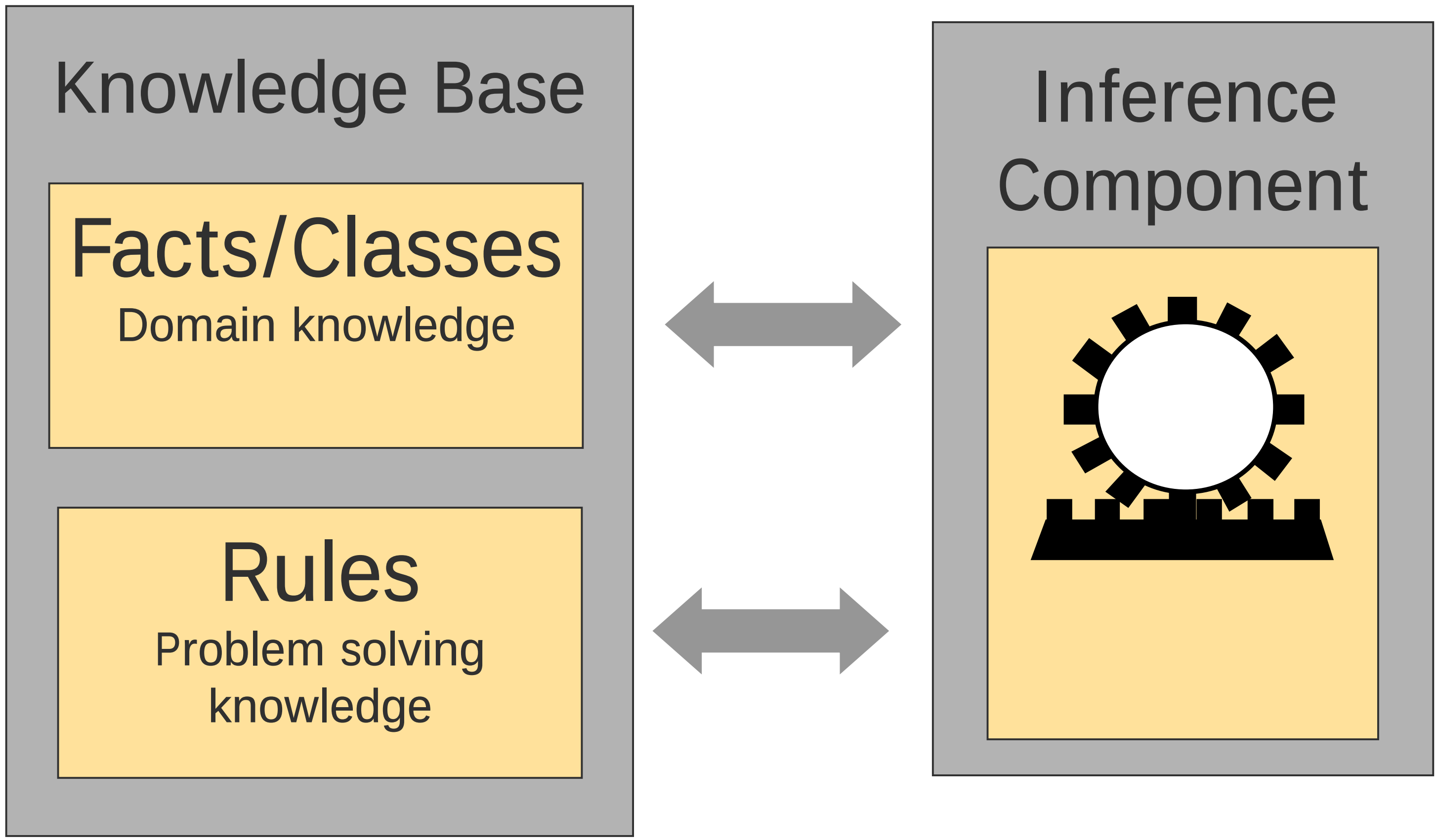
University of Applied Sciences and Arts Northwestern Switzerland
School of Business

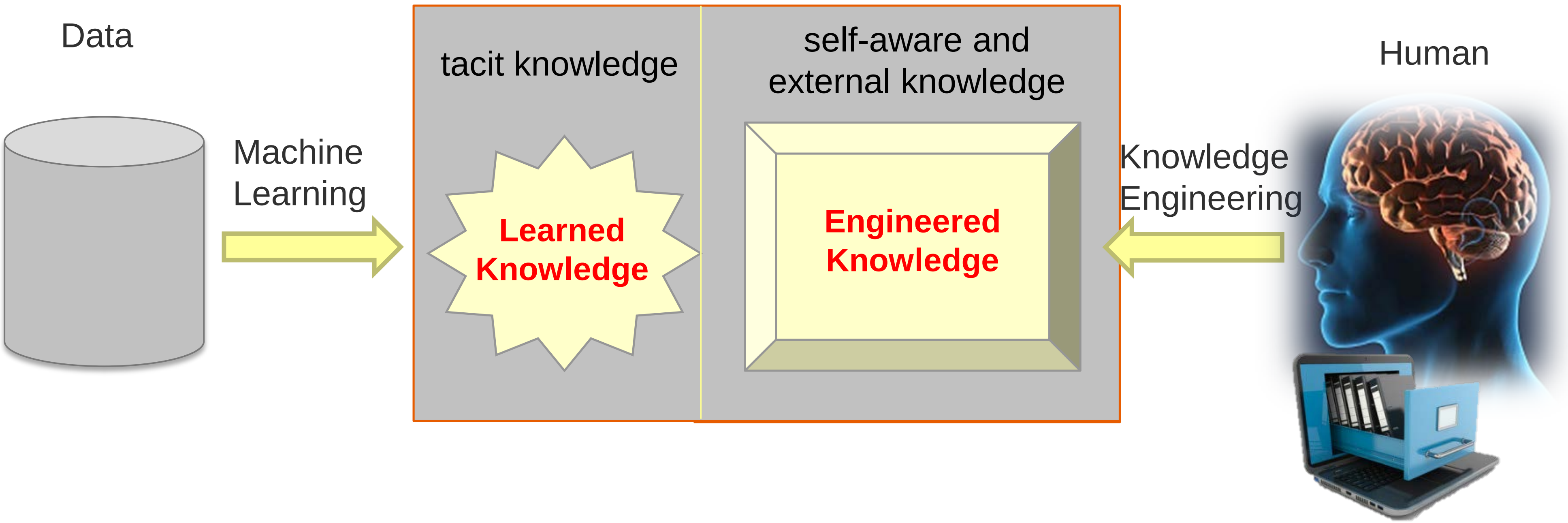
Machine Learning: Learning Rules

Knut Hinkelmann

Knowledge-Based Systems (Rules & Facts)



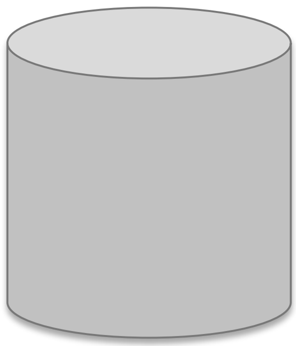
Knowledge Sources in a Knowledge Base



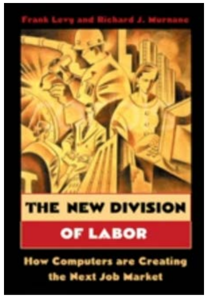
Creating Knowledge Bases

- **Knowledge Engineering:** Human experts build knowledge base
  - For knowledge we are aware of
  - Knowledge is comprehensible
  - For knowledge we need to be sure of (e.g. compliance rules)

- **Machine Learning:** automatic creation of knowledge from example data
  - Can solve complex tasks for which
    - knowledge is not known
    - knowledge is tacit
  - Reliance on real-world data instead of pure intuition
  - Requires large sets of data
  - Can adapt to new situations (collect more data)



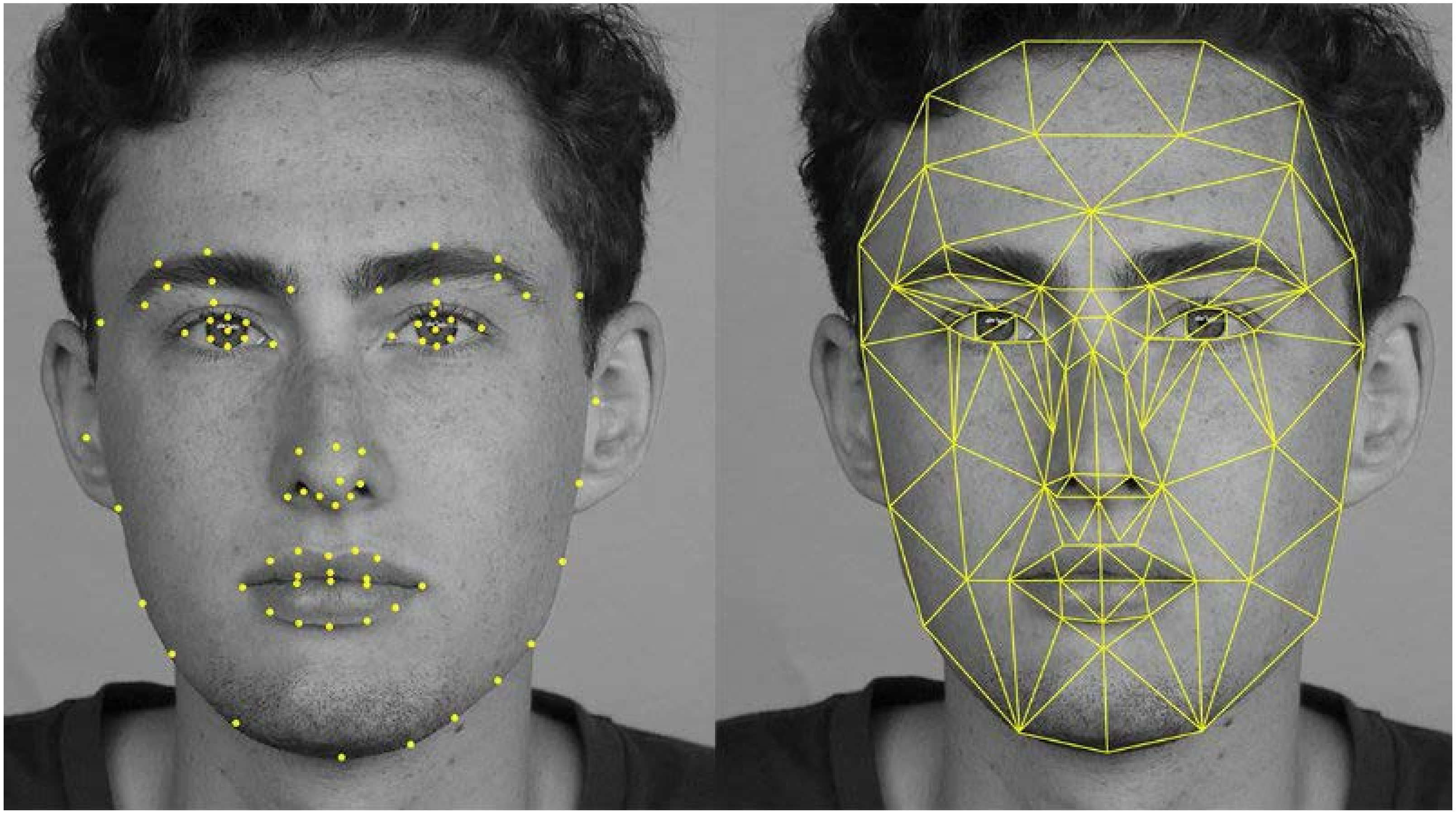
Self-driving Cars



"… it is hard to imagine discovering the set of rules that can replicate the driver's behavior."

