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

Daniel Wehner

SAP SE

October 22, 2019

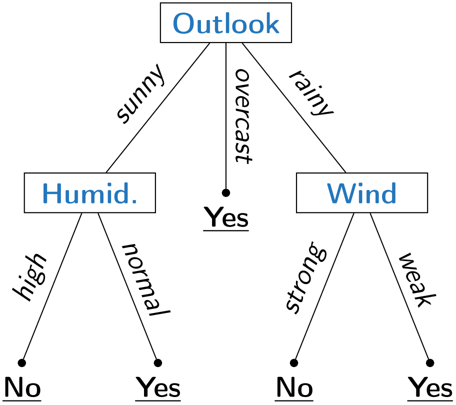
Agenda October 22, 2019

Section:

Introduction

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	???



What are Decision Trees?

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

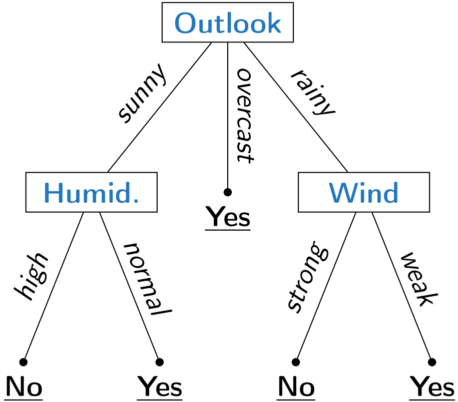
Nodes	Each node corresponds to an attribute test
Edges	One edge per possible test outcome
Leaves	Class label to predict

Classifying new Instances

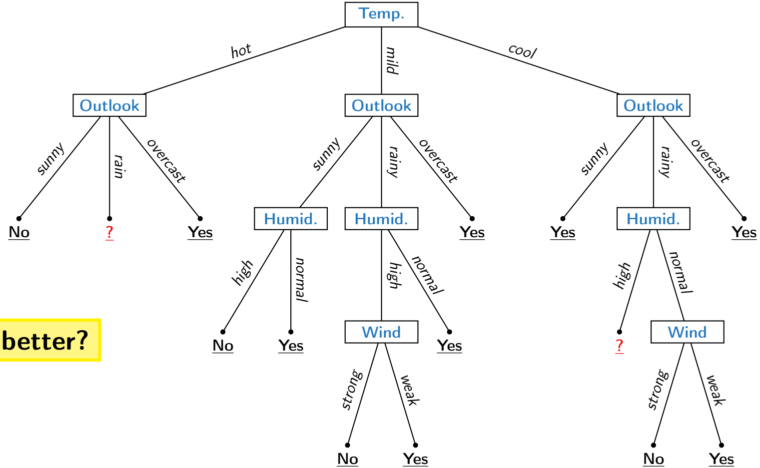
- Suppose we get a new instance:

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

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



Another Decision Tree...



Is this one better?

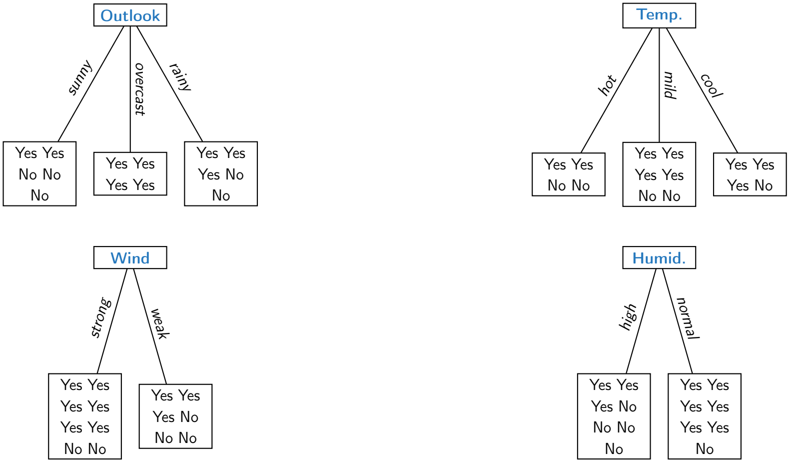
Inductive Bias of Decision Trees

- 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.'

