the default index if no index is passed is `np.arrange(n)`.

6.2 Simple Decision Tree with Python

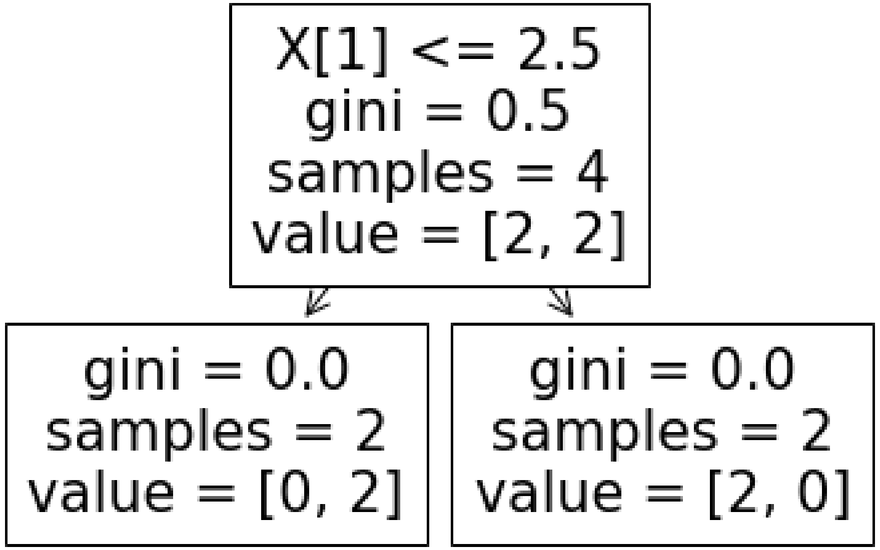
- Build your own decision tree with Python

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=0)
clf = clf.fit(X_train, Y_train)
```

- After being fitted, the model can then be used to predict the class of samples

```
clf.predict(X_test, Y_test)
```

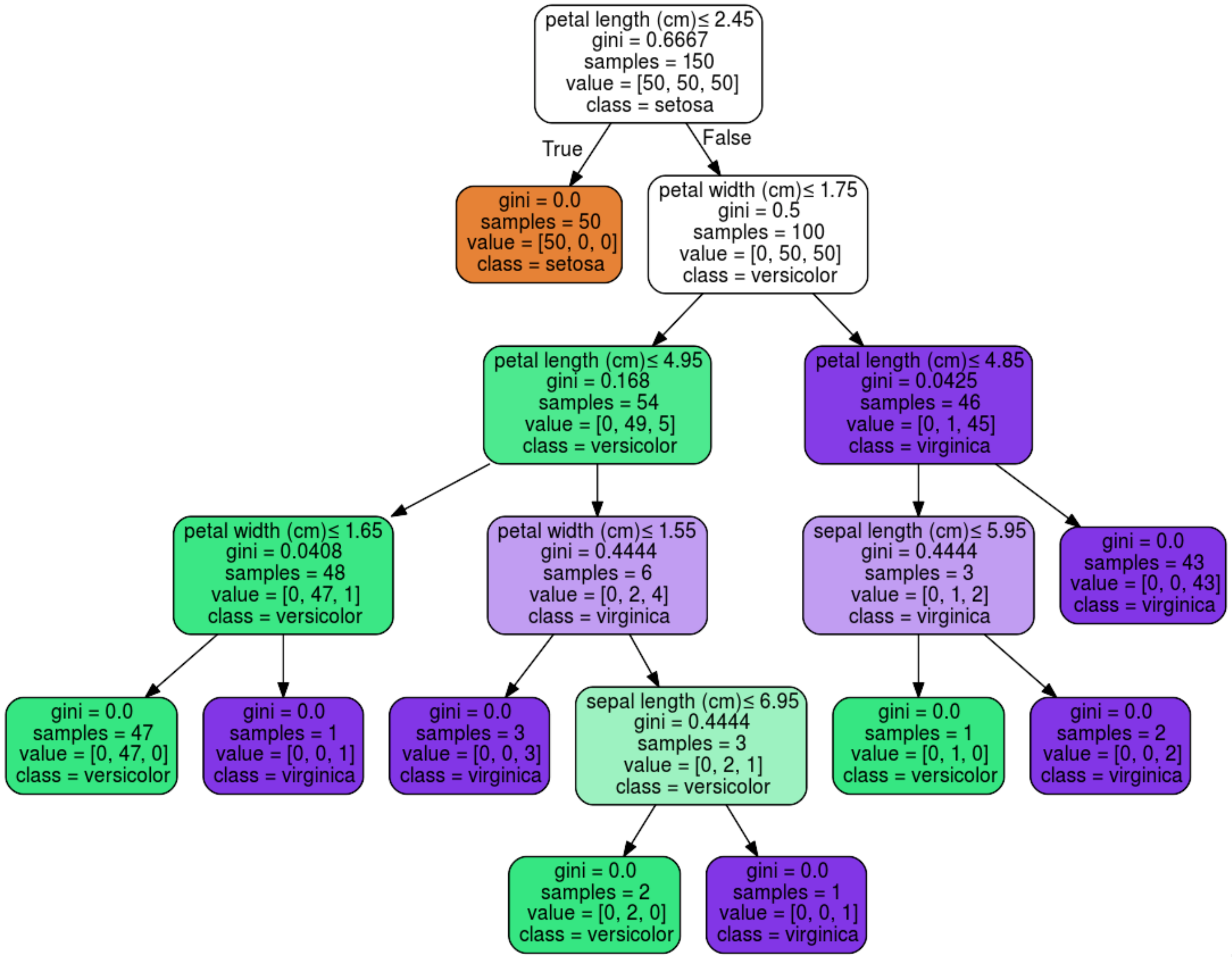
6.2 Plot Decision Trees with the `plot_tree` Function

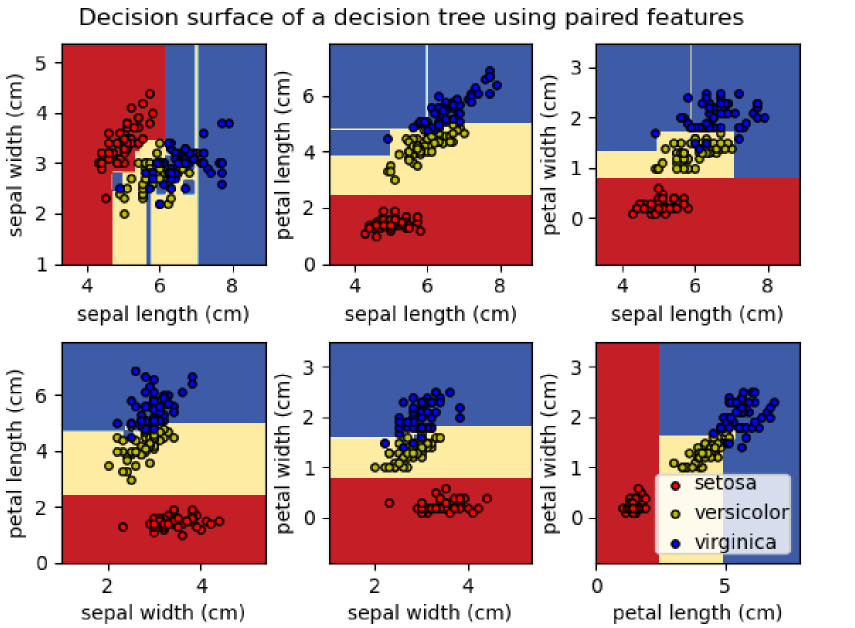



```
X = array([[1, 1, 3],
           [3, 3, 5],
           [1, 2, 5],
           [2, 3, 2]])

Y = array([1, 0, 1, 0])

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)

tree.plot_tree(clf)
```

You find all code examples in the ↗ code folder of the lecture repository

6.2 Scitkit-learn has Many Tools to Visualize Decision Trees





Adapted from Scikit-learn (2021)

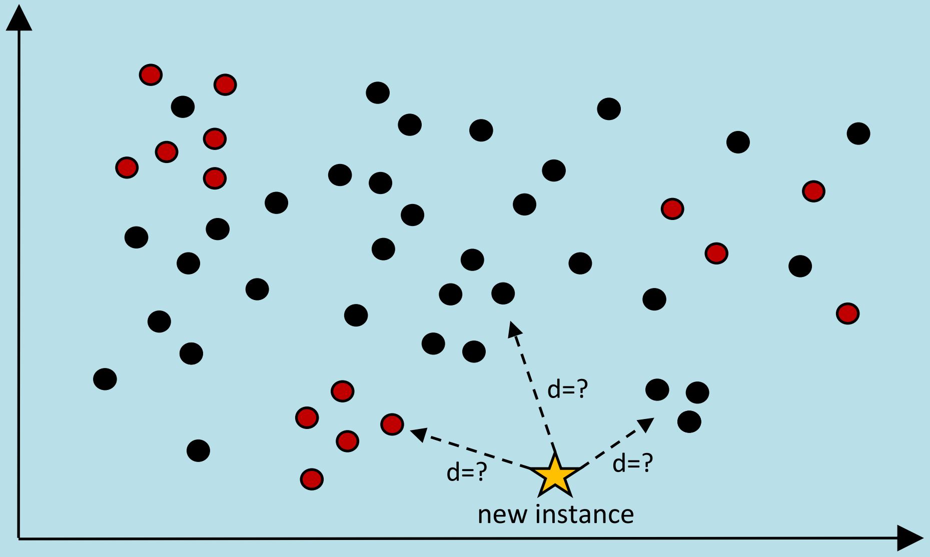
You find all code examples of scikit-learn on the userguide (↗ here)

6.2 Other Useful Decision Tree Functions

- The scikit-learn module provides many useful decision tree related functions

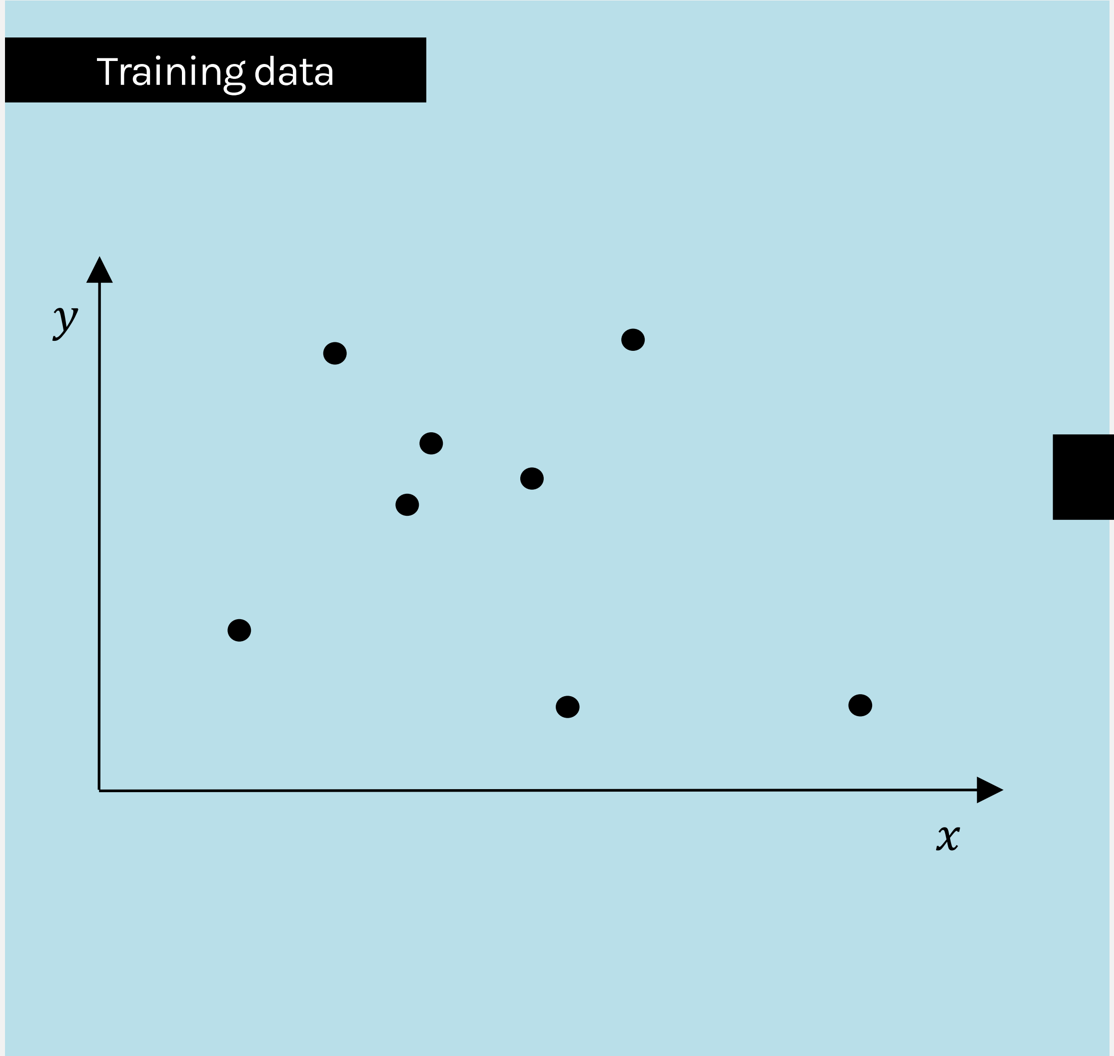
`cost_complexity_pruning_path(X, y[, …])`	Compute the pruning path during Minimal Cost-Complexity Pruning.
`decision_path(X[, check_input])`	Return the decision path in the tree.
`fit(X, y[, sample_weight, check_input, …])`	Build a decision tree classifier from the training set (X, y).
`get_depth()`	Return the depth of the decision tree.
`get_n_leaves()`	Return the number of leaves of the decision tree.
`predict(X[, check_input])`	Predict class or regression value for X.
`predict_proba(X[, check_input])`	Predict class probabilities of the input samples X.
`score(X, y[, sample_weight])`	Return the mean accuracy on the given test data and labels.

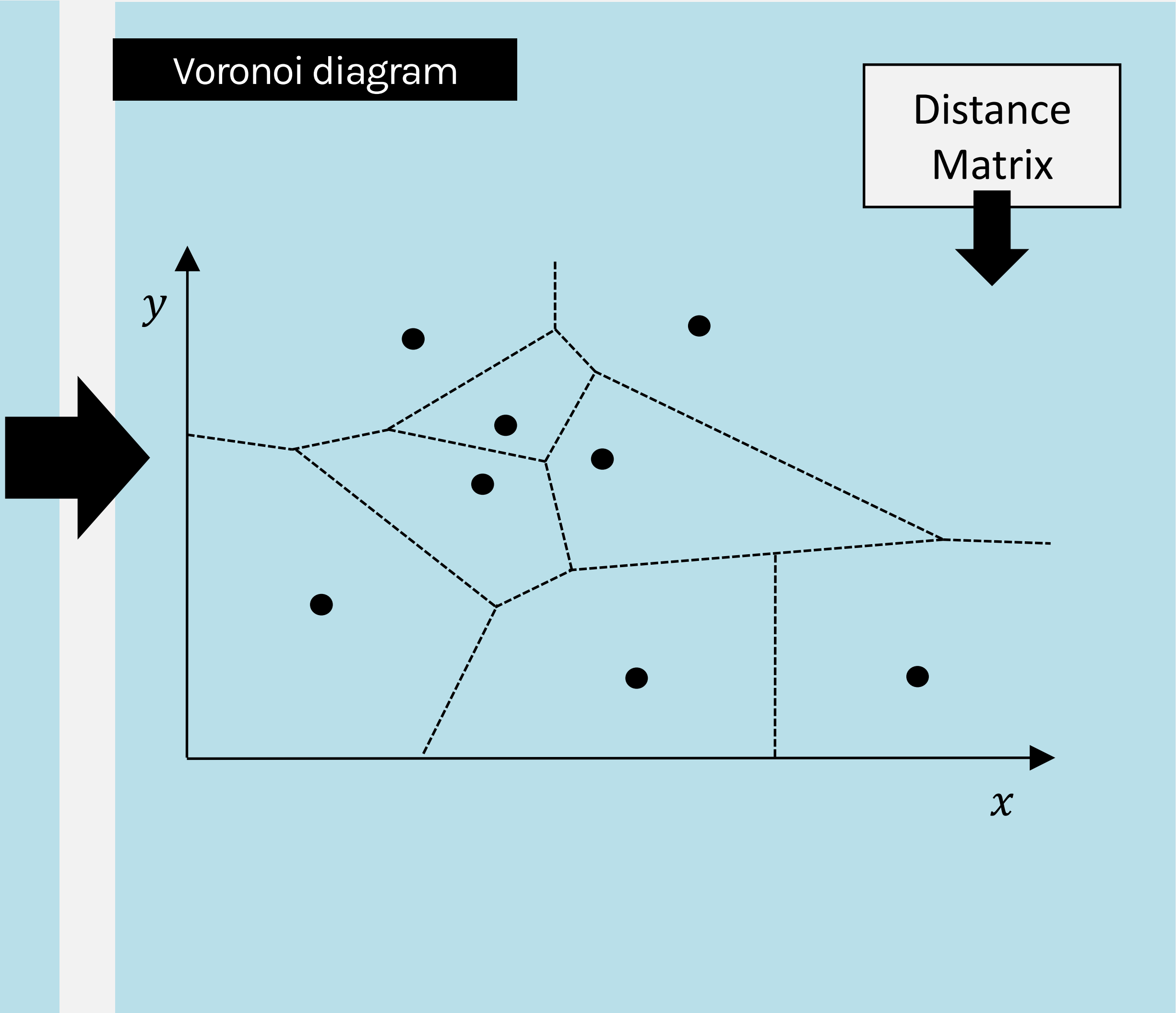
6.2 K-Nearest-Neighbor (KNN)



- Classify an object by his k nearest neighbours
- K-Nearest-Neighbor approach is no „real model" ist just a type of simple classification heuristic (there is no learning)
- When a new instance comes in find the k closest neighbors and use the majority class to classify the new instance

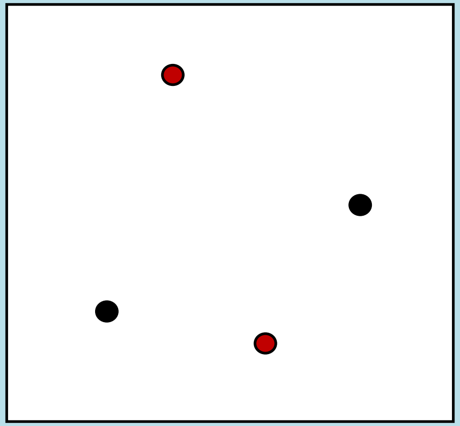
6.2 Geometric Interpretation: Voronoi diagram





6.2 K-Nearest Neighbor Procedure

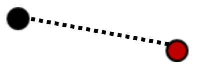
Initial dataset



Compute the distance of each point-to-point relation (use heuristics to speed up)

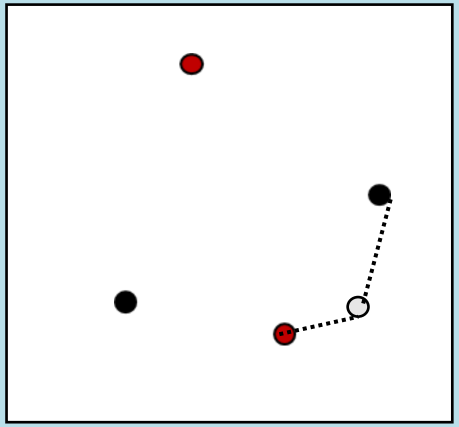
1. Distance Matrix

Object	O1	O2	On
O1	0	1	4
O2	2	0	3
…	…	…	…



Use distance matrix to label new objects

Label new subsets



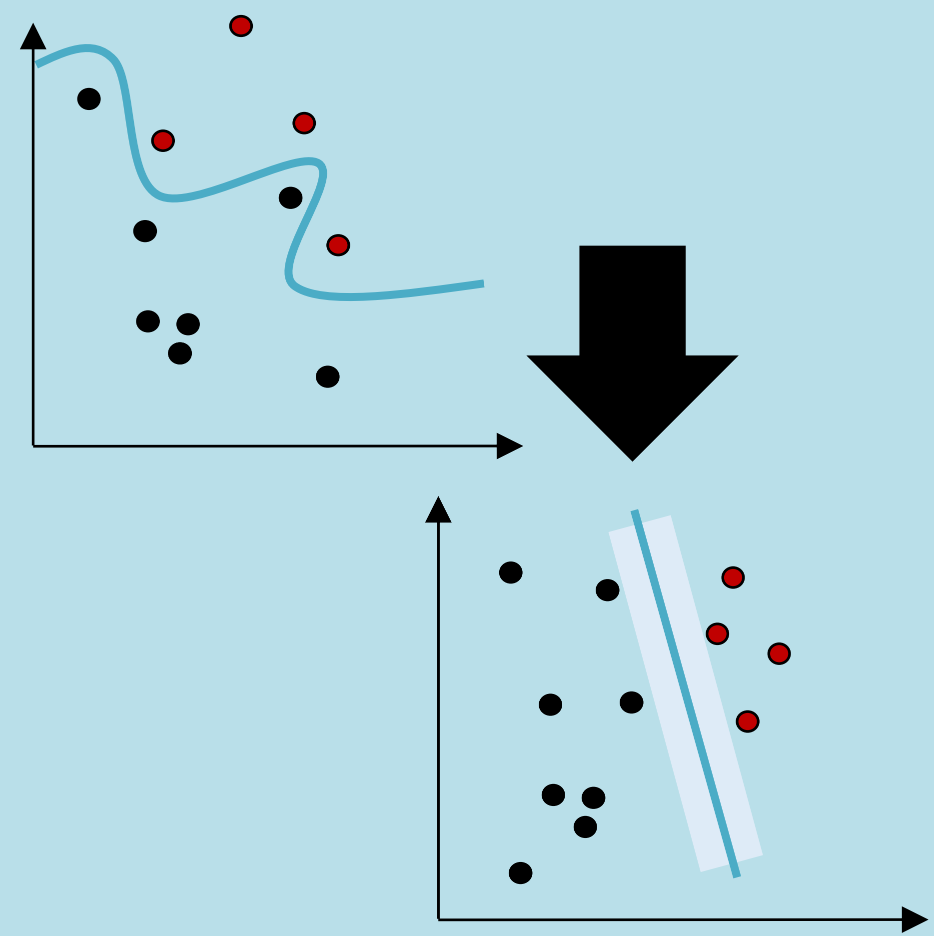
The distance between two objects (e.g. X, Y) is calcuated by different types of distance

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

- Build your own KNN-classifier with Python

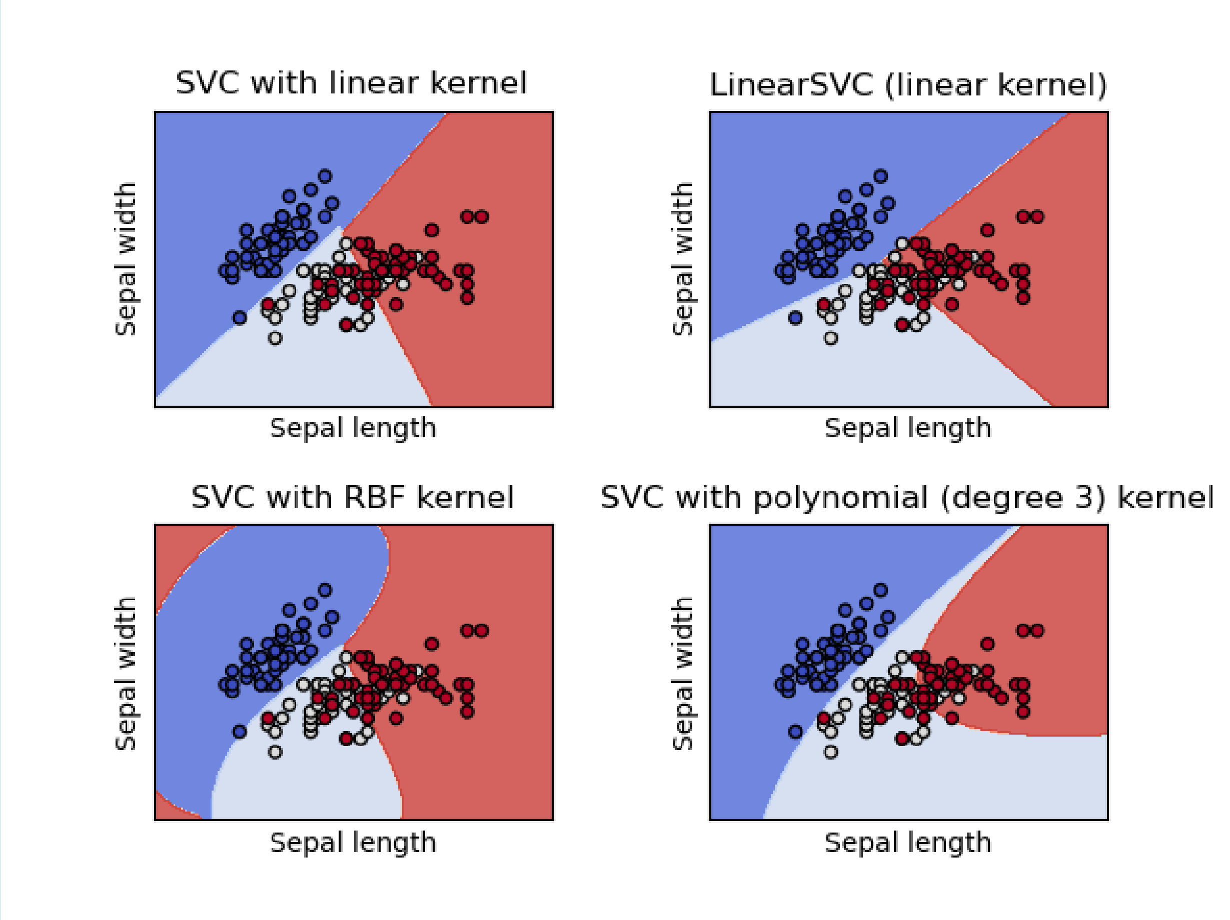
```
>>> import numpy as np
>>> from sklearn.impute import KNNImputer

