Cyber-Physical Systems
in Mechanical Engineering TU Berlin

Applied Machine Learning in Engineering

Lecture 05 summer term 2025

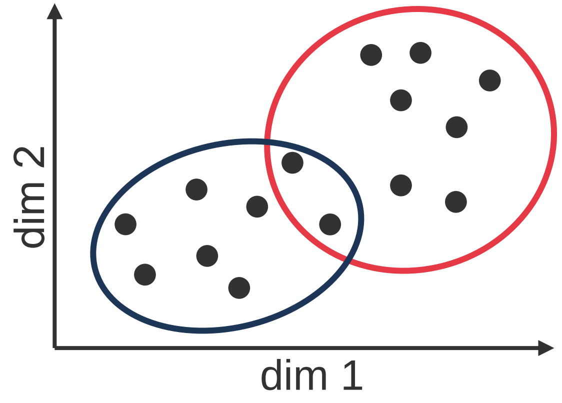
Prof. Merten Stender

Cyber-Physical Systems in Mechanical Engineering, Technische Universität Berlin

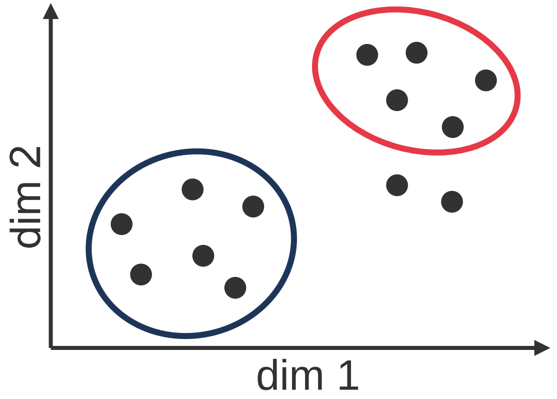
www.tu.berlin/cpsme
merten.stender@tu-berlin.de

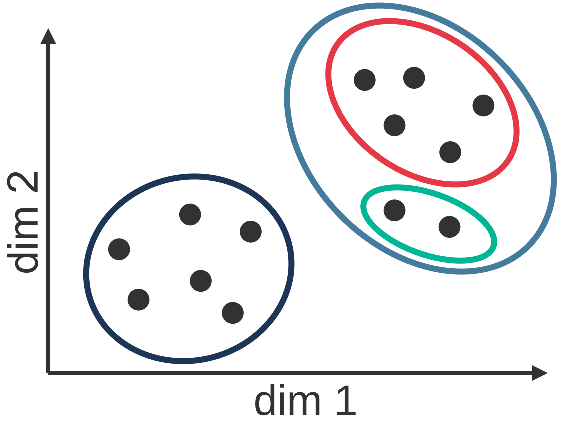
Recap: Lecture 04

Clustering = the entity of clusters = the overall result of a clustering process



1. **Nesting**
2. **Exclusiveness**
3. **Completeness**



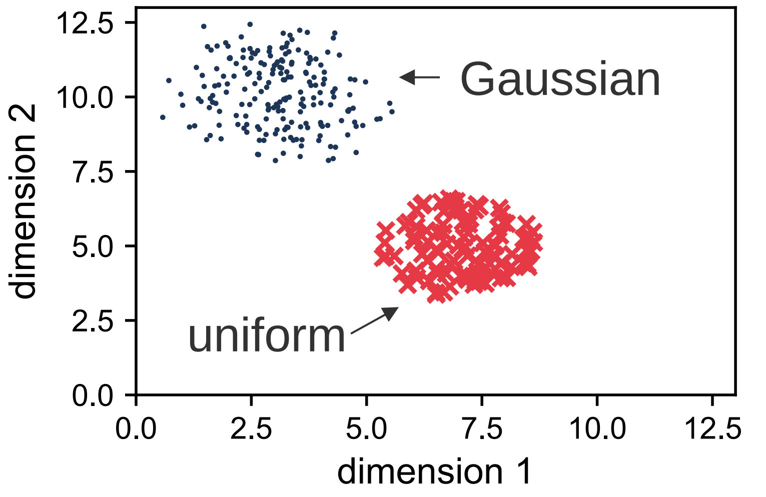


Recap: Lecture 04

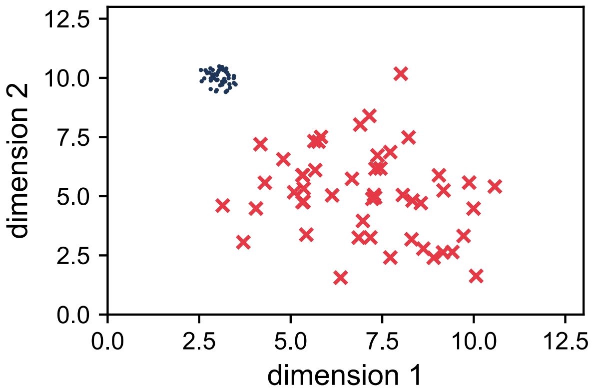
Cyber-Physical Systems
in Mechanical Engineering TU Berlin

Characteristics of Clusters

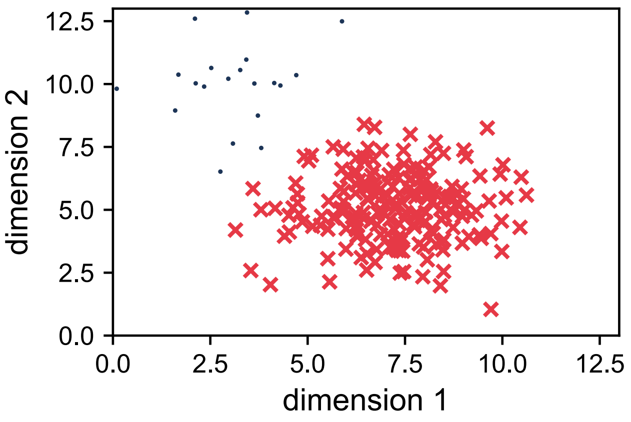
- **Distribution** of points within a cluster
  - Examples:
  - Gaussian distribution
  - Uniform distribution



- **Density**
  - High density clusters
  - Low density clusters



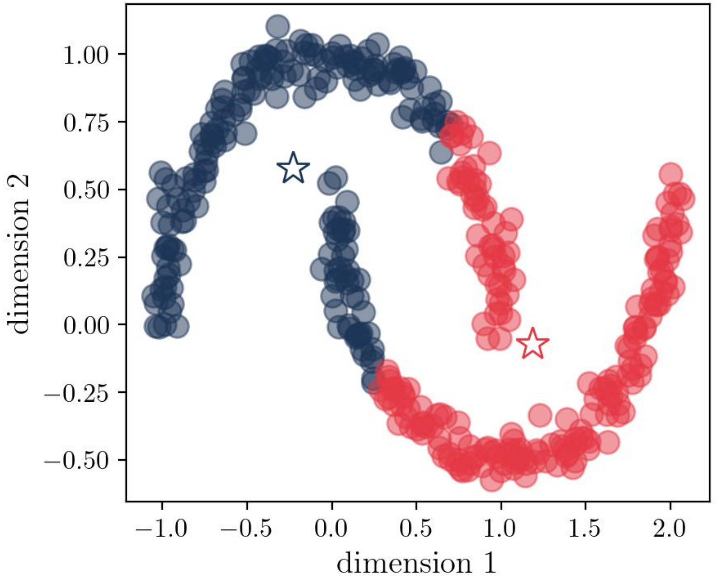
- **Size / variance**
  - Hypervolume consumed by cluster
  - Number of cluster members



Recap: Lecture 04

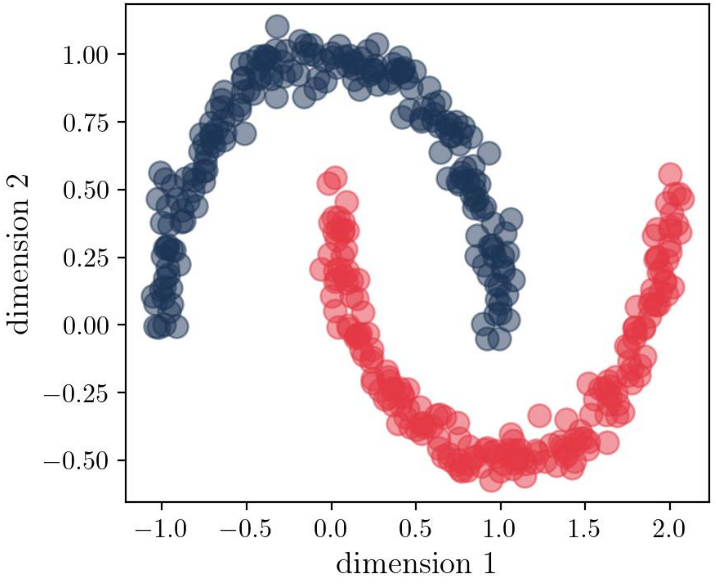
K-means

- Built for prototype-based clusters (globular shape)
- No outlier handling (*exclusive* and *complete* clustering)



DBSCAN*

- Built for density-based clusters (any shape)
- Allows for incomplete clusterings (outliers without cluster assignment)



* **D**ensity-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise

DBSCAN: Definition

- DBSCAN builds on 3 different types of data points:

- **Core point $\mathbf{x}_{\text{core}}$** has at least N_{min} points within ϵ neighborhood (incl. itself). Interior of a cluster

- **Edge point $\mathbf{x}_{\text{edge}}$** is reachable from a core point within ϵ, but is not a core point. Edge of a cluster

- **Outliers $\mathbf{x}_{\text{outlier}}$** is no core point and is not ϵ -reachable from any core point. Not a cluster member

