

# Decision Trees and Ensemble Methods

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025



Find all slides on [GitHub](https://github.com/DaWe1992/Applied_ML_Fundamentals) (DaWe1992/Applied\_ML\_Fundamentals)

# Lecture Overview

- I** Machine Learning Introduction
- II** Optimization Techniques
- III** Bayesian Decision Theory
- IV** Non-parametric Density Estimation
- V** Probabilistic Graphical Models
- VI** Linear Regression
- VII** Logistic Regression
- VIII** Deep Learning
- IX** Evaluation
- **X** Decision Trees
- XI** Support Vector Machines
- XII** Clustering
- XIII** Principal Component Analysis
- XIV** Reinforcement Learning
- XV** Advanced Regression

# Agenda for this Unit

① Introduction

② Iterative Dichotomizer (ID3)

③ Extensions and Variants

④ Ensemble Methods

⑤ Wrap-Up

## Section: Introduction

What are Decision Trees?  
An exemplary Decision Tree  
An alternative Decision Tree

# What are Decision Trees?

- Decision trees are induced in a **supervised fashion**
- The ID3 algorithm was originally proposed by ROSS QUINLAN in 1986
- Decision trees are grown **recursively** (*divide-and-conquer*)
- Decision trees are **easily interpretable**  
(*unlike other methods like e. g. neural networks*)
- A decision tree consists of:

<b>Nodes</b>	Each node corresponds to an <b>attribute test</b>
<b>Edges</b>	One edge per possible test outcome ( <i>attribute value</i> )
<b>Leaves</b>	Class label to predict



## Portrait: ROSS QUINLAN

**JOHN ROSS QUINLAN** is an Australian computer science researcher in data mining and decision theory. He has contributed extensively to the development of decision tree algorithms, including inventing the canonical C4.5 and ID3 algorithms.

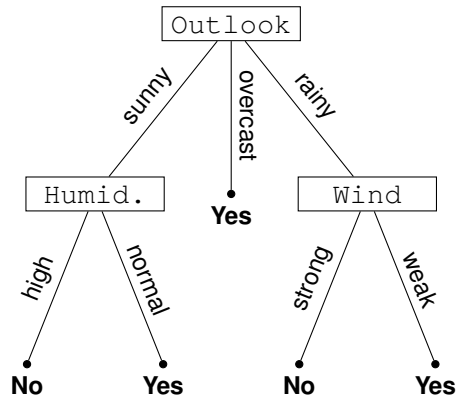
He is currently running the company RuleQuest Research which he founded in 1997.



*(Wikipedia)*

## What we want...

Outlook	Temperature	Humidity	Wind	PlayGolf
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rainy	mild	high	weak	yes
rainy	cool	normal	weak	yes
rainy	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rainy	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rainy	mild	high	strong	no
rainy	mild	normal	strong	???



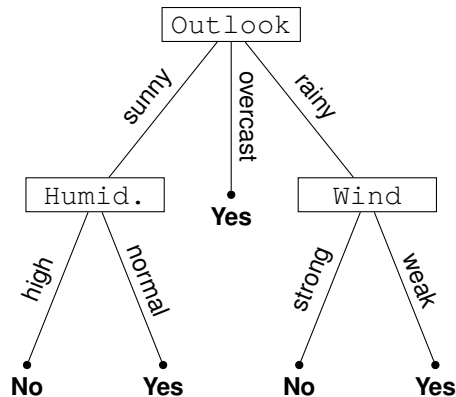
## Classification of new Instances

- Suppose we get a new instance:

Outlook	rainy
Temperature	mild
Humidity	normal
Wind	strong

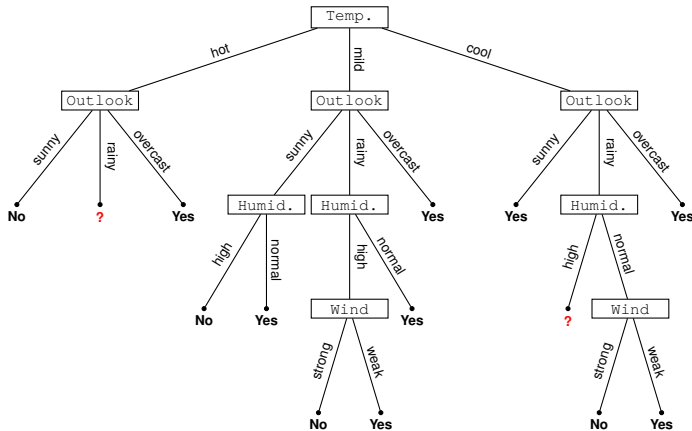
- Question:** What is its class?
- Answer:** No

**Please note:** Not all attributes must be considered for the classification!





# Another Decision Tree...



## Questions:

- Is this one better?
- What problems do you see?