>>> nan = np.nan
>>> X = [[1, 2, nan], [3, 4, 3], [nan, 6, 5], [8, 8, 7]]

>>> imputer = KNNImputer(n_neighbors=2, weights="uniform")
>>> imputer.fit_transform(X)

array([[1. , 2. , 4. ],
       [3. , 4. , 3. ],
       [5.5, 6. , 5. ],
       [8. , 8. , 7. ]])
```

6.2 Support-Vector-Machines (SVM)



- From chapter 4 we know that we can convert difficult separtion problems to easier problems by adding more dimension to it
- SVM takes a low-dimensional input space and transforms it into a higher dimensional space, and retransform it afterwards (Kernel trick)
- Select a hyperplane with the maximum possible margin between support vectors in the given dataset

6.2 Support-Vector-Machines Graphical Illustration



Adapted from Scikit-learn (2021)

6.2 `SupportVectorClassification` in Python

`SVC()`

```
SVC(kernel, C)
```

Parameters

`kernel`	Specifies the kernel type to be used in the algorithm like e.g. ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’.
`C`	Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty

Evaluation of Classification Learning Models

- Classification accuracy (percentage of instances classified correctly).
  - Measured on an independent test data.

- Training time (efficiency of training algorithm).

- Testing time (efficiency of subsequent classification).

Adapted from Mitchell, T. M. (1997); Rusell, S., & Norvig, P. (2016)

6.2 Evaluation of Classification Models

Confusion Matrix

		True class	
		Positive	*Negative*
Predicted class	*Positive*	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

D

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$True\ Positive Rate = \frac{TP}{TP + FN}$$

$$True\ Negative Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Example Confusion Matrix

Task: Car classification, $positive\ class\ = Taycan$

	Taycan	911	718
Taycan	4	2	2
911	2	2	8
718	1	0	3

<table>
<tr><td></td><td>Taycan</td><td>911</td><td>718</td></tr>
<tr><td>Taycan</td><td>TP (4)</td><td colspan="2">FN (4)</td></tr>
<tr><td>911</td><td rowspan="2">FP (3)</td><td colspan="2" rowspan="2">TN (13)</td></tr>
<tr><td>718</td></tr>
</table>

Accuracy: Amount of correct model predictions

$$A = \frac{(TP + TN)}{Total} = \frac{4 + 13}{4 + 4 + 3 + 13} = 70{,}83\ \%$$

- 4 actually positive cases were correctly assigned
- 13 other cases were correctly assigned as negative

- The model is correct in 70% of the Taycan detection tasks

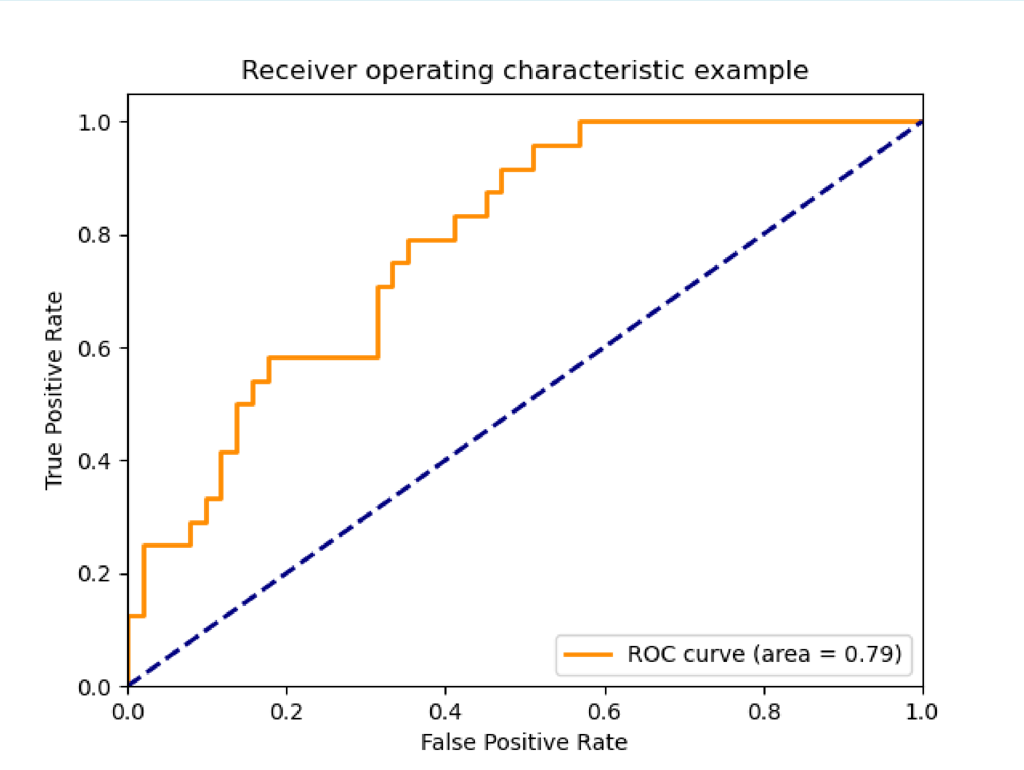
Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Confusion Matrix in Python

- In Python you can compute confusion matrizes easily with scikit-learn

```
>>> from sklearn.metrics import confusion_matrix

>>> y_true = ["Taycan", "911 ", "Taycan ", "Taycan ", "911", "718"]
>>> y_pred = ["911", "911", "Taycan ", "Taycan ", "911", "Taycan "]

>>> confusion_matrix(y_true, y_pred, labels=["911 ", "718", "Taycan"])


array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]]))
```

6.2 Visualization of the Accuracy – Receiver Operating Characteristic




```
from sklearn import metrics
metrics.roc_curve()
```

- Receiver Operating Characteristic (ROC) is used to assess the accuracy of a continuous measurement for predicting a binary outcome
- ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis
- The top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Area Under the Curve (AUC)

- The `roc_auc_score` function computes the area under the receiver operating characteristic curve, which is also denoted by AUC or AUROC.

```
>>> import numpy as np
>>> from sklearn.metrics import roc_auc_score

>>> y_true = np.array([0, 0, 1, 1])
>>> y_scores = np.array([0.1, 0.4, 0.35, 0.8])
>>> roc_auc_score(y_true, y_scores)