Recap: Exercise 04

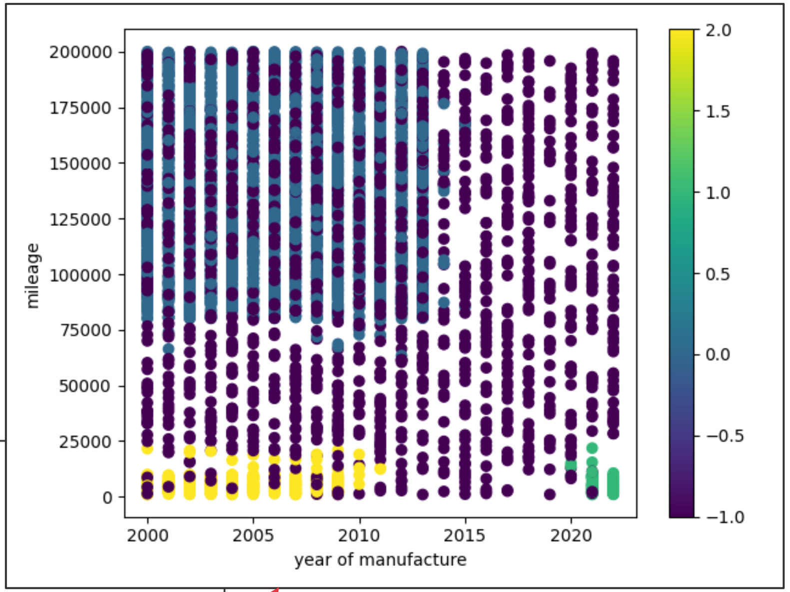
Cyber-Physical Systems
in Mechanical Engineering TU Berlin

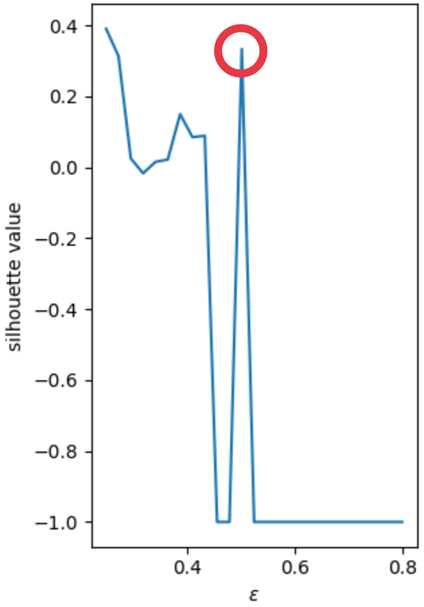
- From scratch implementation of Z-scoring
- initialization
- .fit method: calculate mean and std. deviation of the given data set
- .transform: z-scoring operation
- .inverse_transform: reverse z-scoring operation

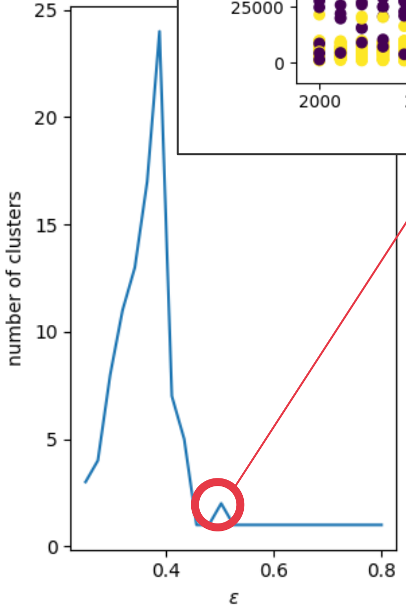
```python
class ZScorer:

    def __init__(self):
        self._means: np.ndarray
        self._stds: np.ndarray

    def fit(self, data: np.ndarray):
        # Fits the class object to the data set,
        # extracts mean and std per feature column

        self._means = np.mean(data, axis=0)
        self._stds = np.std(data, axis=0)

    def transform(self, data: np.ndarray) -> np.ndarray:
        # returns a zscored data set

        # 1. subtract the mean from the data set, columnwise
        data_transformed = data - self._means

        # 2. devide by standart deviations (columnwise)
        data_transformed = data_transformed / self._stds

        return data_transformed
    def inverse_transform(self, data: np.ndarray) -> np.ndarray:
        # reverse the zscoring transformation

        # 1. multiply with std
        data_reversed = data * self._stds

        # 2. add the means
        data_reversed = data_reversed + self._means

        return data_reversed
```

Recap: Exercise 04

- Clustering data using **DBSCAN**
- **Hyperparameter optimization** through grid search for optimum silhouette value








```python
# hyperparameter search for N_min, epsilon
nmin_grid = np.arange(10, 11)
epsilon_grid = np.linspace(0.25, 0.8, num=25)

silhouette_values = []
number_of_clusters = []
for _nmin in nmin_grid:
    for _epsilon in epsilon_grid:

        _dbscan = DBSCAN(eps=_epsilon, min_samples=_nmin)
        _labels = _dbscan.fit_predict(data_scaled)

        _num_clusters = num_clusters(_labels)
        if _num_clusters > 1:
            _silhouette_value = sil_coeff(data_scaled, _labels)
        else:
            _silhouette_value = -1

        silhouette_values.append(_silhouette_value)
        number_of_clusters.append(_num_clusters)

silhouette_values = np.array(silhouette_values)
number_of_clusters = np.array(number_of_clusters)
```

Questions?

Agenda

- Introduction to supervised learning
- Decision and Regression Trees
- Entropy and class purity measures

Learning outcomes

Learn to …

- Estimate the value of high-quality labels for supervised learning
- Build a decision tree classifier
- Find optimal binary feature space segmentations

Know about …

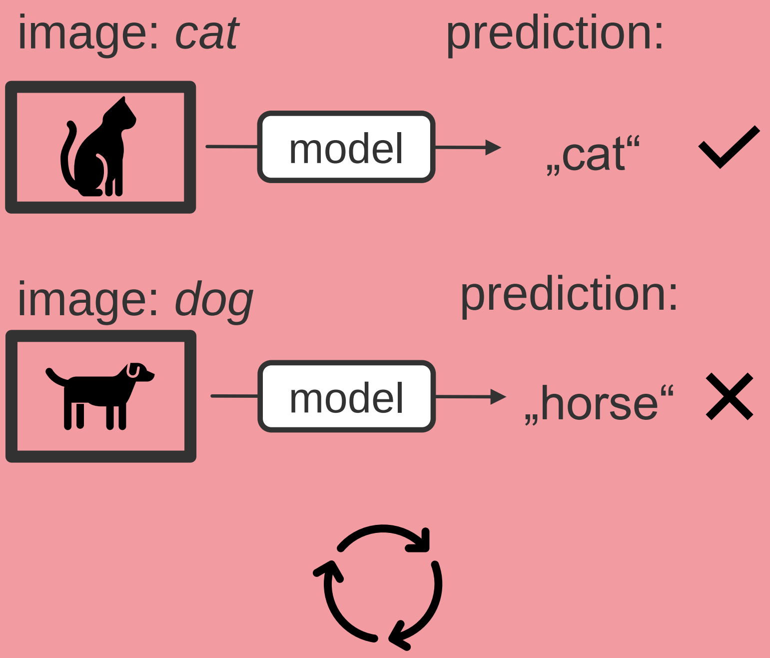
- Measures of purity in populations of discrete events
- Shannon entropy
- Information gain and Gini Index
- Python: Recursive Functions

Supervised Learning

Supervised and Unsupervised Learning

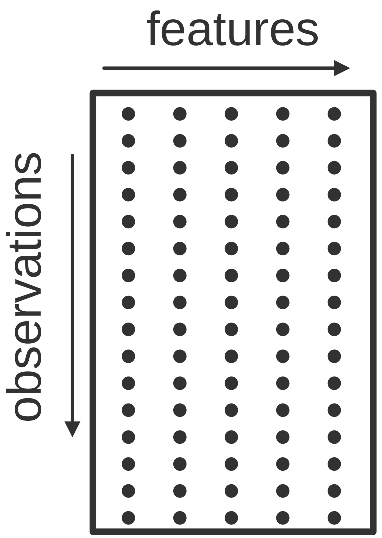
Cyber-Physical Systems
in Mechanical Engineering TU Berlin

supervised learning
(predictive task)

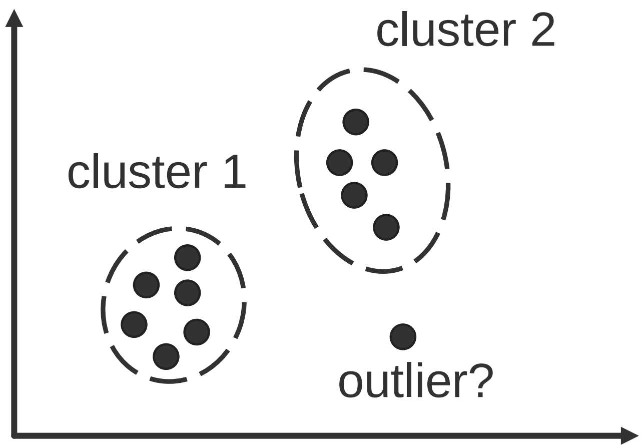


model training = reduce prediction error

data
(tabular)



unsupervised learning
(descriptive task)



finding clusters, groups and anomalies

Supervised Learning

Supervised learning = fitting prediction models to data for which ground truth targets exist

$$\mathcal{M}_{\boldsymbol{\theta}}: \ \mathbf{X} \mapsto \mathbf{y} \ , \mathbf{X} \in \mathbb{R}^{N\times n}, \mathbf{y} \in \mathbb{R}^{N\times m}$$

- **Ground truth** data (‘*labels*’)
  - Desired target quantities y_i

  → Allows comparing y_i against model predictions $\hat{y}_i$, prediction error $E = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|$

  → Quantitative statements about model prediction quality

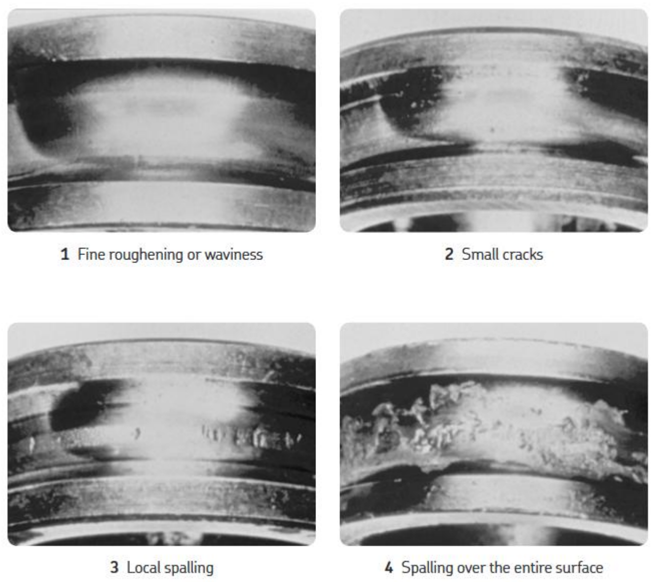
- **Model fitting**:
  - Reduction of error on training data set $\min_{\boldsymbol{\theta}} E(D_{\text{train}}, \ \boldsymbol{\theta})$ through optimization of $\boldsymbol{\theta}$
  - Model validation on hold-out validation data set D_{val}
  - Under- and overfitting as potential issues

