*** Applied Machine Learning Fundamentals ***
Decision Trees and Ensembles

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2023/2024

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	**Classification I**
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

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 Tree
An alternative Tree

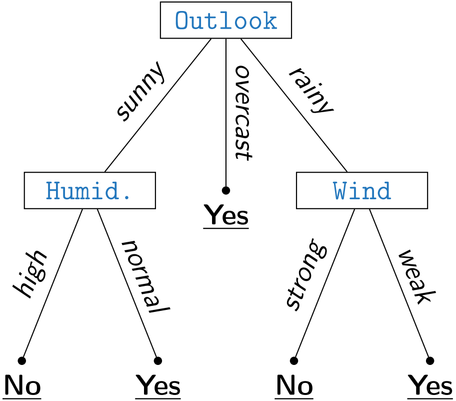
What are Decision Trees?

- Decision trees are induced in a **supervised** fashion
- Originally invented by *Ross Quinlan* (1986)
- Decision trees are grown **recursively** → *'divide-and-conquer'*
- A decision tree consists of:

Nodes	Each node corresponds to an attribute test
Edges	One edge per possible test outcome
Leaves	Class label to predict

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	???

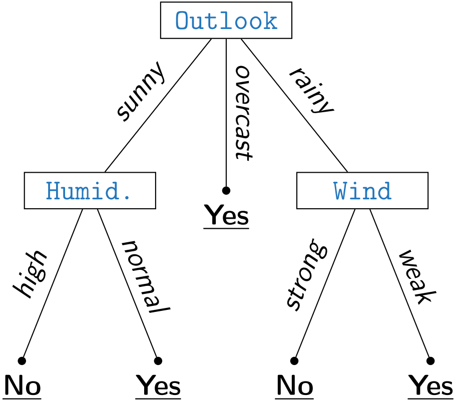


Classifying new Instances

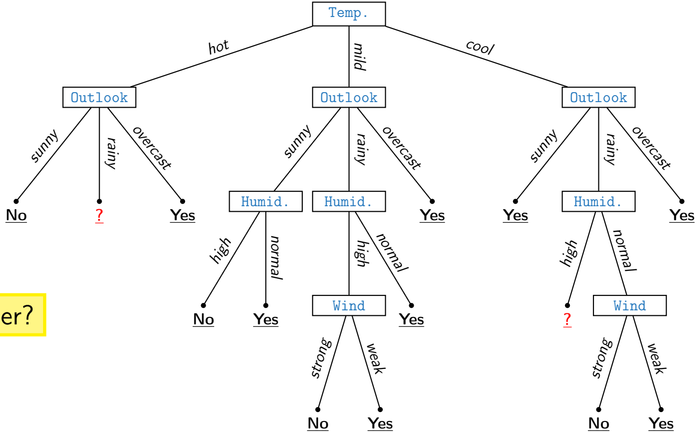
- Suppose we get a new instance:

`Outlook`	rainy
`Temperature`	mild
`Humidity`	normal
`Wind`	strong

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



Another Decision Tree...



Is this one better?

Section:

Iterative Dichotomizer (ID3)

Inductive Bias
Entropy and Information Gain
ID3 Algorithm

Important

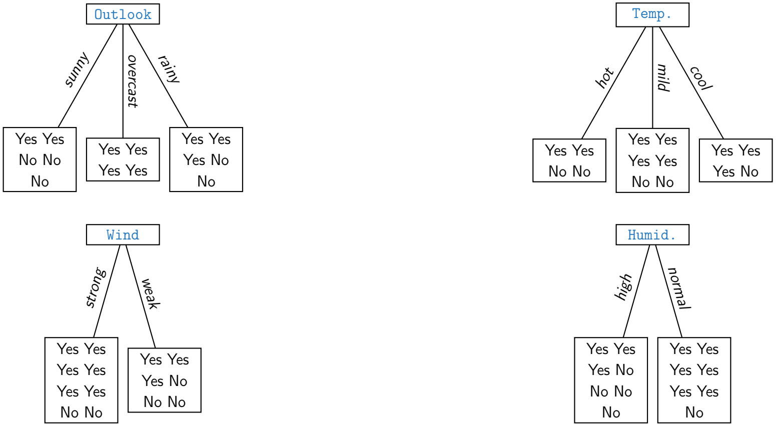
Inductive Bias of Decision Trees

- Complex models tend to **overfit** the data and **do not generalize well**
- Small decision trees are preferred