0.75
```

- By computing the area under the roc curve, the curve information is summarized in one number

6.2 More Model Evaluation Metrics on Scikit-learn

3.3.1.1. Common cases: predefined values

For the most common use cases, you can designate a scorer object with the `scoring` parameter; the table below shows all possible values. All scorer objects follow the convention that **higher return values are better than lower return values**. Thus metrics which measure the distance between the model and the data, like `metrics.mean_squared_error`, are available as neg_mean_squared_error which return the negated value of the metric.

Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multi[illegible]mple
'neg_log_loss'	`metrics.log_loss`	requires [illegible]
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'

https://scikit-learn.org/

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Regression Problem

But what to do, when the prediction is numeric?

ID	X1	…	Xn	Y
1				0.3
…				
n				1.1

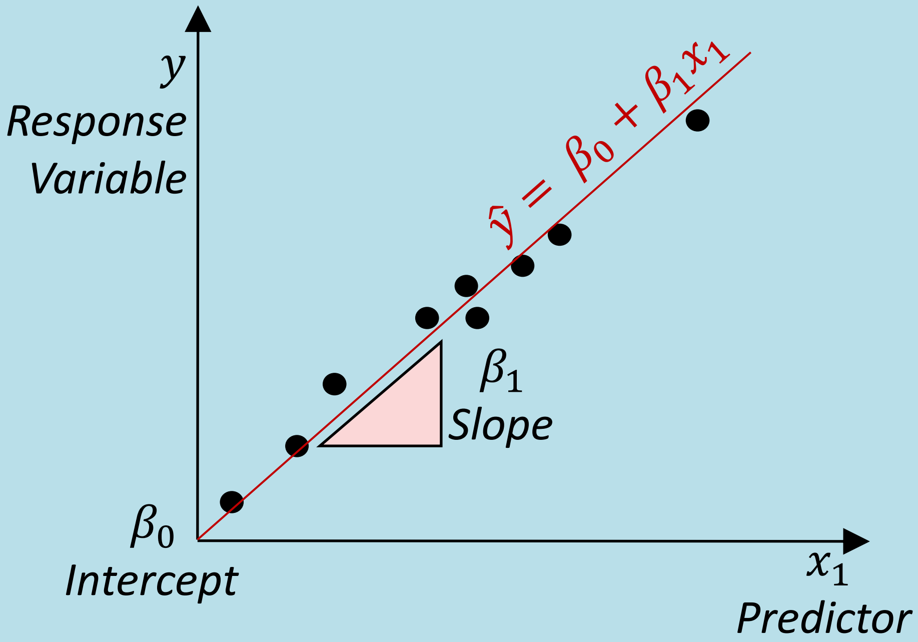
$$f : X \to Y$$

Estimate a continuous value

Common real-life problems:

- Compute scores: Fraud detection
- Predict values in finance and investing like product prices, stock prices etc.
- Estimate demographic characteristics in marketing like salary, age, size etc.

6.2 Linear Regression



- Linear regression fits a linear model with coefficients $w = (w_1, \ldots, w_n)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation

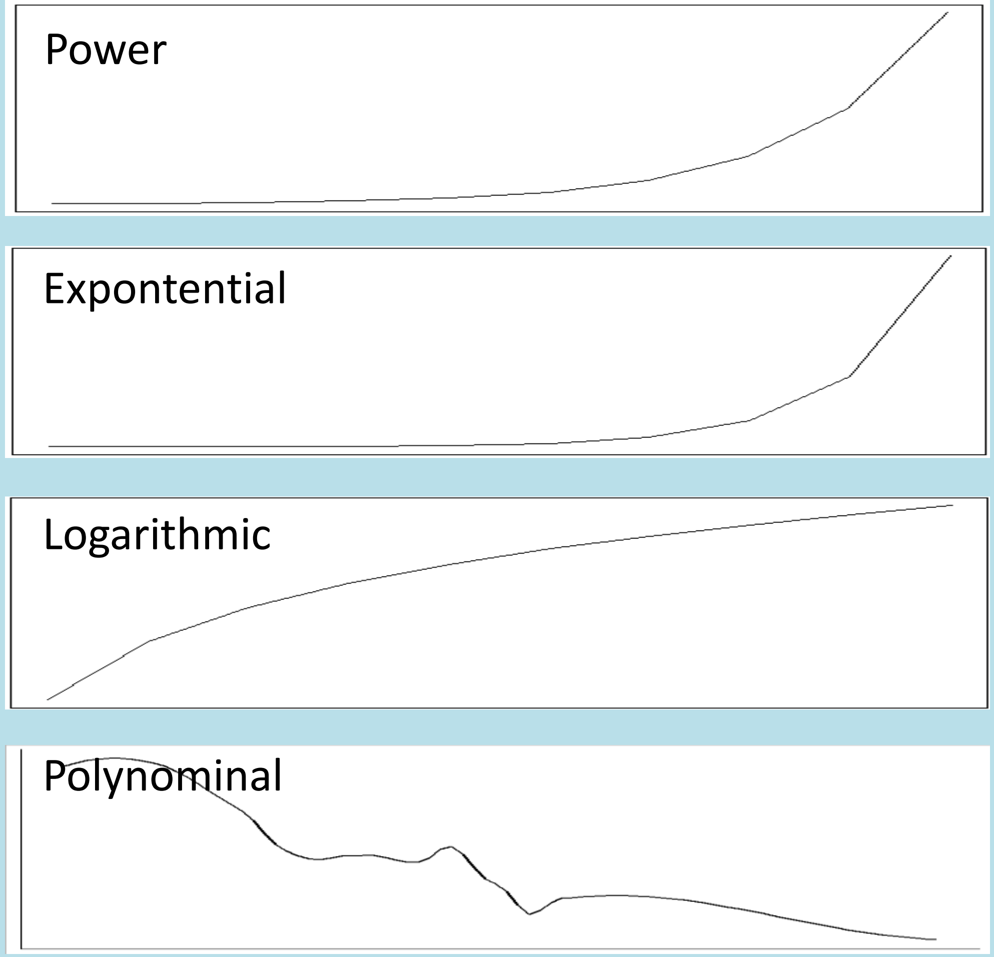
- **Univariate (single attribute)**

$$\bar{y} = \beta_0 + \beta_1 x_1$$

- **Multivariate (many attributes)**

$$\bar{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

6.2 Non-Linear Regression Methods



Method properties

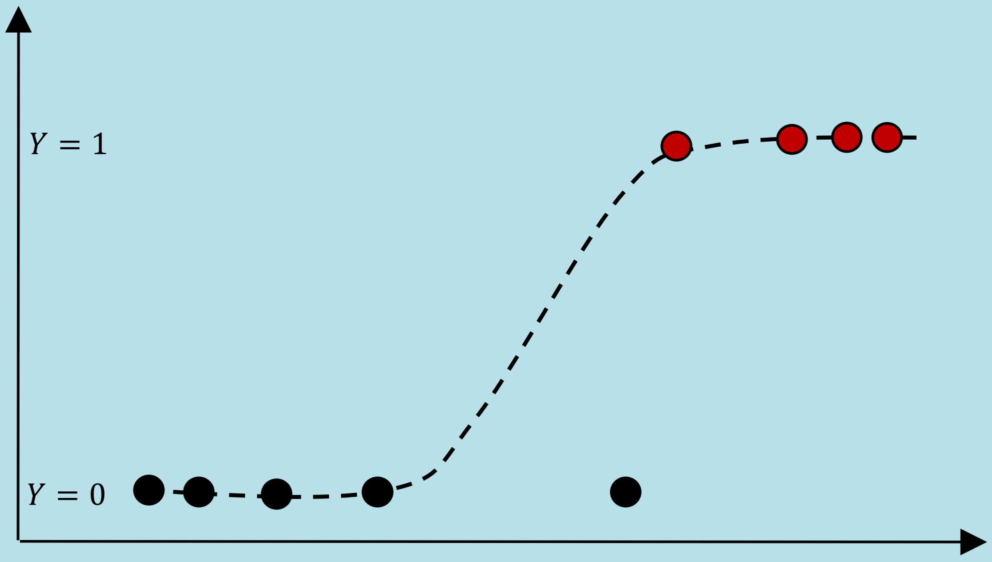
- Analogous to linear regression model
- Specify nonlinear function must be preselcted (risk of misspecification)
- Computationally more complex than linear regression
- Sensitive to outliers

Data requirements

- Complete numeric, independent predictors

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Logistic Regression



$$Y = \frac{1}{1 + e^{-z}}$$
$$z = \beta_0 + \beta_1 X_1$$

- Regression model in which the dependent variable is binary
- The model calculates the probability of the binary outcome variable taking on a 0/1 value.
- By determining a threshold the regression can be used for binary classification problems

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Evaluation of Regressions Models

Regression Error

Regression Error / Residuals: Measures how far off the predicted value is from the actual known value

Loss Function

$Error: e_t = y_t - \hat{y}_t$

$Absolute\ Error: |y_t - \hat{y}_t|$

$Squared\ Error: (y_t - \hat{y}_t)^2$

$Observation: y_t$

$Prediction: \hat{y}_t$

Accuracy Measures

Measure	Formula
Mean Absolute Error	$MAE = average(\lvert e_t \rvert)$
Mean Squared Error	$MSE = average(\lvert e_t^2 \rvert)$
Mean Absolute Percentage Error	$MAPE = 100 \cdot average(\left\lvert \frac{e_t}{y_t} \right\rvert)$
Mean Absolute Scaled Error	$MASE = \frac{MAE}{Q}$

Q: Scaling constant

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Decision Tree Learning and Continous Data (Classes or Features)

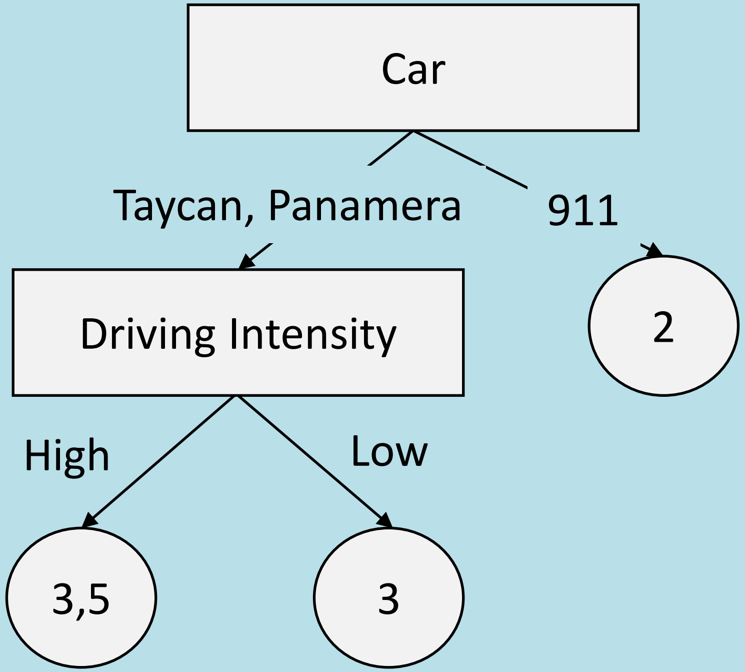
- Continuous features can be handled by allowing nodes to split a real valued feature into two ranges based on a threshold (e.g. length < 3 and length ≥ 3)

- Classification trees have discrete class labels at the leaves, regression trees allow real-valued outputs at the leaves.

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Basic Idea of Regression Trees

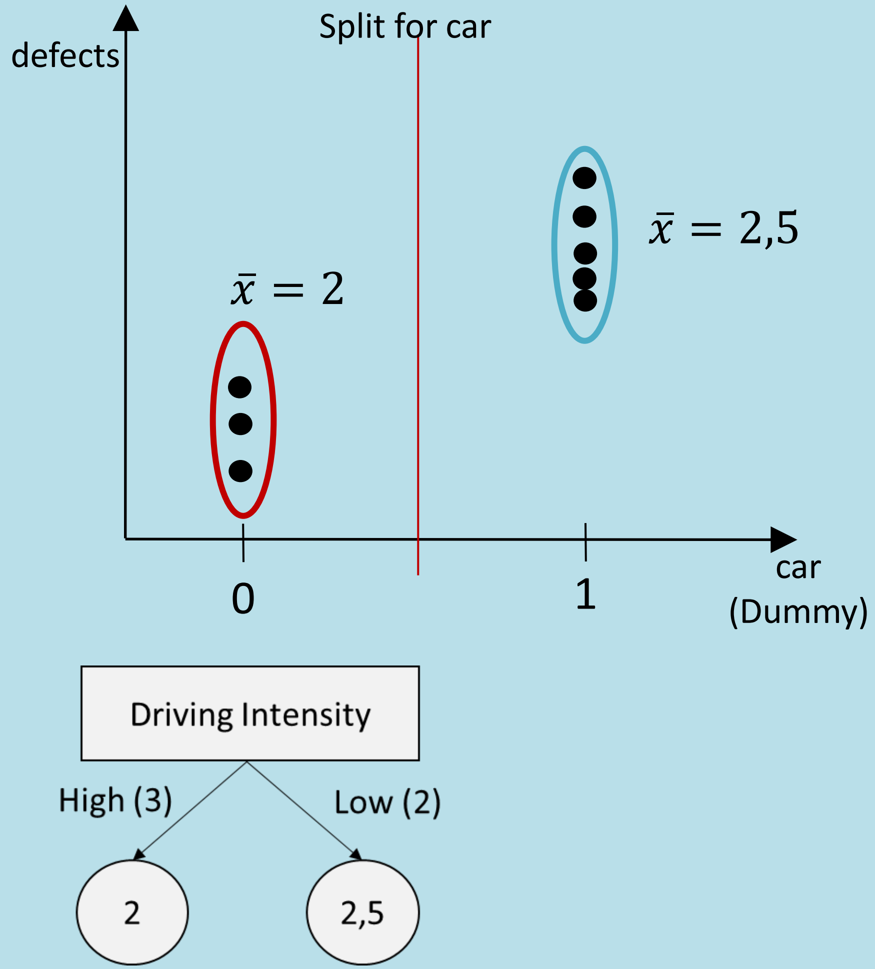
Concept similar to decision tree, but store continous value at each leaf representing the prediction



Car	Driving_intensity	Num_of_defects
Taycan	High	3
Taycan	High	4
911	Low	2
911	High	2
Panamera	Low	3
….	….	…

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Computation of Regression Trees



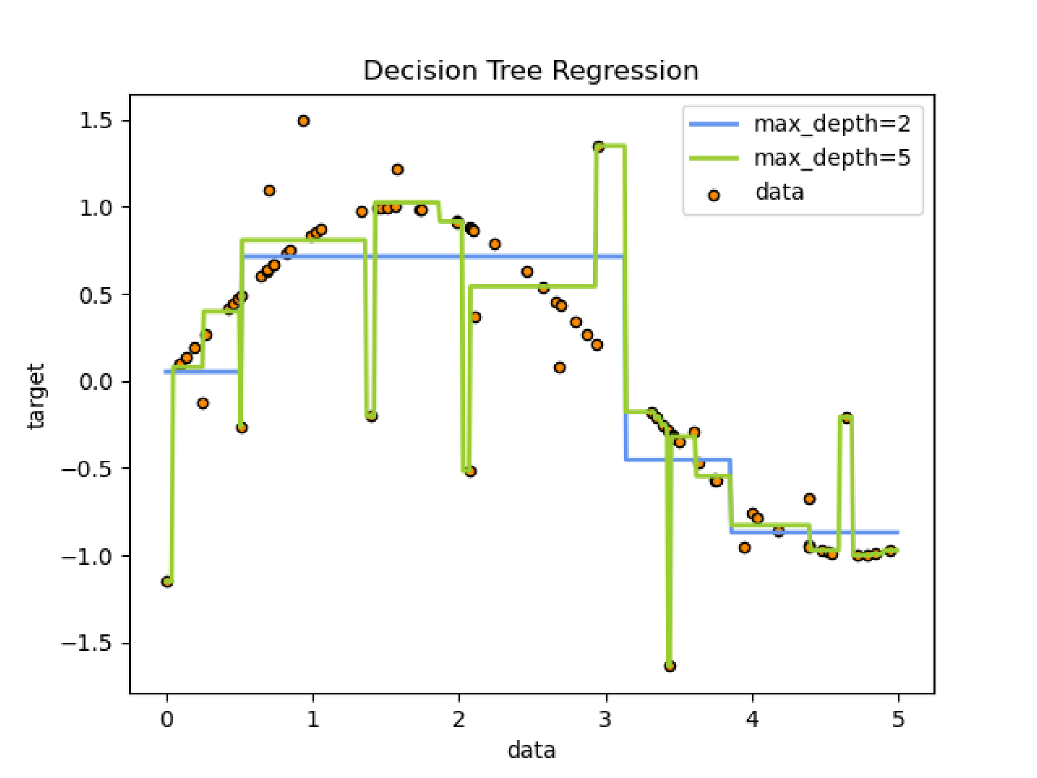
- Partition data into smaller sets and then fit a simple model (constant) for each subgroup
- Use reduction of standard deviation for attribute sleection

Algorithm: Regression Tree Induction

For each **node**
    For each **attribute**
        Calculate standard deviation of predicition in child nodes
        Chosse split that minimizes intrasubset variation/standard deviation

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Regression Trees with Python




```
>>> from sklearn import tree

>>> X = [[0, 0], [2, 2]]
>>> y = [0.5, 2.5]

>>> clf = tree.DecisionTreeRegressor()
>>> clf = clf.fit(X, y)
>>> clf.predict([[1, 1]])