- **Classification** and **regression tasks**

Application Cases in Engineering

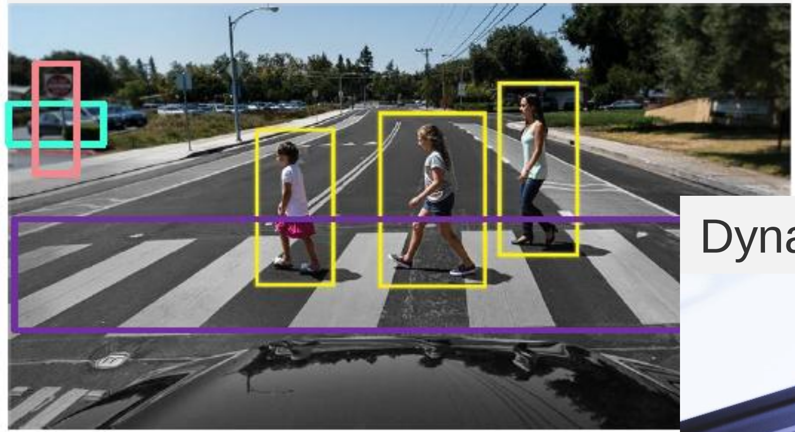
Cyber-Physical Systems
in Mechanical Engineering TU Berlin

Structural Health Monitoring
Predictive Maintenance
Remaining lifetime prediction



SKF® Bearing damage and failure analysis

Computer vision



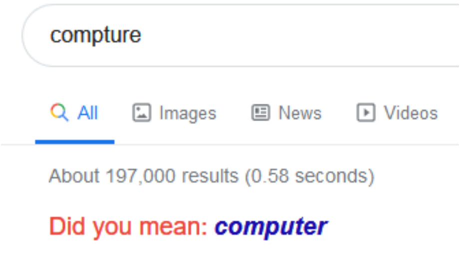
Dynamical behavior prediction

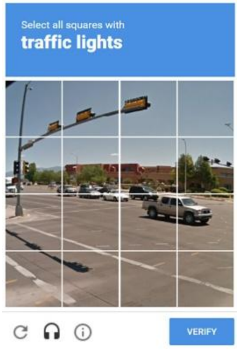
… and many more

Generation of Targets

- **Extremely important** (‚*trash in – trash out*'), yet tedious and expensive
  - Correct and high-quality ground truth targets are of crucial importance
  - Less but high-quality data should always be preferred over large and less-quality data

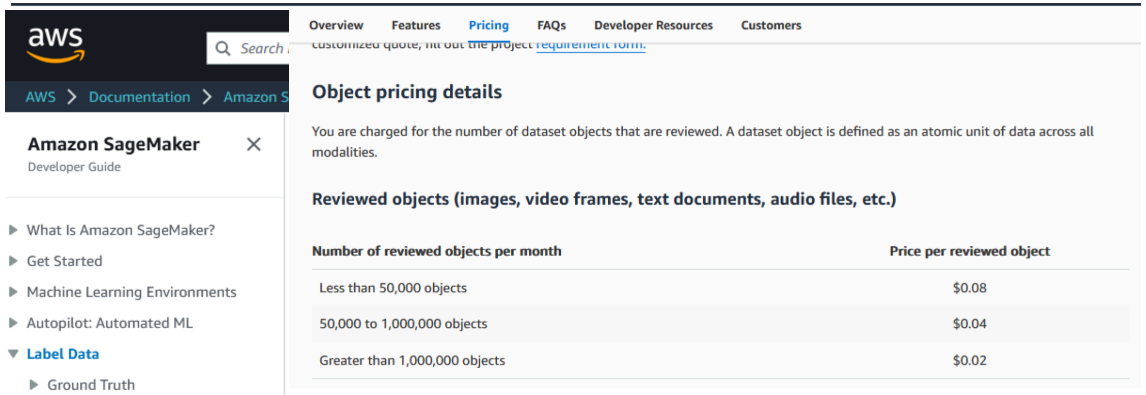
- Creative ways to generate labels:
  - Did you mean … ? → grammar / language models
  - reCAPTCHA - *are you a robot?* → computer vision





- Professional data labeling services

- Read newspaper article in ZEIT: https://www.zeit.de/2023/25/bild-annotation-kuenstliche-intelligenz-auto-indien



Amazon, 06 April 2023

Decision Trees

Decision Trees

input features $\boldsymbol{x} = [x_1, x_2, x_3]$

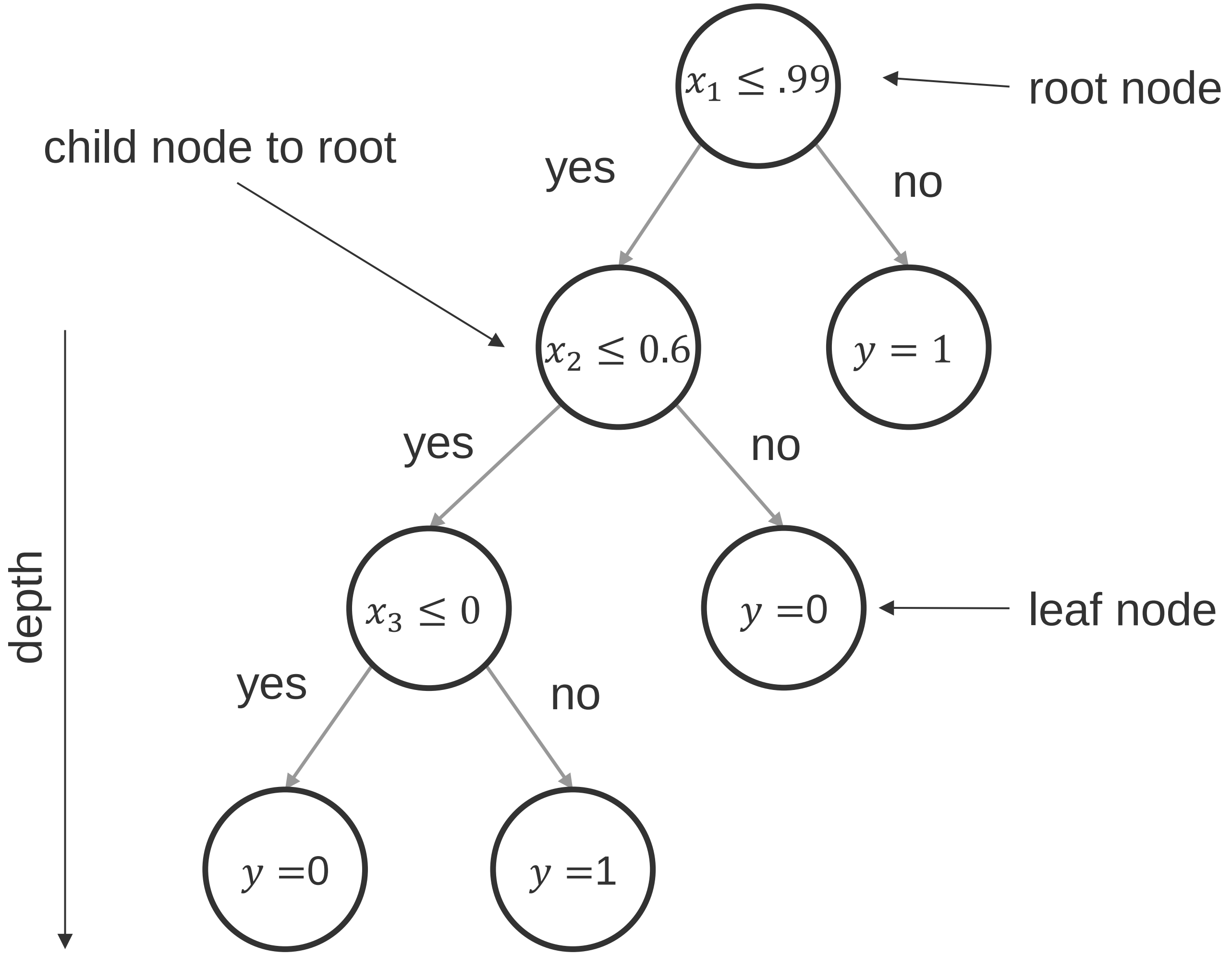
x_1: sun is shining: {0, 1}

x_2: probability of rain: [0,1.0]