(Levy & Murnane 2006)

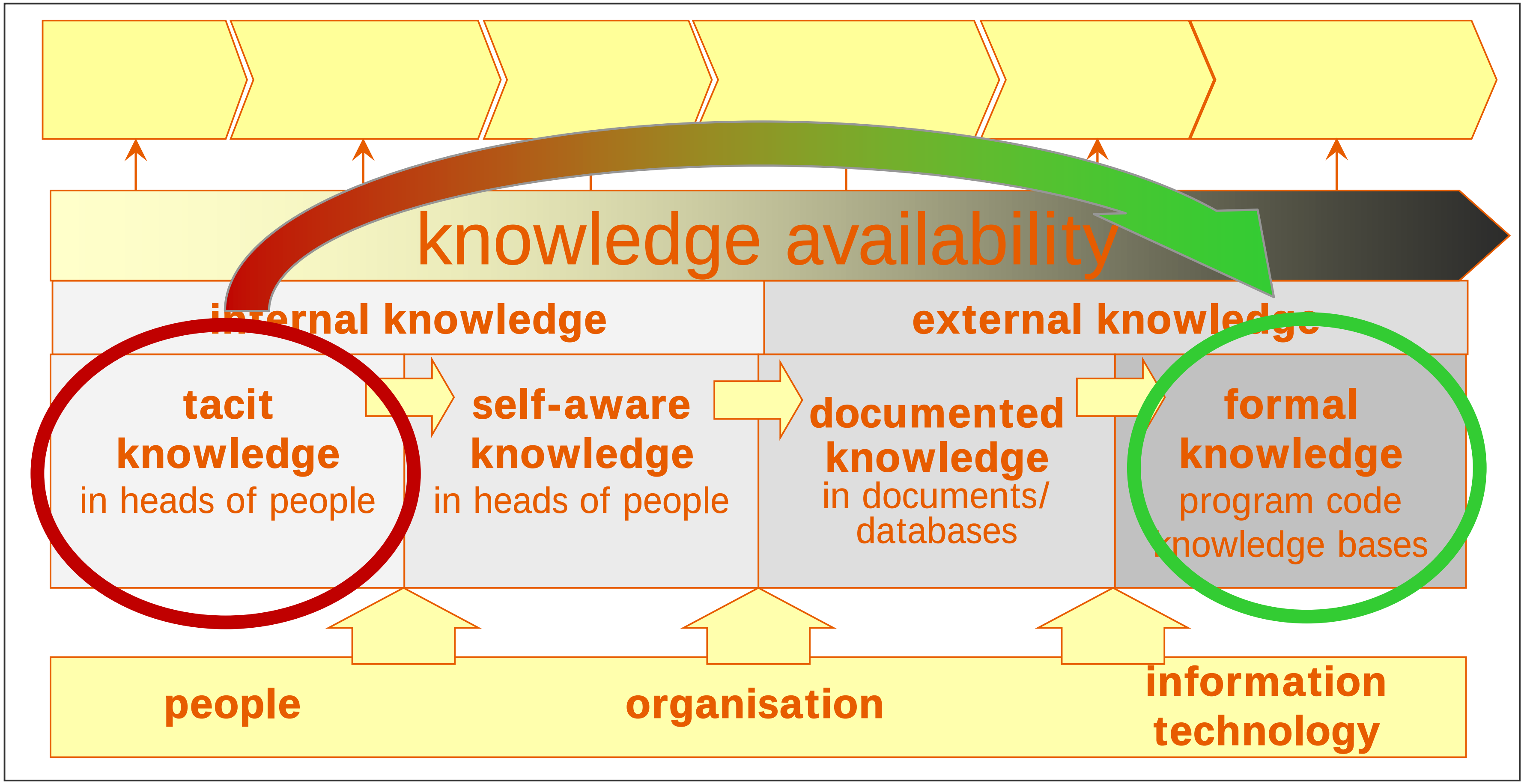
Face Recognition



Spam Filter

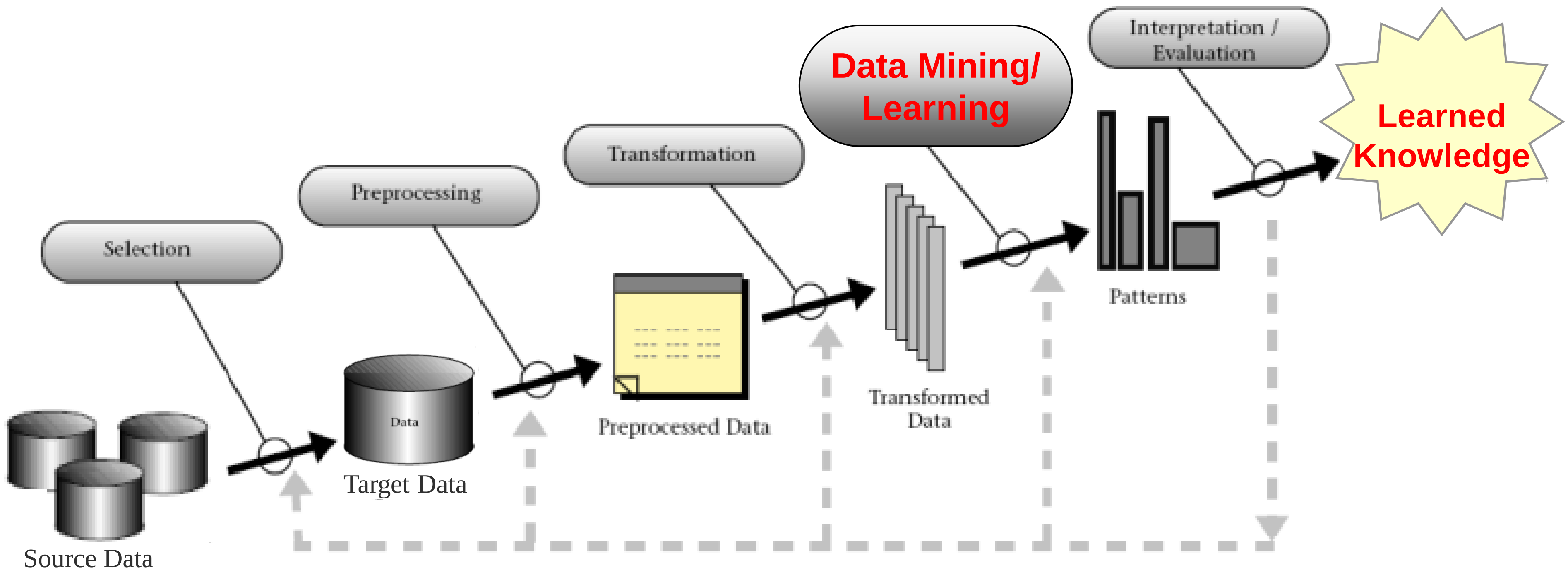
"It's not the most sophisticated Spam blocker I've tried, but it's the only one that works!"

Machine Learning:
Make Knowledge explicit with the Use of Data



Machine Learning in Context

- Machine Learning (Data Mining) is a step to discover knowledge in data

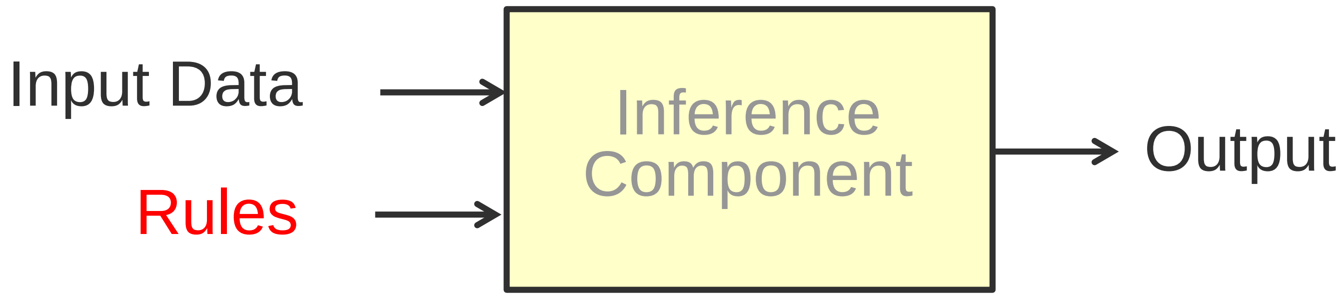


Knowledge is then used in processes and applications.

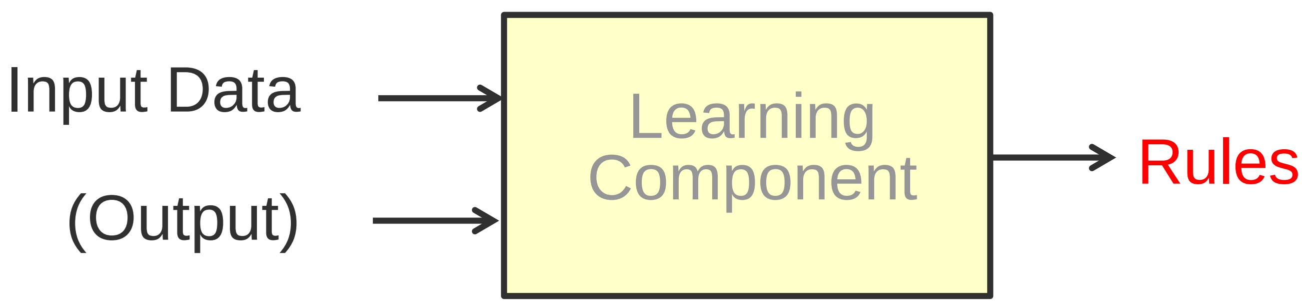
(Fayyad et al., 1996)

Machine Learning vs. Knowledge-based Systems

Knowledge-based System



Machine Learning



Source: Vibhav Gogate, UT Dallas

Input and Output for Machine Learning

	Input *i*	Output **o**
Spam filtering	An email	{spam, non-spam}
Face recognition	An image	Identified faces
Machine translation	A sentence in language A	A sentence in language B
Speech recognition	A speech signal	A (text) sentence
Fraud detection	A financial transaction	{fraud, non-fraud}
Robot motion	Sensory data	Motor control

(Lison, 2012)

Types of Learning

- The learning method depends on the kind of data that we have at our disposal
  - The data contains sets of inputs and corresponding outputs: (i,o)
  - No prior knowledge: The data contains only the inputs i: output has to be determined
  - The data contains sets of inputs without corresponding «correct» output, but we can get some measure of the quality of an output o for input i.
    Rewards for good output quality.

Supervised Learning

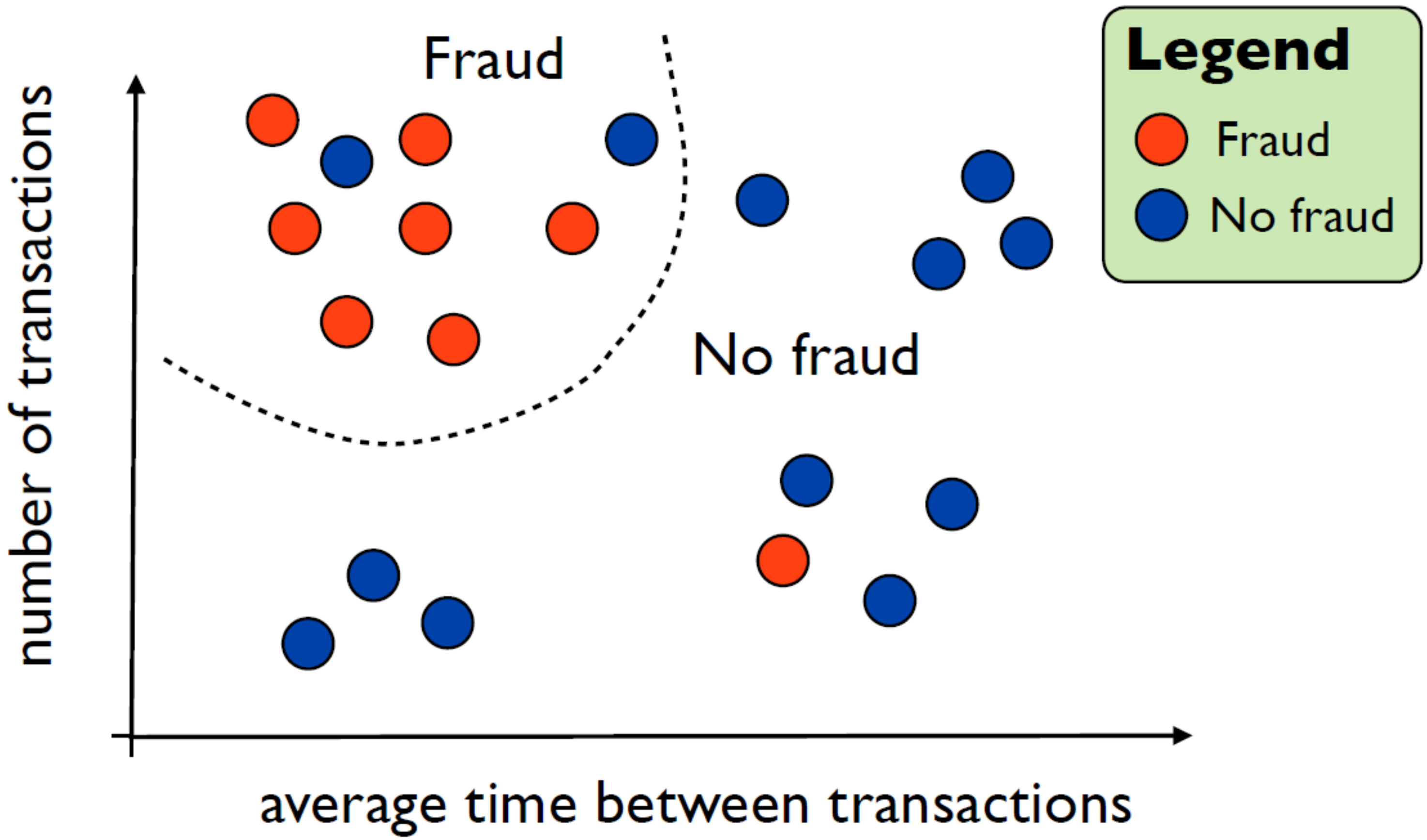
Unsupervised Learning

Reinforcement Learning

(Lison, 2012)