> **Occam's razor**:
> **'More things should not be used than are necessary.'**

- **Prefer the simplest hypothesis that fits the data!**

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



Outlook
sunny: Yes Yes No No No
overcast: Yes Yes Yes Yes
rainy: Yes Yes Yes No No

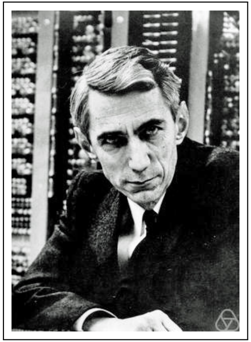
Temp.
hot: Yes Yes No No
mild: Yes Yes Yes Yes No No
cool: Yes Yes Yes No

Wind
strong: Yes Yes Yes Yes Yes Yes No No
weak: Yes Yes Yes No No No

Humid.
high: Yes Yes Yes No No No No
normal: Yes Yes Yes Yes Yes Yes No

Finding a proper Attribute

- Simple and small trees are preferred
  - Data in successor node should be **as pure as possible**
  - I. e. nodes containing one class only are preferable
- **Question:** How can we express this thought as a mathematical formula?
- **Answer:**
  - **Entropy** (*Claude E. Shannon*)
  - Originates in the field of **information theory**



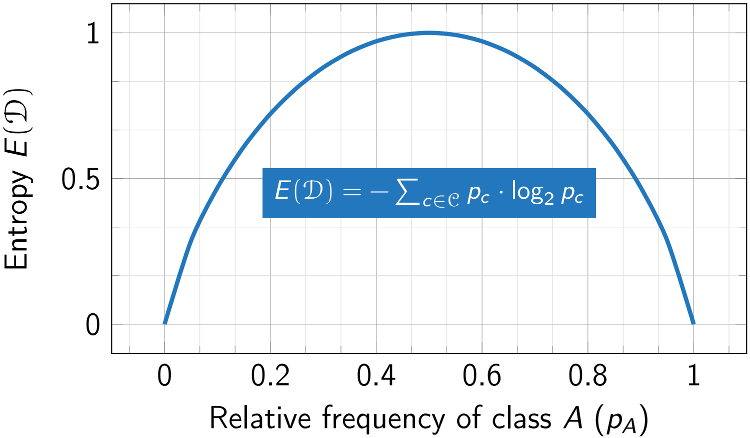
Measure of Impurity: Entropy

- Entropy is a measure of chaos in the data (measured in bits)
- **Example:** Consider two classes A and B ($\mathcal{C} = \{A, B\}$)

$$\begin{array}{lll}
E(\{\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}\}) & \rightarrow 0 & Bits \\
E(\{\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, B, B\}) & \rightarrow 0.81 & Bits \\
E(\{\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, \boldsymbol{A}, B, B, B, B\}) & \rightarrow 1 & Bit \\
E(\{\boldsymbol{A}, \boldsymbol{A}, B, B, B, B, B, B\}) & \rightarrow 0.81 & Bits \\
E(\{B, B, B, B, B, B, B, B\}) & \rightarrow 0 & Bits
\end{array}$$

If both classes are equally distributed, the entropy function E reaches its maximum. Pure data sets have minimal entropy.

Measure of Impurity: Entropy (Ctd.)



$$E(\mathcal{D}) = -\sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c$$

Measure of Impurity: Entropy (Ctd.)