array([0.5])
```

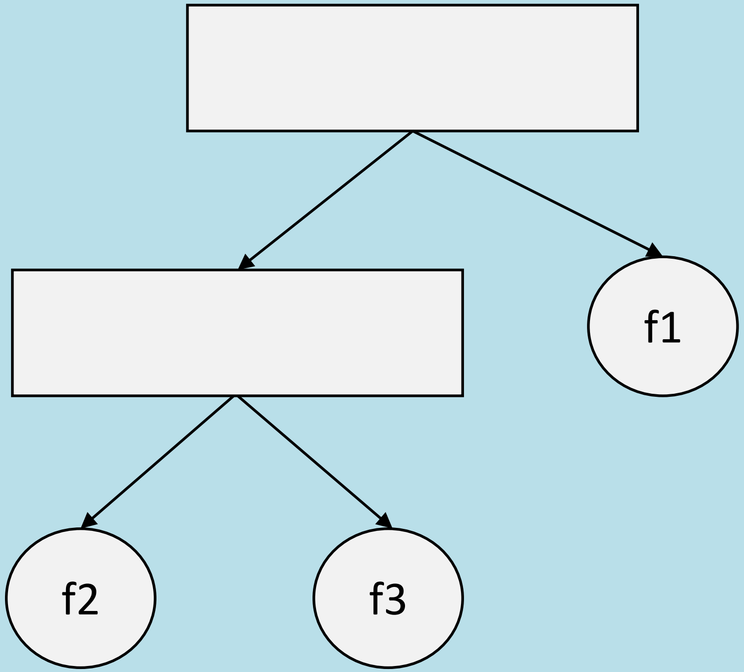
- As in the classification setting, the fit method will take as argument arrays X and y,
- y is expected to have floating point values instead of integer values

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.2 Basic Idea of Model Trees

- Concept similar to decision tree, but store regression models in the leaf node

- Basic algorithm named M5



Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

Your turn!

Task

Please construct and draw a decision tree to predict “`Buy = yes`” with minimal leaf size 1.

- Try out the scikit-learn `DecisionTreeClassifier()` to train your own decision tree in Python

Brand	Model	Engine	Buy
Porsche	Taycan	Electric	y
Mercedes	S-Class	Gasoline	n
Tesla	Model 3	Electric	n
VW	Golf	Gasoline	n
Porsche	Cayenne	Hybrid	y
Renault	Zoe	Electric	n
Porsche	911	Gasoline	n

Outline

Image source: ↗ Pixabay (2019) / ↗ CC0

▶ **What we will learn:**

- General concepts of AI modelling and what types of problems match which models
- Get an intuition for which model approach fits bets for a particular learning problem
- Know general problems of machine learning and how to optimize learning models

▶ **Duration:**

- 270 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 6.1 – 6.4

- Machine Learning in a nutshell: Build a statistical model to make predictions about your dataset (the reality) with as few **errors** as possible

- The prediction errors of your final model can be decomposed into two main subcomponents:
  - **Error due to Bias**: difference between the expected prediction and the true value
  - **Error due to Variance**: variability of a model prediction for a given data point

? What do you think: What is the relationship between error, bias and variance?

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Fitting Dilemma

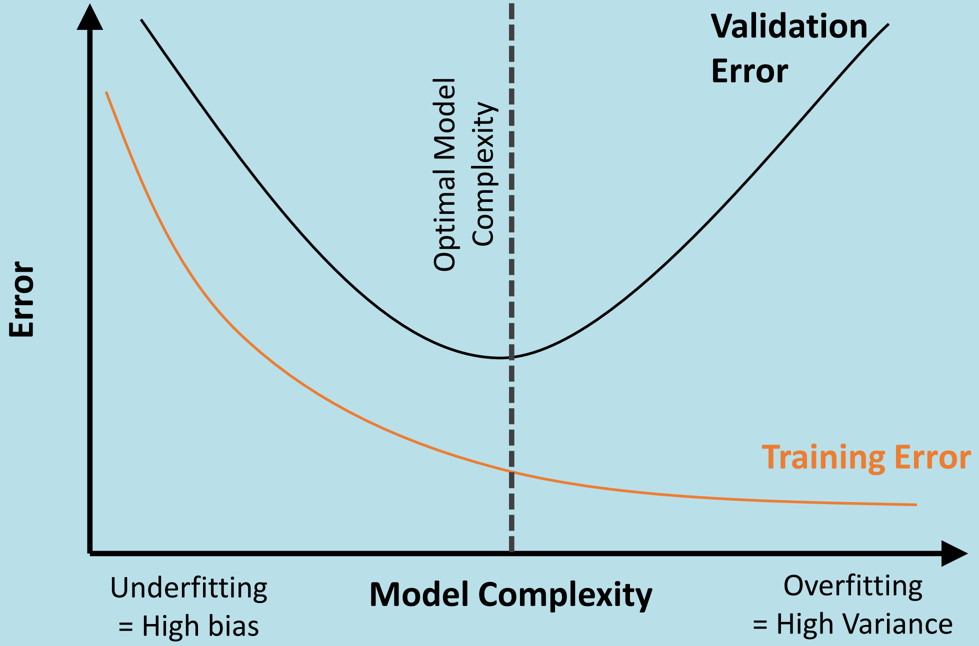
- We have seen bias and variance influence each other: the bias **decreases** with the complexity of the model, while the variance **increases** with the complexity of the model

- As a consequence, there are two opposite effects:
  - **Underfitting**: When the model has high bias and low variance, i.e. is too general (high total error)
  - **Overfitting**: When the model has low bias and high variance, i.e. is to specific (high total error)

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Variance Bias Trade-off

- We need to make a trade-off between "too specific" and "too general"



Bias = Unaccuracy of the model

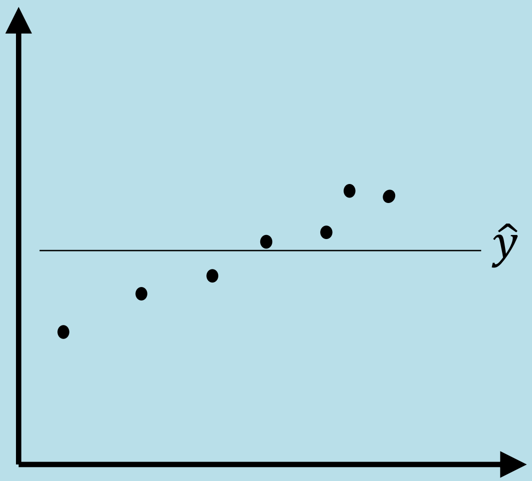
Variance = Imprecision of the model

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Bias, Variance and Model Complexity

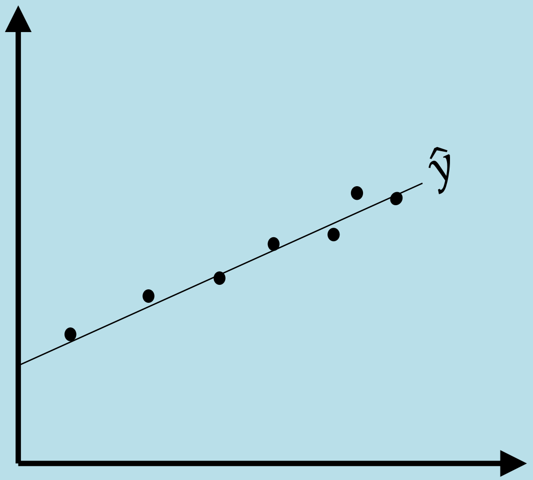
- We can vary our model's complexity by adjusting the number of features, number of parameters, model type etc.

Low model complexity



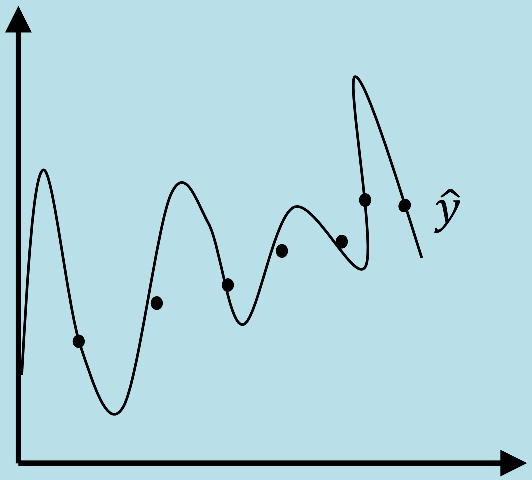
- High bias, low variance
- Too simple model

Medium model complexity



- Medium bias, Medium variance
- Best trade-off

High model complexity



- Low bias, high variance
- Overfitting!

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Occam's Razor

Image source: ↗ William of Ockham, from stained glass window at a church in Surrey. (2007) by Moscarlop from Wikimedia ↗CC-BY-SA-3.0

"Entia non sunt multiplicanda praeter necessitatem "

- More things should not be used than are necessary

- If you have two models with the same generalization error, the simpler model (with fewer nodes, elements, predictors, etc.) is preferable.

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

- Penalize to complex models
  - Example: linear regression
- Regularization term $\Omega(w)$
- Used to enforce small weights:
  - $\Omega(w) = \|w\|_2^2$; many weights 0: $\Omega(w) = \|w\|_1$
- Parameter λ controls strength of regularization

D **Regularization**

$$w^* = \arg\min_{w \in R^d} \sum_{i=1}^{n} \|w^T x_i - y_i\|_2^2 + \lambda\Omega(w)$$

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Pessimistic Error Rate $e_g(T)$

- **Example**: Decision tree
- To the sum of all misclassifications $e(t_i)$ at the leaf nodes above the training data one adds a malus ("penalty") $\Omega(t_i)$ for each leaf node t_i in the tree and refers the result to the number of observations in the training data.

D **Pessemistic Error Rate**

$$e_g(T) = \frac{\sum_{j=1}^{k}[e(t_i) + \Omega(t_i)]}{\sum_{i=1}^{k} n(t_i)} = \frac{e(T) + \Omega(T)}{N_i}$$

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Minimum Description Length Principle

- For each misclassification a measure is added to the binary coding to punish the complexity of the model.

$$cost(fit, data) = cost(data|fit) + cost(fit)$$

*Example: for 16 observations (4 bits) and 3 errors, cost (data | fit) = 3*4 = 12*

- Mostly used for decision trees

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Parameter Tuning

- Optimize your model's learning parameter (also known as hyperparameter) to control the learning process to reduce error.

- New modelling approach: Holdout Validation

- Then we can use the following tuning methods:
  - Random search
  - Grid search
  - Evolutionary search

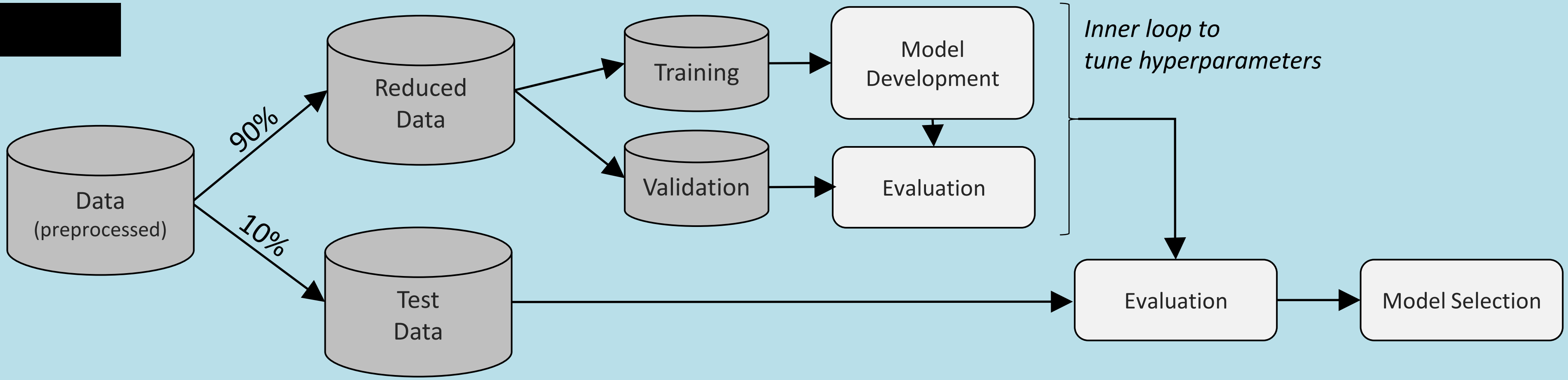
Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Holdout Validation

Procedure

- Hold out a part of your data to evaluate several candidate models and select the best one
- You train multiple models with various hyperparameters and select the best one based on a validation set (or development set). Train the best model on the full reduced data, your traditional training data (training + validation)
- Evaluate your final model on the test set to get an estimate of the performance

Visualization



Adapted from Géron, A. (2017)

6.3 Parameter Tuning with Random Search

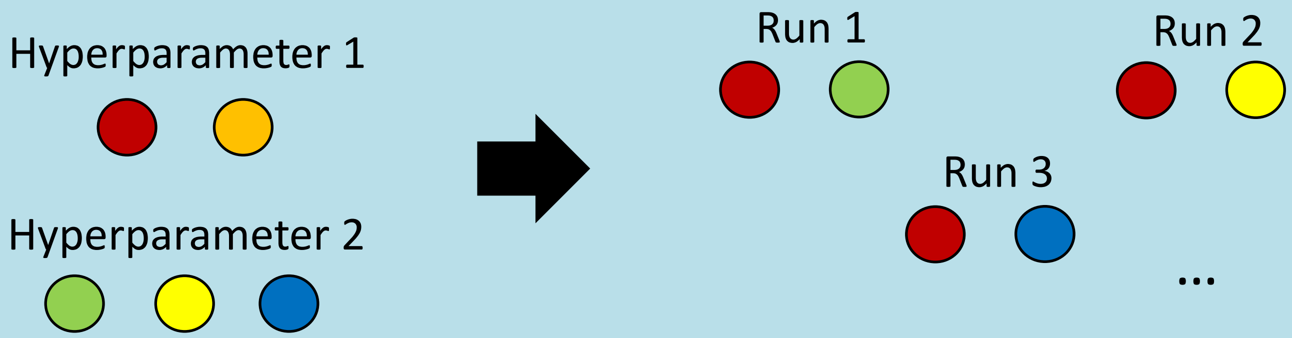
- Compute n random samples of hyperparameters, repeat a fixed number of times

- Efficient in high dimensional feature spaces

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Parameter Tuning with Grid Search

- Simple, systematic search for optimal hyperparameters
- Test all combinations for n hyperparameters and k attributes in a specific range which has to be set manually
- Computional expensive, results in a total of n^k combinations



Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Grid Search with Python

- The `sklearn` module provides many parameter tuning methods implementations like the `GridSearch`.

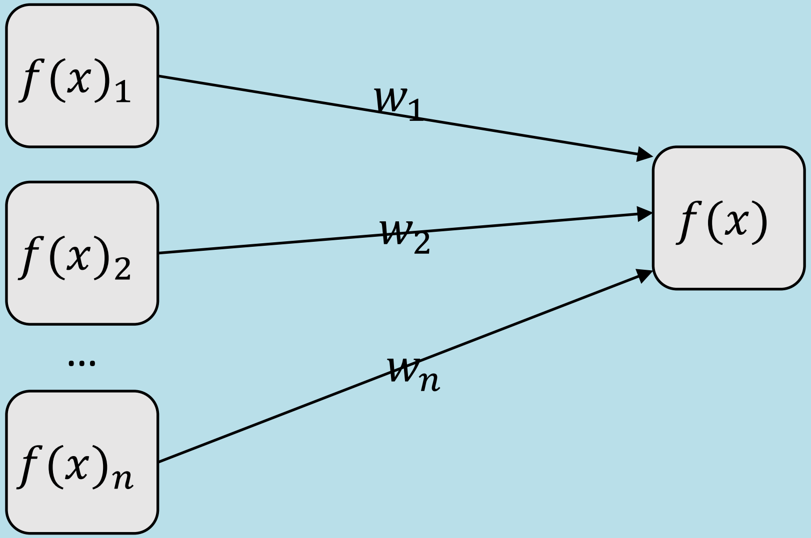
```
from sklearn.model_selection import GridSearchCV
param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2,
    3, 4]},
]
forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(housing_prepared, housing_labels)
```

Adapted from Géron, A. (2017)

6.3 Ensemble Methods

D **Ensembling**

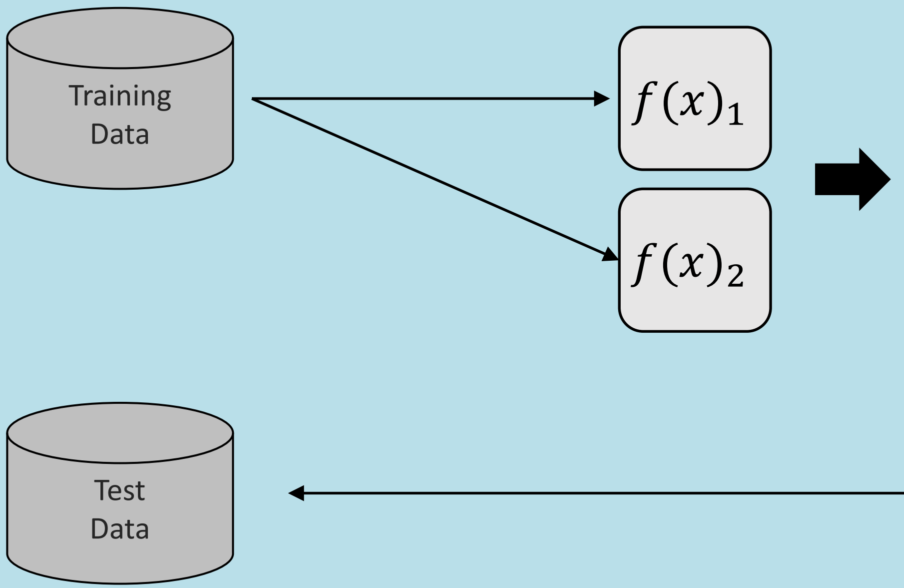
Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions (Dietrich TG, p.1, 2000)



- By combining *(mode, mean etc.)* the predictions from multiple models $f(x)_1, \ldots f(x)_n$ together we can make more accurate predictions than any individual model $f(x)_i$

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016); Dietrich T.G. (2000)

6.3 Example: Random forest algorithm with i = 2



$$f(x) = \frac{f(x)_1 + f(x)_2}{2}$$

- The random forest algorithm makes use of the concept of ensemble learning
- It predicts by averaging the different learns (ensemble of weak learners)

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016); Dietrich T.G. (2000)

6.3 Ensemble Methods

- Ensemble methods try to obtain better predictive performance by combining different learning algorithms

- In the next step we will take a look at different methods for such model combinations:
  - Bagging: Train independent model ensembles
  - Boosting: Train dependent model ensembles

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Bagging

Training

- In each iteration
  - Randomly sample with replacement N samples from training set
  - Train a chosen “base” model (e.g. decision tree) on the samples

Testing

- For each test example
  - Start all trained base models
  - Predict by combining results of all trained models
  - Voting: Regression → Averaging, Classification → majority vote

Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

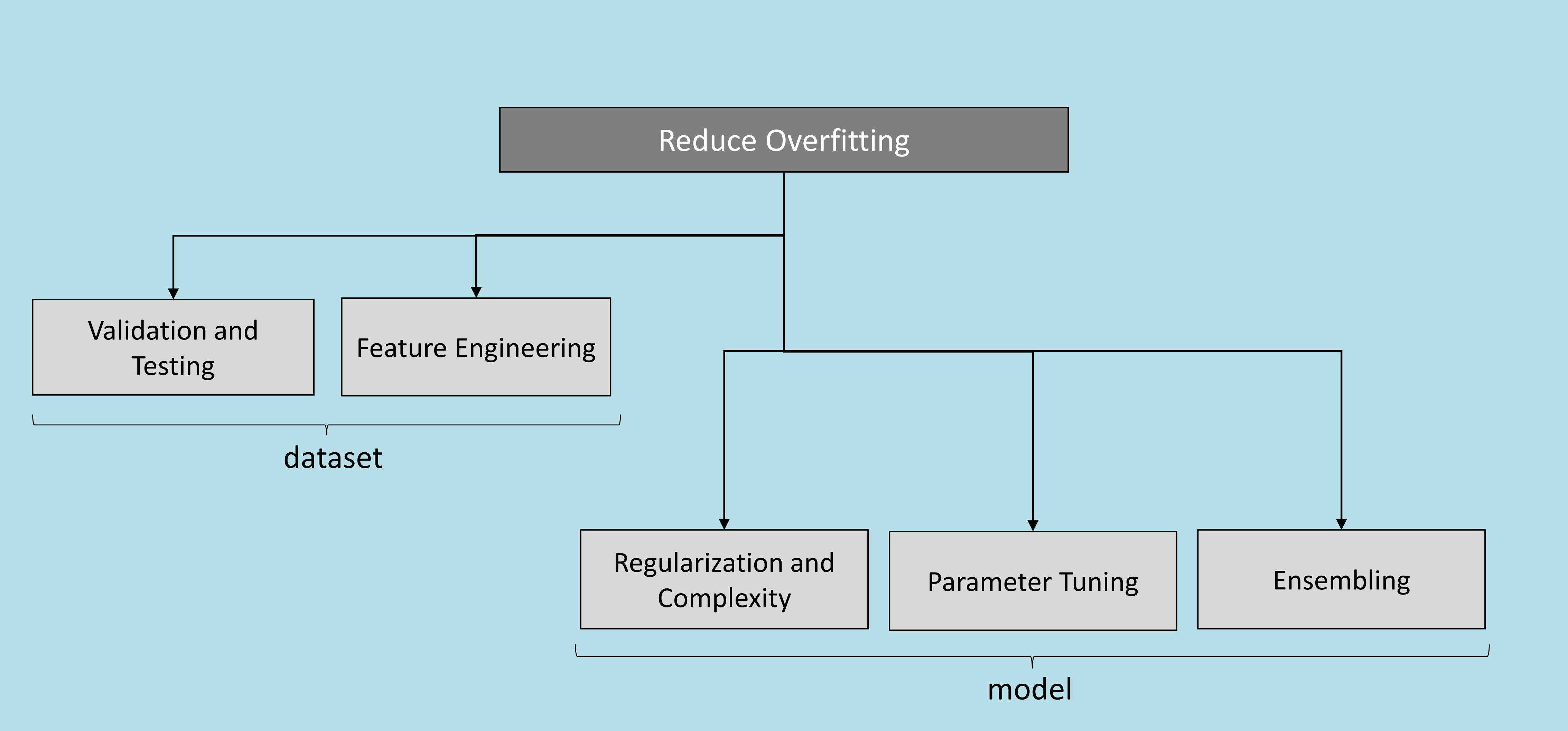
Training

- Train a sequence of T base models on T different sampling distributions defined over the training set (D)
  - A sample distribution $D(t)$ for building the model t is constructed by modifying the sampling distribution $D(t-1)$ from the $(t-1)th$ step
  - Example classified incorrectly in the previous step receive higher weights in the new data (attempts to cover misclassified samples)

- Classify according to the weighted majority of classifiers

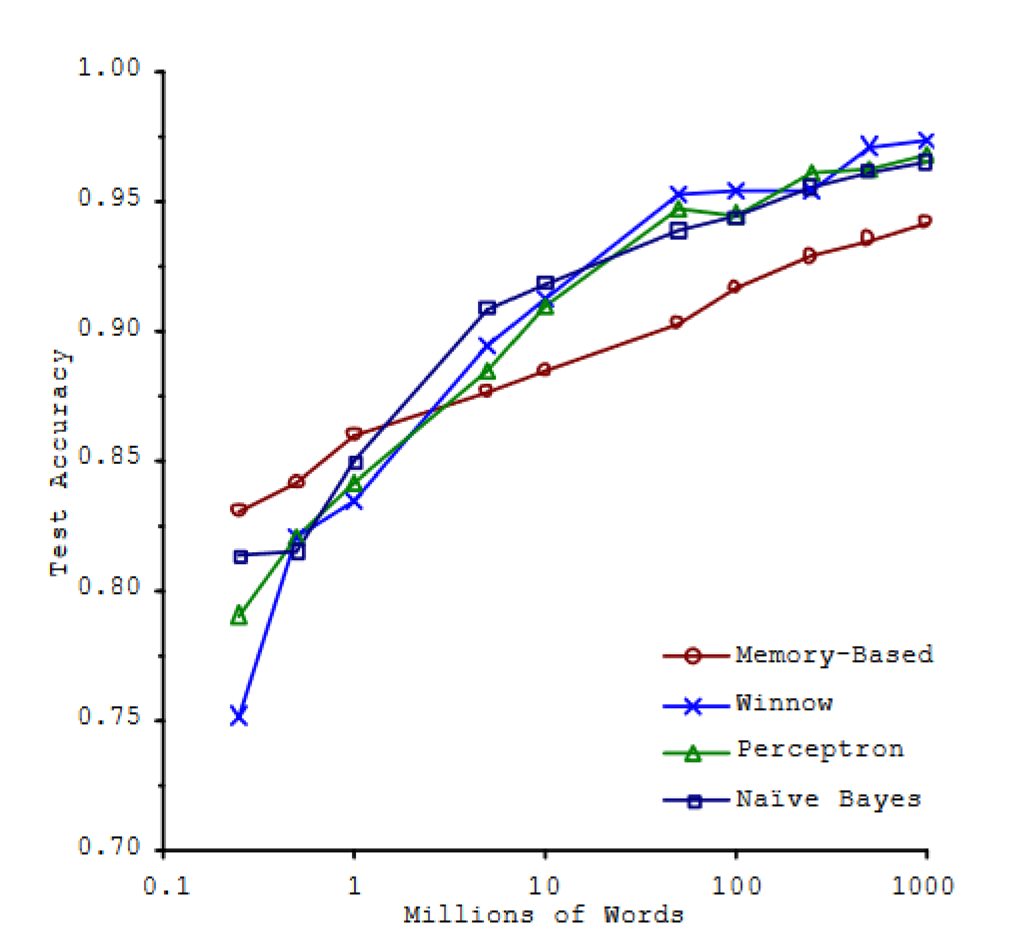
Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 Summary: How to handle Overfitting



Adapted from Géron, A. (2017); Rusell, S., & Norvig, P. (2016)

6.3 The Unreasonable Effectiveness of Data (2001)



EXPERT OPINION

Contact Editor: **Brian Brannon**, bbrannon@computer.org

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences"[1] examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.[2] Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

One of us, as an undergraduate at Brown University, remembers the excitement of having access to the Brown Corpus, containing one million English words.[3] Since then, our field has seen several notable corpora that are about 100 times larger, and in 2006, Google released a trillion-word corpus with frequency counts for all sequences up to five words long.[4] In some w… is a step backwards from the Brow… from unfiltered Web pages an… …ete sentences, spelling er- ro… …nd all sorts of other er- ro… pa… tim… drawb… Web-derive… lions of links, vid… actions—captures

behavior. So, this corpus could serve as the basis of a complete model for certain tasks—if only we knew how to extract the model from the data.

Learning from Text at Web Scale

The biggest successes in natural-language-related machine learning have been statistical speech recognition and statistical machine translation. The reason for these successes is not that these tasks are easier than other tasks; they are in fact much harder than tasks such as document classification that extract just a few bits of information from each document. The reason is that translation is a natural task routinely done every day for a real human need (think of the operations of the European Union or of news agencies). The same is true of speech transcription (think of closed-caption broadcasts). In other words, a large training set of the input-output behavior that we seek to automate is available to us *in the wild*. In contrast, traditional natural language processing problems such as document classification, part-of-speech tagging, named-entity recognition, or parsing are not routine tasks, so they have no large corpus available in the wild. Instead, a corpus for these tasks requires skilled human annotation. Such annotation is not only slow and expensive to acquire but also difficult for experts to agree on, being bedeviled by many of the difficulties we discuss later in relation to the Semantic Web. The first lesson of Web-scale learning is to use available large-scale data rather than hoping for annotated

8

The unreasonable Effectiviness of Data:

Online available from ↗ [Institute of Electrical and Electronics Engineers (ieeexplore)]

Adapted from Banko, M., & Brill, E. (2001, July); Halevy, A., Norvig, P., & Pereira, F. (2009); Rusell, S., & Norvig, P. (2016) |Image source: ko, M., & Brill, E. (2001, p. 2);

Your turn!

Task

One of your colleagues says “I heart that boosting relays on dependent model ensembles. But I don’t get it why!”. Is this true? Why are the models dependent from each other?

Outline

Image source: ↗ Pixabay (2019) / ↗ CC0

▶ **What we will learn:**

- General concepts of AI modelling and what types of problems match which models
- Get an intuition for which model approach fits bets for a particular learning problem
- Know general problems of machine learning and how to optimize learning models

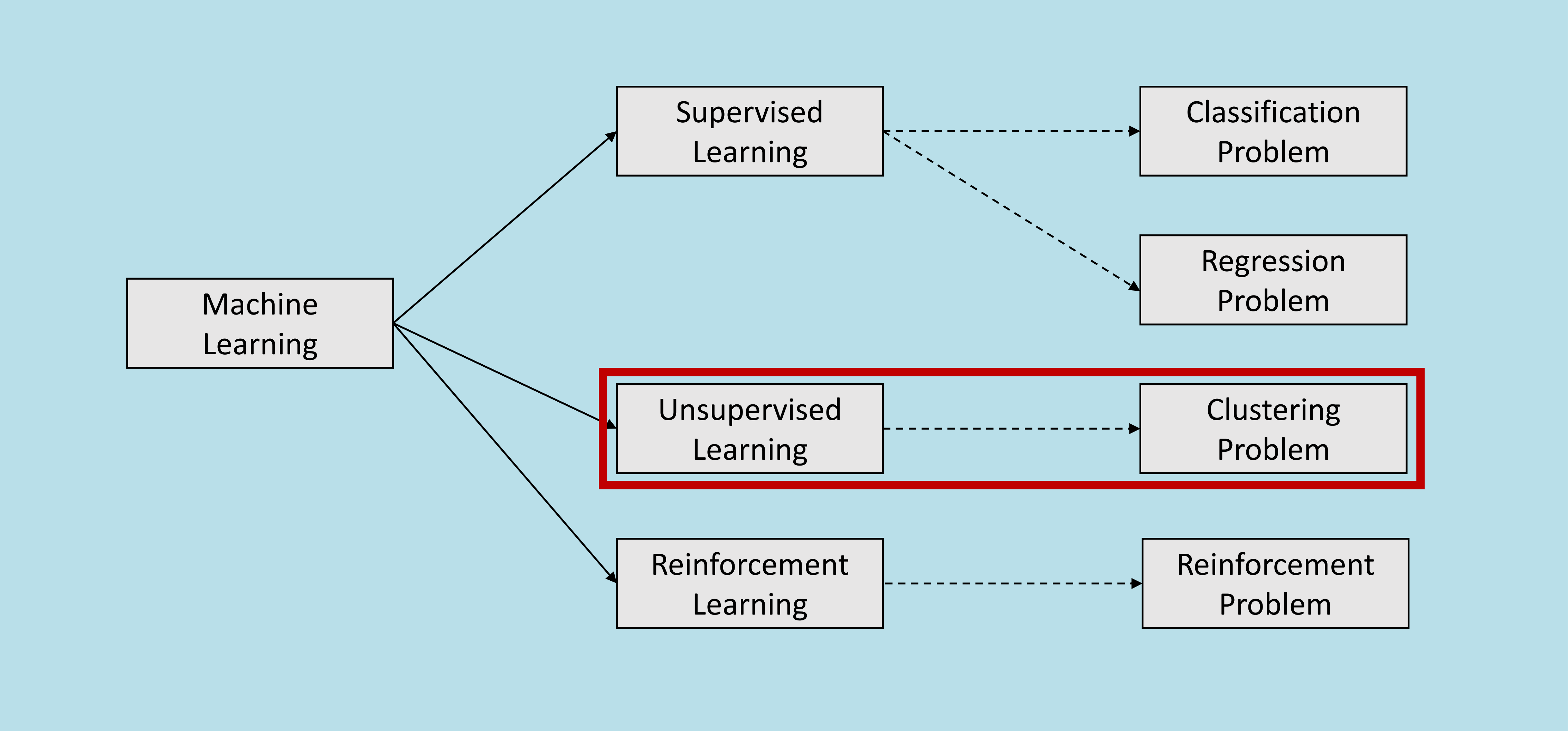
▶ **Duration:**

- 270 min + 90 (Lectorial)

▶ **Relevant for Exam:**

- 6.1 – 6.4

6.4 Problem Types in Machine Learning (High-Level)



6.4 What is Clustering about?

- Grouping a set of data objects into groups of data objects, similar to one another within the same group, and dissimilar to the objects in other groups

- Clustering = unsupervised “classification“ (no predefined classes)

- Typical usage in artificial intelligence:
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

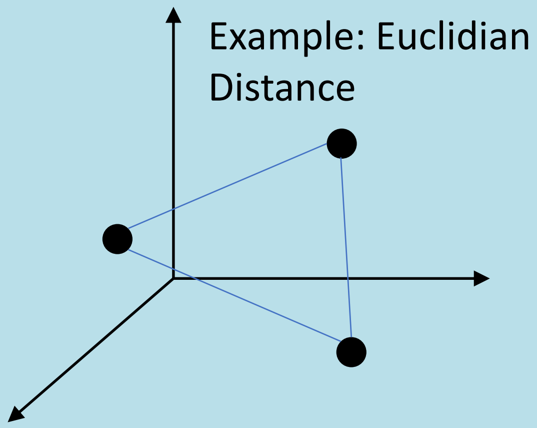
6.4 What is Similarity?

Image Source: ↗ Pixabay (2019) / ↗ CC0

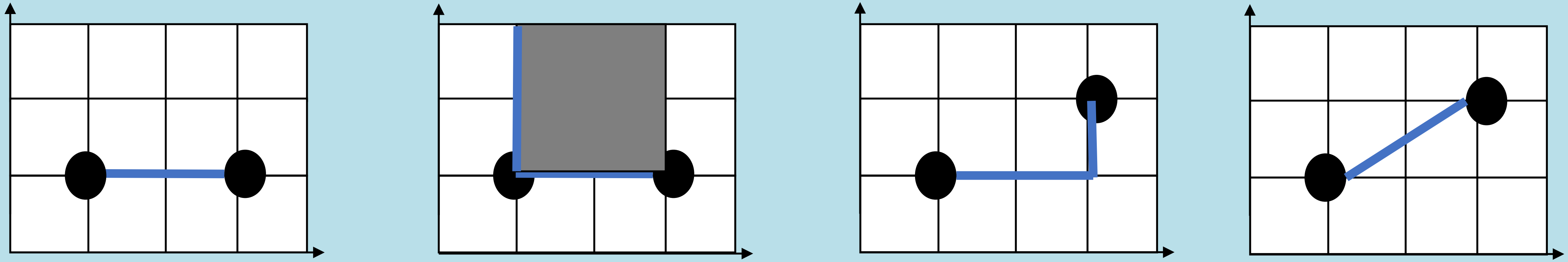
6.4 Definition of a Metric

D

- **Nonnegativity**: $\forall xy: d(x,y) \geq 0$
- **Symmetry**: $\mathrm{d(x,y) = d(y,x);\ s(x,y) = s(y,x)}$
- **Triangle inequity**: $\mathrm{d(x,z)} \leq d(x,y) + d(y,z)$



6.4 Most Popular Metric: Minkowski Distance



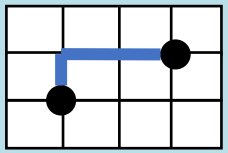
6.4 L1 and L2 Norm

$$d(x,y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r\right)^{\frac{1}{r}}$$

- Generalization of Euclidian Distance

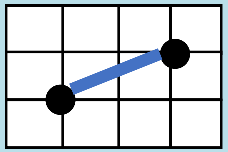
Mannhatten Distance

- $r = 1$
- Named Cityblock distance, or L_1-norm
- For binary attributes also named Hamming Distance
- Sum of distance about all dimensions 1…n



Euclidian Distance

- $r = 2$
- L_2-norm
- For numeric variables (ordinal or rational)
- Describes the geometric distance between two points



Supremum

- $r = \infty$
- L_∞-norm
- Biggest difference in the dimensions

6.4 Difference L1 and L2 Norm

- L_1: Impact of a difference proportial to difference itself

- L_2: Higher relative impact of larger distances!
  - **Example**: $x = (3,3), y = (4,5)$
  - L1 norm distance: $|3-4| + |3-5| = 1 + 2 = 3$
  - L2 norm distance: $\sqrt{|3-4|^2 + |3-5|^2} = \sqrt{1 + 4} = 2.2$

- L_2 norm is smaller than L_1 but the individual difference has more relative weight

6.4 Beyond Minkowski

Binary Objects

- Simple Matching Coefficient (SMC)
- Jaccard Coefficient

Correlation

- Pearson
- Speerman

Vectors

- Simple Matching
- Cosinus Coefficient
- Dice Coefficient
- Tanimoto Coefficient
- Overlap Coefficient

You find a comprehensive overview of further similarity concepts at „Cha, S. H. (2007). Comprehensive survey on distance/similarity measures between probability density functions"

6.4 Fundamental Clustering Approaches

Clustering

Partitioning

- Subdivide the data sets into a set of k groups, where k is the number of groups pre-specified
- Find k partitions, minimizing some objective function
- Find mutually exclusive clusters of spherical shape
- Ignores outliers

Density-based

- Find arbitary shaped clusters
- Based on density and reachability
- Assumes the presence of outliers (sparse regions)

Hierarchical

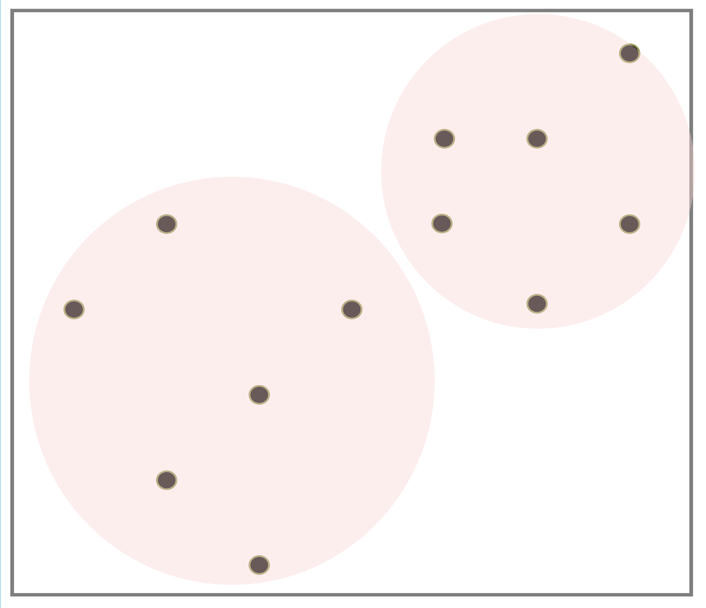
- Finds hierarchies of clusters
- Based on distance
- Can detect outliers (singleton clusters at cut)

Grid-based

- Improves speed of partitioning clustering throug grids
- Based on grids at different scales
- Assumes the presence of outliers (non dense grid calls)

6.4 Partitioning Clustering: K-Means

▶ Forming groups of objects in a way that the same group (clusters) are more similar to each other than to those in other groups



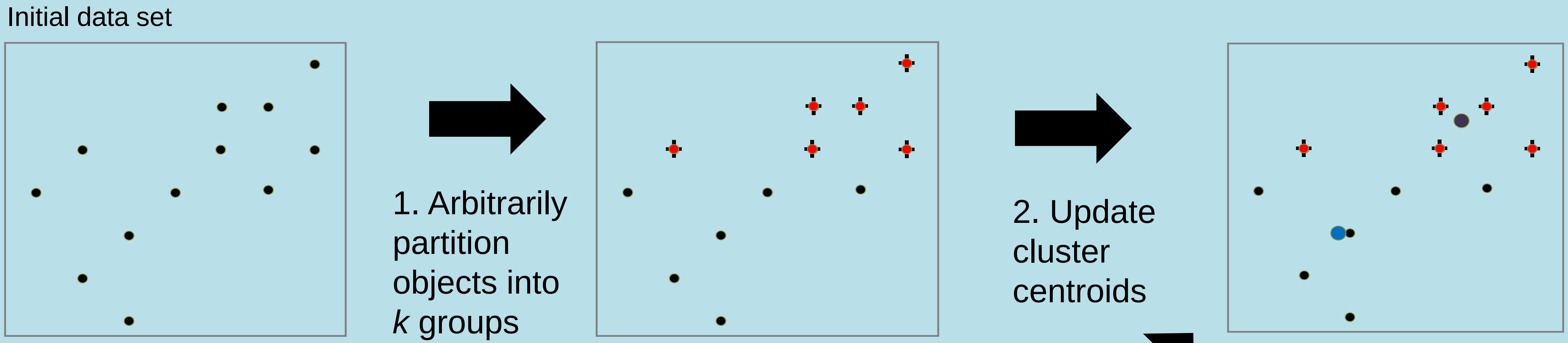
- Most popular clustering technique today
- Based on centroids, clusters instances based on the closest centroid
- Generally used for exploratory rather than confirmatory analysis

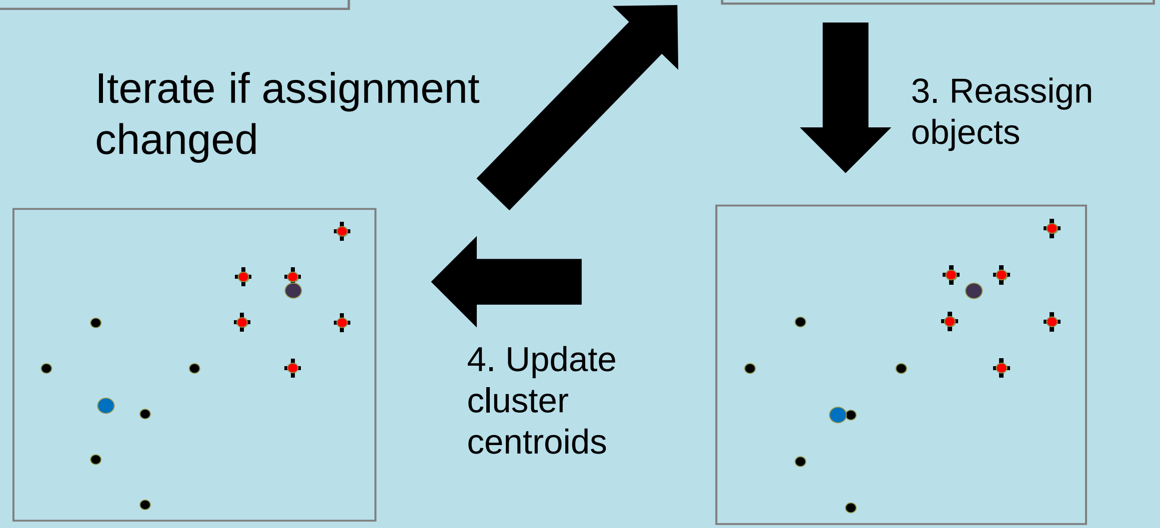
Example
- Cluster different log-files etc. for further analysis

\+ Only one parameter (k), execution is very fast, resulting clusters are convex

\- Applicable only when mean is defined, outliers have a strong influence on the result

6.4 K-means procedure (with k = 2)





- **Terminate if assignment does not change** or if no significant reduction in **total squared distances**
- → $C_1, \ldots, C_k$

6.4 KMeans in Python

KMeans()

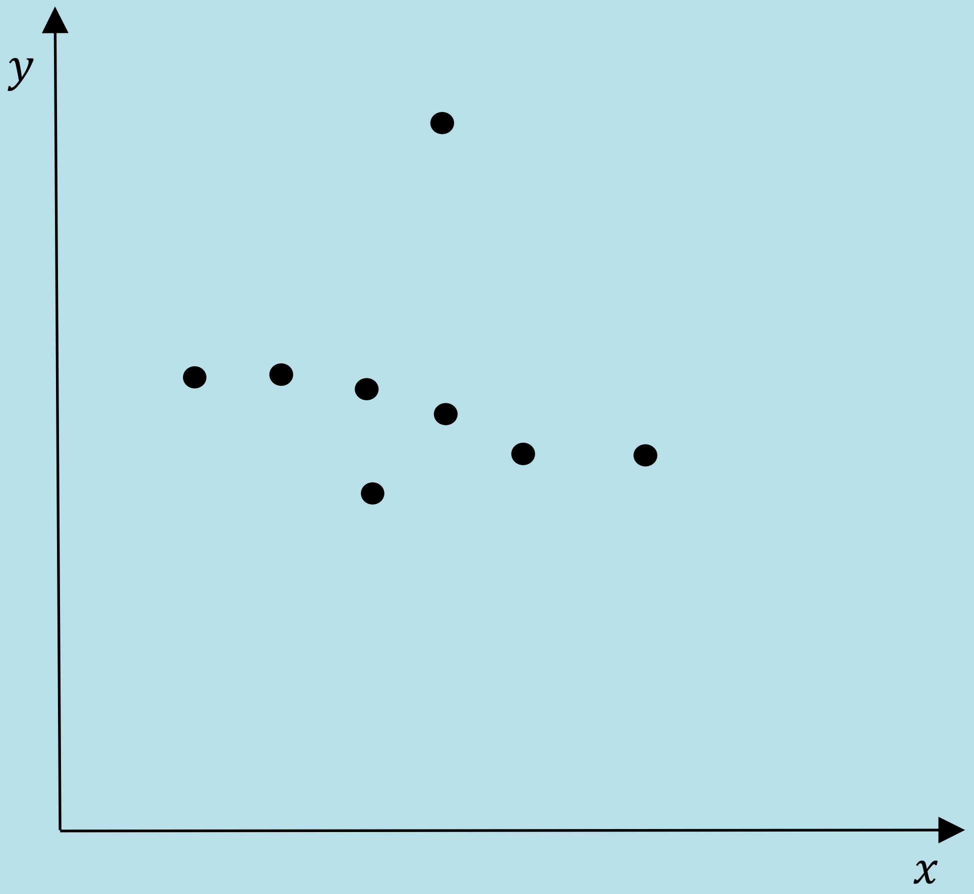
```
KMeans(n_clusters=8, n_init=10, max_iter=300
```

Parameters

`n_clusters`	The number of clusters to form as well as the number of centroids to generate.
`n_init`	Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.
`max_iter`	Maximum number of iterations of the k-means algorithm for a single run.

6.4 Density-based Clustering

▶ Find arbitary shaped clusters based on density and reachability

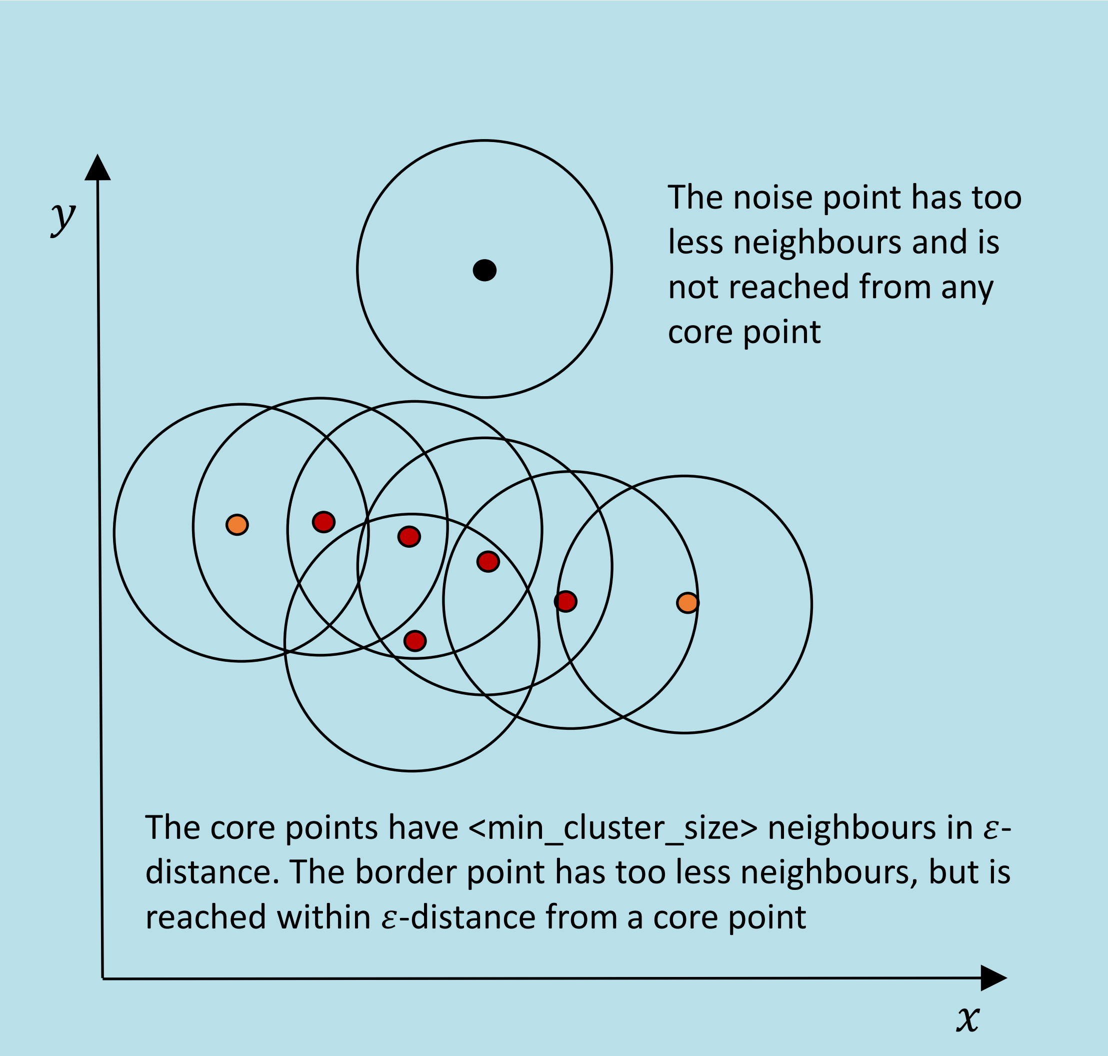


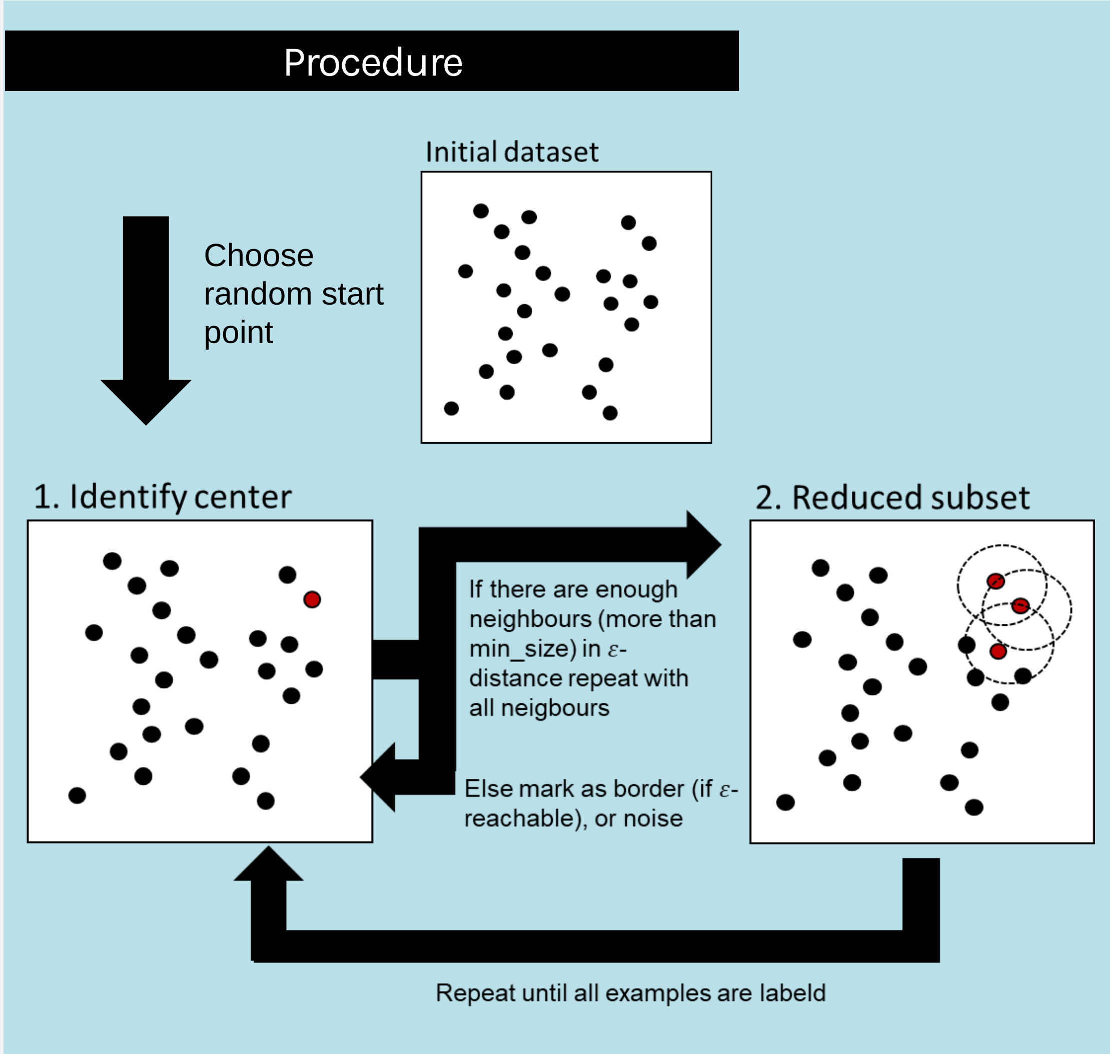
- Based on ε-rechability. A point is ε-reachable from another one, if they lie not more than ε-distance away
- There are core points, border points and noise points

\+ Number of clusters can be unspecified. Clusters are arbitrary-shaped. Can handel noise (assumes there is), hence very outlier resistance

\- Clusters can not have different densities, parameter tuning for ε is difficult

6.4 Density-based Clustering (DBSCAN)





6.4 DBSCAN in Python

DBSCAN()