Supervised Learning

Example: Classification

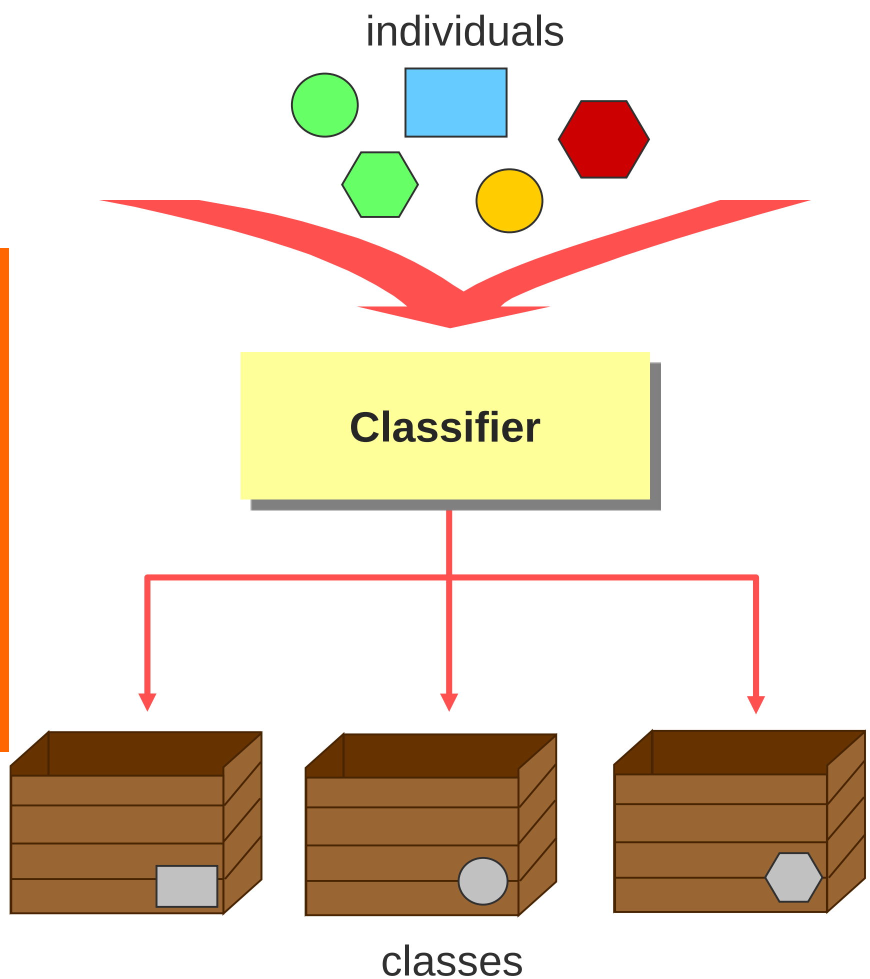


(Lison, 2012)

Classification

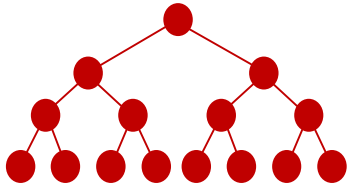
- Assign objects (input) to known classes (output)
- Examples:
  - credit assessment
    Input: customers of a bank
    Classes: credit worthy
    not credit worthy
  - Spam filtering
    Input: email
    Classes: spam
    non-spam
  - optical character recognition (OCR)
    Input: scanned pixel image
    Classes: ASCII characters

Supervised Learning: Classification Criteria



- The classifier decides, which individual belongs to which class
- Problem:
  - The criteria for the decision are not always obvious
- Supervised Learning:
  - Learn the classification criteria from known examples

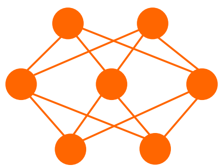
Classification Methods



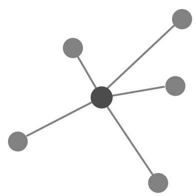
Decision Trees

IF ...
THEN ...

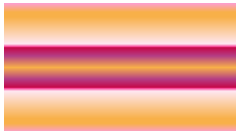
Rules



Neuronale Netze



k-Nearest Neighbor



Genetic Algorithms

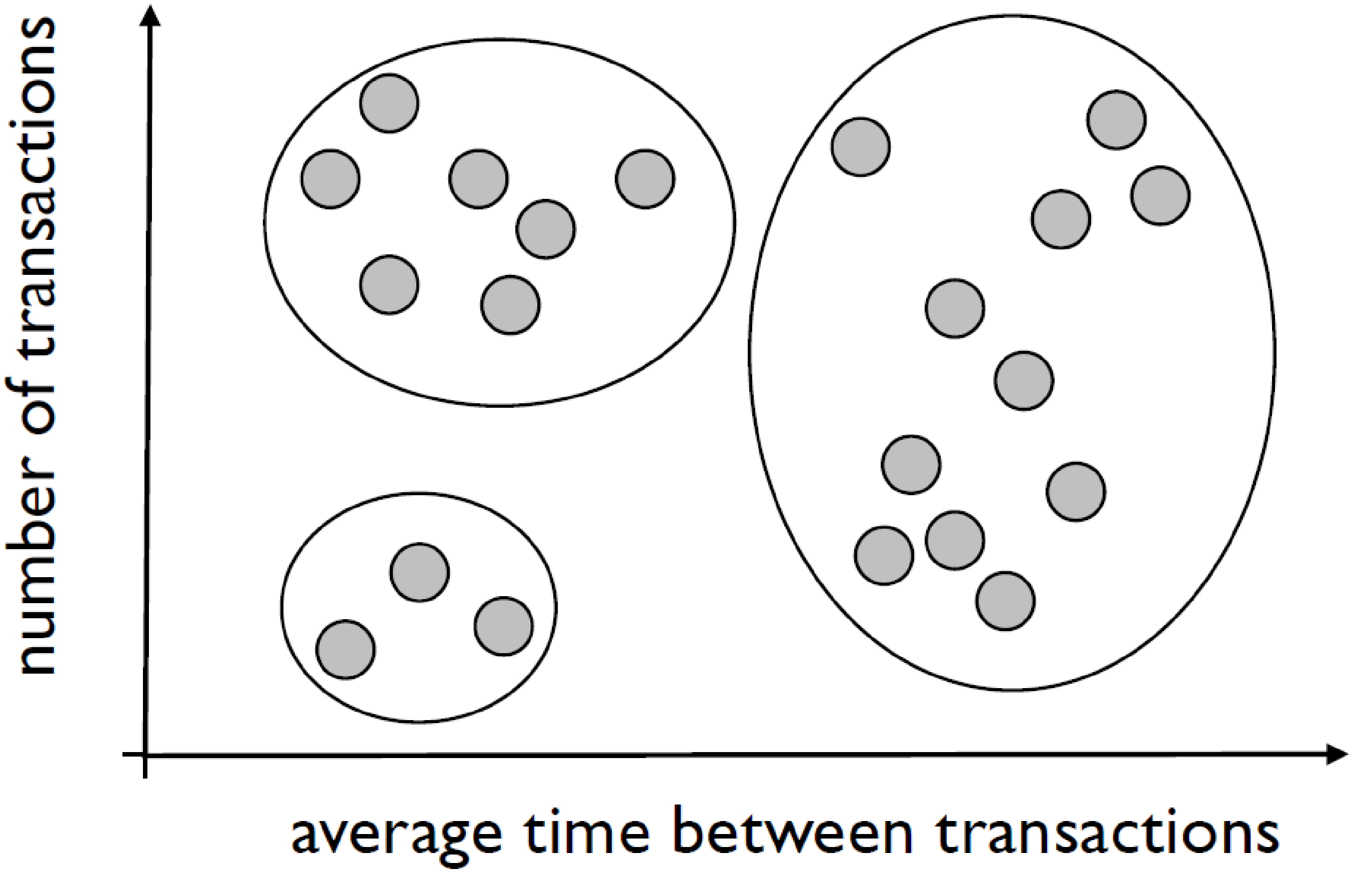
Unsupervised Learning

- Sometimes, we don't have access to any output value **o**, we simply have a collection of input examples **i**
- In this case, what we try to do is to learn the underlying patterns of our data
  - are there any *correlations* between attributes?
  - can we *cluster* our data set in a few groups which behave similarly, and detect *outliers*?

(Lison, 2012)

Unsupervised Learning

Example: Clustering (= identify new classes)



(Lison, 2012)

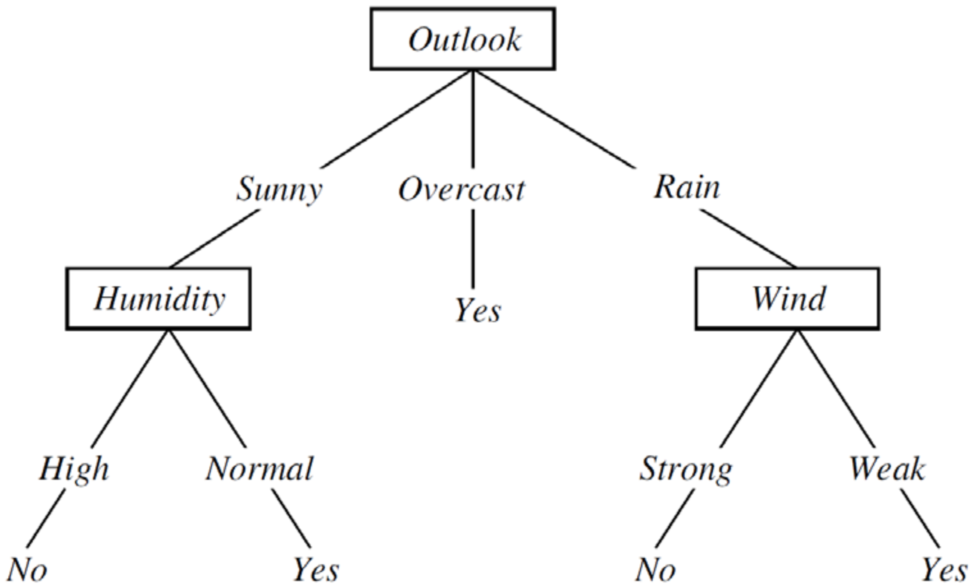
Reinforcement Learning

- Sometime we don’t have direct access to «the» correct output ***o*** for an input ***i***
- But we can get a measure of «how good/bad» an output is
  - Often called the *reward* (can be negative or positive)
- The goal of the agent is to learn the behaviour that maximises its expected cumulative reward over time
  - To learn how to flip pancakes, the reward could for instance be +3 if the pancake is flipped, -1 if the pancake stays in the pan, and -5 if it falls