## Section: Iterative Dichotomizer (ID3)

Inductive Bias of Decision Trees  
Split Heuristics: Entropy and Information Gain  
ID3 Algorithm

# Inductive Bias of Decision Trees

- Complex models tend to **overfit** the training data and hence **do not generalize well** to unseen data points
- **Therefore:** Prefer the simplest hypothesis that fits the data!
- This leads to:

**OCCAM's razor:** *'More things should not be used than are necessary'*

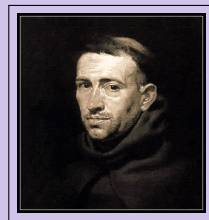
**Please note:** OCCAM's razor is a general methodology in machine learning and not limited to decision trees! Also, it is a guiding principle in other sciences.



## Portrait: WILLIAM OF OCKHAM

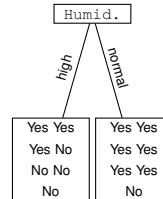
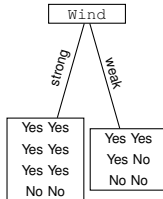
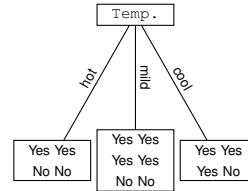
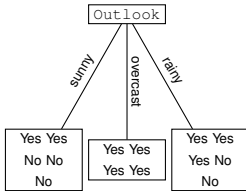
**WILLIAM OF OCKHAM** (circa 1287 – 1347) was an English Franciscan friar who is believed to have been born in Ockham, a small village in Surrey. He is considered to be one of the major figures of medieval thought.

He is commonly known for OCCAM's razor, the methodological principle that bears his name, and also produced significant works on logic, physics and theology.



*(Wikipedia)*

# The Root of all Evil... Which Attribute to choose?



# Finding a proper Split Attribute

- Simple and small trees are preferred:
  - Data in successor nodes should be **as pure as possible**  
*(with respect to the class labels)*
  - This means, nodes containing only one class are preferable
- To learn small trees, we have to split by attributes which **provide the most information** and produce the least successor nodes

## Question:

**How can we express this thought as a mathematical formula?**

# Measure of Impurity: Entropy

## Answer:

- **Entropy**  $H$  (greek capital  $\eta$ ) introduced by CLAUDE E. SHANNON
- The idea of entropy originates in the field of **information theory**

## Properties of entropy:

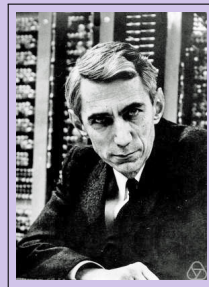
- Entropy reaches its minimum if both classes are equally distributed
- **Pure datasets have minimal entropy**
- Therefore, split by attributes which reduce the entropy the most



## Portrait: CLAUDE SHANNON

**CLAUDE ELWOOD SHANNON** (April 30, 1916 – February 24, 2001) was an American mathematician, electrical engineer, computer scientist and cryptographer. He is commonly known as the "father of information theory".

Shannon contributed to the field of cryptanalysis for national defense of the United States during World War II, and his mathematical theory of information became very well cited and laid the foundation for the field of information theory.



*(Wikipedia)*



# Computation of the Entropy

- The entropy  $H$  is a measure of chaos in the data and is measured in bits
- Example:** Consider the two classes  $\mathcal{C}_1 = A$  and  $\mathcal{C}_2 = B$

$$H(\{A, A, A, A, A, A, A, A\}) \rightarrow 0 \quad \text{Bits}$$

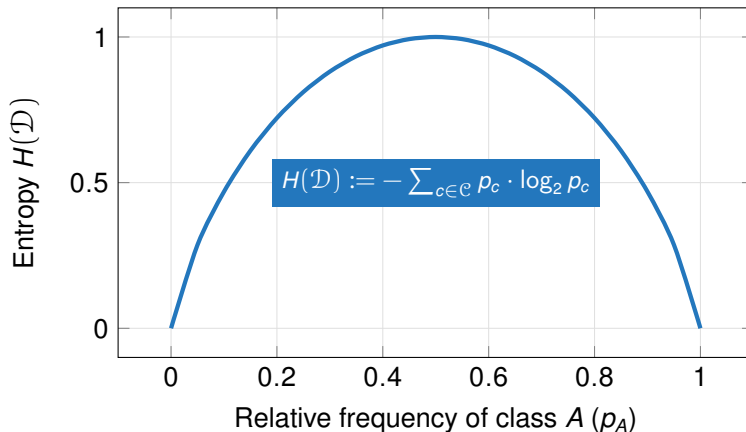
$$H(\{A, A, A, A, A, A, B, B\}) \rightarrow 0.81 \quad \text{Bits}$$

$$H(\{A, A, A, A, B, B, B, B\}) \rightarrow 1 \quad \text{Bit}$$

$$H(\{A, A, B, B, B, B, B, B\}) \rightarrow 0.81 \quad \text{Bits}$$

$$H(\{B, B, B, B, B, B, B, B\}) \rightarrow 0 \quad \text{Bits}$$

## Computation of the Entropy (Ctd.)



# Computation of the Entropy (Ctd.)

## Entropy formula:

$$H(\mathcal{D}) := - \sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c \quad (1)$$

( $p_c$  denotes the relative frequency of class  $c \in \mathcal{C}$ )