```
DBSCAN(eps, min_samples)
```

Parameters

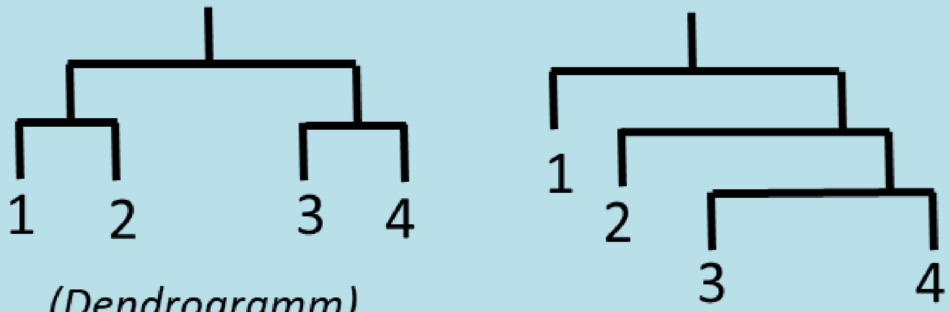
`eps`	The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.
`min_samples`	The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

6.4 Hierarchical Clustering

▶ Create a hierarchical decomposition of the set of objects

Method of cluster analysis which seeks to build a hierarchy of clusters

- **Agglomerative**: "bottom up" approach; each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy
- **Divisive**: "top down" approach; all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy



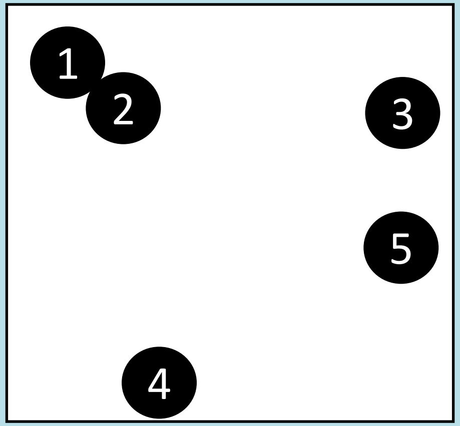
(Dendrogramm)

\+ Do not have to assume any particular number of clusters, any desired number of clusters can be obtained by ‘cutting’ the dendrogram

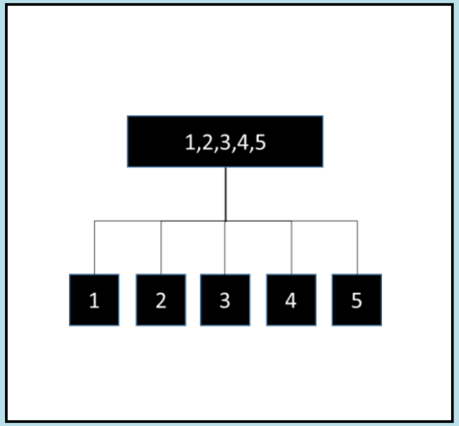
\- Can not handle complex shapes, new points can not be added without recalculation of the model, single-link-effect (“chains” depending on the metric)

6.4 Agglomerative Clustering

Initial dataset

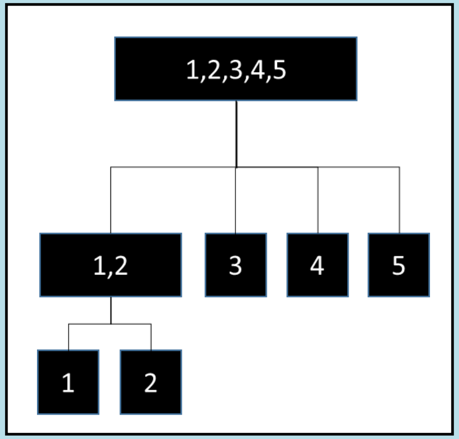


Define distance measure and put every observation in a single cluster

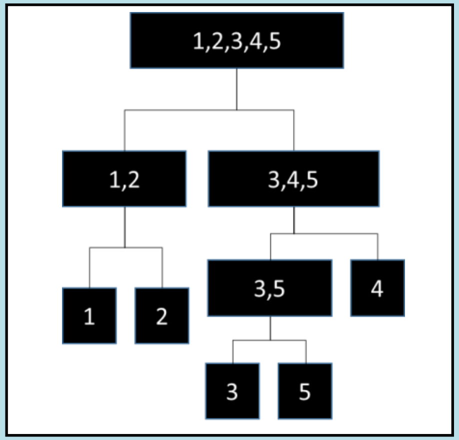


Until only one cluster left:

1. Select the two clusters with minimum distance
2. Join these two clusters



- Define distance measure and linkage criteria



6.4 Linkage Criteria

Single

Minimal distance between two elements of each cluster

$$D(X,Y) := \min_{x \in X,\, b \in B} \{d(x,y)\}$$

Complete

Maximum distance between two elements of each cluster

$$D(X,Y) := \max_{x \in X,\, b \in B} \{d(x,y)\}$$

Average

Average distance between two elements of each cluster

$$D(X,Y) := \frac{1}{|X||Y|} \sum_{x \in X,\, b \in B} \{d(x,y)\}$$

Ward

Growth of cluster variance

$$D(X,Y) := \frac{d(\bar{a},\bar{b})^2}{\frac{1}{|A|}+\frac{1}{|B|}}$$

$\bar{a}$, $\bar{b}$ are cluster centroids

6.4 `AgglomerativeClustering` in Python

AgglomerativeClustering()