x_3: ambient temperature: [-20, 40] °C

target y

y: ride bike to work? {0, 1}

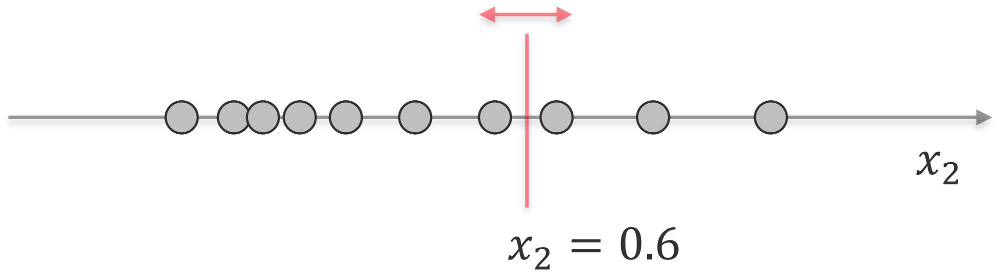


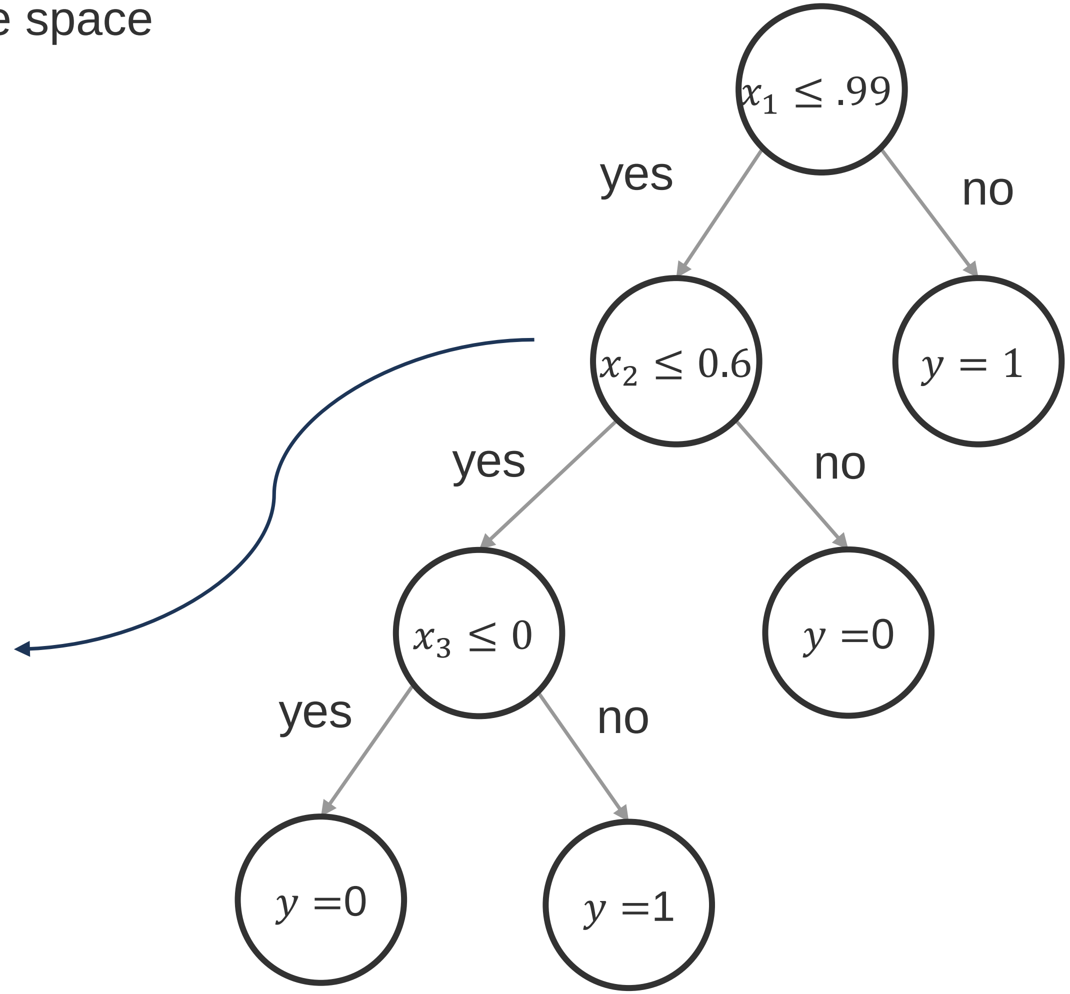
Decision Trees

Cyber-Physical Systems
in Mechanical Engineering TU Berlin

- Sequential decision rules partitioning the feature space
- Greedy algorithm
  - Previous splits not affected by current split
  - Algorithm does not ‘look ahead‘ or back

- Recursive binary **feature space segmentation**

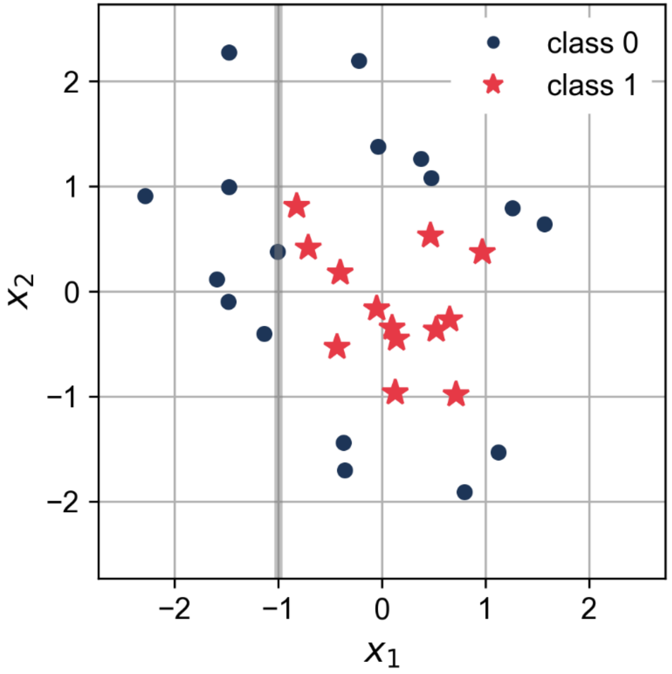




Decision Trees: Example

Cyber-Physical Systems
in Mechanical Engineering TU Berlin

- Aim: classify 2-dimensional data set with two classes (binary classification task)



Which split to do?

- Feature dimension (x_1, x_2)?
- Feature value (x^*) ?

- Split condition 1: $x_1 \leq -1.0$

child 1 (condition true)

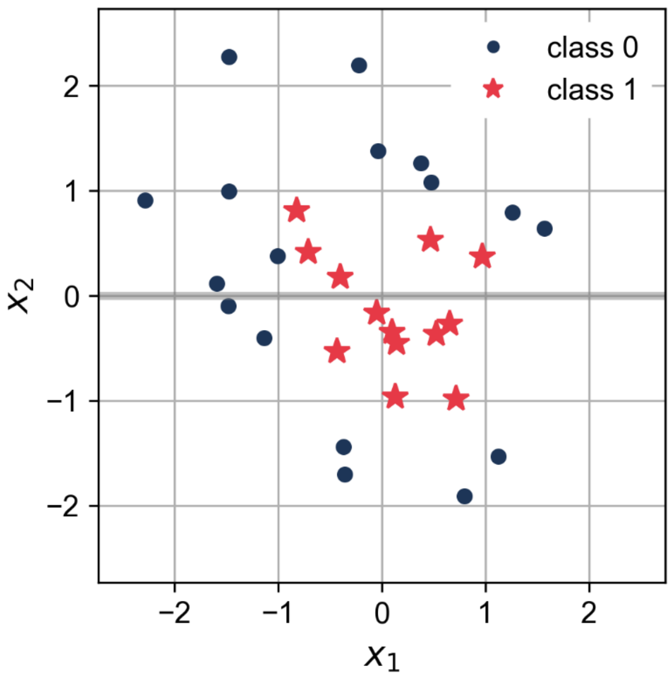
$7 \times$ ●
$0 \times$ ★

child 2 (condition false)

$10 \times$ ●
$13 \times$ ★

Decision Trees: Example

- Aim: classify 2-dimensional data set with two classes (binary classification task)



Which split to do?

- Feature dimension (x_1, x_2)?
- Feature value (x^*) ?

- Split condition 2: $x_2 \leq 0$

child 1 (condition true)

$6 \times$ ●

$8 \times$ ★

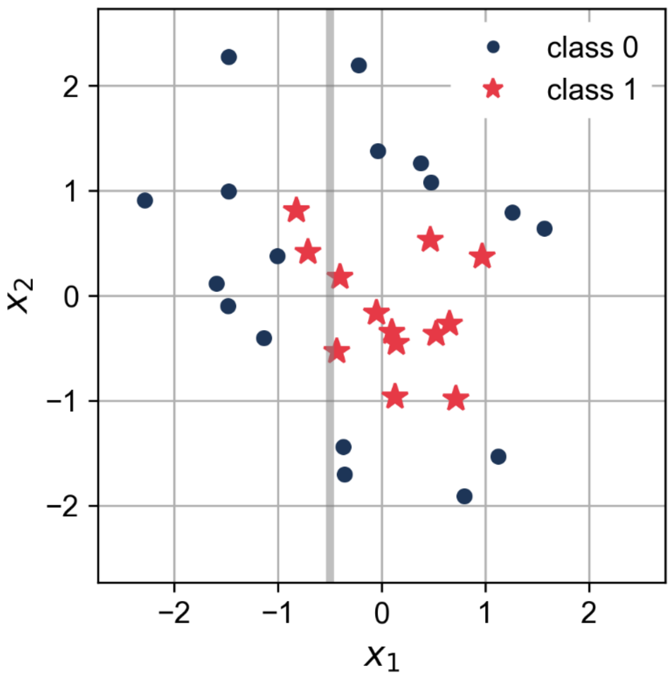
child 2 (condition false)

$11 \times$ ●

$5 \times$ ★

Decision Trees: Example

- Aim: classify 2-dimensional data set with two classes (binary classification task)



Which split to do?

- Feature dimension (x_1, x_2)?
- Feature value (x^*) ?

- Split condition 3: $x_1 \leq -0.5$

child 1 (condition true)

$7 \times$ ●

$2 \times$ ★

child 2 (condition false)