Entropy formula:

$$E(\mathcal{D}) = -\sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c \tag{1}$$

- p_c denotes the relative frequency of class $c \in \mathcal{C}$
- **Weather data:**

$$\mathcal{C} = \{yes, no\} \qquad \text{i. e.} \qquad p_{yes} = {}^{9}/_{14} \quad \text{and} \quad p_{no} = {}^{5}/_{14}$$

$$E(\mathcal{D}) = -\sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c = -\left({}^{9}/_{14} \cdot \log_2 {}^{9}/_{14} + {}^{5}/_{14} \cdot \log_2 {}^{5}/_{14}\right) = 0.9403$$

Quality of the Split: Average Entropy

- We still don't know which attribute to use for the split
- Calculate the entropy after each potential split
- **Average Entropy** after splitting by attribute $\mathtt{A}$:

$$E(\mathcal{D}, \mathtt{A}) = \sum_{v \in \mathrm{dom}(\mathtt{A})} \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \cdot E(\mathcal{D}_{\mathtt{A}=v}) \tag{2}$$

- Legend:

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

Quality of the Split: Average Entropy (Ctd.)

Example: Attribute `Outlook`

$$E(\mathcal{D}, \texttt{Outlook}) = \sum_{v \in \text{dom}(\texttt{Outlook})} \frac{|\mathcal{D}_{\texttt{Outlook}=v}|}{|\mathcal{D}|} \cdot E(\mathcal{D}_{\texttt{Outlook}=v})$$

$$= {}^5/_{14} \cdot 0.9710 + {}^5/_{14} \cdot 0.9710 + {}^4/_{14} \cdot 0 \qquad = 0.6936$$

$$E(\mathcal{D}_{\texttt{Outlook}=sunny}) = -({}^2/_5 \cdot \log_2({}^2/_5) + {}^3/_5 \cdot \log_2({}^3/_5)) \qquad = 0.9710$$

$$E(\mathcal{D}_{\texttt{Outlook}=rainy}) = -({}^3/_5 \cdot \log_2({}^3/_5) + {}^2/_5 \cdot \log_2({}^2/_5)) \qquad = 0.9710$$

$$E(\mathcal{D}_{\texttt{Outlook}=overcast}) = -({}^4/_4 \cdot \log_2({}^4/_4) + {}^0/_4 \cdot \log_2({}^0/_4)) \qquad = 0$$

Information Gain

- We have calculated the entropy before and after the split
- The difference of both is called the **information gain (IG)**
- Select the attribute with the highest IG

Attribute	E_{before}	E_{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

- Attribute `Outlook` maximizes IG
- After the split: Remove attribute `Outlook`

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

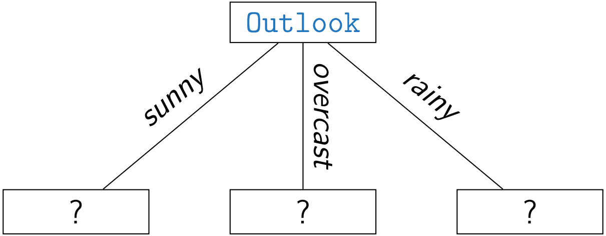
- Data set $\mathcal{D}$ after the split
- We obtain three subsets (one per attribute value)
- Attribute `Outlook` is removed

Important

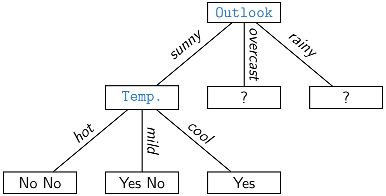
How to proceed?

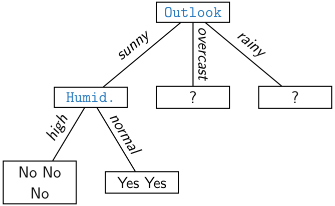
- The algorithm is recursively applied to the resulting subsets
  1. Calculate entropy (before and after the split)
  2. Calculate information gain for each attribute
  3. Choose the attribute with max. information gain for the split
  4. In the current branch: Do not consider the attribute any more
  5. **Recursion** ↺ (`Go to 1`)
- Recursion stops as soon as the subset is pure
- In the example above the subset $\mathcal{D}_{\texttt{Outlook}=\mathit{overcast}}$ is already pure
- This algorithm is referred to as ID3 (Iterative Dichotomizer)