```
AgglomerativeClustering(n_cluster, linkage,)
```

Parameters

`n_clusters`	The number of clusters to find. It must be None if distance_threshold is not None
`linkage`	Which linkage criterion to use. The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.

Overview of Clustering Methods

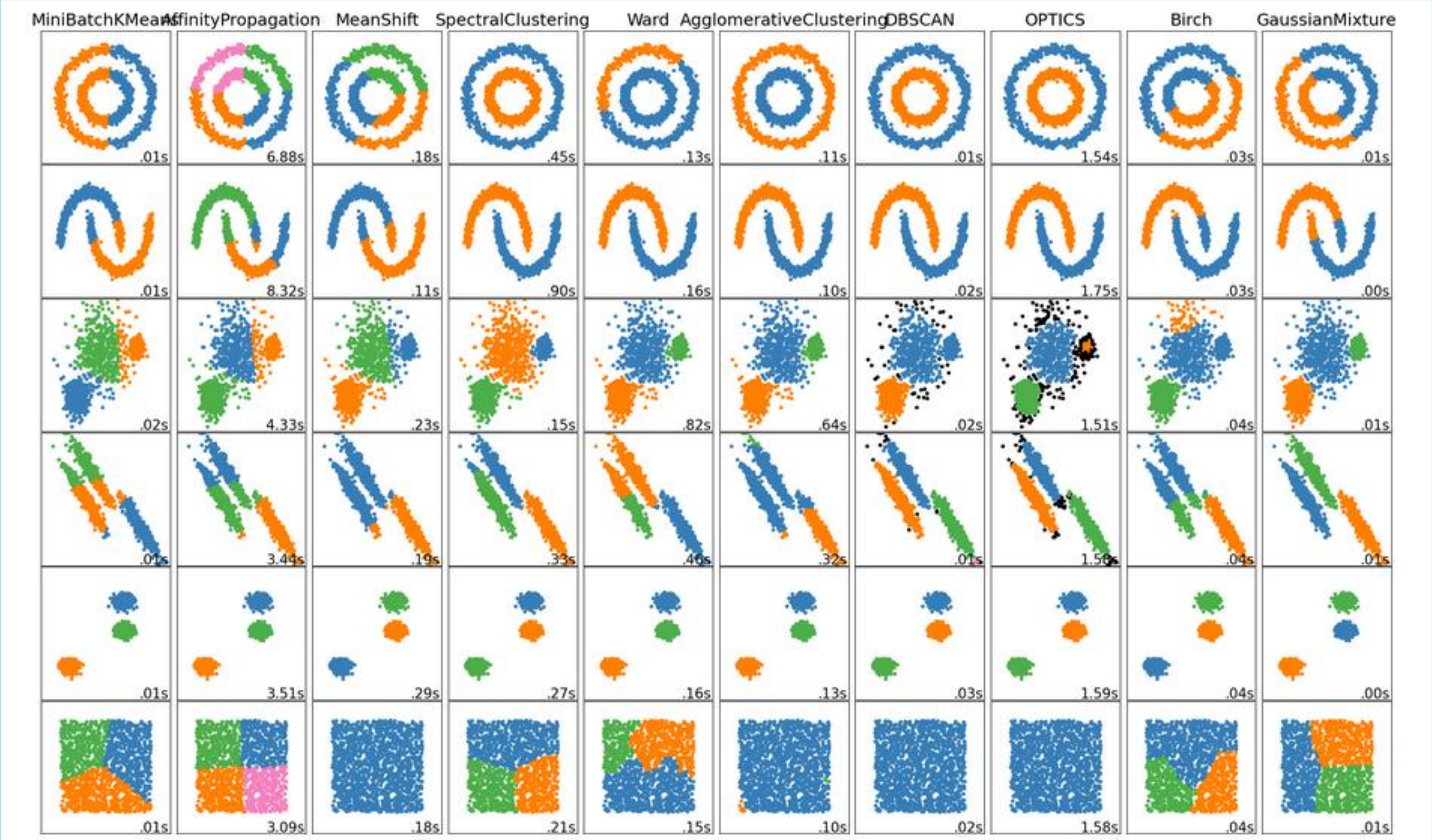


Image source: Scikit-learn ↗ Clustering (2021)

6.4 Evaluation of Clustering Models: Silhouette-Value

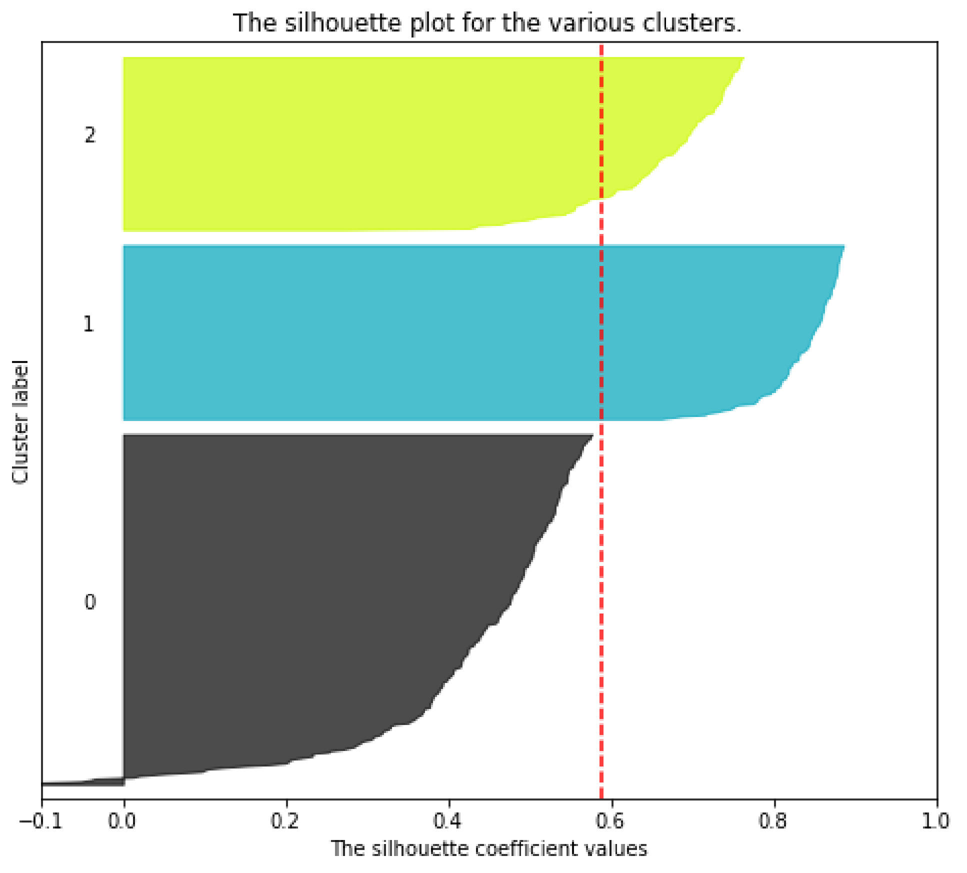
Silhouette Value [-1,1]:

D

$$S(v_k) = \frac{b(v_k) - a(v_k)}{\max\{a(v_k), b(v_k)\}}$$

- $a(v_k)$: mean dissimilarity of the silhouette value to all other elements of the same clusters
- $b(v_k)$: mean dissimilarity of the silhouette value to all other elements to the nearest cluster
- A negative silhouette value indicates that an element would better fit into another cluster
- The greater the value the better the model