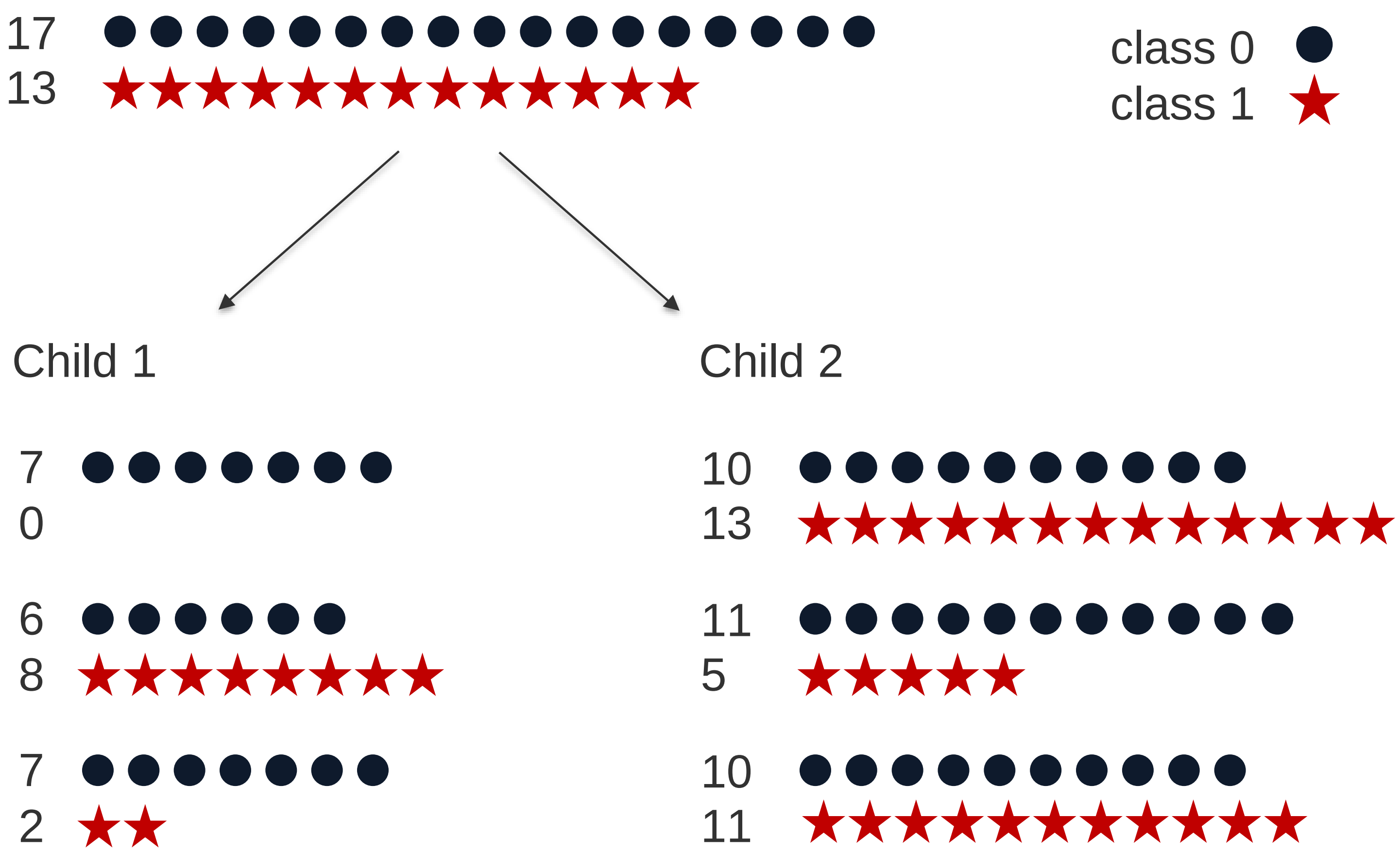
$10 \times$ ●

$11 \times$ ★

Selecting the Best Split

- Full data set

- Data split
  - Which to select?

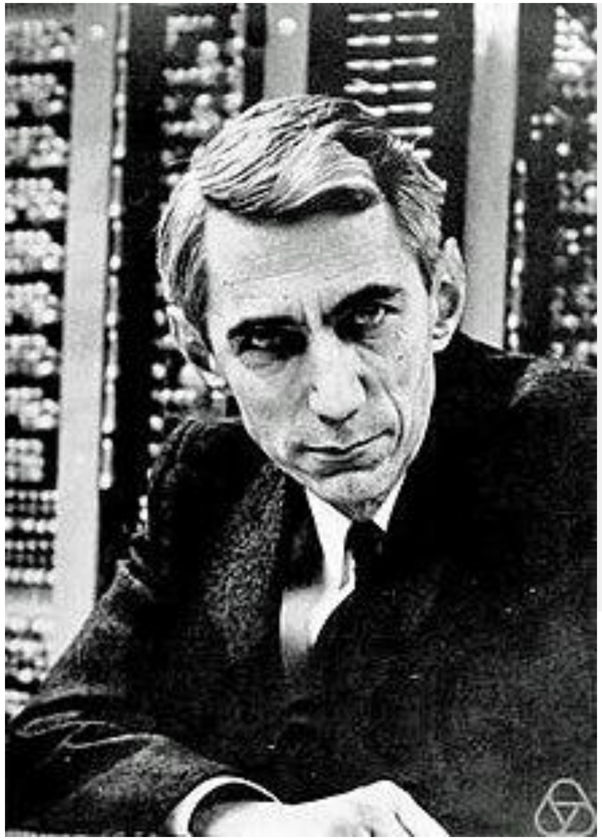


$$H(x) = -\sum_{i \in C} P(x_i) \log P(x_i)$$

Claude E. Shannon (1916-2001)

Founder of information theory

1948: A Mathematical Theory of Communication



Wikipedia, GNU Free Documentation License

Shannon Entropy: Definition

- Also denoted *information entropy* or *entropy index*

$$H(x) = -\sum_{i \in C} P(x_i) \log_a(P(x_i)) = \sum_{i \in C} P(x_i) \log_a\left(\frac{1}{P(x_i)}\right) \in [0,]$$

- $C = \{c_1, c_2, c_3\}$ set of distinct classes

- $P(x_i)$ probability of a single event *i:*
  - fraction of population composed of a single species i

- $H(x)$ amount of information gained by observing an event of probability $P(x_i)$

a base of the logarithm

The unit of entropy depends on the base of the logarithm

- Computer science: $a = 2$
  → unit *bits*

- Euler's number: $a = e$
  → unit *nats*

Shannon Entropy: Intuition

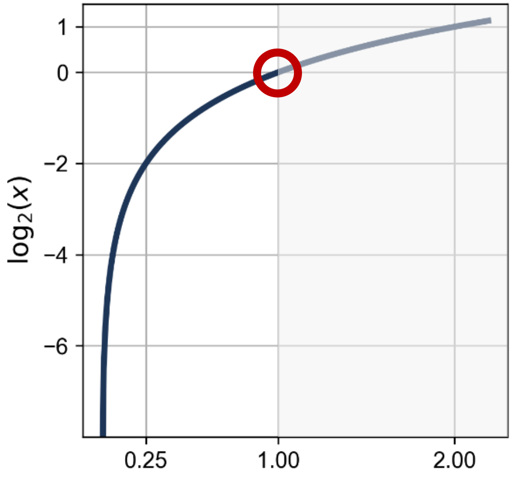
- Intuition and edge cases

$$H(x) = -\sum_{i \in C} P(x_i) \cdot \log_2(P(x_i))$$

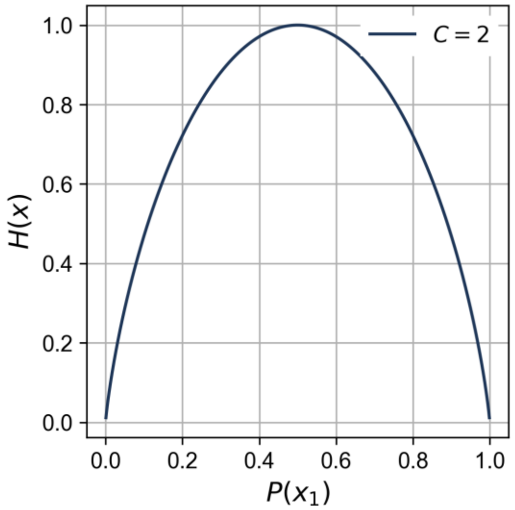
$C = 2$ (binary classification): $P(x_1) + P(x_2) = 1$

$$H(x) = -P(x_1) \cdot \log_2\big(P(x_1)\big) - P(x_2) \cdot \log_2\big(P(x_2)\big)$$
$$H(x) = -P(x_1) \cdot \log_2\big(P(x_1)\big) - (1 - P(x_1)) \cdot \log_2\big(1 - P(x_1)\big)$$

Log-2



$x = P(x_i)$



- $P(x_i) = 1 \rightarrow$ entropy$= 0$
- $\max\big(H(x)\big)_{C=2} = 1.0$ for $P(x_1) = P(x_2)$

Arbitrary C:

- $\max\big(H(x)\big)_C = -C \cdot \left(\frac{1}{C} \log \frac{1}{C}\right) = -\log \frac{1}{C}$

Shannon Entropy: Example

$$H(x) = -\sum_{i \in C} P(x_i) \log_2\big(P(x_i)\big)$$

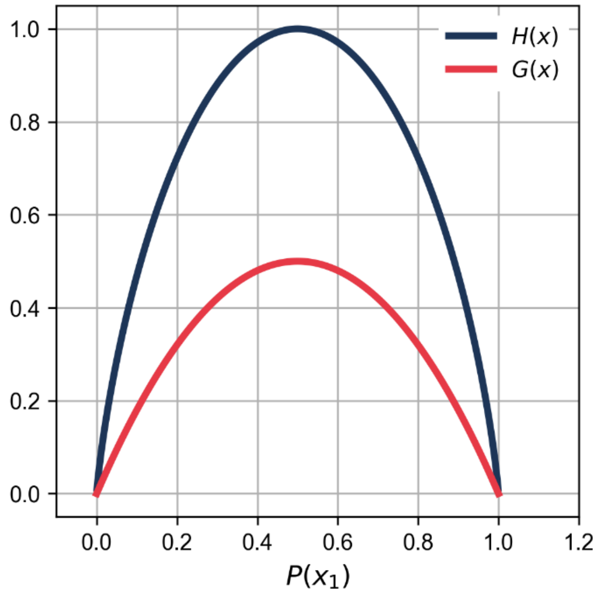
- Example: **calculate** the uncertainty coming with a certain character appearing next
  - sequence of numbers $[2\ 3\ 0\ 2\ 7\ 1\ ]$
  - probabilities: $P(0) = 1/6$, $P(1) = 1/6$, $P(2) = 2/6$, $P(3) = 1/6$, $P(7) = 1/6$
  - entropy $H(x) = 2.2516$

- Edge case 1: Outcome is certain: vanishing entropy $H(x) = 0$, e.g., $[2\ 2\ 2\ 2\ 2\ 2\ ]$
- Edge case 2: The more proportional the frequencies of occurrence are, the harder it gets to make a prediction, hence the larger the entropy $H(x) \gg 0$, e.g., $[1\ 2\ 3\ 4\ 5\ 6\ ]$

Measuring purity of a population

Cyber-Physical Systems
in Mechanical Engineering TU Berlin

- Information entropy $H(X) = -\sum_{i=1}^{K} P_i \log_2(P_i)$ P_i : probability of class i out of K classes
- Gini index $G(X) = 1 - \sum_{i \in C} P(x_i)^2$