Supervised Learning: Learning Decision Trees

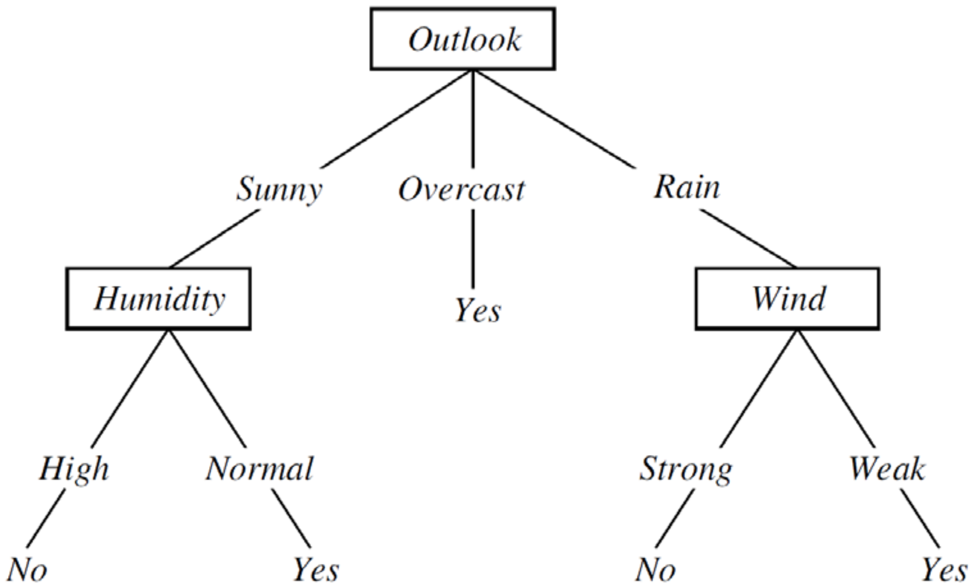
Decision Trees

Example: Decision tree for playing tennis



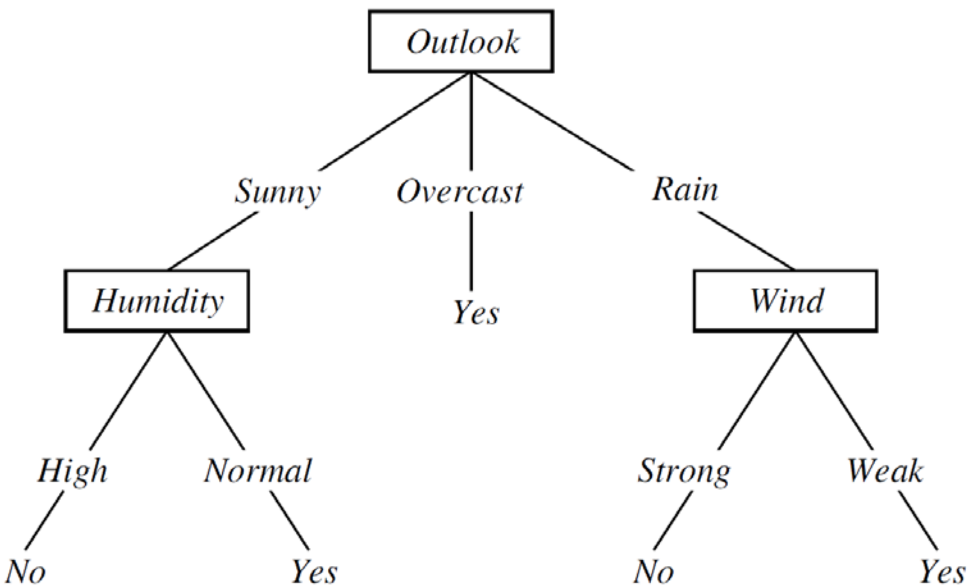
- Decision trees are primarily used for classification
- Decision trees represent classification rules
- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification
- Decision trees classify instances by sorting them down the tree from the root to some leaf node,

Decision Trees represent Rules



- Each path from root to a leaf is a rule
- Each path/rule is a conjunction of attribute tests:
  - **IF** Outlook = Sunny AND
    Humidity = High
    THEN No
  - **IF** Outlook = Sunny AND
    Humidity = Normal
    THEN Yes
  - **IF** Outlook = Overcast
    THEN Yes
  - **IF** Outlook = Rain AND
    Wind = Strong
    THEN No
  - **IF** Outlook = Rain AND
    Wind = Weak
    THEN Yes

Decision Trees represent Rules

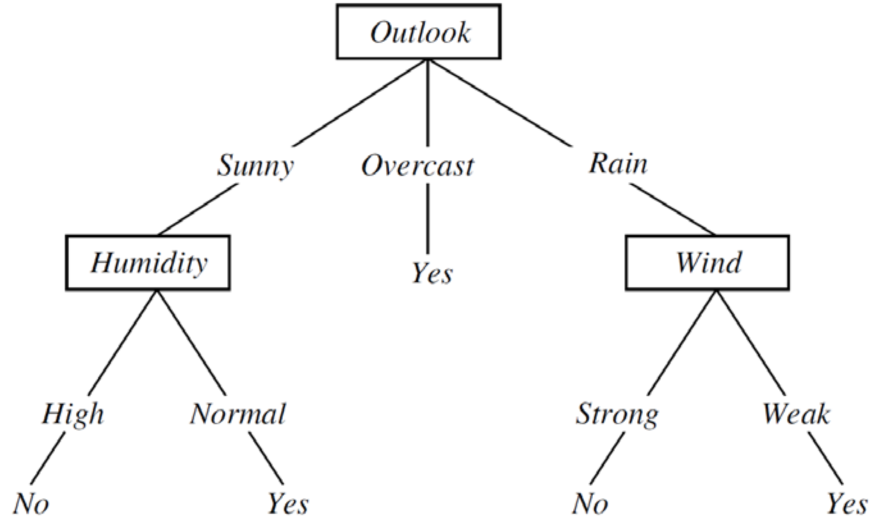


- If the classes are boolean, a path can be regarded as a conjunction of attribute tests.
- The tree itself is a disjunction of these conjunctions

$$
\begin{gathered}
(\ \text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal}\ ) \\
\vee \\
(\ \text{Outlook} = \text{Overcast}\ ) \\
\vee \\
(\ \text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak}\ )
\end{gathered}
$$

Example: Decision Tree – Decision Table

The decision tree can be represented as a decision table.

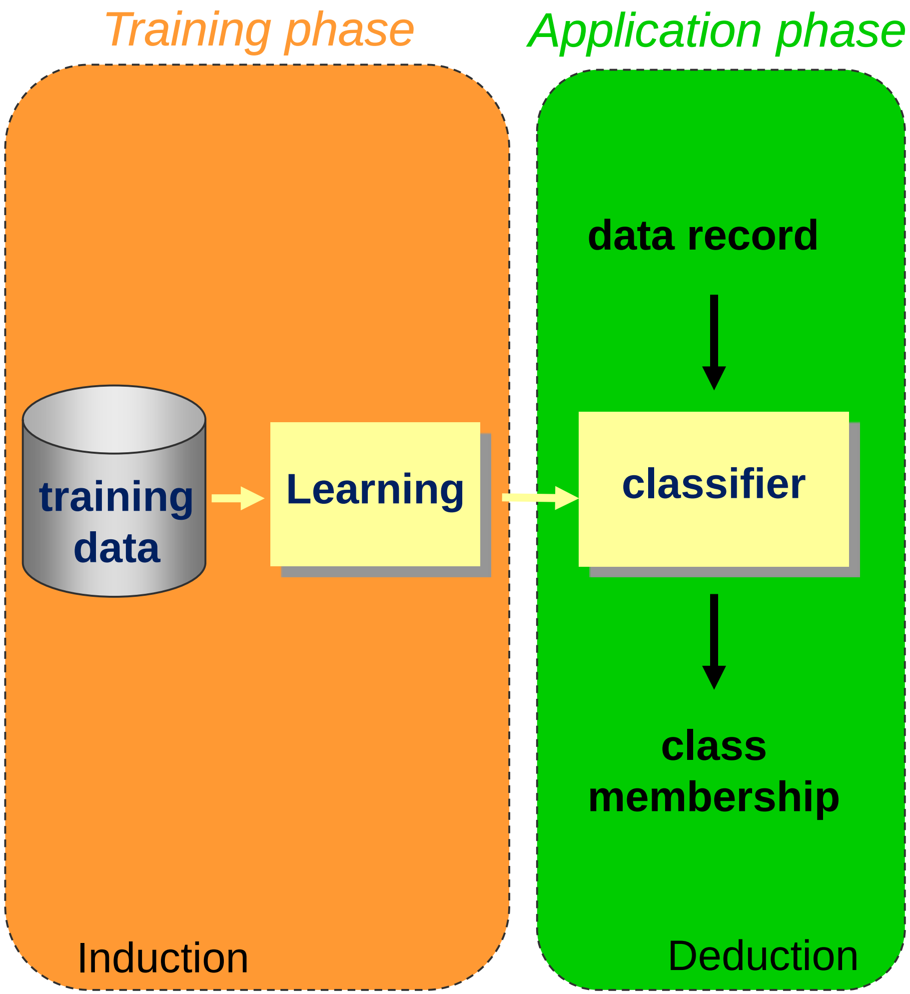


Playing Tennis

	Outlook	Humidity	Wind	Tennis
	Sunny, Overcast, Rain	*High, Normal*	*Strong, Weak*	*Yes, No*
1	Sunny	High		No
2	Sunny	Normal		Yes
3	Overcast			Yes
4	Rain		Strong	No
5	Rain		Weak	Yes

Learning Rules / Decision Trees

Training and Application Phase



- **Application:** Classification
  - Goal: assign a class to previously unseen records of input data as accurately as possible
- **Training:** Learning the classification criteria
  - Given: sample set of training data records
  - Result: Decision logic to determine class from values of input attributes
    (decision tree, rules, model)

Supervised Learning

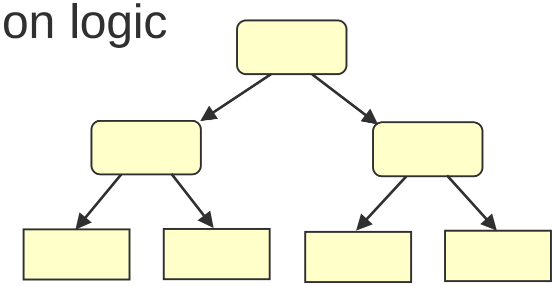
Example: Learning Decision Logic

Training data

input			***output***
...	...	...	...
...	...	...	...
...	...	...	...

Learning

Decision logic



Each record consists of several input attributes and one output attribute, which is the decision

Generalisation if training set does not cover all possible cases or if data are too specific (= induction)

Predictive Model for Classification

- Given a collection of training records (*training set*)
  - Each record consists of *attributes*, one of the attributes is the *class*
  - The class is the dependent attribute, the other attributes are the independent attributes
- Find a *model* for the class attribute as a function of the values of the other attributes.
- Goal: to assign a class to **previously unseen records** as accurately as possible.
- Generalisation of data if training set does not cover all possible cases or data are too specific
  - → Induction

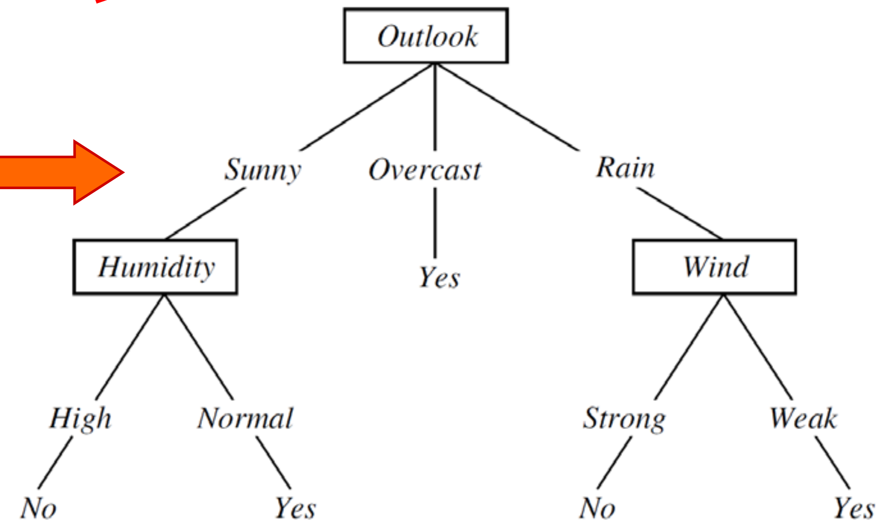
Example

The dependent variable „Tennis“ determines if the weather is good for tennis („Yes“) or not („No“).

Training Data

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

~~Naive approach: Each example data set represents a rule~~



Induction generalizes the data set → prediction of future case

The result of the induction algorithms classifies the data with only three of the four attributes into the classes „Yes“ and „No“.

Discussion

What is the difference between the table with the Training Data and the Decision Table?

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Playing Tennis

	Outlook	Humidity	Wind	Tennis
	Sunny,Overcast, Rain	*High, Normal*	*Strong,Weak*	*Yes, No*
1	Sunny	High		No
2	Sunny	Normal		Yes
3	Overcast			Yes
4	Rain		Strong	No
5	Rain		Weak	Yes

Training Data vs. Decision Tables (Rules)

- Training Data could also be used as decision tables, but
  - Training data are incomplete: only a subset of all possible situations
  - Training data are too specific: they contain input variables, which are not necessary to determin the output
- Decision Tree shall be general, i.e. allow decisions/ predictions for unknown situations
  - Rules only consider combinations of input values, which are necessary to determine the output
  - As a consequence, the decision table does not contain variables, which are not necessary at all
    (e.g. playing tennis does not depend on the temperature)

Induction of Decision Tree

- Enumerative approach
  - Create all possible decision trees
  - Choose the tree with the least number of questions

  This approach finds the best classifying tree, but it is inefficient.

- Heuristic approach:
  - Start with an empty root and extend the tree step by step with new decision nodes
  - Stop, if the desired homogenity is achieved

  This approach is efficient, but does not necessariy find the best classifying tree.