```
Import sklearn
silhouette_score(X, cluster_labels)
```

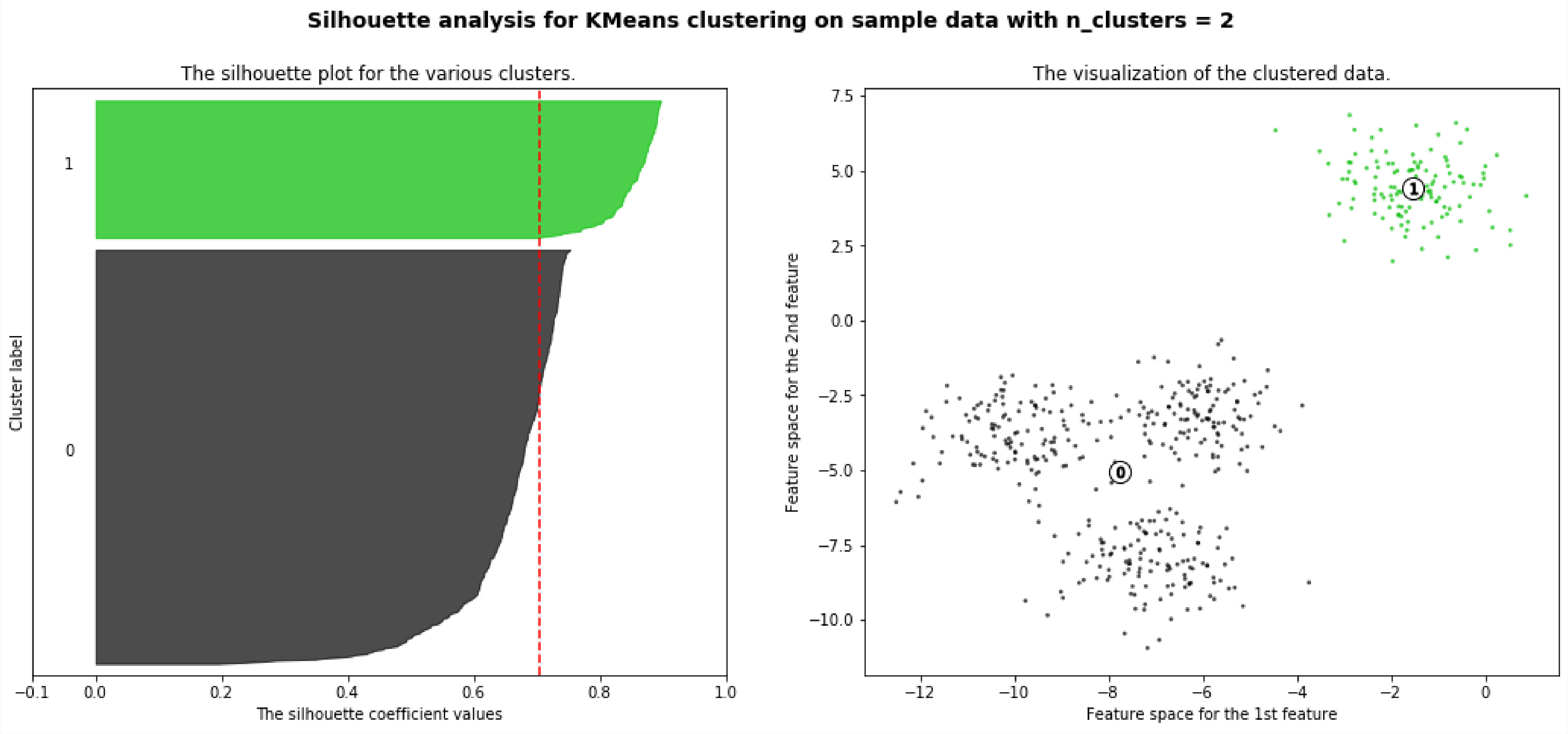


Silhouette plot: Visualizes the silhouettes for all data and the mean value

Adapted from Géron, A. (2017)

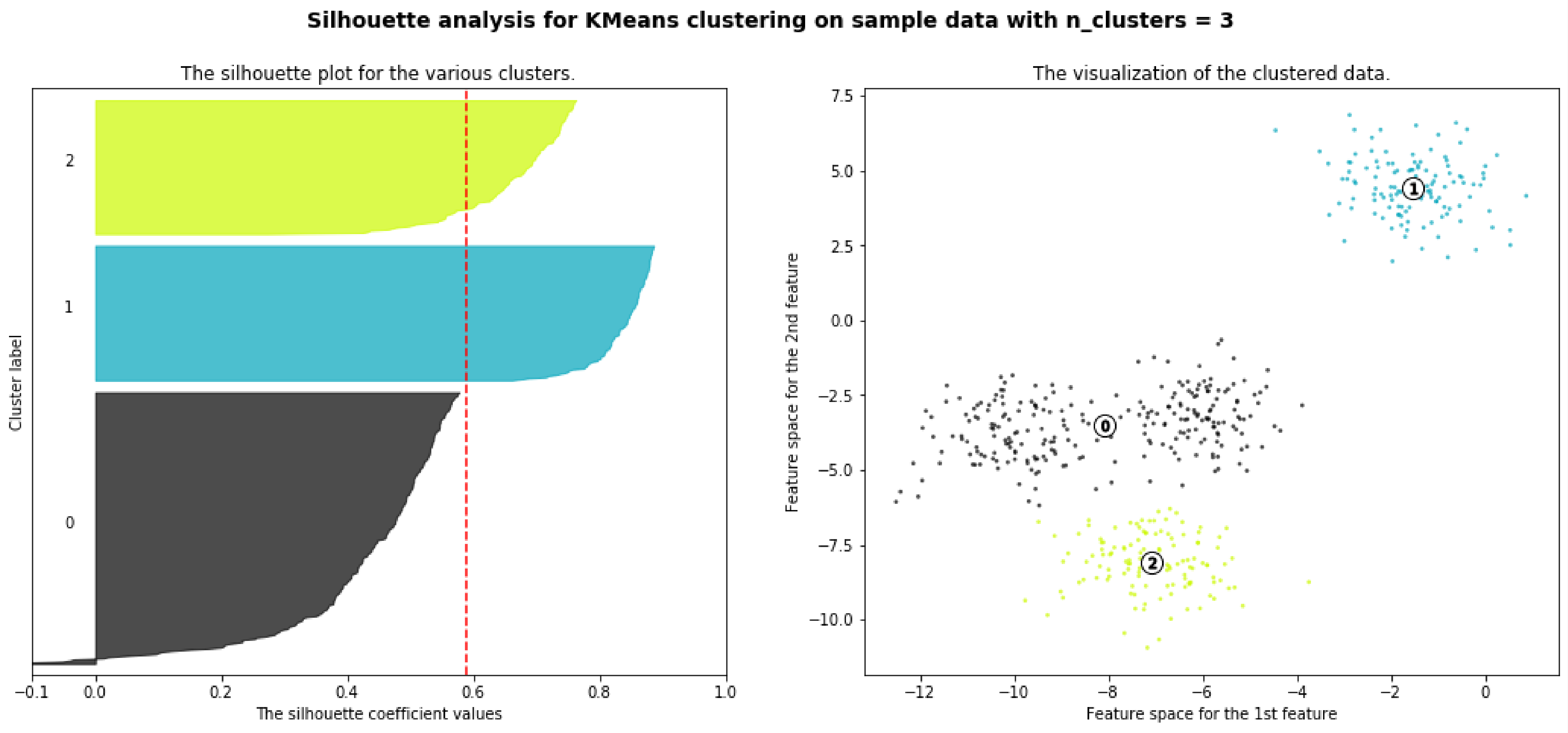
6.4 Compare Silhouette-Values with Python

```
For n_clusters = 2 The
average silhouette_score
is : 0.7049787496083262
```



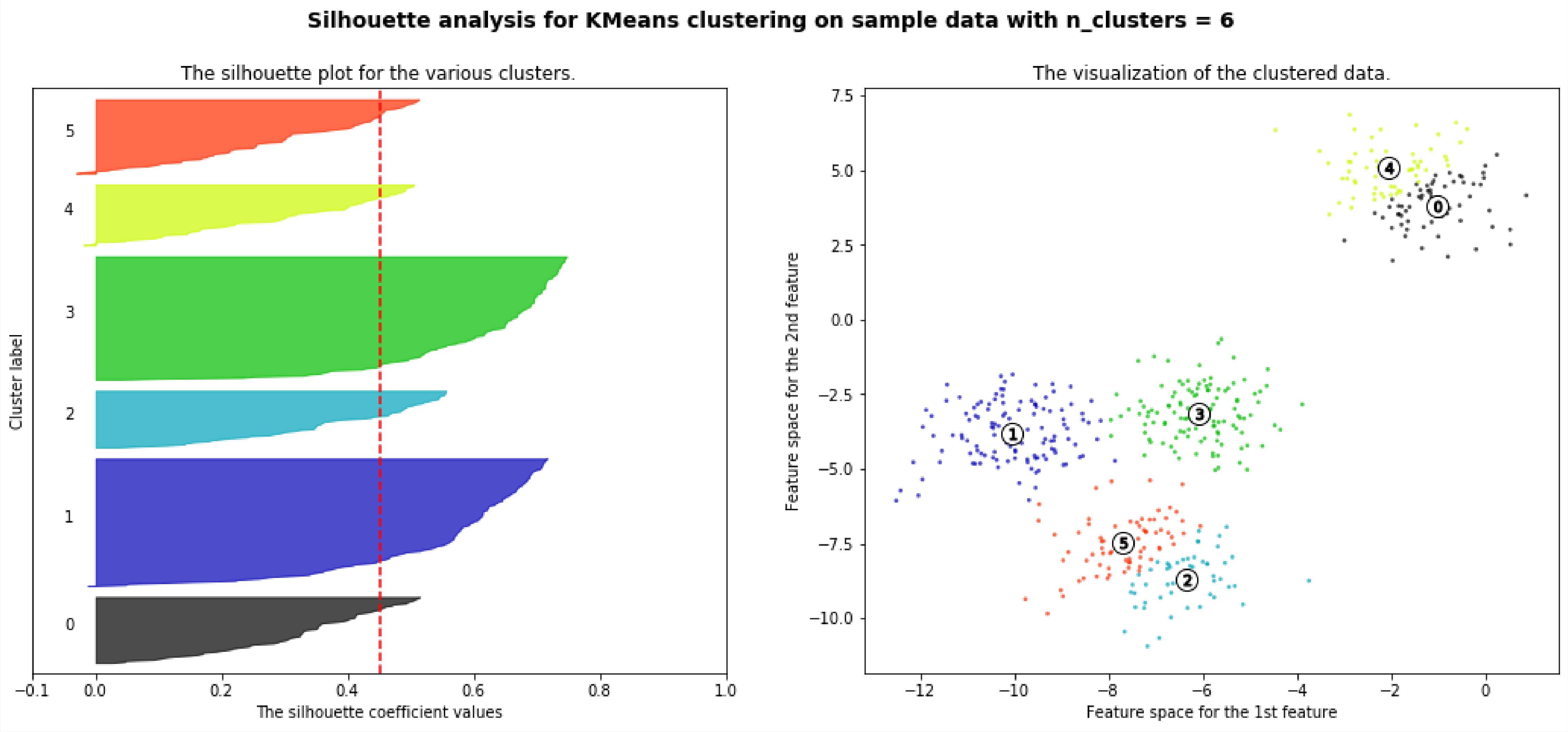
6.4 Compare Silhouette-Values with Python