**Weather data:**  $\mathcal{C} := \{\text{yes}, \text{no}\} \Rightarrow p_{\text{yes}} = 9/14$  and  $p_{\text{no}} = 5/14$

$$H(\mathcal{D}) = - \sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c = -(9/14 \cdot \log_2 9/14 + 5/14 \cdot \log_2 5/14) = \mathbf{0.9403}$$

## Quality of the Split: Average Entropy

- We still do not know which attribute to use for the first split
- **Idea:** Calculate the entropy after each potential split and compare

**Average Entropy** after splitting by attribute  $A$ :

$$H(\mathcal{D}|A) := \sum_{v \in \text{dom}(A)} \frac{|\mathcal{D}_{A=v}|}{|\mathcal{D}|} \cdot H(\mathcal{D}_{A=v}) \quad (2)$$

$A$	Attribute
$\text{dom}(A)$	Possible values attribute $A$ can take (domain of $A$ )
$ \mathcal{D}_{A=v} $	Number of examples satisfying $A = v$

## Example: Average Entropy

### Weather data:

- Potential split by attribute `Outlook`
- We compute the entropy of the resulting subsets:

$$H(\mathcal{D}_{\text{Outlook=sunny}}) = -\left(\frac{2}{5} \cdot \log_2\left(\frac{2}{5}\right) + \frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right)\right) = 0.9710$$

$$H(\mathcal{D}_{\text{Outlook=rainy}}) = -\left(\frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right) + \frac{2}{5} \cdot \log_2\left(\frac{2}{5}\right)\right) = 0.9710$$

$$H(\mathcal{D}_{\text{Outlook=overcast}}) = -\left(\frac{4}{4} \cdot \log_2\left(\frac{4}{4}\right) + \frac{0}{4} \cdot \log_2\left(\frac{0}{4}\right)\right) = 0$$

We directly see that  $H(\mathcal{D}_{\text{Outlook=overcast}}) = 0$  (*do not compute it explicitly!*)

## Example: Average Entropy (Ctd.)

### Weather data:

- Potential split by attribute Outlook
- We compute a weighted average:

$$\begin{aligned} H(\mathcal{D}|\text{Outlook}) &= \sum_{v \in \text{dom}(\text{Outlook})} \frac{|\mathcal{D}_{\text{Outlook}=v}|}{|\mathcal{D}|} \cdot H(\mathcal{D}_{\text{Outlook}=v}) \\ &= 5/14 \cdot 0.9710 + 5/14 \cdot 0.9710 + 4/14 \cdot 0 \\ &= \mathbf{0.6936} \end{aligned}$$

# Information Gain

- We have to calculate the average entropy for all attributes
- The difference of entropy before and after split is called the **information gain (IG)**
- Select the attribute with the highest IG

Attribute	$H_{\text{before}}$	$H_{\text{after}}$	IG
Outlook	0.9403	0.6936	0.2464
Temperature	0.9403	0.9111	0.0292
Humidity	0.9403	0.7885	0.1518
Wind	0.9403	0.8922	0.0481

- **Here:** Attribute Outlook maximizes the information gain

# Training Data after the Split by Attribute Outlook

Outlook	Temperature	Humidity	Wind	PlayGolf
sunny	hot	high	weak	no
sunny	hot	high	strong	no
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
sunny	mild	normal	strong	yes
rainy	mild	high	weak	yes
rainy	cool	normal	weak	yes
rainy	cool	normal	strong	no
rainy	mild	normal	weak	yes
rainy	mild	high	strong	no
overcast	cool	normal	strong	yes
overcast	hot	high	weak	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes

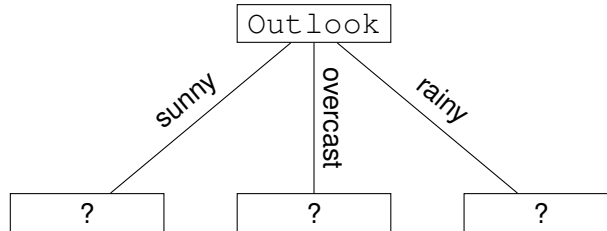
- The table on the left displays the dataset  $\mathcal{D}$  after the split by attribute Outlook
- We obtain three subsets (*one per attribute value*)
- Attribute Outlook is removed in the current branch of the tree (**Why?**)



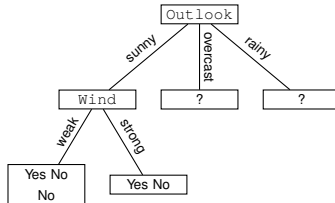
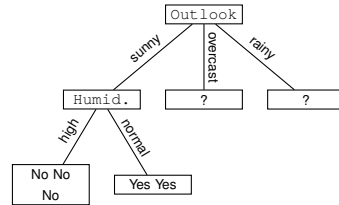
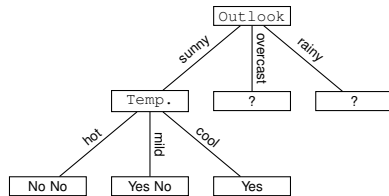
# How to proceed?