Sketch of an Induction Algorithmus

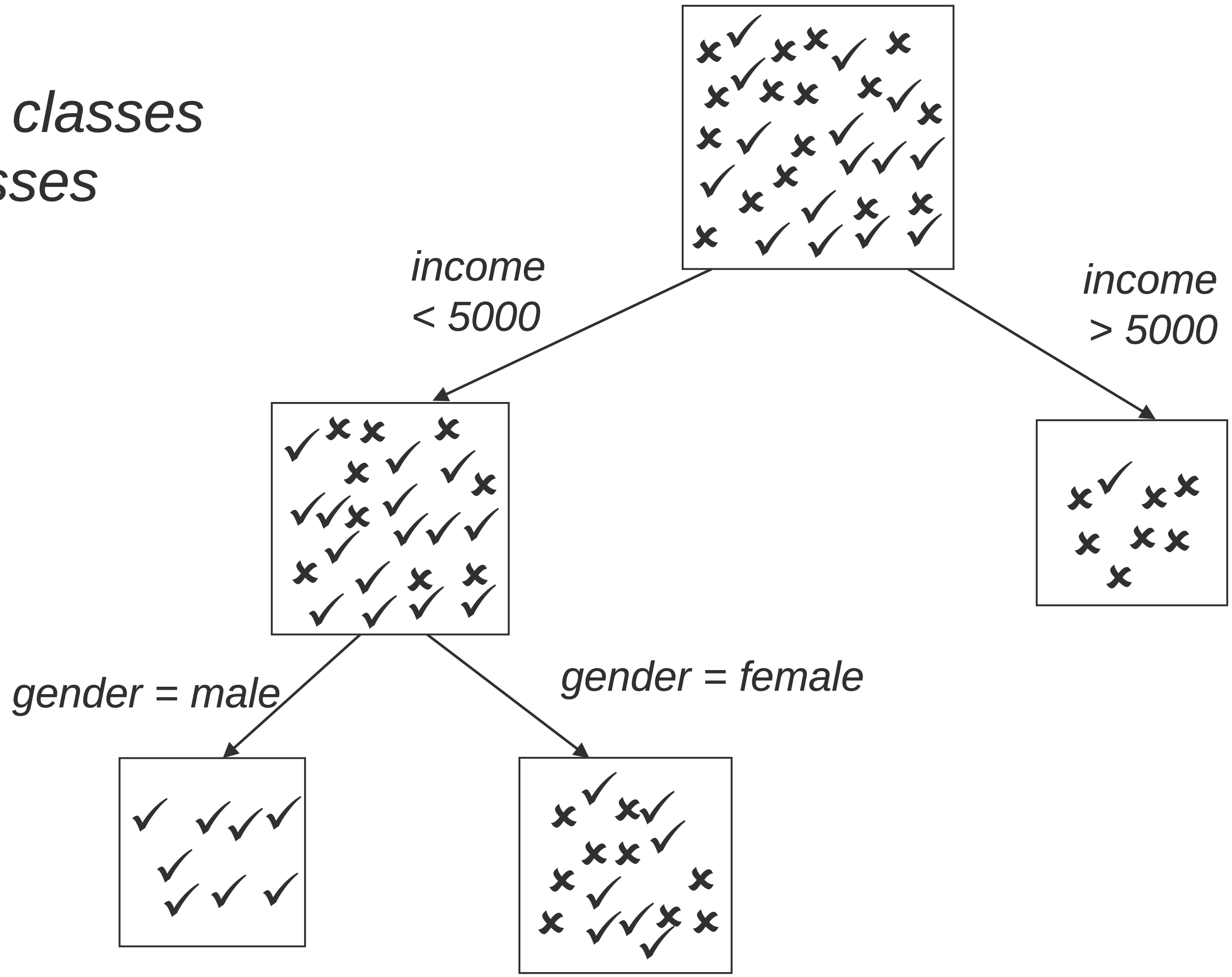
Heuristic Approach

Learning a Decision Tree

- Calculate for each attribute, how *good* it classifies the elements of the training set
- Classify with the *best* attribute
- *Repeat* for each resutling subtree the first two steps
- Stop this recursive process as soon as a *termination condition* is satisfied

Learning a Decision Tree

Principle:
From heterogeous classes
to homogeous classes

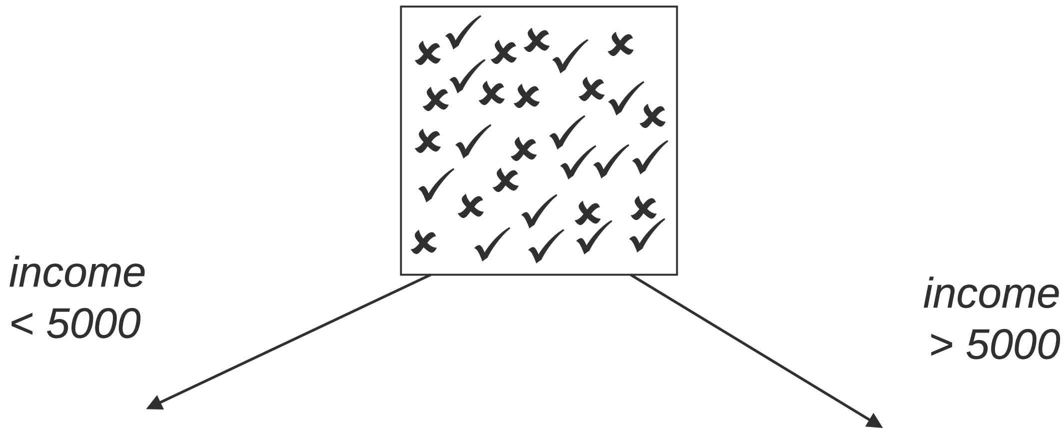


Types of Data

- Discrete: final number of possible values
  - Examples: marital status, gender
  - Splitting: selection of values or groups of values
- Numeric infinite number of values on which an order is defined
  - Examples: age, income
  - Splitting: determine interval boundaries

For which kind of attributes is splitting easier?

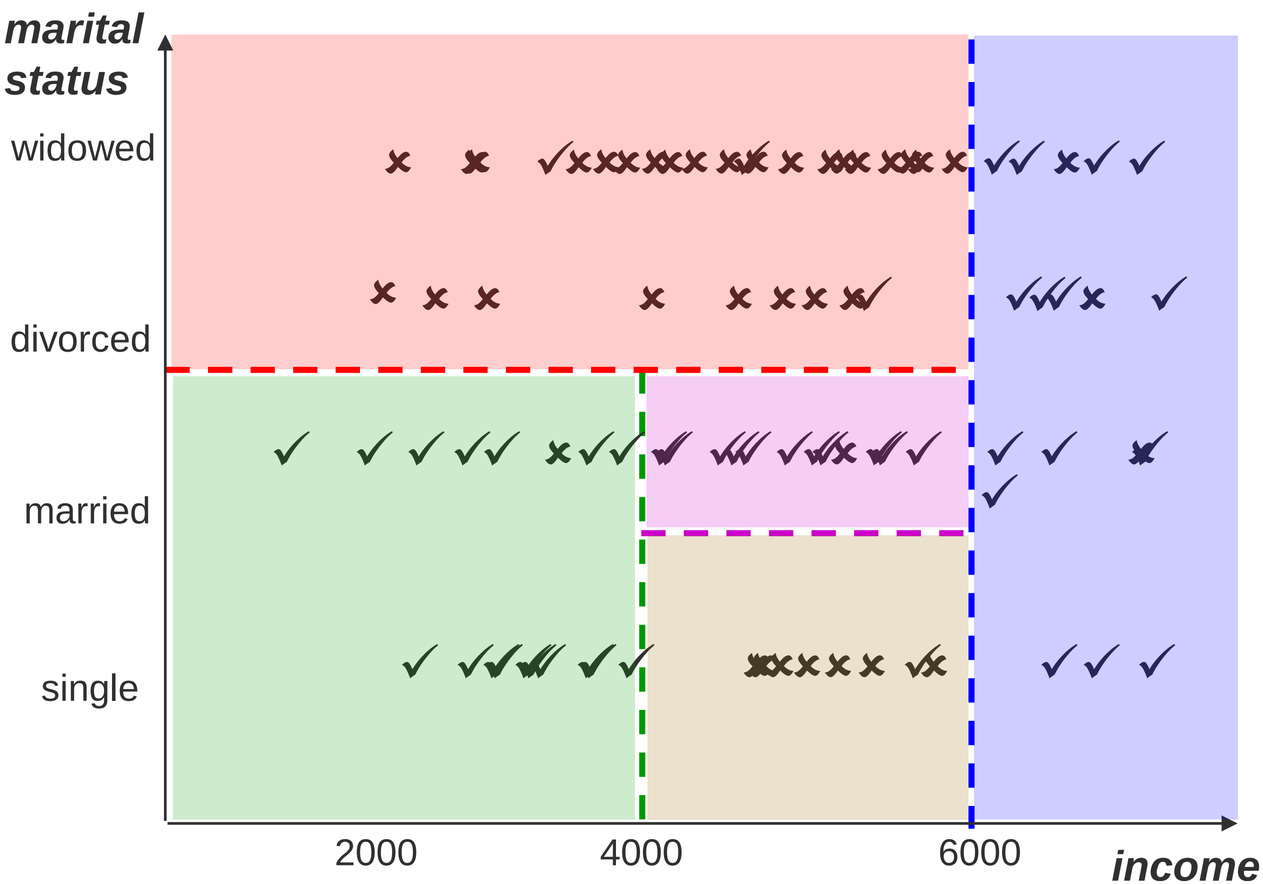
Determine how to split the Records in a Decision Tree



- **Attribute selection**
  - Which **attributes** separate best in which order?
    - e.g. income before marital status
- **Test condition**
  - Which **values** separate best?
    - Discrete: select value, e.g. single or married
    - Number: determine splitting number, e.g. income < 5000

Creation of Decision Trees

Each decision divides the area in sections



IF income > 6000
THEN accept

IF income <= 6000 and marital status = widowed or marital status = divorced
THEN reject

IF income <= 4000 and marital status = single or marital status = married
THEN accept

IF income > 4000 and income <= 6000 and marital status = married
THEN accept

IF income > 4000 and income <= 6000 and marital status = single
THEN reject

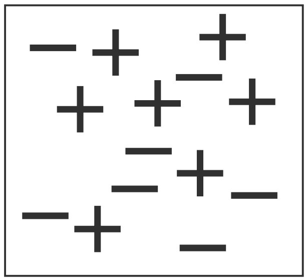
Generating Decision Trees

- ID3 is a basic decision learning algorithm.
- It recursively selects test attributes and begins with the question "*which attribute should be tested at the root of the tree?* "
- ID3 selects the attribute with the highest
  - **Information Gain**
    (this is the attribute with reduces entropy the most)
- To calculate the information gain of an attribute A one needs
  - the **Entropy** of a classification
  - the **Expectation Entropy** of the attribute A

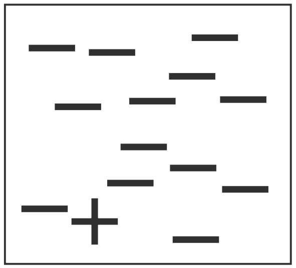
Entropy („disorder“)

- Entropy is a measure of *(im)purity* of *a collection S of examples*.
- The higher the homogeneity of the information content, the lower the entropy
- Let + denote a possible example for a class C in S and – denote a negative example for a class C in S.
- Let p be the frequency of + and n be the frequency of –
- The entropy is smaller the more + and -- are different from equal distribution, i.e. *the more unequal p and n, the smaller is the entropy*

high entropy



low entropy



Calculation of the Entropy for binary Classification

- Assume a decision tree which classifies the training set into to classes + (positive) and – (negative)
- The entropy is calculated by

  Entropy (S) = $- p_+ * \log_2 (p_+) - p_- * \log_2 (p_-)$

  S = p + n is the number of all elements

  p frequency of elements of class +

  n frequency of elements of class –

  $p_+ = p / S$ and $p_- = n / S$ are the relative frequencies, i.e. the proportions of values of classes + and –

Entropy Calculation for different Distributions

- The more different p and n, the lower is the entropy

p	*n*	*p+*	*ld(p+)*	*p-*	*ld(p-)*	*Entropy(p+n)*
7	7	0.5	-1	0.5	-1	1
6	8	0.43	-1.22	0.57	-0.81	0.99
5	9	0.36	-1.49	0.64	-0.64	0.94
4	10	0.29	-1.81	0.71	-0.49	0.86
3	11	0.21	-2.22	0.79	-0.35	0.75
2	12	0.14	-2.81	0.86	-0.22	0.59
1	13	0.07	-3.81	0.93	-0.11	0.37

ld = $\log_2$ (logarithmus dualis)
ld(0) cannot be calculated, but for p = 0 or n = 0 no classification is necessary