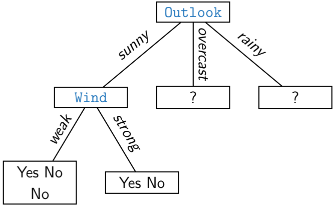
Step by Step: Construction of the Tree



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

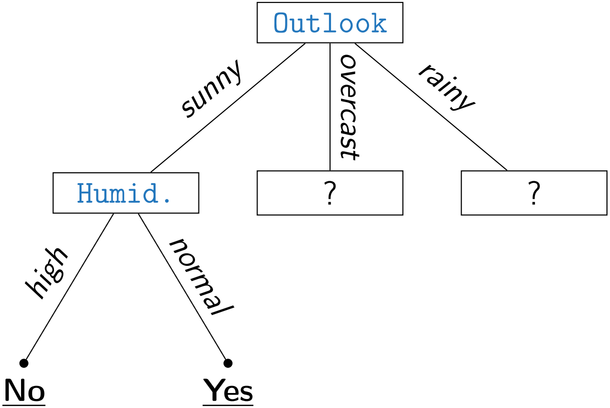




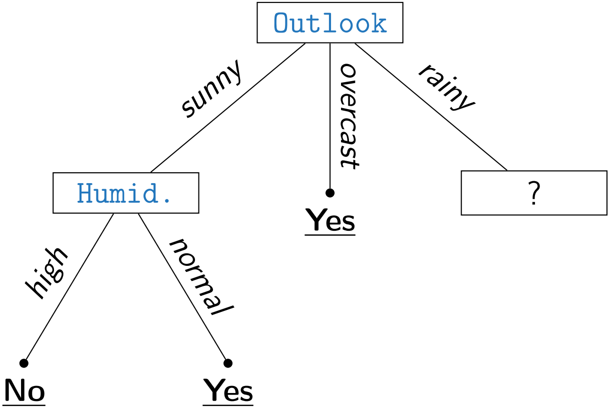


- $IG(\texttt{Temperature}) = 0.571$
- $IG(\texttt{Humidity}) = \mathbf{0.971}$
- $IG(\texttt{Wind}) = 0.020$

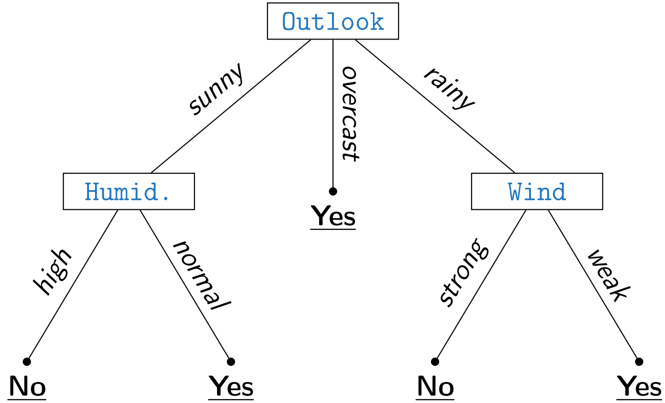
Step by Step: Construction of the Tree (Ctd.)



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



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



Algorithm 1: ID3 Algorithm (Iterative Dichotomizer)

```
Input: Training set $\mathcal{D}$, Attribute list $Attr\_List$
Create a node $N$
if all tuples in $\mathcal{D}$ have class $c$ then
    return $N$ as leaf node labeled with class $c$ // subset is pure
if $|Attr\_List| = 0$ then
    return $N$ as leaf node labeled with majority class in $\mathcal{D}$ // no attribute left
Find best split attribute $\mathtt{A}^*$ and label node $N$ with $\mathtt{A}^*$
$Attr\_List \longleftarrow Attr\_List \setminus \{\mathtt{A}^*\}$ // remove attribute from list
forall $v \in dom(A^*)$ do
    Let $\mathcal{D}_{\mathtt{A}^*=v}$ be the set of tuples in $\mathcal{D}$ that satisfy $\mathtt{A}^* = v$
    if $|\mathcal{D}_{A^*=v}| = 0$ then
        Attach leaf labeled with majority class in $\mathcal{D}$ to node $N$
    else
        Attach node returned by $ID3(\mathcal{D}_{\mathtt{A}^*=v}, Attr\_List)$ // recursion
return $N$
```