- Cases:
  - All members of the data set belong to a single class

    $10\times$, $0\times$ $H(X) = -\frac{10}{10}\cdot\log\frac{10}{10} - \frac{0}{10}\cdot\log\frac{0}{10} = 0 - 0 = 0$

  - Even distribution of members per class

    $5\times$, $5\times$ $H(X) = -\frac{5}{10}\cdot\log\frac{5}{10} - \frac{5}{10}\log\frac{5}{10} = 0.5 + 0.5 = 1$

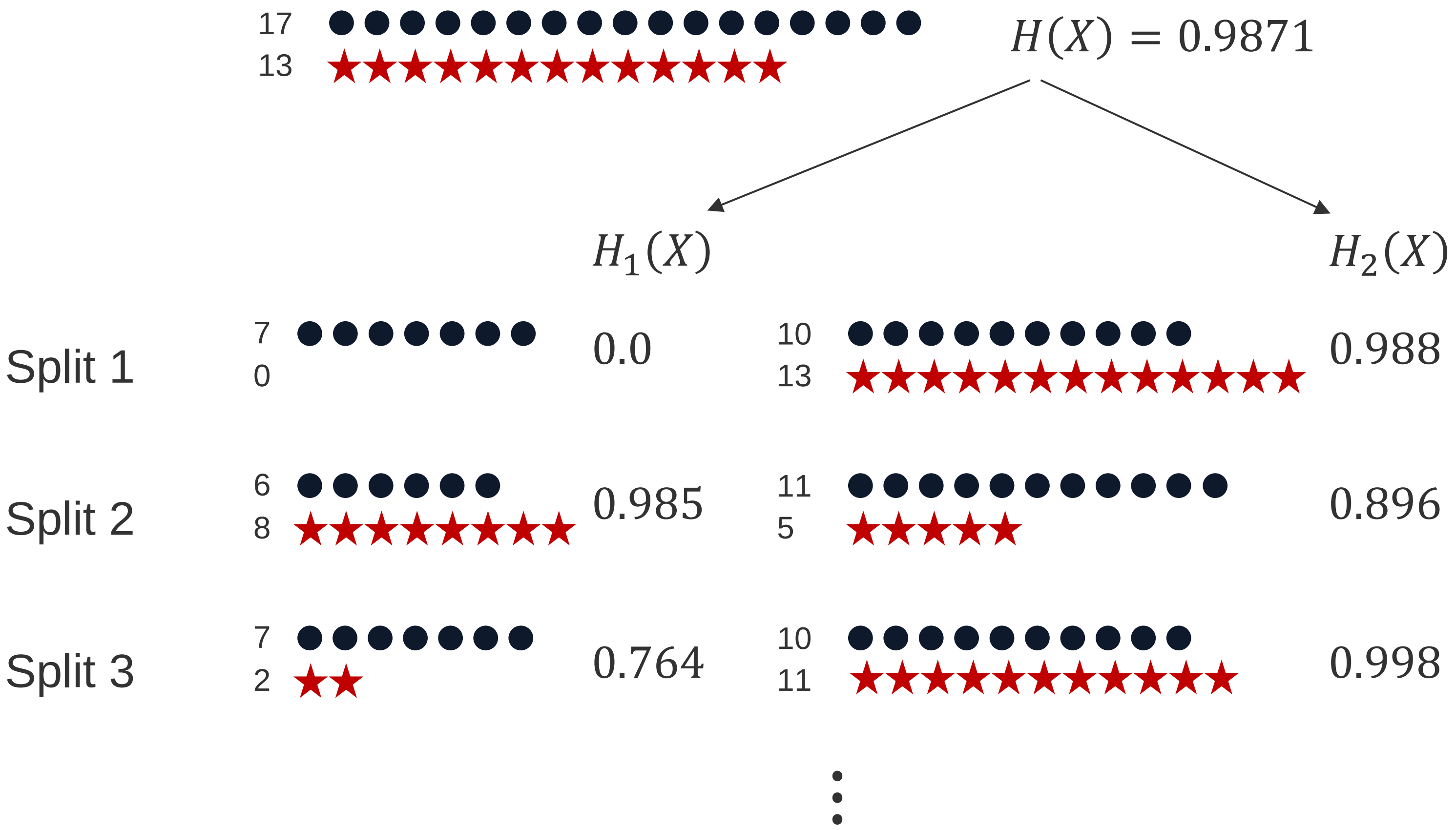
- Our data set at root:

17
13

$$H(X) = -\frac{17}{30}\log\frac{17}{30} - \frac{13}{30}\log\frac{13}{30} = 0.9871$$

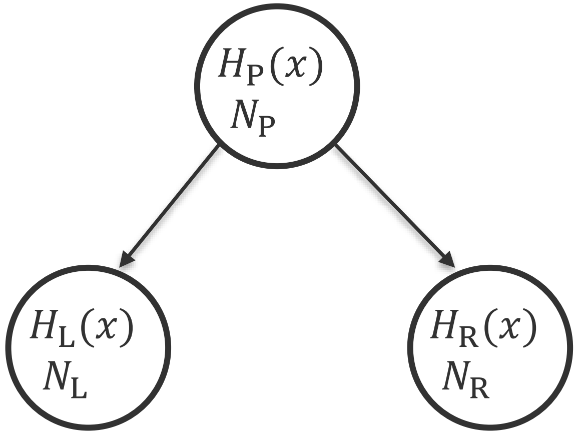
Selecting the best split

- Root (parent) data set
- **Compute** the entropy of every possible data split
- What now? Which split to select?
  → **Information gain**



Information Gain

- DT: segmentation of the feature space
- Aim: maximal class purity of the sub-data set
- Purity metric: entropy
  - Entropy at parent node: $H_{\mathrm{p}}(x)$
  - Entropy at children: $H_{\mathrm{L,\,R}}(x)$
  - Number of samples: $N_{\mathrm{P,\,L,\,R}}$



- Information gain

$$I(x) = H_{\mathrm{P}}(x) - \underbrace{\left(\frac{N_{\mathrm{L}}}{N_{\mathrm{P}}} \cdot H_{\mathrm{L}}(x) + \frac{N_{\mathrm{R}}}{N_{\mathrm{P}}} \cdot H_{\mathrm{R}}(x)\right)}$$

- Split for maximum information gain

$$x^* = \max(I(x))$$

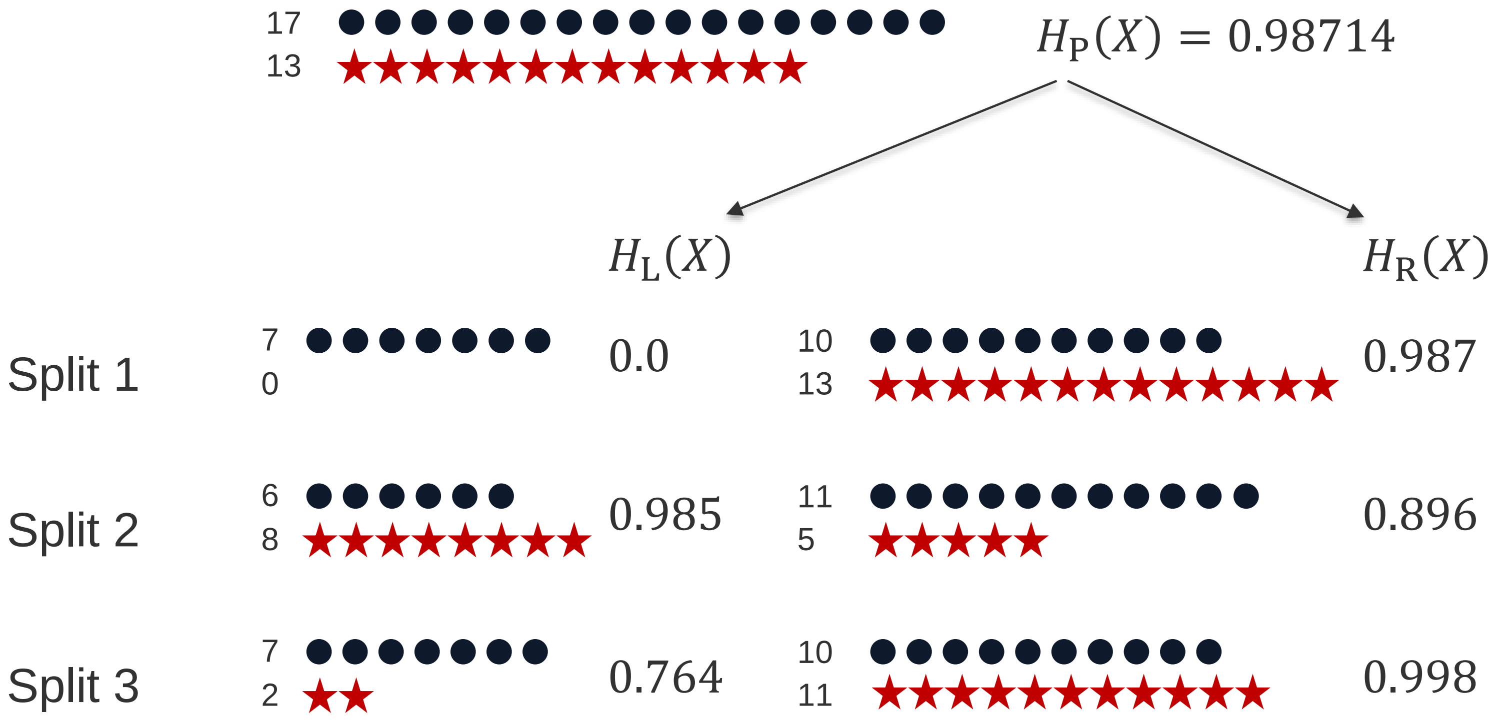
Maximum gain: $H_{\mathrm{L}} = H_{\mathrm{R}} = 0$

entropy in both child nodes vanishes (pure child notes)

Best Split: Information Gain

Information gain

$$I(x) = H_\mathrm{P}(x) - \left(\frac{N_\mathrm{L}}{N_\mathrm{P}} \cdot H_\mathrm{L}(x) + \frac{N_\mathrm{R}}{N_\mathrm{P}} \cdot H_\mathrm{R}(x)\right)$$