Information Gain

- The information gain for an attribute A is the expected reduction in entropy caused be partitioning the example according to the attribute A
- The information gain is calculated by subtracting the expectation entropy of the subtrees created by A from the current entropy

$$GAIN\ (S, A) = Entropy\ (S) - EE(A)$$

Expected Entropy

- Let A be an attribute with m possible values v_1, ..., v_i, ... v_m
  - *Values(A)* is the set of all possible values for attribute A
  - S_v is the subset of S for which attribute A has value *v*
- The attribute A divides the elements into m partitions (subtrees)
- Entropy(S_v) is the entropy of the subtree for which the attribute A has value v
- The **Expected Entropy EE_A** for an attribute A is the weighted average of the entropies of the subtrees created by the values v_i of A

$$EE(A) := \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Formula for the Information Gain

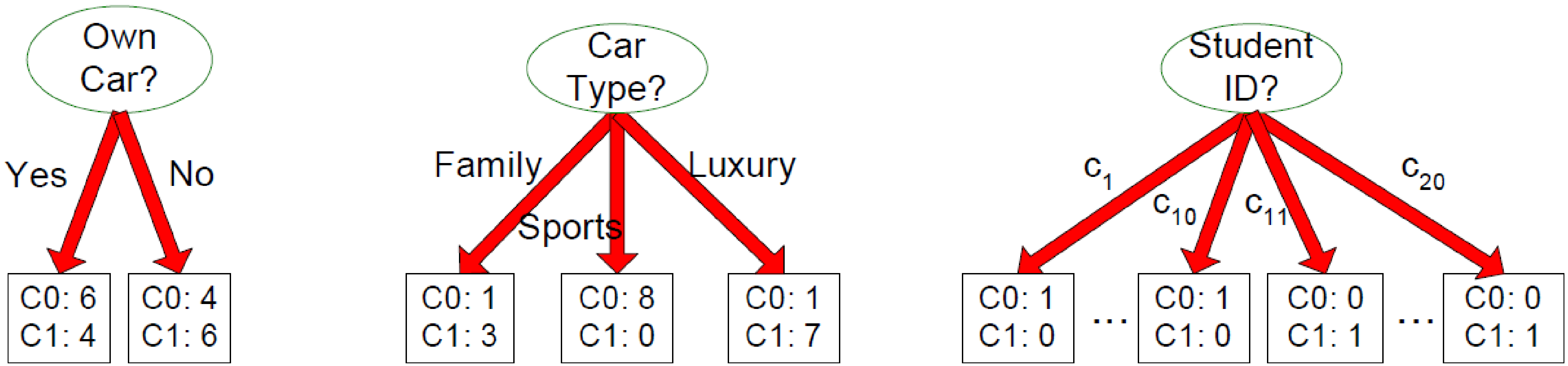
- The information gain for an attribute A is the expected reduction in entropy caused be partitioning the example according to the attribute A

$$GAIN(S, A) = Entropy(S) - \left(\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \right)$$

Exercise

Entropy (S) = 1

Before Splitting: 10 records of class 0,
10 records of class 1



- Which test condition is the best?
- Does it make sense?

Thanks to Nadeem Qaisar Mehmood

ID3: Information Gain for Attribute Selection

- The goal of learning is to create a tree with minimal entropy
- ID3 uses the Information Gain to select the test attribute

On each level of the tree select the attribute with the highest information gain

- The recursive calculation of the attributes stops when either
  - all partitions contain only positive or only negative elements (i.e. entropy is 0) or
  - a user-defined threshold is achieved

An Illustrative Example (1)

The dependent variable „Tennis“ determines if the weather is good for tennis („Yes“) or not („No“).

Element	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Tennis*
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (2): Entropy of the Decision Tree

$$\text{Entropy}(S) = -\,9/14 * \log_2(9/14) - 5/14 * \log_2(5/14) = 0{,}94$$

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

positive frequency (Yes)
negative frequency (No)

Selection of the topmost Node

- In order to determine the attribute that should be tested first in the tree, the information gain for attributes (*Outlook* , *Temperature* ,*Humidity* and *Wind*) are determined.
  - Gain(S,Outlook) = 0.246
  - Gain(S,Humidity) = 0.151
  - Gain(S,Wind) = 0.048
  - Gain(S,Temperature) = 0.029
- Since *Outlook* attribute provides the best prediction, it is selected as the decision attribute for the root node.

An Illustrative Example (3): Selection of the topmost Node

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

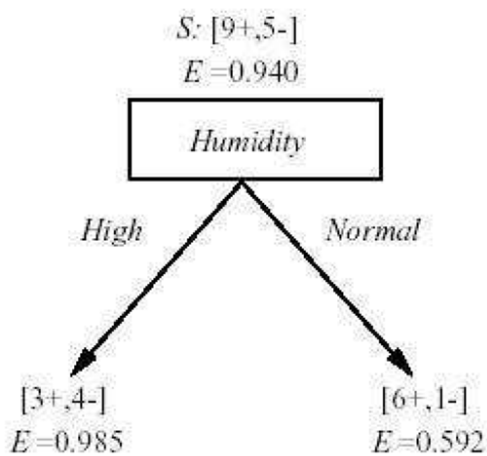
- In order to determine the attribute that should be tested first in the tree, the information gain for each attribute (*Outlook* , *Temperature, Humidity* and *Wind*) is determined.
  - Gain(S,Outlook) = **0.246**
  - Gain(S,Humidity) = **0.151**
  - Gain(S,Wind) = **0.048**
  - Gain(S,Temperature) = **0.029**
- Since *Outlook* attribute provides the best prediction, it is selected as the decision attribute for the root node.

An Illustrative Example (4): Computation of Information Gain

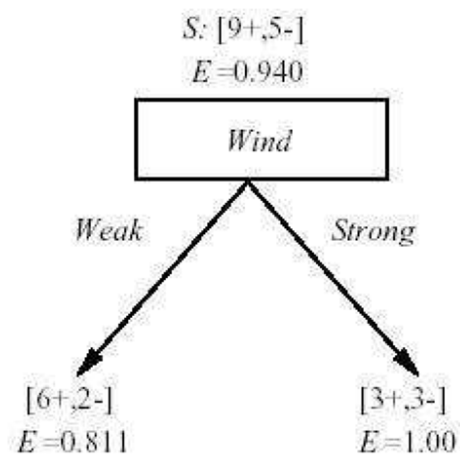
- The computation of Information Gain for Outlook:

$$
\begin{aligned}
GAIN(S,Outlook) &= Entropy(S) - EE(Outlook) \\
&= 0.94 - 0.694 = \mathbf{0.246}
\end{aligned}
$$

- The computation of information gain for *Humidity* and *Wind:*



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

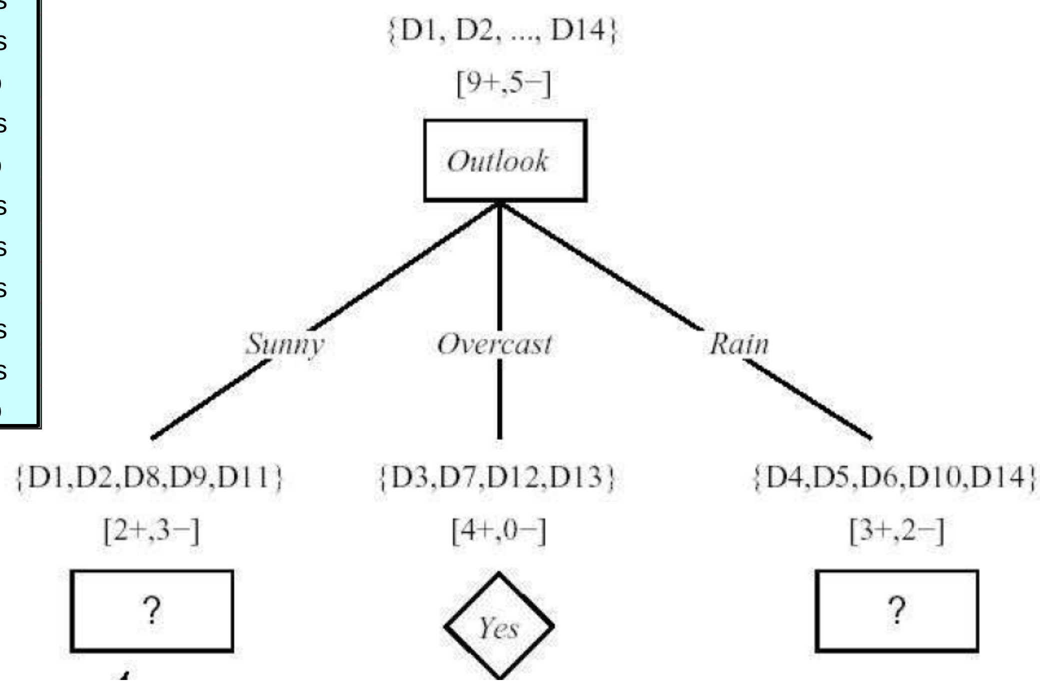


Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

An Illustrative Example (5): Resulting Subtree

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- The partially learned decision tree resulting from the first step of ID3:

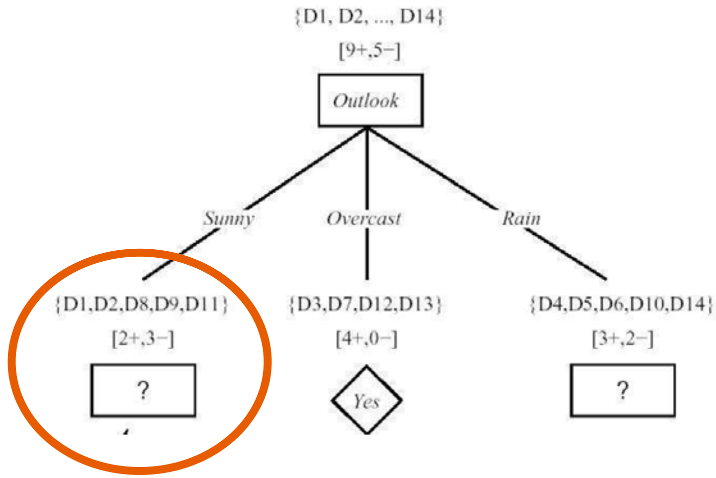


An Illustrative Example (6): Entropie of a Subtree

The subtree with root Sunny:

$$\text{Entropy(Sunny)} = -\ 2/5 \log_2 (2/5) - 3/5 \log_2 (3/5)$$
$$= 0{,}970$$

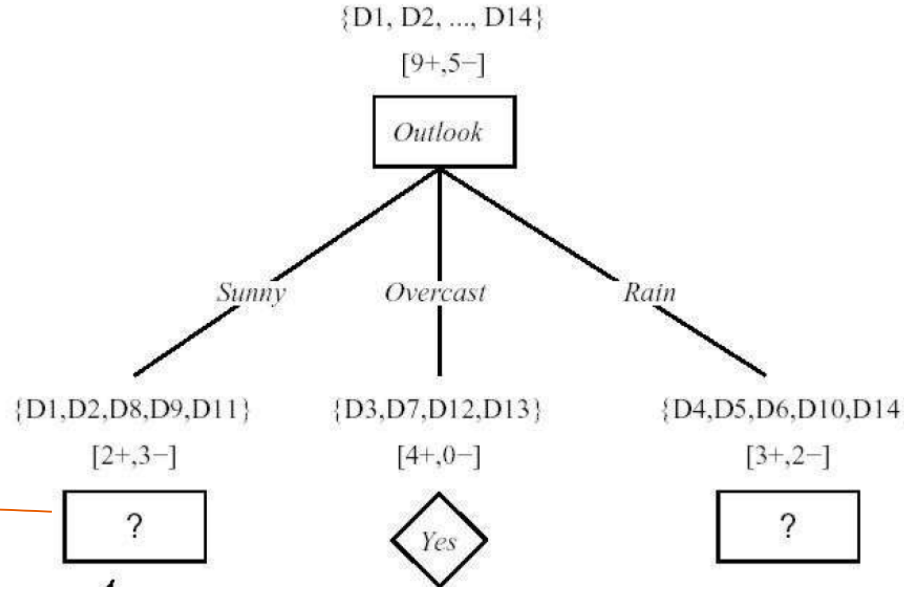
Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



The more **up** in the decision tree, the smaller the entropy of the subtree

An Illustrative Example (7): Selectiong Next Attribute

Which attribute should be tested here?



$$S_{sunny} = \{D1,D2,D8,D9,D11\}$$

$$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$$

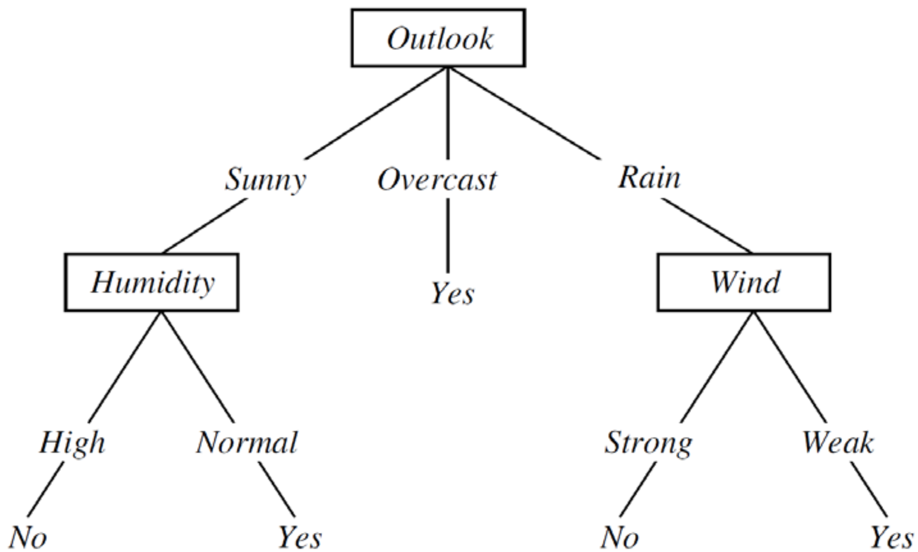
$$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$$

$$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$$

An Illustrative Example (8): The Resulting Decision Tree

The dependent variable „Tennis“ determines if the weather is good for tennis („Yes“) or not („No“).