Section:

Extensions and Variants

Other Measures of Impurity

Highly-branching Attributes

Numeric Attributes

Regression Trees

An Alternative to Information Gain: Gini Index

Gini index:

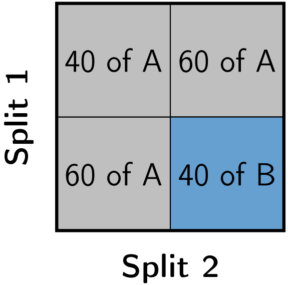
$$Gini(\mathcal{D}) = \sum_{c \in \mathcal{C}} p_c \cdot (1 - p_c) = 1 - \sum_{c \in \mathcal{C}} p_c^2 \tag{3}$$

- Used e. g. in **CART (Classification and Regression Trees)**
- **Gini gain** could be defined analogously to IG

  (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 without splitting:

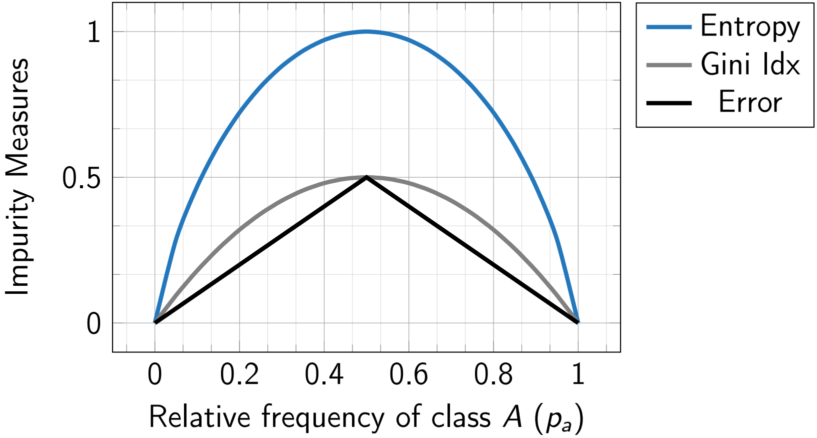
20 %

Error after splitting:

20 %

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

Summary: Impurity Measures



Highly-branching Attributes

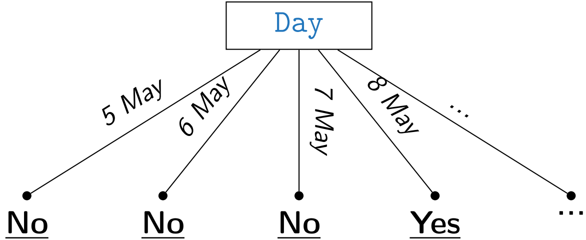
Attributes with a large number of values are problematic, since the leaves are not 'backed' with sufficient data examples.

In extreme cases 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)

Highly-branching Attributes (Ctd.)



- Entropy before was 0.9403, Entropy after split is 0
- $\text{IG}(\mathcal{D}, \texttt{Day}) = 0.9403$
- Attribute Day would be chosen for the split ⇒ **Bad for prediction** ☠

Highly-branching Attributes (Ctd.)

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

$$IntI(\mathcal{D}, \mathtt{A}) = -\sum_{v \in \mathrm{dom}(\mathtt{A})} \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \cdot \log_2 \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \tag{4}$$

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

$$GR(\mathcal{D}, \mathtt{A}) = \frac{IG(\mathcal{D}, \mathtt{A})}{IntI(\mathcal{D}, \mathtt{A})} \tag{5}$$

Highly-branching Attributes (Ctd.)