- The algorithm is recursively applied to the resulting subsets:
  - 1 Calculate entropy before and after each potential split
  - 2 Calculate the information gain for each attribute
  - 3 Choose the attribute with maximum information gain for the split
  - 4 In the current branch: Do not consider the attribute chosen anymore
  - 5 **Recursion** ↻ (go to 1)
- Recursion stops as soon as the subsets are pure  $\Rightarrow$  **Danger:** ☠ **Overfitting** ☠
- In the example above, the subset  $\mathcal{D}_{\text{Outlook}=\text{overcast}}$  is already pure
- This algorithm is referred to as **ID3 (Iterative Dichotomizer)**

# Step by Step: Construction of the Tree

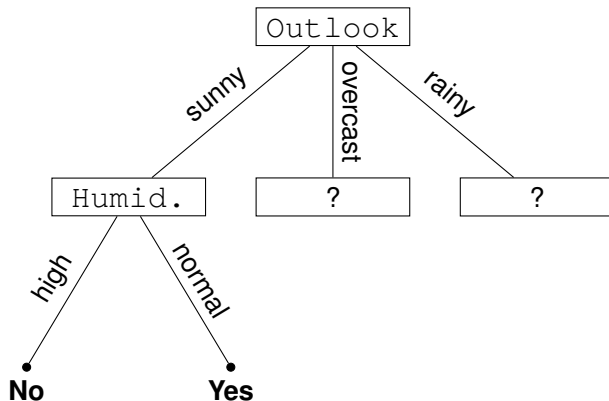


## Step by Step: Construction of the Tree (Ctd.)

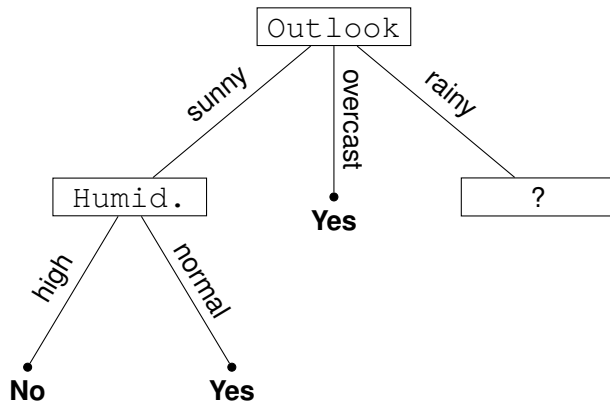


- $IG(\text{Temperature}) = 0.571$
- $IG(\text{Humidity}) = \mathbf{0.971}$
- $IG(\text{Wind}) = 0.020$

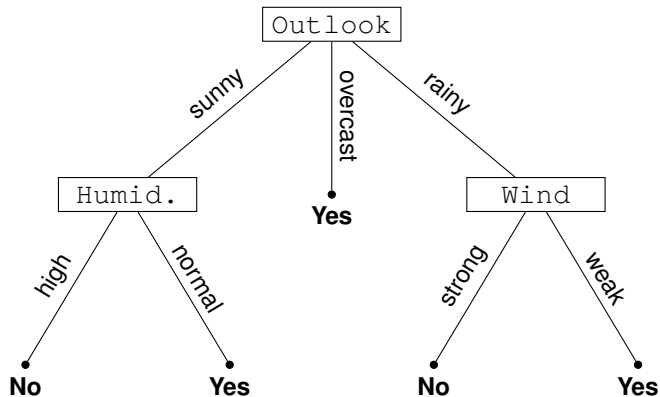
## Step by Step: Construction of the Tree (Ctd.)



## Step by Step: Construction of the Tree (Ctd.)



## Step by Step: Construction of the Tree (Ctd.)



## Section: Extensions and Variants

Other Measures of Impurity  
Highly-branching Attributes  
Numeric Attributes  
Regression Trees

# An Alternative to Information Gain: Gini Index

## Gini index:

$$\text{Gini}(\mathcal{D}) := \sum_{c \in \mathcal{C}} p_c \cdot (1 - p_c) = 1 - \sum_{c \in \mathcal{C}} p_c^2 \quad (3)$$

- Gini index and entropy always **produce the same decision tree**
- Often used as a default in machine learning libraries (**Why?**)
- Used e. g. in **CART (Classification and Regression Trees)**
- **Gini gain** could be defined analogously to IG (*this is usually not done*)





## Why not use the Error as a splitting Criterion?

- The bias towards pure leaves is **not strong enough**
- **Example:**

Split 1	40 of A	60 of A
	60 of A	40 of B
Split 2		

Error before the split:

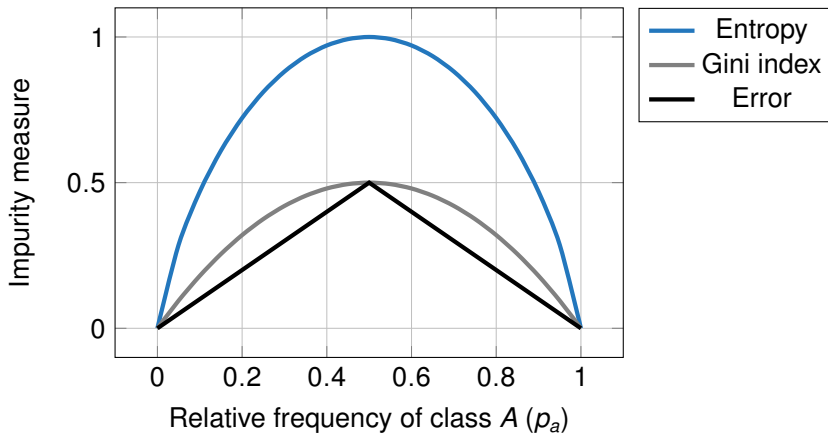
**20 %**

Error after the split:

**20 %**

**Both splits don't improve the error.  
But together they give a perfect split!**

## Summary: Impurity Measures



# Highly-branching Attributes

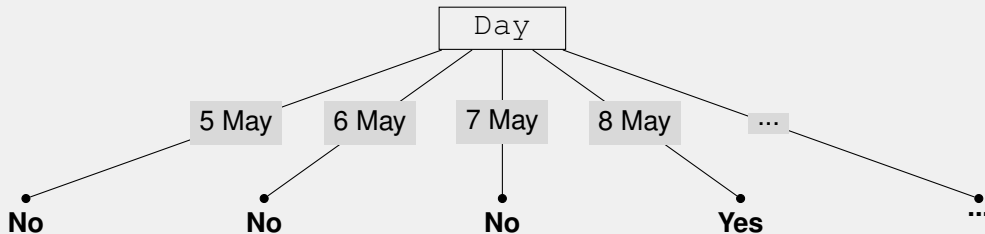
**Problem: Attributes with a large number of values are problematic, since the leaves are not 'backed' with sufficient data examples.**

In extreme cases there is only one example per node (e. g. *IDs*)

This may lead to:

- **Overfitting** (*selection of attributes which are not optimal for prediction*)
- **Fragmentation** (*data is fragmented into too many small sets*)

## Example: Highly-branching Attributes



- Entropy before the split is 0.9403, entropy after the split is 0
- $IG(\mathcal{D}, \text{Day}) = 0.9403$
- Attribute `Day` would be chosen for the split  $\Rightarrow$  **Bad for prediction** 🦴

# Intrinsic Information

**Solution:** Calculate the **intrinsic information (IntI)**:

$$\text{IntI}(\mathcal{D}, A) := - \sum_{v \in \text{dom}(A)} \frac{|\mathcal{D}_{A=v}|}{|\mathcal{D}|} \cdot \log_2 \frac{|\mathcal{D}_{A=v}|}{|\mathcal{D}|} \quad (4)$$

- Attributes with high *IntI* are **less useful** as they result in high fragmentation
- New splitting heuristic: **Gain ratio (GR)**

$$\text{GR}(\mathcal{D}, A) := \frac{\text{IG}(\mathcal{D}, A)}{\text{IntI}(\mathcal{D}, A)} \quad (5)$$

## Example: Intrinsic Information

- Intrinsic information for attribute `Day`:

$$\text{IntI}(\mathcal{D}, \text{Day}) = 14 \cdot \left(-1/14 \cdot \log_2(1/14)\right) = \mathbf{3.807} \quad (6)$$

- Gain ratio for attribute `Day`:

$$\text{GR}(\mathcal{D}, \text{Day}) = \frac{0.9403}{3.807} = \mathbf{0.246} \quad (7)$$

**Attention:** In this case, attribute `Day` would still be chosen. Be careful which features you include in the training dataset! **(Feature engineering is important!)**

## Handling of numeric Attributes

- Usually, only **binary splits** are considered, e. g.:
  - Temperature  $< 48$
  - **Not:**  $24 \leq \text{Temperature} \leq 31$  (*produces three subsets*)
- To support non-binary splits, the attribute is **not removed** and can be chosen again

**Problem:** There is an **infinite number** of possible splits! **Splitting on numeric attributes is computationally more demanding!**

**Solution:** Discretize the range using a fixed step size

## Example I: Handling numeric Attributes

- Consider the attribute `Temperature`
- Unlike before, the attribute takes **numerical values** instead of discrete ones:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- $\text{Temperature} < 71.5$   
 #yes: 4 | #no: 2
- $\text{Temperature} \geq 71.5$   
 #yes: 5 | #no: 3

$$H(\mathcal{D}|\text{Temp.}) = \frac{6}{14} \cdot H(\text{Temp.} < 71.5) + \frac{8}{14} \cdot H(\text{Temp.} \geq 71.5) = \mathbf{0.939}$$



## Example II: Handling numeric Attributes

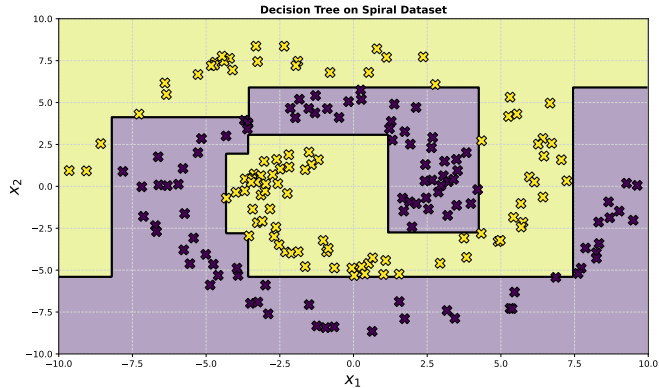
### Dataset:

Taxable income	60	70	75	85	90	95	100	120	125	220
Class label	No	No	No	Yes	Yes	Yes	No	No	No	No

### Evaluation of splits:

Split point	55		65		72		80		87		92		97		110		122		172		230	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini index	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400		0.420	

# Decision Tree on a Spiral Dataset





# Regression Trees

- We can use decision trees to predict **continuous target variables**
- Predict the average value of all examples in the leaf
- Split the data such that the variance in the leaves is minimized

## Standard deviation reduction (SDR):

$$\text{SDR}(\mathcal{D}, A) := \text{SD}(\mathcal{D}) - \sum_{v \in \text{dom}(A)} \frac{|\mathcal{D}_{A=v}|}{|\mathcal{D}|} \cdot \text{SD}(\mathcal{D}_{A=v}) \quad (8)$$



## Regression Trees (Ctd.)

**Caveat: It is important to specify a termination criterion when using decision trees in regression tasks, otherwise we get a single data point per leaf!**

**For example:** Continue splitting the dataset until the standard deviation falls below a specified threshold for the first time

## Section: Ensemble Methods

Introduction to Ensembles  
Bootstrap Aggregating (Bagging)  
Randomization  
Random Forests and ExtraTrees  
Boosting

# Introduction Ensemble Methods

**Key Idea:** Do not learn a single classifier, but a **set of classifiers**. Then combine the predictions of the base classifiers to obtain the ensemble prediction

**Problem:** How can we induce multiple classifiers from a single dataset without getting the same classifier over and over again? **We want to have diverse classifiers, otherwise the ensemble is useless!**

- Basic techniques are: **Bagging**, **boosting**, and **stacking**
- We shall have a closer look into the bagging technique

# What is the Advantage of an Ensemble?

- Let 25 **independent** base classifiers be given
- **Independence assumption:** The probability of a single classifier misclassifying an example **does not** depend on other classifiers in the ensemble
- This condition is usually not fully satisfied in practice (**Why?**)
- Each individual classifier in the ensemble is assumed to have an error rate of  $\varepsilon := 0.35$

**Question: What is the error rate of the ensemble?**

## What is the Advantage of an Ensemble? (Ctd.)

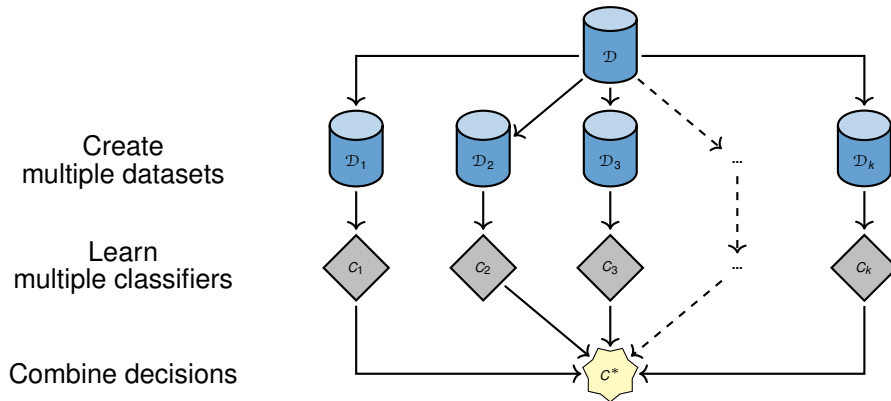
- The prediction of the ensemble is given by the **majority vote**,  
i. e. the class with the most votes is predicted
- The ensemble makes a wrong prediction **if the majority is wrong**,  
i. e. at least 13 base classifiers misclassify the example
- This probability is computed according to the **binomial distribution**

$$\varepsilon_{\text{ensemble}} := \sum_{k=13}^{25} \binom{25}{k} \cdot \varepsilon^k \cdot (1 - \varepsilon)^{25-k} \approx 0.06 \quad (9)$$

- We see:  $\varepsilon_{\text{ensemble}} \approx 0.06 \ll 0.35 = \varepsilon$



# Bootstrap Aggregating (Bagging)



# Creating the Bootstrap Samples

- **Question:** How to generate multiple datasets which are different?
- **Solution:** Use sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Some examples may appear **in more than one set**
- Some examples may appear **more than once** in one set
- Some examples may **not appear at all** in a set

# Bagging Algorithm

**Input:** Training set  $\mathcal{D}$ , number of base classifiers  $K$

1 **Training phase:**

2 **forall**  $k \in \{1, 2, \dots, K\}$  **do**

3     Draw a bootstrap sample  $\mathcal{D}_k$  with replacement from  $\mathcal{D}$

4     Learn a base classifier  $C_k$  (e. g. a decision tree) from  $\mathcal{D}_k$

5     Add the classifier  $C_k$  to the ensemble

6 **end**

7 **Prediction phase:**

8 **forall** *unlabeled instances* **do**

9     Get predictions from all classifiers  $C_k$  ( $1 \leq k \leq K$ )

10 **end**

11 **return** *Class which receives the majority of votes (combined classifier  $C^*$ )*