Element	Outlook	Temperature	Humidity	Wind	Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cold	Normal	Weak	Yes
6	Rain	Cold	Normal	Strong	No
7	Overcast	Cold	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



The result of the induction algorithms classifies the data with only three of the four attributes into the classes „Yes“ and „No“.

How to specify Attribute Test Conditions

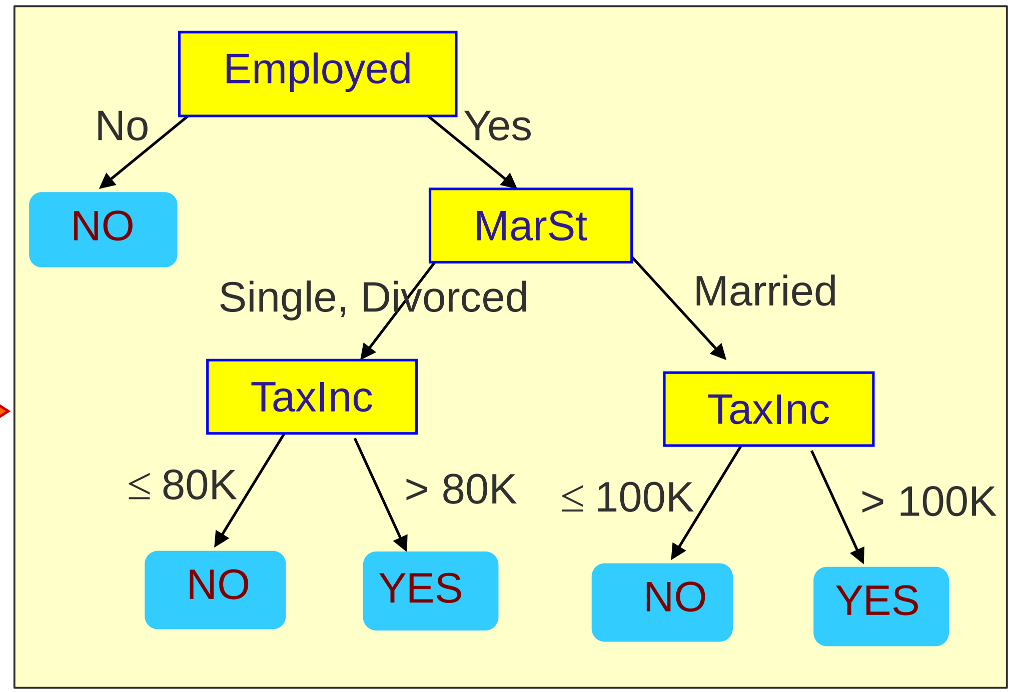
Specification of the test condition depends on

- attribute types
  - Nominal
  - Ordinal
  - Continuous
- number of ways to split
  - 2-way split
  - Multi-way split

Learning Decision Trees: Generalisation of Data

categorical categorical continuous class

Tid	**Employed**	**Marital Status**	**Taxable Income**	**accept**
1	No	Single	125K	**No**
2	Yes	Married	160K	**Yes**
3	Yes	Single	70K	**No**
4	No	Married	120K	**No**
5	Yes	Divorced	95K	**Yes**
6	Yes	Married	60K	**No**
7	No	Divorced	220K	**No**
8	Yes	Single	85K	**Yes**
9	Yes	Married	95K	**No**
10	Yes	Single	90K	**Yes**



Model: Decision Tree

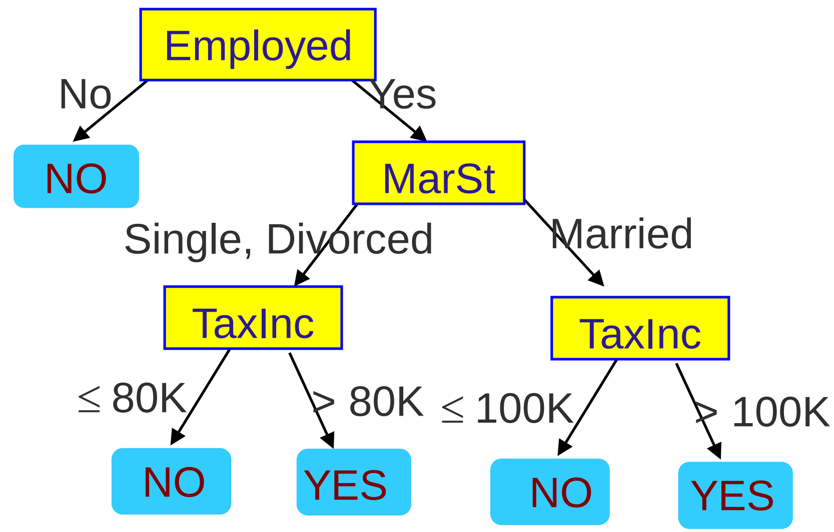
The model uses intervals instead of concrete numerical data

Learning Decision Trees: Generalisation of Data

categorical categorical continuous class

Tid	Employed	Marital Status	Taxable Income	accept
1	No	Single	125K	No
2	Yes	Married	160K	Yes
3	Yes	Single	70K	No
4	No	Married	120K	No
5	Yes	Divorced	95K	Yes
6	Yes	Married	60K	No
7	No	Divorced	220K	No
8	Yes	Single	85K	Yes
9	Yes	Married	95K	No
10	Yes	Single	90K	Yes

Training Data



Credit Worthiness

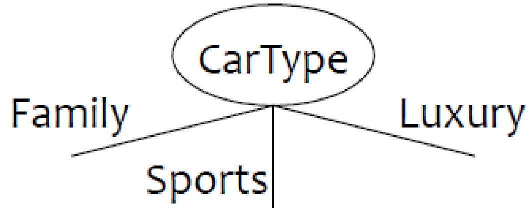
	Employed	Marital Status	Taxable Income	Accept
	Yes, No	*Single, Divorced, Married*	*Integer*	*Yes, No*
1	No			No
2	Yes	Single	> 80K	Yes
3	Yes	Divorced	> 80K	Yes
4	Yes	Single	≤ 80K	No
5	Yes	Divorced	≤ 80K	No
6	Yes	Married	> 100K	Yes
7	Yes	Married	≤ 100K	No

Model: Decision Tree/Table

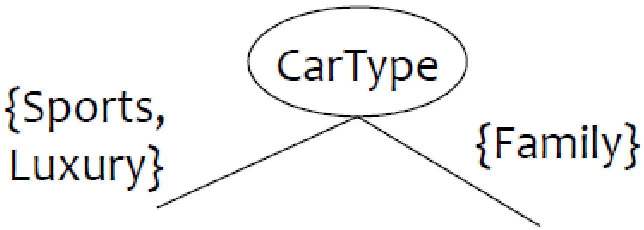
The model uses intervals instead of concrete numerical data

Splitting for Nominal Attributes

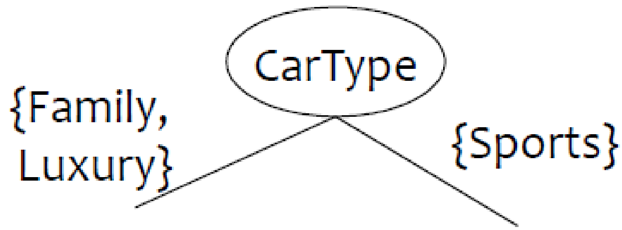
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.

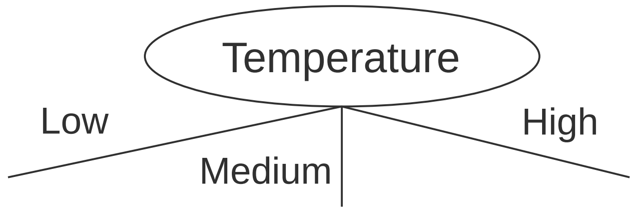


or

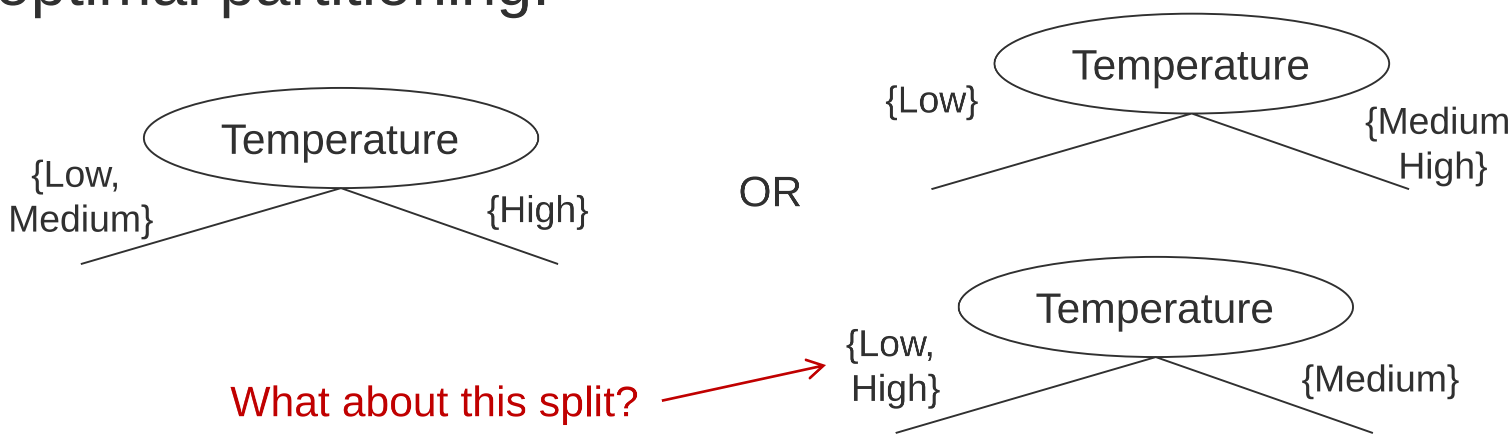


Splitting for Ordinal Attributes

- Multi-way split: Use as many partitions as distinct values.

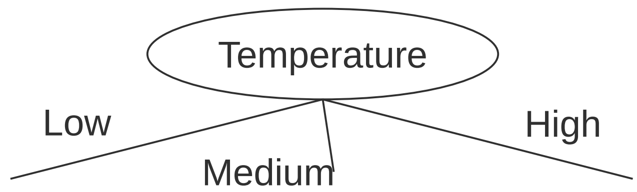


- Binary split: Divides values into two subsets. Need to find optimal partitioning.