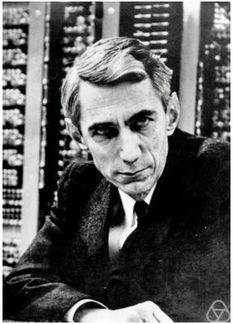
- **Prefer the simplest hypothesis that fits the data!**

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



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**



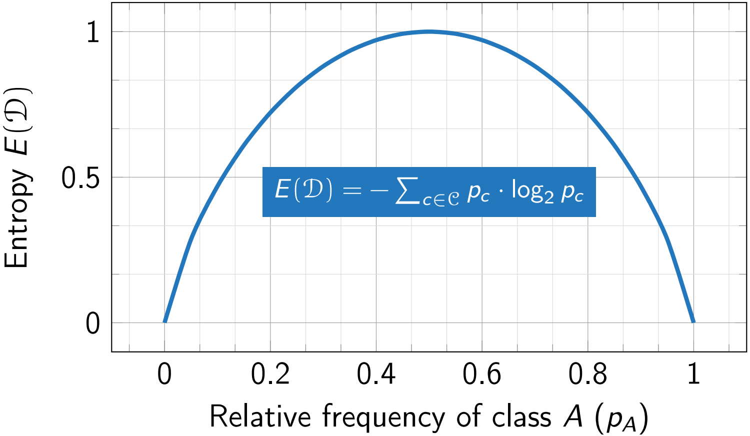
Measure of Impurity: Entropy

- 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.)



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:**

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

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

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`

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

$$= 5/14 \cdot 0.9710 + 5/14 \cdot 0.9710 + 4/14 \cdot 0 \quad = \mathbf{0.6936}$$

$$E(\mathcal{D}_{\texttt{Outlook}=sunny}) = -(2/5 \cdot \log_2(2/5) + 3/5 \cdot \log_2(3/5)) \quad = 0.9710$$

$$E(\mathcal{D}_{\texttt{Outlook}=rainy}) = -(3/5 \cdot \log_2(3/5) + 2/5 \cdot \log_2(2/5)) \quad = 0.9710$$

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

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`

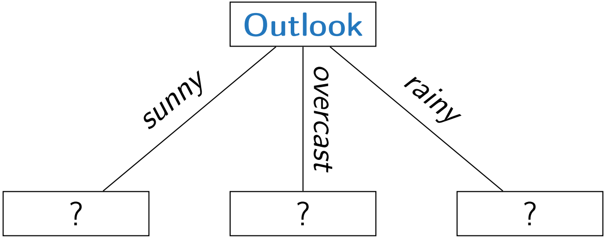
~~Outlook~~	Temperature	Humidity	Wind	PlayGolf
~~sunny~~	hot	high	weak	no
~~sunny~~	hot	high	strong	no
~~sunny~~	mild	high	weak	no
~~sunny~~	cool	normal	weak	yes
~~sunny~~	mild	normal	strong	yes
~~rainy~~	mild	high	weak	yes
~~rainy~~	cool	normal	weak	yes
~~rainy~~	cool	normal	strong	no
~~rainy~~	mild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

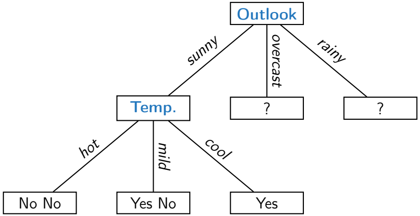
How to proceed?

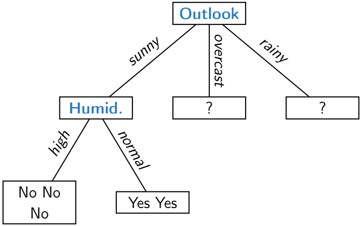
- The algorithm is recursively applied to the resulting subsets
  1. Calculate entropy (before and after the split)
  2. Calculate information gain for each attribute
  3. Choose the attribute with max. information gain for the split
  4. In the current branch: Do not consider the attribute any more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