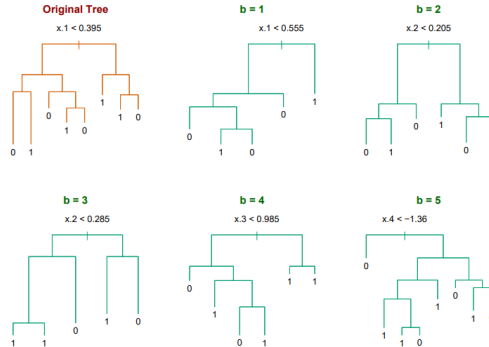
# Bagging Variations

We have considered bootstrap samples of equal size, drawn with replacement

Also conceivable are the following scenarios:

- ❶ **Varying the size** of the bootstrap samples
- ❷ Sampling **without replacement** (**Pasting**)
- ❸ **Sampling of features**, not instances (*not all features are available*)
- ❹ Creating **heterogeneous ensembles** comprising different types of base classifiers (*e. g. neural networks, decision trees, support vector machines*)

# Bagged Decision Trees



cf. [HASTIE.2008], page 284

# Randomization

- Why not **randomizing the algorithm** instead of the data?
- Some algorithms already do that: For example neural networks randomly initialize the weights before training
- Especially greedy algorithms can be randomized:
  - Pick from the options **randomly**, instead of picking the best one
  - E. g. decision trees: Do not choose the attribute with the highest information gain, but select a split attribute randomly

**A random forest combines randomization and bagging**

# Random Forest Algorithm

- A **random forest** is an ensemble of decision trees
- It combines **bagging** and **random attribute subset selection**
- We grow a decision tree from each bootstrap sample
- We select the best splitting attribute among a random subset of attributes

At each step a random forest selects the best splitting attribute from a randomly chosen subset of features, but the globally best feature **may not** be available.

# Random Forest Algorithm

**Input:** Training set  $\mathcal{D}$ , number of base classifiers  $K$

1 **Training phase:**

2 **for**  $k \in \{1, 2, \dots, K\}$  **do**

3     Create a bootstrap sample from  $\mathcal{D}$  (e. g. with replacement)  $\Rightarrow$  **Bagging**

4     **begin**

5         Grow the tree

6         At every node: Randomly choose a subset of attributes to be considered for the split

7          $\Rightarrow$  **Randomization**

8     **end**

9     Add tree  $C_k$  to the ensemble

10 **end**

11 **Prediction phase:**

12 **forall** *unlabeled instances* **do**

13     Get predictions from all classifiers  $C_k$  ( $1 \leq k \leq K$ )

14 **end**

15 **return** *Class which receives the majority of votes (combined classifier  $C^*$ )*



## ExtraTrees (Randomization 2.0)

- One more step of randomization  $\Rightarrow$  **Extremely Randomized Trees** (ExtraTrees)
- The general approach is the same as for random forests, **but:**
  - Instead of choosing the optimal split point...
  - ...it is selected randomly
  - Each decision tree is grown without having to calculate entropy
- It is **much faster** due to less computation

**The large number of classifiers compensates for suboptimal splits**

# General Idea of Boosting

- **Boosting** is a powerful ensemble method
- We introduce the **AdaBoost** (*adaptive boosting*) algorithm
- Again we train a specified number  $K$  of base classifiers, sometimes also referred to as **weak learners** because a single classifier usually has high bias
- In contrast to bagging, the base learners have to be trained **in sequence**
- Each classifier is trained on a weighted form of the dataset, where the weights depend on the performance of the previous classifiers (*misclassified points get a higher weight*)
- In the following consider a binary classification problem with labels  $y_n \in \{-1, +1\}$

# AdaBoost Algorithm – Training Phase

**Input:** Training set  $\mathcal{D}$  of size  $N$ , number of base classifiers  $K$

1 Initialize the weights  $w_n^{(1)} \leftarrow 1/N$  for all  $n = 1, 2, \dots, N$

2 **foreach**  $k \in \{1, 2, \dots, K\}$  **do**

3     Fit a classifier  $h_k(\mathbf{x})$  to the training data using the current example weights  $w_n^{(k)}$

4     Compute the weighted error measure  $\varepsilon_k$  and the classifier weight  $\alpha_k$  according to

$$\varepsilon_k \leftarrow \sum_{n=1}^N w_n^{(k)} \mathbf{1}(h_k(\mathbf{x}_n) \neq y_n) \quad \text{and} \quad \alpha_k \leftarrow \frac{1}{2} \log \left( \frac{1 - \varepsilon_k}{\varepsilon_k} \right)$$