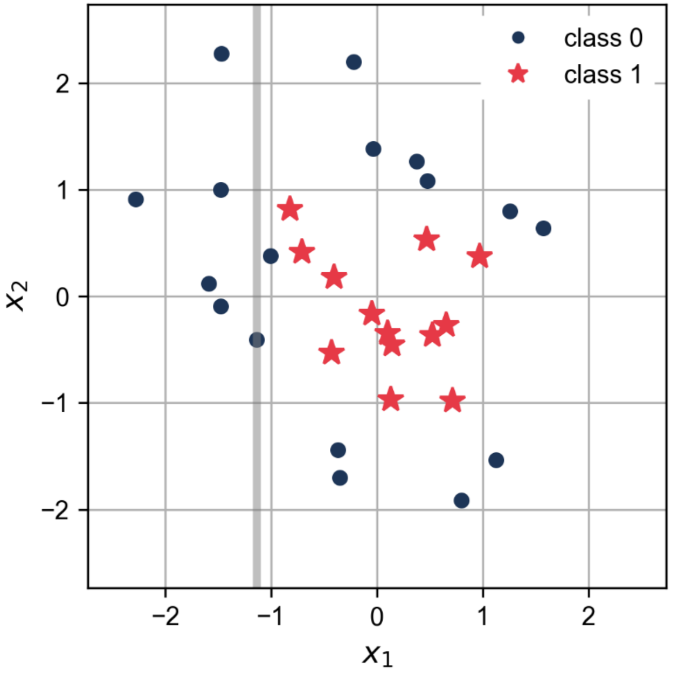
$I(x) = 0.230$

$I(x) = 0.049$

$I(x) = 0.059$

Decision Trees: Example

- Previous slides: 3 examplary data splits
- Actual: 58 splits possible (29 for 1st feature dimension, 29 for 2nd feature dimension)
  [excluding the splits that would put the complete data set into a single child, hence assuming a non-optimal purity in the data set to be split]

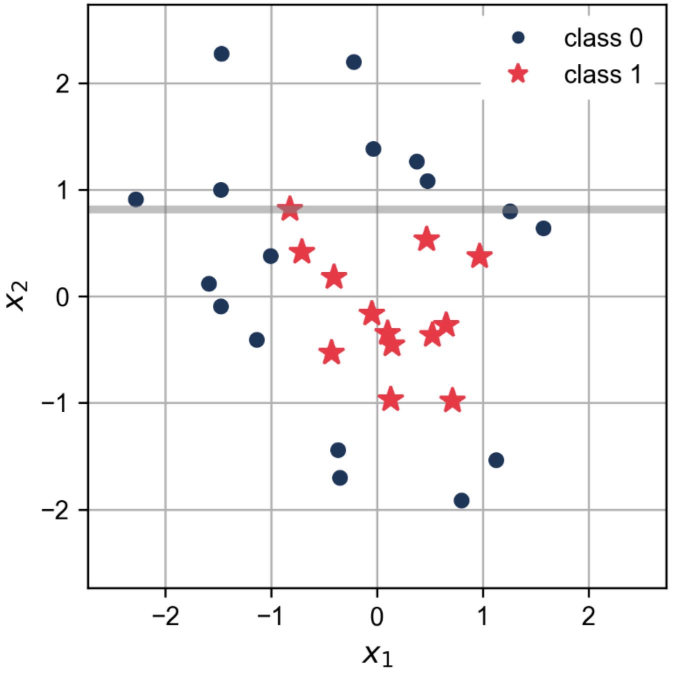


Greedy approach:

- Computation of all information gains
- Top 3 splits:
  - (3) condition: $x_1 \leq -1.137$ → information gain $I(x) = 0.191$

Decision Trees: Example

- Previous slides: 3 examplary data splits
- Actual: 58 splits possible (29 for feature dimension 1, 29 for feature dimension 2)

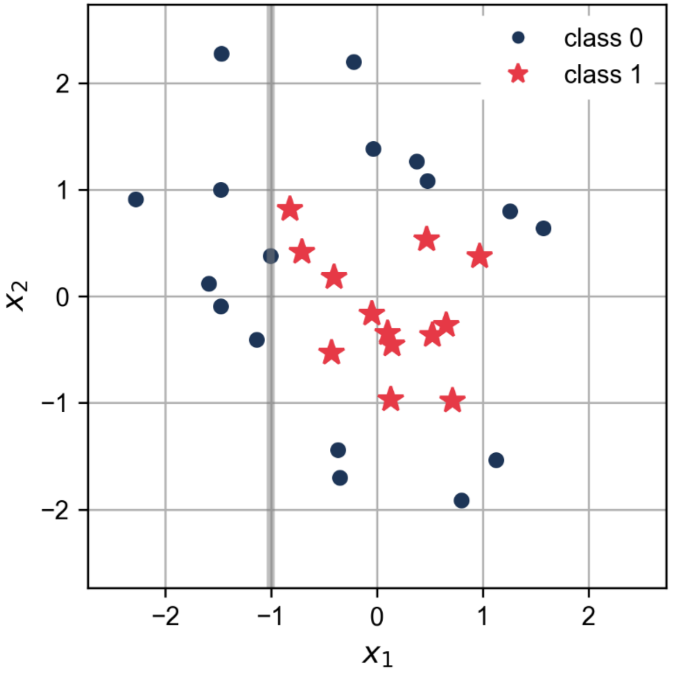


Greedy approach:

- Computation of all information gains
- Top 3 splits:
  - (3) condition: $x_1 \leq -1.137$ → information gain $I(x) = 0.191$
  - (2) condition: $x_2 \leq 0.813$ → information gain $I(x) = 0.229$

Decision Trees: Example

- Previous slides: 3 examplary data splits
- Actual: 58 splits possible (29 for feature dimension 1, 29 for feature dimension 2)



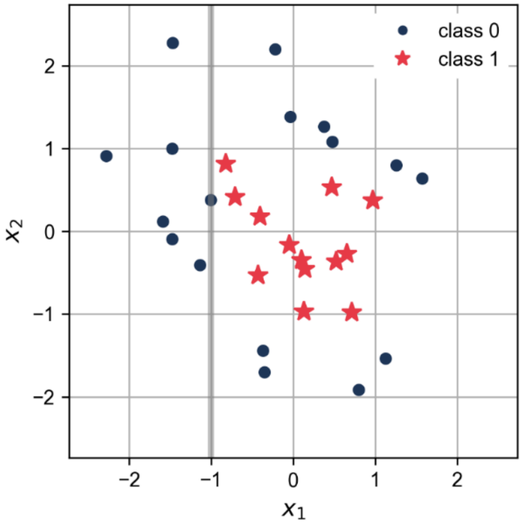
Greedy approach:

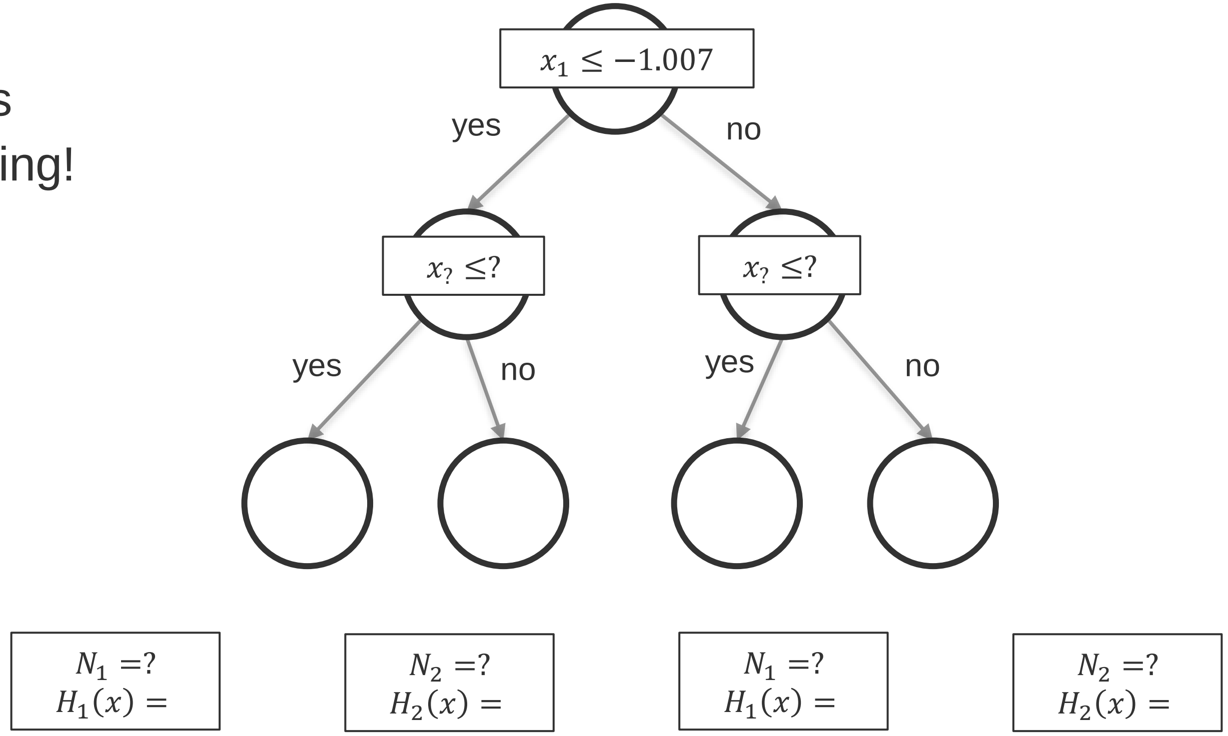
- Computation of all information gains
- Top 3 splits:
  - (3) condition: $x_1 \leq -1.137$ → information gain $I(x) = 0.191$
  - (2) condition: $x_2 \leq 0.813$ → information gain $I(x) = 0.229$
  - (1) condition: $x_1 \leq -1.007$ → information gain $I(x) = 0.229$

Decision Trees: Example

- **First split** of the data set:
  - Left child node: $x_1 \leq -1.007$ — 7 members of class 0
  - Right child node: $x_1 > -1.007$ — 10 members of class 0, 13 members of class 1

- **Second split**:
  - Optimal (information gain) split of child nodes
  - Here: left child node is pure → no more splitting!