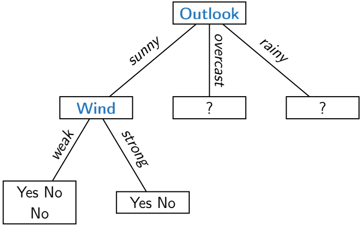
Step by Step: Construction of the Tree



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

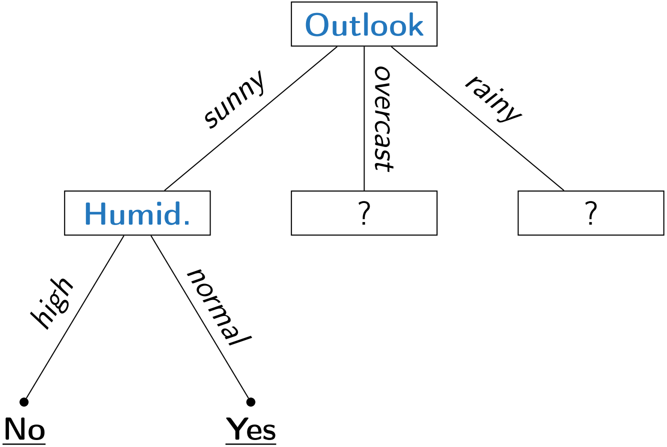




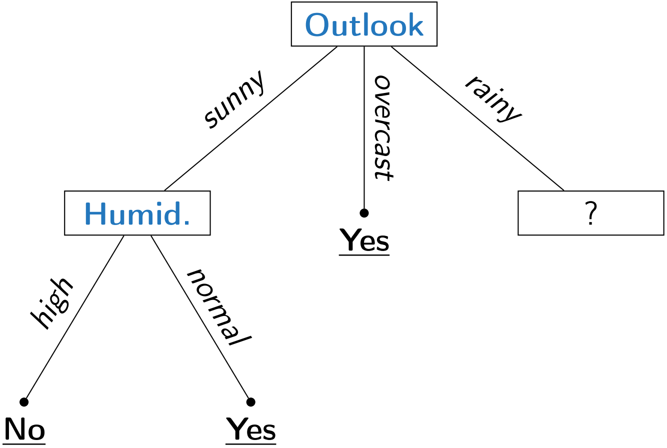


- $IG(\texttt{Temperature}) = 0.571$
- $IG(\texttt{Humidity}) = \mathbf{0.971}$
- $IG(\texttt{Wind}) = 0.020$

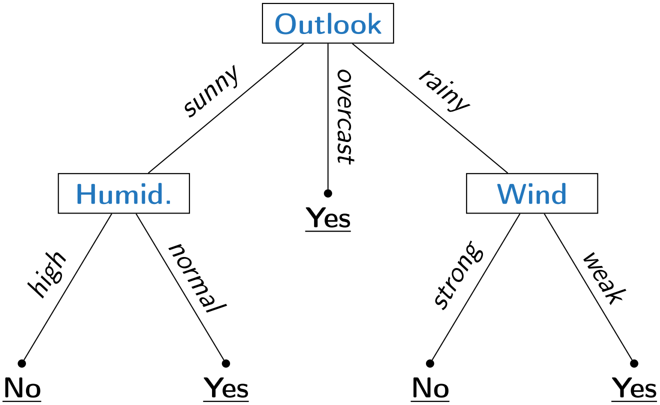
Step by Step: Construction of the Tree (Ctd.)



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

if $|Attr\_List| = 0$ **then**

  return N as leaf node labeled with majority class in $\mathcal{D}$

Find best split attribute $\mathtt{A}^*$ and label node N with $\mathtt{A}^*$

$Attr\_List \longleftarrow Attr\_List \setminus \{\mathtt{A}^*\}$

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 $ID3(\mathcal{D}_{\mathtt{A}^*=v}, Attr\_List)$

return N

An Alternative to Information Gain: Gini Index

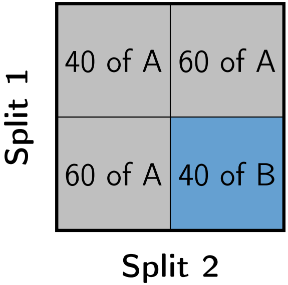
Gini index:

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

- 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:**



Error without splitting:

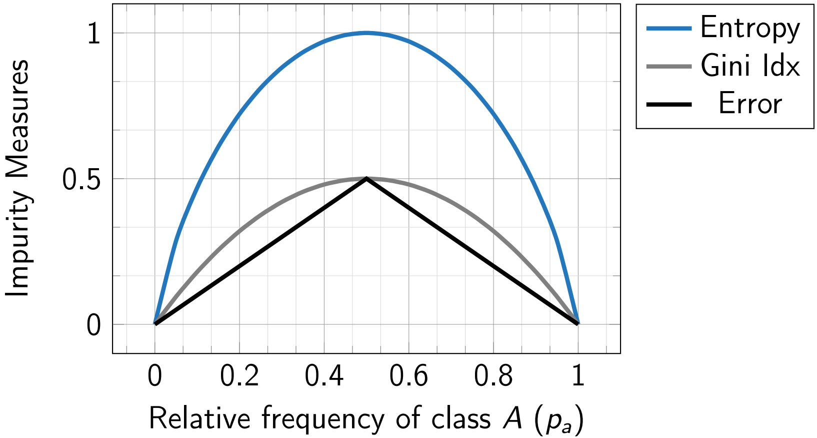
20 %

Error after splitting:

20 %

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

Summary: Impurity Measures



Highly-Branching Attributes

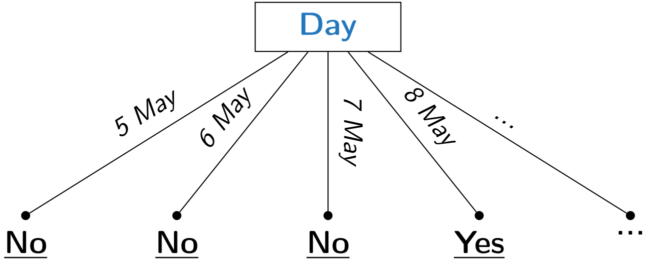
Attributes with a large number of values are problematic, since the leaves are not ‘backed’ with sufficient data examples.

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
- $\mathrm{IG}(\mathcal{D}, \mathtt{Day}) = 0.9403$
- Attribute Day would be chosen for the split $\Rightarrow$ **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`:

$$IntI(\mathcal{D}, \texttt{Day}) = 14 \cdot \left(-{}^{1}\!/{}_{14} \cdot \log_2({}^{1}\!/{}_{14})\right) = \mathbf{3.807} \tag{6}$$

- Gain ratio for attribute `Day`:

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

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!**

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

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

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	
	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$
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):

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

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
  - Stacking

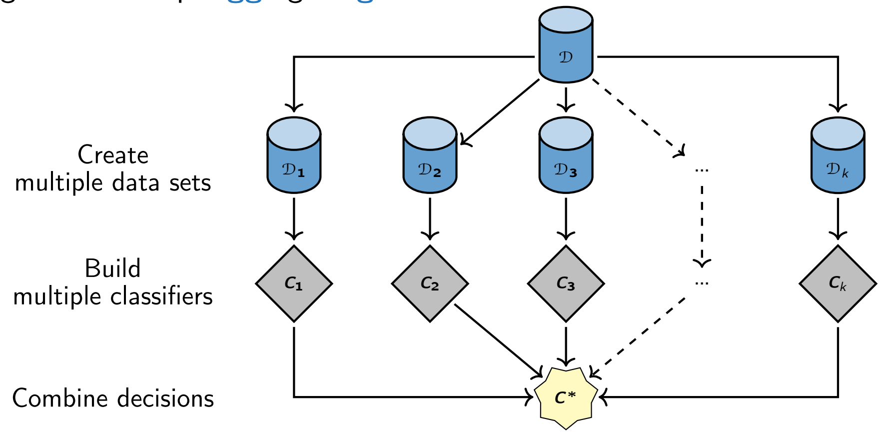
What is the Advantage?

- 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 $\epsilon = 0.35$
- The ensemble makes a wrong prediction **if the majority is wrong** ($\Rightarrow$ i. e. at least 13)

$$\epsilon_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \cdot \epsilon^i \cdot (1-\epsilon)^{25-i} \approx \mathbf{0.06} \ll \epsilon \tag{9}$$

Bagging: General Approach

Bagging $\widehat{=}$ **B**ootstrap **Agg**regat**ing**



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 n

Training:

forall $i \in \{1, 2, \ldots, k\}$ **do**

- Draw a bootstrap sample $\mathcal{D}_i$ with replacement from $\mathcal{D}$
- Learn a classifier C_i from $\mathcal{D}_i$
- Add classifier C_i to the ensemble

Prediction:

forall *unlabeled instances* **do**

- Get predictions from all classifiers C_i

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** $\Rightarrow$ **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, ...)

Section:

Wrap-Up

Summary

Lecture Overview

Unit I: Machine Learning Introduction

Self-Test Questions

Recommended Literature and further Reading

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Decision Trees and Ensembles
Date: October 22, 2019

Contact:
Daniel Wehner (D062271)
SAP SE
daniel.wehner@sap.com

Do you have any questions?