5     Update the data weight coefficients for all  $n = 1, 2, \dots, N$ :

$$w_n^{(k+1)} \leftarrow w_n^{(k)} \exp(-\alpha_k y_n h_k(\mathbf{x}_n))$$

6     Normalize the weights so that their sum is equal to 1:

$$w_n^{(k+1)} \leftarrow \frac{w_n^{(k+1)}}{\sum_{j=1}^N w_j^{(k+1)}}$$

7 **end**

## AdaBoost Algorithm – Prediction Phase

- The predictions are made using the final model function

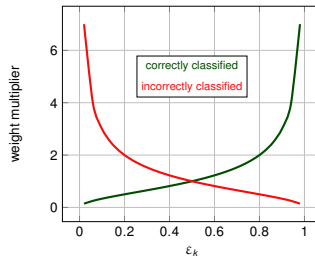
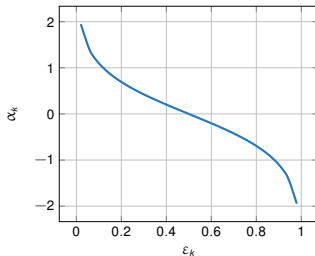
$$h_{\theta}(\mathbf{x}) := \text{sign} \left( \sum_{k=1}^K \alpha_k h_k(\mathbf{x}) \right)$$

- We see that the ensemble prediction is obtained by weighting the prediction of each base classifier with its individual classifier weight  $\alpha_k$

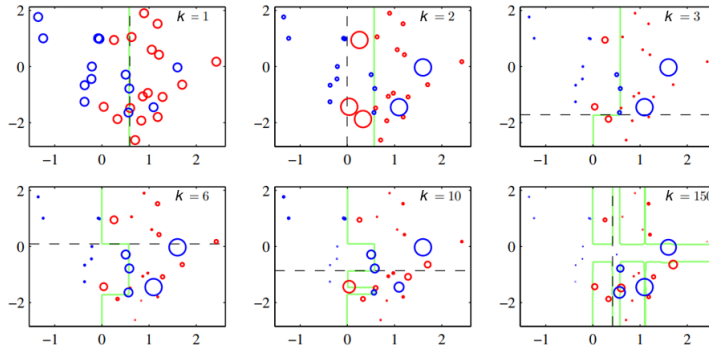
**Please note:** According to its definition, the value of  $\alpha_k$  can also be negative, i. e. if the base classifier has poor performance we still consider it, but with inverted sign (prediction)!

# AdaBoost Algorithm – Prediction Phase

- The classifier weights  $\alpha_k$  grow and shrink exponentially
- The classifier weight  $\alpha_k$  is 0 if  $\varepsilon_k = 1/2$  (*random guessing*)
- For high error rates  $\Rightarrow$  **Do the opposite of what the classifiers predicts**



# AdaBoost Example



cf. [BISHOP.2006], page 660

## Section: Wrap-Up

Summary  
Recommended Literature  
Self-Test Questions  
Lecture Outlook

# Summary

## 1 Decision trees:

- The construction of decision trees is guided by an **impurity measure**, e. g. the entropy measure  $H$  or the Gini index
- Recursively select features which **maximize the information gain**
- Decision trees can handle **numeric attributes** and **continuous output**
- Be careful with highly-branching attributes

## 2 Ensembles:

- Usually, ensembles allow for a significant error reduction
- **Bagging**: Sample diverse datasets from an underlying dataset
- **Random forests** combine bagging and randomization
- **Boosting**: Train classifiers in sequence on a weighted form of the dataset



# Recommended Literature

1 [MURPHY.2012], chapter 16.2,  
pages 546 – 554

2 [HASTIE.2008], chapter 9.2,  
pages 305 – 317

3 [QUINLAN.1993]

(For free PDF versions, see list in GitHub readme!)



# Self-Test Questions

- 1 What is the inductive bias? What is the inductive bias of decision trees?
- 2 Explain what OCCAM's razor is.
- 3 What does entropy measure? How do you compute the information gain?
- 4 True or false? *'Pure datasets have maximal entropy.'*
- 5 What is the advantage of ensemble methods?
- 6 What is bagging?
- 7 Explain how a random forest works.
- 8 How does AdaBoost work? Explain the algorithm.

# What's next...?

- |             |                                   |             |                              |
|-------------|-----------------------------------|-------------|------------------------------|
| <b>I</b>    | Machine Learning Introduction     | <b>IX</b>   | Evaluation                   |
| <b>II</b>   | Optimization Techniques           | <b>X</b>    | Decision Trees               |
| <b>III</b>  | Bayesian Decision Theory          | • <b>XI</b> | Support Vector Machines      |
| <b>IV</b>   | Non-parametric Density Estimation | <b>XII</b>  | Clustering                   |
| <b>V</b>    | Probabilistic Graphical Models    | <b>XIII</b> | Principal Component Analysis |
| <b>VI</b>   | Linear Regression                 | <b>XIV</b>  | Reinforcement Learning       |
| <b>VII</b>  | Logistic Regression               | <b>XV</b>   | Advanced Regression          |
| <b>VIII</b> | Deep Learning                     |             |                              |

# Thank you very much for the attention!

**\*\*\* Artificial Intelligence and Machine Learning \*\*\***

**Topic:** Decision Trees and Ensemble Methods

**Term:** Summer term 2025

**Contact:**

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

[daniel.wehner@sap.com](mailto:daniel.wehner@sap.com)

## Do you have any questions?