- Intrinsic information for attribute Day:

$$\mathit{IntI}(\mathcal{D}, \mathtt{Day}) = 14 \cdot (-{}^{1}/_{14} \cdot \log_2({}^{1}/_{14})) = 3.807 \tag{6}$$

- Gain ratio for attribute Day:

$$\mathit{GR}(\mathcal{D}, \mathtt{Day}) = \frac{0.9403}{3.807} = 0.246 \tag{7}$$

In this case the attribute Day would still be chosen. Be careful what features to include into the training data set! **(feature engineering is important!)**

Handling numeric Attributes

- Usually, only **binary splits** are considered, e. g.:
  - $\texttt{Temperature} < 48$
  - $\texttt{CPU} > 24$
  - **Not:** $24 \leqslant \texttt{Temperature} \leqslant 31$
- To support multiple splits, the attribute is **not removed**
  (the same attribute can be used again for another split)
- **Problem:** There is an **infinite number** of possible splits!
- **Solution:** Discretize range (fixed step size, ...)
- **Splitting on numeric attributes is computationally demanding!**

Important

Handling numeric Attributes (Ctd.)

- Consider the attribute `Temperature`:
  Use **numerical values** instead of discrete values like *cool*, *mild*, *hot*:

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

- $\texttt{Temperature} < 71.5$
  yes: 4 | no: 2

- $\texttt{Temperature} \geqslant 71.5$
  yes: 5 | no: 3

$$E(\mathcal{D}, \texttt{Temp.}) = {}^{6}\!/\!{}_{14} \cdot E(\texttt{Temp.} < 71.5) + {}^{8}\!/\!{}_{14} \cdot E(\texttt{Temp.} \geqslant 71.5) = 0.939$$

Handling numeric Attributes (Ctd.)

Sorted Values									
No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income									
60	70	75	85	90	95	100	120	125	220

Split	55		65		72		80		87		92		97		110		122		172		230	
	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>	⩽	>
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	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400		0.420	

Regression Trees

- Prediction of continuous variables
- Predict average value of all examples in the leaf
- Split the data such that variance in the leaves is minimized
- **Termination criterion is important, otherwise single point per leaf!**

Standard deviation reduction (SDR):

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

Section:
Ensemble Methods

Introduction to Ensembles
Bagging and Randomization

Introduction Ensemble Methods

- **Key Idea**: Don't learn a single classifier, but a **set of classifiers**
- Combine the predictions of the single classifiers to obtain the final prediction

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

- Basic techniques:
  - Bagging
  - Boosting (not covered)
  - Stacking (not covered)

What is the Advantage?

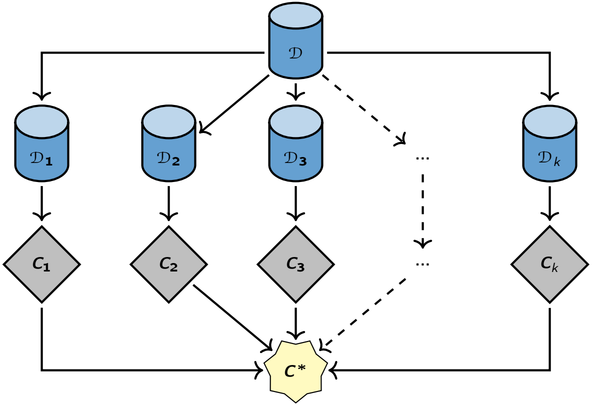
- Consider the following:
  - There are 25 **independent** base classifiers
  - Independence assumption: Probability of misclassification **does not** depend on other classifiers in the ensemble
  - Usually, this assumption does not fully hold in practice
  - Each classifier has an error rate of $\varepsilon = 0.35$
- The ensemble makes a wrong prediction **if the majority is wrong** ($\Rightarrow$ i. e. at least 13)

$$\varepsilon_{ensemble} = \sum_{u=13}^{25} \binom{25}{u} \cdot \varepsilon^{u} \cdot (1 - \varepsilon)^{25-u} \approx 0.06 \ll \varepsilon \tag{9}$$

Important

Bagging: General Approach

Bagging ≙ **B**ootstrap **Agg**regat**ing**



Create
multiple data sets

Build
multiple classifiers

Combine decisions

Creating the Bootstrap Samples

- How to generate multiple data sets 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**