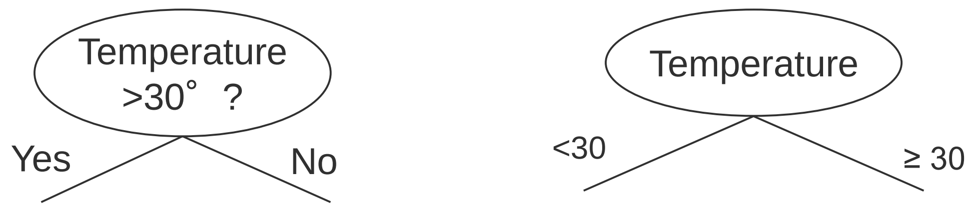


Splitting for Continuous Attributes

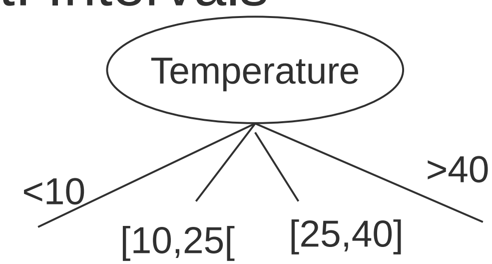
- Different ways of handling
  - Discretization to form an ordinal categorical attribute



  - Binary Decision: (A < v) or (A ≥ v)



  - Multi-way Split: Intervals



considering all possible splits and finding the best cut can be computing intensive

ID3 Algorithm in English

The algorithm looks at each attribute within the attributelist and determines the attribute **X** which provides the largest information gain. Once **X** is found it can be removed from the list of candidates to be considered.

A **newattributelist** and a **newdata_subset** are created which are subsets of the original **attributelist** and **newdata_subset** respectively (excluding attribute **X**). Each possible value of the attribute **X** is recursively called with the **newattributelist** and the narrowed down examples of **newdata_subset**, so the algorithm will continue performing the steps indicated.
The base case is reached when a **attributelist** is provided that has no attributes in it (so the attributes have been exhausted), or where the entropy is equal to 0 (there's complete certainty). For these cases, the algorithm returns a leaf node consisting of the most probable outcome.

https://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-decision-trees-and-entropy/

attribute = feature = independent variable

Building the Decision Tree

Decision trees can be constructed using the ID3 algorithm that splits the data by the attribute with the maximum information gain recursively for each branch.

```
maketree  ( attributelist, examples )  returns  tree
{
BASE CASE: if attributelist is empty, or entropy = 0
return an empty tree with leaf = majority answer in examples

RECURSION:
find the attribute X with the largest information gain,
list_subset = remove X from the attributelist

create an empty tree T
for each possible value ‘x’ of attribute X
data_subset = get all examples where X = ‘x’
t = maketree( list_subset, data_subset )
add t as a new sub-branch to T
endfor

return T
}
```

https://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-decision-trees-and-entropy/

A basic Decision Tree Learning Algorithm

ID3(Examples, Target-attribute, Attributes)

/* Examples: The training examples; */
/* Target-attribute:The attribute whose value is to be predicted by the tree; */
/* Attributes: A list of other attributes that may be tested by the learned decision tree. */
/* Return a decision tree that correctly classifies the given Examples */
Step 1: Create a Root node for the tree
Step 2: If all *Examples* are positive, Return the single-node tree *Root*,with label = +
Step 3: If all *Examples* are negative, Return the single-node tree *Root*,with label = -
Step 4: If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target-attribute* in *Examples*
Step 5: Otherwise Begin

- A ← the attribute from *Attributes* that best (i.e., highest information gain) classifies *Examples*;
- The decision attribute for *Root* ← A;
- For each possible value, v_i , of A,
  - Add a new tree branch below *Root*, corresponding to the test A=v_i;
  - Let $Examples_{(v_i)}$ be the subset of *Examples* that have value v_i for A;
  - If $Examples_{(v_i)}$ is empty
    - Then below this new branch add a leaf node with label = most common value of *Target-attribute* in *Examples*
    - Else below this new branch add the subtree ID3($Examples_{(v_i)}$, *Target-attribute*, *Attributes*- A))

End

Return *Root*

Preference for Short Trees

- Preference for short trees over larger trees, and for those with high information gain attributes near the root
- **Occam's Razor:** Prefer the simplest hypothesis that fits the data.
- Arguments in favor:
  - a short hypothesis that fits data is unlikely to be a coincidence – compared to long hypothesis
- Arguments opposed:
  - There are many ways to define small sets of hypotheses

Overfitting

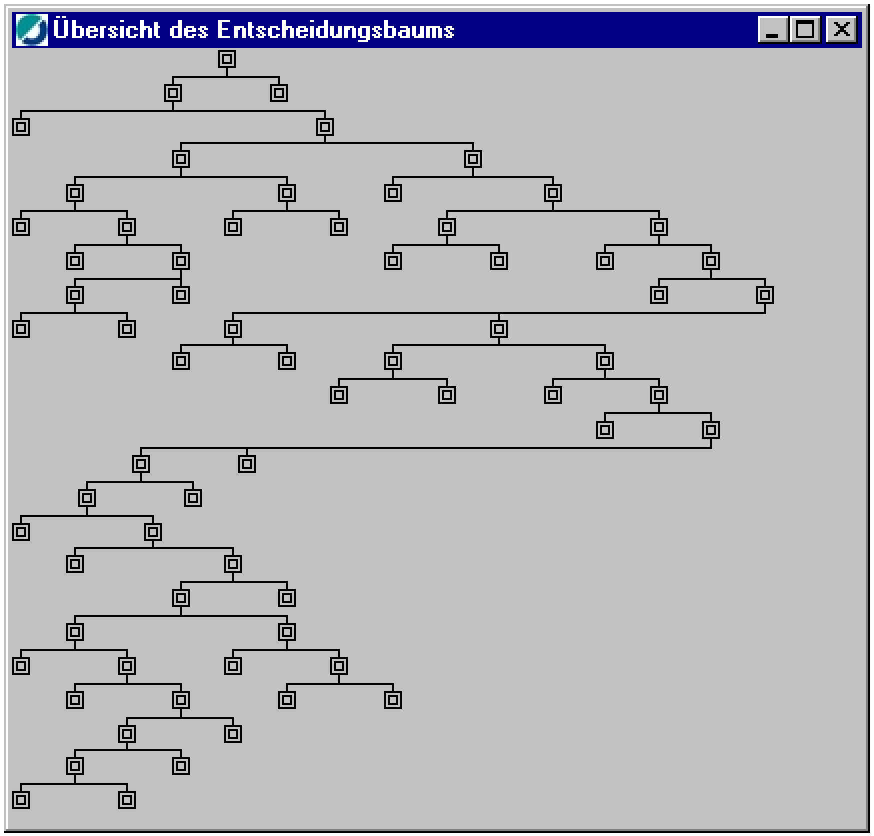
- When there is ***noise in the data***, or when the number of training ***examples is too small*** to produce a representative sample of the true target function, the rule set (hypothesis) ***overfits*** the training examples!!
- Consider error of hypothesis *h* over
  - training data: *errortrain*(h)
  - entire distribution D of data: *errorD*(h)
- Hypothesis *h* OVERFITS training data if there is an alternative hypothesis h*0* such that
  - *errortrain*(*h*) < *errortrain*(*h0*)
  - *errorD*(*h*) > *errorD*(*h0*)

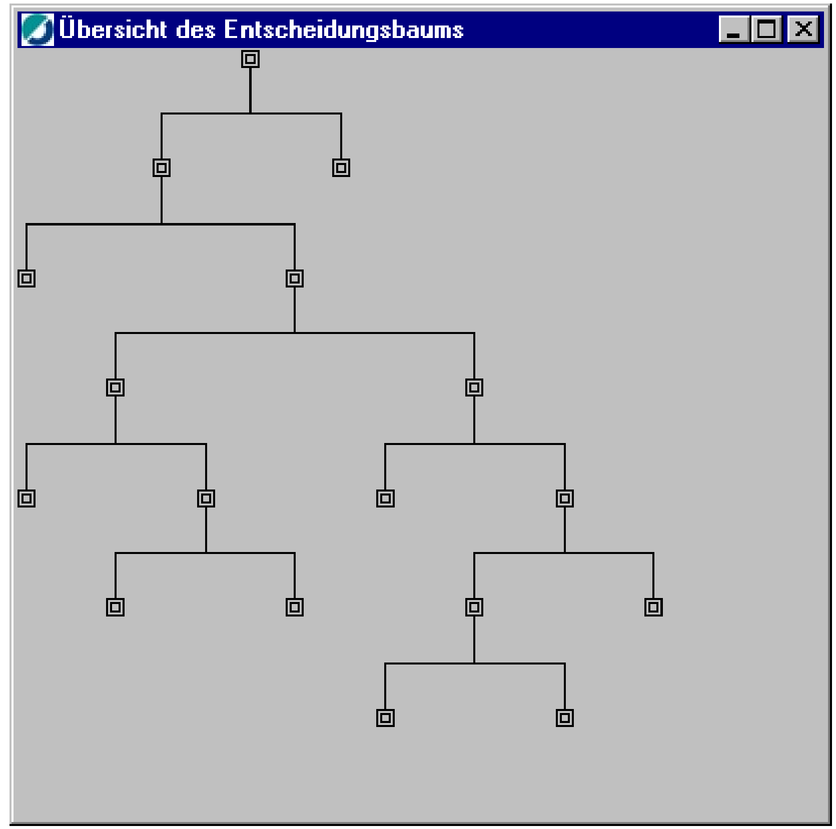
Avoiding Overfitting by Pruning

- The classification quality of a tree can be improved by cutting weak branches
- Reduced error pruning
  - remove the subtree rooted at that node,
  - make it a leaf,
  - assign it the most common classification of the training examples afiliated with that node.
- To test accuracy, the data are separated in training set and valication set. Do until further pruning is harmful:
  - Evaluate impact on *validation* set of pruning each possible node
  - Greedily remove the one that most improves *validation* set accuracy

Pruning

These figures shoe the structure of a decision tree before and after pruning





Training and Validation

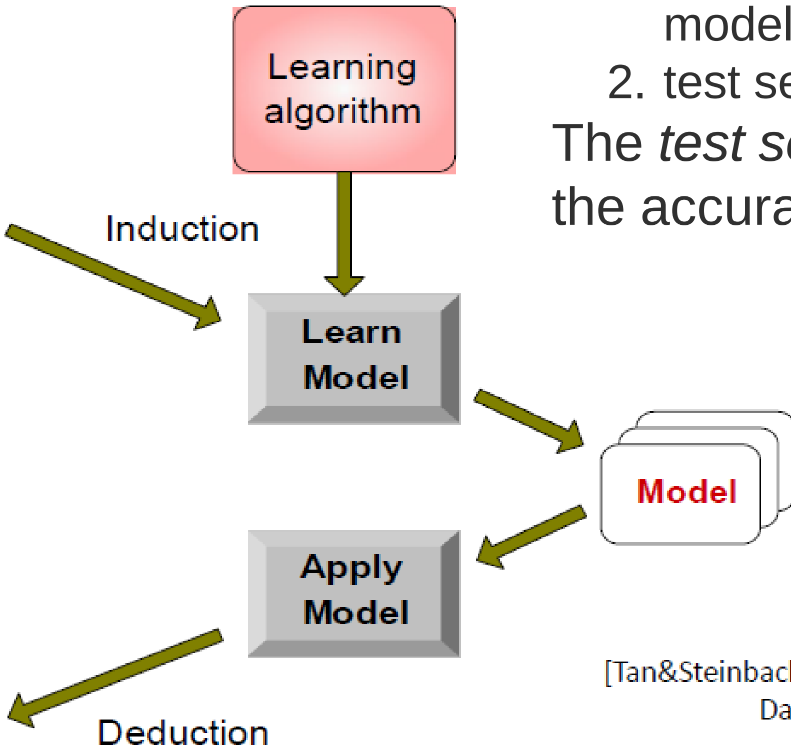
Usually, the given data set is divided into

1. training set (used to build the model)
2. test set (used to validate it)

The *test set* is used to determine the accuracy of the model.

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

[Tan&Steinbach's "Intro to Data Mining"]

Generalisations

- Multiple Classes
  - Although the examples had only two classes, decision tree learning can be done also for more than two classes
  - Example: Quality
    - okay, rework, defective
- Probability
  - The examples only had Boolean decisions
    - Example: **IF** income > 5000 and age > 30
      THEN creditworthy
  - Generalisation: Probabilties for classification
    - Example: **IF** income > 5000 and age > 30
      THEN creditworthy with probability 0.92

Algorithms for Decision Tree Learning

- Examples os algorithms for learning decision trees
  - C4.5 (successor of ID3, predecessor of C5.0)
  - CART (Classification and Regression Trees)
  - CHAID (CHI-squared Automatic Interaction Detection)
- A comparison [1)] of various algorithms showed that
  - the algorithms are similar with respect to classification performance
  - pruning increases the performance
  - performance depends on the data and the problem.

1) D. Michie, D.J. Spiegelhalter und C.C. Taylor: Machine Learning, Neural and Statistical Classificaiton, Ellis Horwood 199