```
For n_clusters = 3 The
average silhouette_score
is : 0.5882004012129721
```



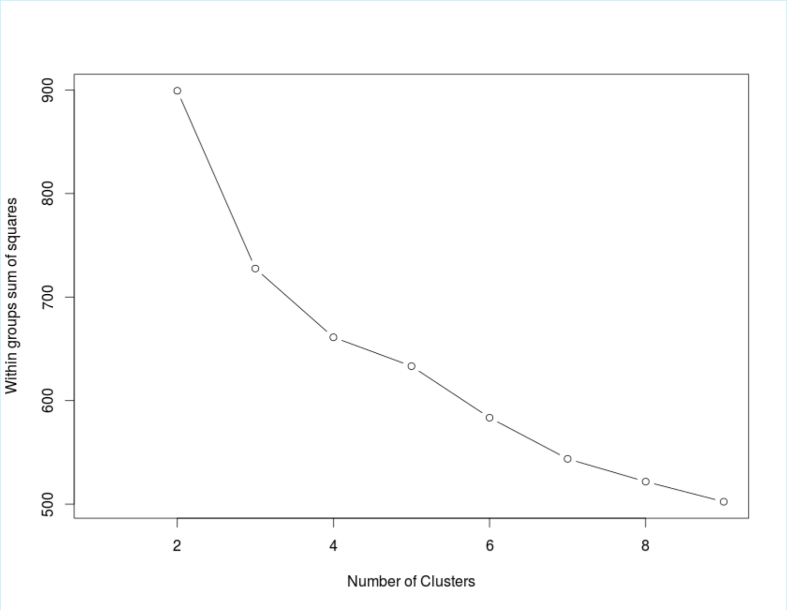
6.4 Compare Silhouette-Values with Python

```
For n_clusters = 5 The
average silhouette_score
is : 0.56376469026194
```



6.4 Evaluation of Clustering Models: Elbow criterion

▶ Choose a number of clusters so that adding another cluster doesn't give much better modeling of the data



What could be possible error measures to compute the quality of a parameterization?

- Run k-means clustering on the dataset for a range of values of k (here 1 to 10 in the examples above)
- For each value of k calculate the cluster’s errors, e.g. sum of squared errors
- Plot a line chart of the total error for each value of k
- If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best

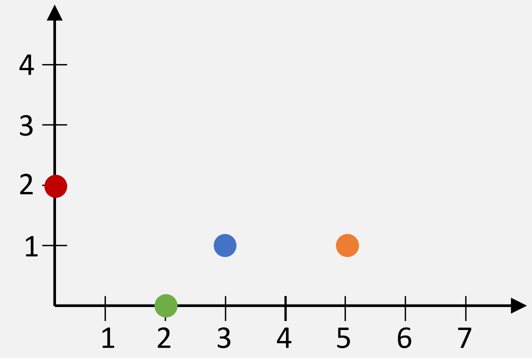
6.4 Classroom Task

Your turn!

Task

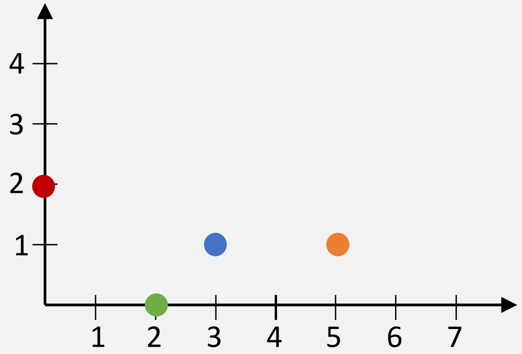
Please compute the Minkowski distances (L_1, L_2, and L_3) for the following dataset:

Observation	X	Y
	0	2
	2	0
	3	1
	5	1



6.4 Classroom Task

Observation	X	Y
●	0	2
●	2	0
●	3	1
●	5	1



L_1	●	●	●	●
●				
●				
●				
●				

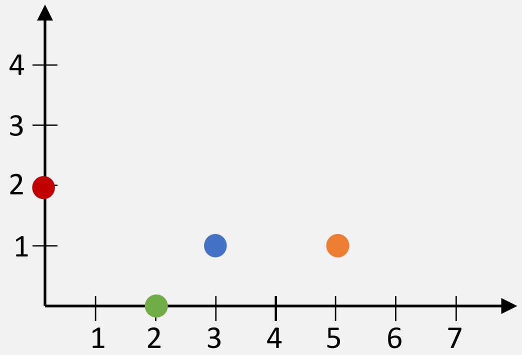
$$d(x,y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r\right)^{\frac{1}{r}}$$

$$d(x,y) = (|(0-0)|^1 + |(2-2)|^1)^{\frac{1}{1}}$$

$$d(x,y) = (|0-2| + |2-0|) = 2 + 2 = 4$$

6.4 Classroom Task

Observation	X	Y
🔴	0	2
🟢	2	0
🔵	3	1
🟠	5	1



$$d(x,y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r\right)^{\frac{1}{r}}$$

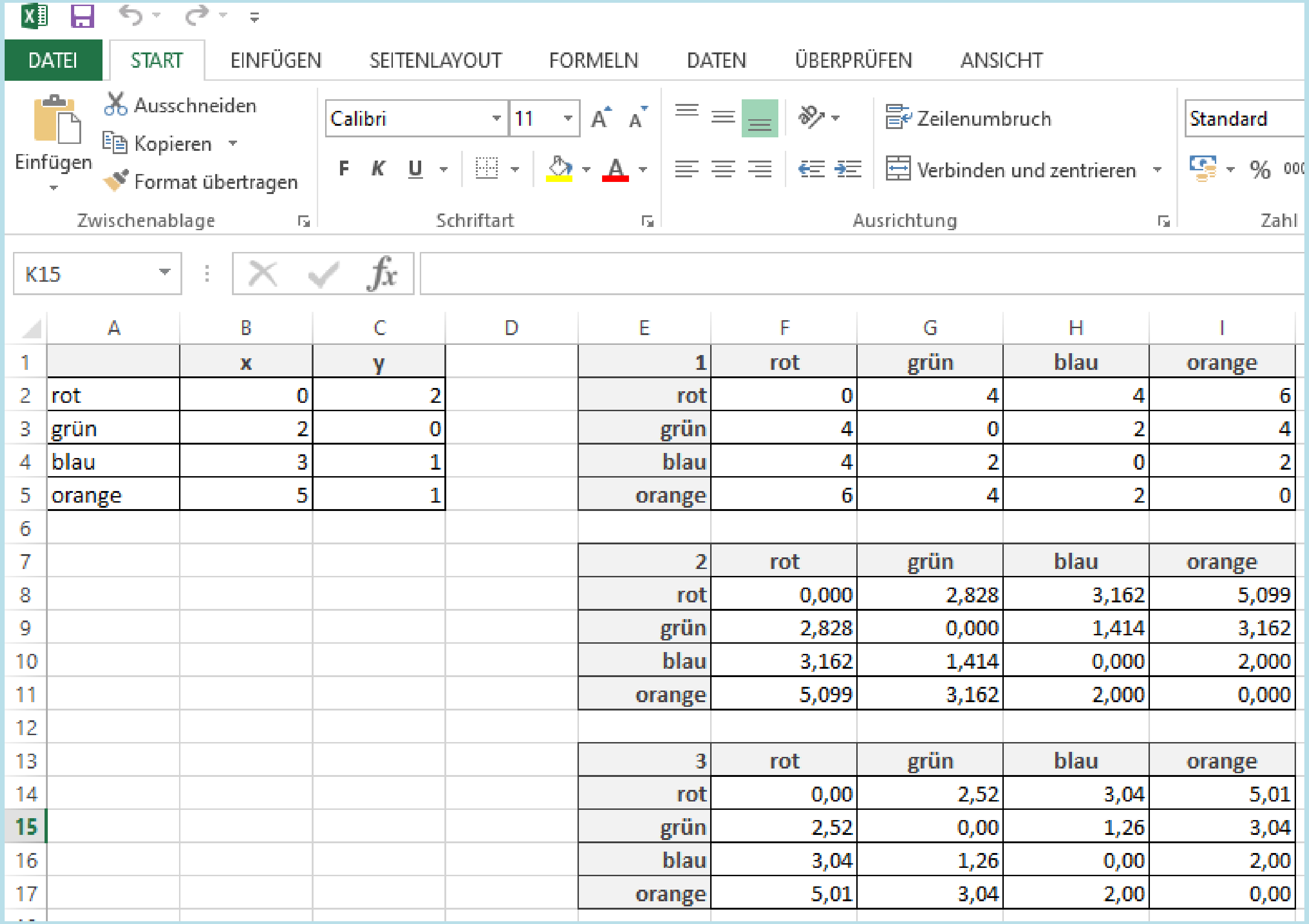
L_1	🔴	🟢	🔵	🟠
🔴	0	4	4	6
🟢	4	0	2	4
🔵	4	2	0	2
🟠	6	4	2	0

L_2	🔴	🟢	🔵	🟠
🔴	0	3	3	5
🟢	3	0	1	3
🔵	3	1	0	2
🟠	5	3	2	0

L_3	🔴	🟢	🔵	🟠
🔴	0	2	3	5
🟢	2	0	1	3
🔵	3	1	0	2
🟠	5	3	2	0

*Results are rounded

6.4 Minkowski Distance Exercise Sheet



	x	y
rot	0	2
grün	2	0
blau	3	1
orange	5	1

1	rot	grün	blau	orange
rot	0	4	4	6
grün	4	0	2	4
blau	4	2	0	2
orange	6	4	2	0

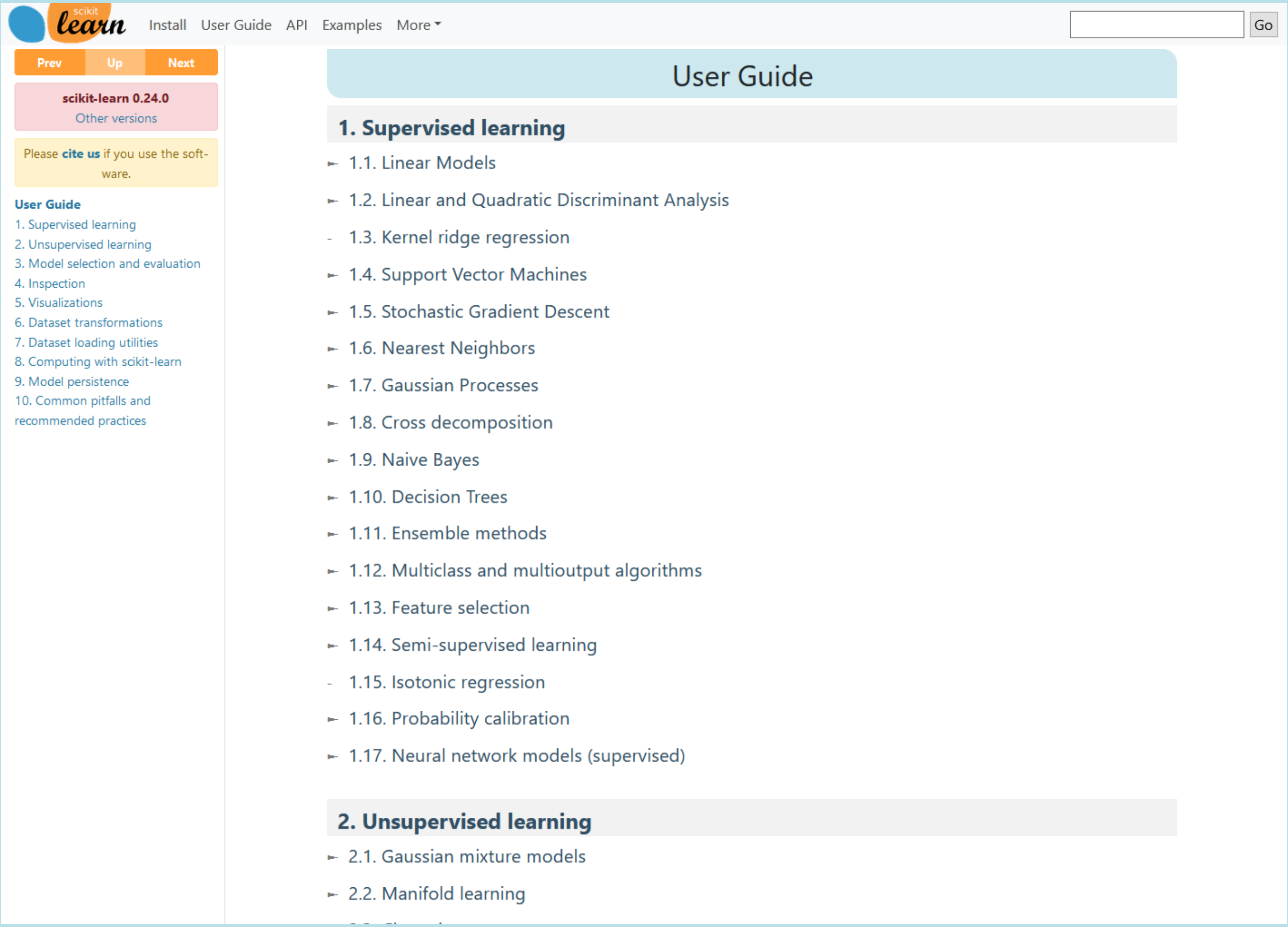
2	rot	grün	blau	orange
rot	0,000	2,828	3,162	5,099
grün	2,828	0,000	1,414	3,162
blau	3,162	1,414	0,000	2,000
orange	5,099	3,162	2,000	0,000

3	rot	grün	blau	orange
rot	0,00	2,52	3,04	5,01
grün	2,52	0,00	1,26	3,04
blau	3,04	1,26	0,00	2,00
orange	5,01	3,04	2,00	0,00

- Still unsecure with the Minkowski distance computation? Check out the excel exercise sheet!

↗ Code/Lecture 6 - Compute Minkowski distances.xlsx

6. Scikit-Learn Userguide



Check out the scikit-learn user guide for more data and feature engineering algorithms (↗ here).

6. Exercises

Workbook Exercises

- Please read the chapters 18.1-18.6, 18.8, 18.11, 19 and 21 from Rusell, S., & Norvig, P. (2016) to understand the theory behind the concepts of this lecture. Then work through the exercises of each chapter. You can skip the parts about „neuronal networks“ and „probabilistic models“, we will handle these topics in the next chapters.
- Read the chapters 1, 3, 4, 6 and 7 of Géron, A. (2017) to recapitulate the application of the machine learning algorithms we discussed in lecture. Solve the related exercises of each chapter.

Coding Exercises

- Implement the code project from chapter 2 in Géron, A. (2017) for yourself and solve the related coding exercises of this chapter.

6. References

Literature

1. Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics* (pp. 26-33).
2. Beleites, C.; Baumgartner, R.; Bowman, C.; Somorjai, R.; Steiner, G.; Salzer, R. & Sowa, M. G. (2005): Variance reduction in estimating classification error using sparse datasets, Chemom Intell Lab Syst, 79, 91 - 100.
3. Bellman, R. E. (1957): Dynamic Programming. Princeton University Press.
4. Bruner, J. S., & Austin, G. A. (1986). *A study of thinking*. Transaction publishers.
5. Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Springer, Berlin, Heidelberg.
6. Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.*
7. Halevy, A., Norvig, P., & Pereira, F. (2009). The unreasonable effectiveness of data. IEEE Intelligent Systems, 24(2), 8-12.
8. Kohavi, R. (1995): A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Mellish, C. S. (ed.) In *Artificial Intelligence Proceedings 14 th International Joint Conference*, 20 -- 25. August 1995, Montréal, Québec, Canada, Morgan Kaufmann, USA, , 1137 - 1145.
9. Kim, J.-H. (2009): Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap , Computational Statistics & Data Analysis , 53, 3735 - 3745.

6. References

Literature

10. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.
11. Mitchell, T. M. (1997). Machine learning. McGraw Hill.
12. Rusell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach.* Global Edition.
13. Wolpert, D. H., & Macready, W. G. (1995). No free lunch theorems for search (Vol. 10). Technical Report SFI-TR-95-02-010, Santa Fe Institute.

News articles

1. Wired (2017): Artificial Intelligence Is About to Conquer Poker, But Not Without Human Help. Online available at https://www.wired.com/2017/01/ai-conquer-poker-not-without-human-help/.

Images

All images that were not marked other ways are made by myself, or licensed ↗CC0 from ↗Pixabay.

6. References

Further reading

- I strongly recommend to start your own little machine learning project to get more familiar with the workflows and concepts we discussed in the lecture. Start experimenting with real-world datasets and try to solve real-world learning problems. A good point to start are the following websites:
  - Kaggle (↗ www.kaggle.com) provides many challenging tasks where you can win real money or internships at big tech companies like Google or Amazon,
  - or you can take a look at Reddit (↗ datasets subreddit) where many users provide free to use datasets.

6. Glossary

Clustering	*Grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters)*
Confusion Matrix	*Specific table layout to visualize errors in classification tasks*
Error due to Bias	*Difference between the expected prediction and the true value*
Error due to Variance	*Variability of a model prediction for a given data point*
Hypothesis Space	*The space of conjunctive hypotheses represented by a vector of n constraints*
Machine Learning	*A computer program is said to learn from experience 'E', with respect to some class of tasks 'T' and performance measure 'P' if its performance at tasks in 'T' as measured by 'P' improves with experience 'E'. (Mitchell, 1997)*
Metric	*Function that defines the distance between each pair of points of a set*
Overfitting	*When the model has low bias and high variance, i.e. is to specific (high total error)*
Reinforcement Learning	*Learning by using a system of reward and punishment*

6. Glossary

Supervised Learning	*Learning by mapping an input to an output based on example input-output pairs*
Test set	*Data set to test the performance of the learning model*
Train set	*Data set to train the learning model*
Underfitting	*When the model has high bias and low variance, i.e. is too general (high total error)*
Unsupervised Learning	*Learning undetected patterns in a data set with no pre-existing labels*

6. When Machine Learning Fails…