Algorithm 2: Bagging Algorithm

```
Input: Training set $\mathcal{D}$, number of base classifiers $k$
Training:
forall $u \in \{1, 2, \ldots, k\}$ do
    Draw a bootstrap sample $\mathcal{D}_u$ with replacement from $\mathcal{D}$
    Learn a classifier $C_u$ from $\mathcal{D}_u$
    Add classifier $C_u$ to the ensemble
Prediction:
forall unlabeled instances do
    Get predictions from all classifiers $C_u$
return Class which receives the majority of votes (combined classifier $C^*$)
```

Bagging Variations

- The bootstrap samples had equal size and were drawn with replacement
- Also conceivable:
  1. **Varying the size** of the bootstrap samples
  2. Sampling **without replacement** ⇒ Pasting
  3. **Sampling of features**, not instances
     - Not all features are available in all bootstrap samples
     - This is how **random forests** work
  4. Creating **heterogeneous ensembles**
     (neural networks, decision trees, support vector machines, ...)

Bagged Decision Trees

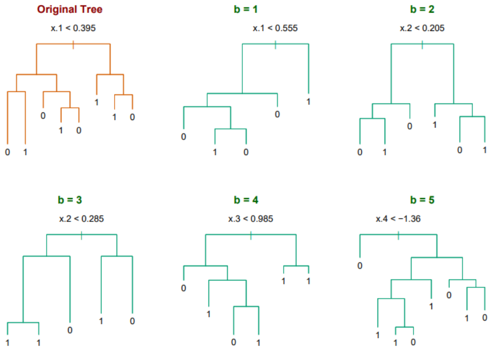


Figure: Bagged decision trees; cf. Hastie 2008, page 284

Randomization

- Why not **randomizing the algorithm** instead of the data?
- Some algorithms already do that: E. g. neural networks (random initialization of weights)
- Especially greedy algorithms can be randomized:
  - Pick from the options **randomly**, instead of picking the best one
  - E. g. decision trees: Do not choose attribute with highest information gain

A random forest combines randomization and bagging.

Random Forest Algorithm

- Ensemble of decision trees
- Combines **bagging** and **random attribute subset selection**
- Build decision tree from a bootstrap sample
- Select best split attribute among a random subset of f attributes

> A random forest selects the best splitting attributes from the set of features available, but the globally best features **may not** be available.

Algorithm 3: Random Forest Algorithm

```
Input: Training set $\mathcal{D}$, number of base classifiers $k$
Training:
for $u \in \{1, 2, \ldots, k\}$ do
    Create a bootstrap sample from $\mathcal{D}$ (e. g. with replacement) $\Rightarrow$ Bagging
    begin
        Grow the tree
        At every node: Randomly choose $f$ attributes to be considered for the split
        $\Rightarrow$ Randomization
        Do not prune tree $C_u$
    Add tree $C_u$ to the ensemble
Prediction:
forall unlabeled instances do
    Get predictions from all classifiers $C_u$
return Class which receives the majority of votes (combined classifier $C^*$)
```

ExtraTrees (Randomization 2.0)

- One more step of randomization ⇒ Extremely Randomized Trees
- The general approach is the same as for random forests
  But:
  - Instead of choosing the optimal split point...
  - ...it is selected randomly
  - The 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.

Section:

Wrap-Up

Summary

Self-Test Questions

Lecture Outlook

Summary

- **Decision trees:**
  - The construction of decision trees is guided by an **impurity measure**, e. g. entropy or Gini
  - Recursively select features which **maximize the information gain**
  - Decision trees can handle **numeric attributes** and **continuous output**
- **Ensembles:**
  - Usually, ensembles allow for a significant error reduction
  - **Bagging:** Sample diverse data sets from underlying data
  - **Random forests** combine bagging and randomization

Important

Self-Test Questions

1. What is an 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 data sets have maximal entropy.'*
5. What is the advantage of ensemble methods?
6. What is bagging?
7. Explain what a random forest does.

What's next...?

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	Classification I
Unit VI	**Evaluation**
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Decision Trees and Ensembles
Term: Winter term 2023/2024

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?