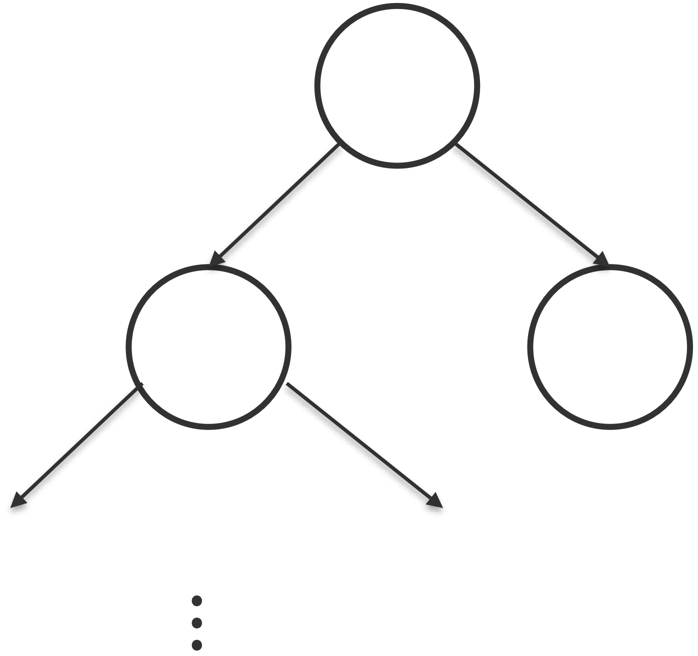
Stopping Criteria

- Aim: maximize purity in leaf nodes
- Without any constrains there is a solution with $H(x) = 0$ in each leaf node
  - Worst case: $N = 1$ samples per leaf
  - Overfitting the data

- Excessive splitting leads to **overfitting**:
  - Very strong performance on training data set
  - Weak performance on new (unseen) data

- **Constraints** to tree growing
  - Minimum number of samples per node N_{min}
  - Maximum depth of tree D_{max}

- **Impure leaves**: return the most-common class label



scikit-learn: DecisionTreeClassifier

Install User Guide API Examples Community More Go

Prev Up Next

scikit-learn 1.2.2
Other versions

Please **cite us** if you use the software.

sklearn.tree.DecisionTreeClassifier
DecisionTreeClassifier
Examples using sklearn.tree.DecisionTreeClass

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.DecisionTreeClassifier`(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Python: Recursive Functions

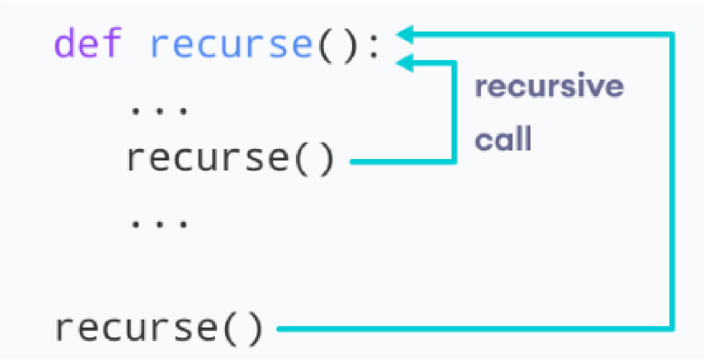
Recursive Functions

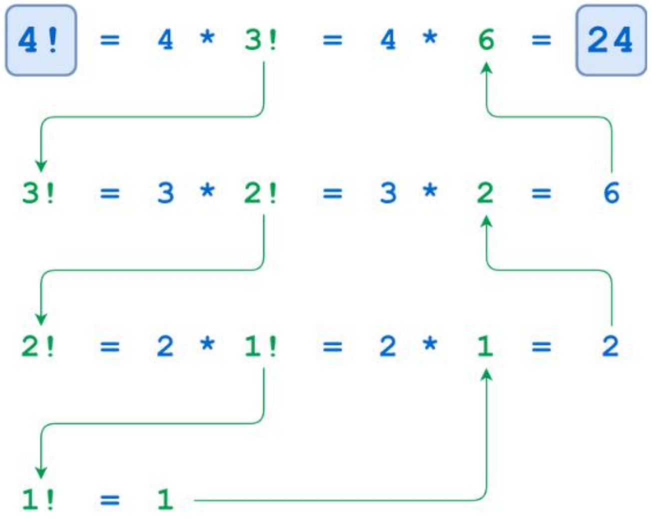
- A recursive function **calls itself** until some break condition (so-called „base condition“) is met
- Let the user created nested function calls or object structures

- Example: computing the factorial of a number

- Get the Python recursion limit (after which Python will stop automatically)

```
from sys import getrecursionlimit
getrecursionlimit()
```





https://realpython.com/python-recursion/

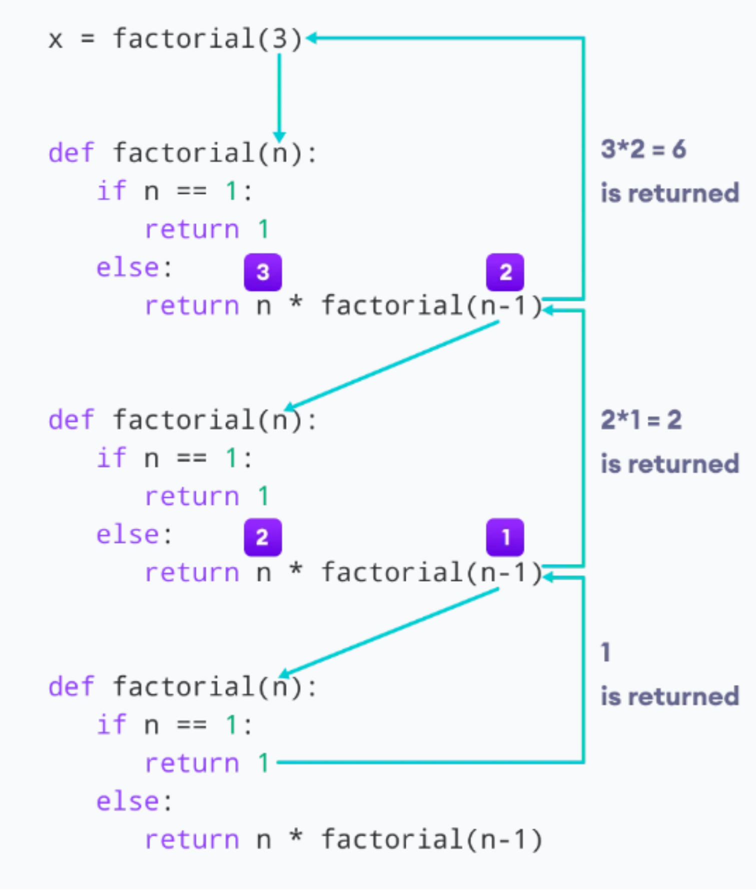
Recursive Functions

Cyber-Physical Systems
in Mechanical Engineering TU Berlin

- A recursive function **calls itself** until some break condition is met
- Example: computing the factorial of a number

```python
def factorial(x: int) -> int:
    """Compute the factorial of an
    integer using a recursive function"""
    if x == 1:
        return 1
    else:
    return x * (factorial(x-1))


print(f'factorial of x=3 is {factorial(3)}')
```

← stopping condition
(‚base condition‘)



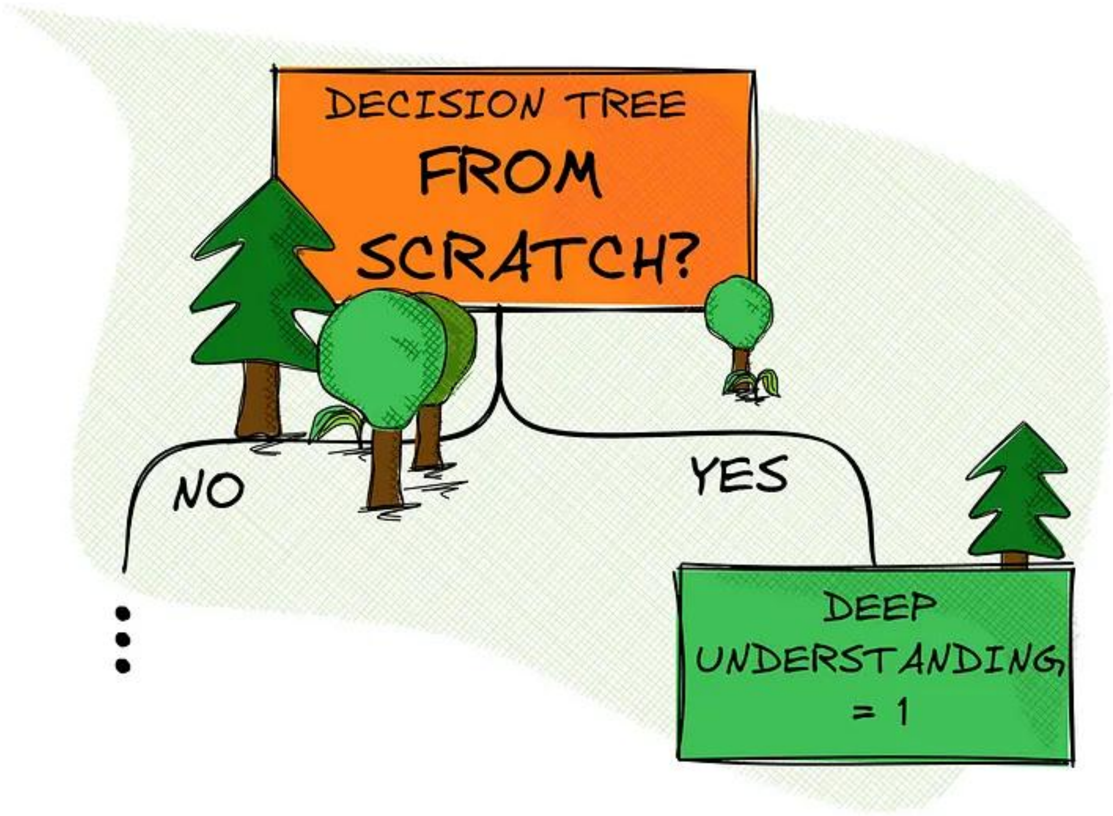
From https://www.programiz.com/python-programming/recursion

Exercise 05

Exercise 05

Cyber-Physical Systems
in Mechanical Engineering TU Berlin

- Implementation of a decision tree from scratch
- Implement entropy
- Implement information gain
- Implement a splitting routine
- Implement a best split routine



Questions?