

# Decision making with decision trees and rules

## Outline:

1. Introduction
2. ID3 (Inductive Dichotomizer 3)
3. CART: classification and regression tree
4. Oblique Classifier 1 (OC1)
5. Decision tree applications

# 1. Introduction

- Decision tree methods are supervised machine learning methods frequently used for automated knowledge acquisition.
- Knowledge acquisition: the transfer and transformation of problem-solving expertise from some knowledge source to a program. It usually involves:
  - problem definition
  - implementation and
  - refinement
- In supervised learning, each example consists of a set of initial conditions and a set of resulting actions/decisions.

# Introduction

**Decision Trees:** The best known and most widely used learning methods in data mining applications.

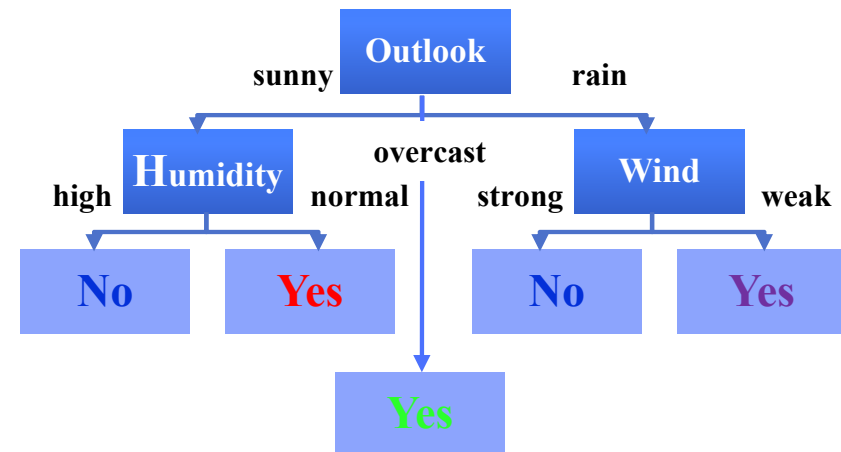
- Reasons: conceptual simplicity, ease of usage, computational speed, robustness with respect to missing data and outliers, interpretability of the generated rules.
- Method to build decision trees is recursive, heuristic, top-down induction:
  1. Initialization phase: all observations are placed in the root of the tree. The root is placed in the active node list  $\mathbf{L}$ .
  2. If the list  $\mathbf{L}$  is empty, stop the procedure. Otherwise, node  $\mathbf{J} \in \mathbf{L}$  is selected, removed from the list and used as the node for analysis.
    - If  $\mathbf{J}$  does not need to be split, node  $\mathbf{J}$  becomes a leaf, target class is assigned according to majority of observations.
    - Otherwise, split node  $\mathbf{J}$ , its children are added to the list. Go to Step 2.

# Introduction

## Components of the top-down induction of decision trees:

- **Splitting rules:** optimal way to split a node (i.e. assigning observations to descendant nodes) and for creating descendant nodes.
- **Stopping criteria:** if the node should be split or not. If not, this node becomes a leaf of the tree.
- **Pruning criteria:** avoid excessive growth of the tree (pre-pruning) during tree generation phase, and reduce the number of nodes after the tree has been generated (post-pruning).

Example: A multisplit classification tree



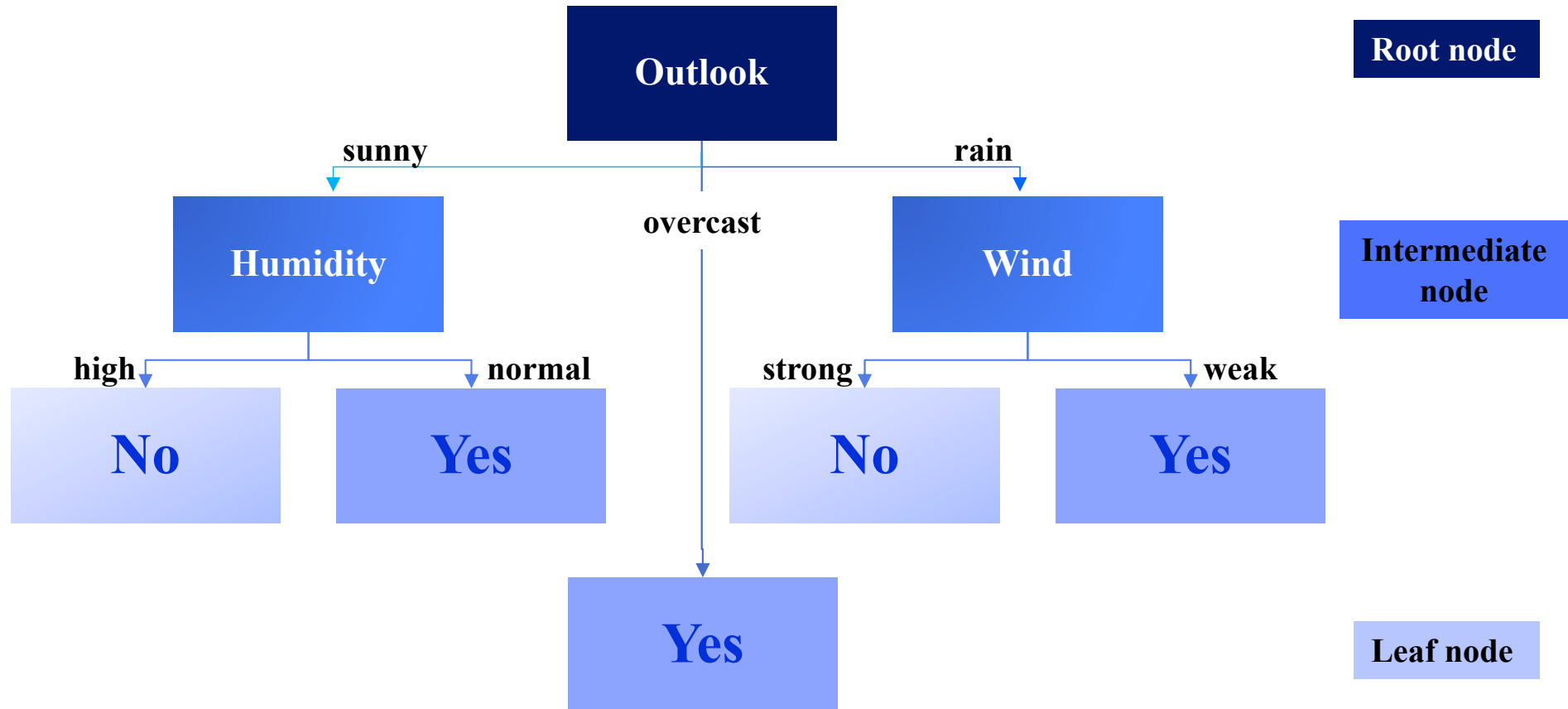
# Introduction

## Two splitting rules according to attribute value selection

- **Binary split:** each node has at most two branches.
  - Example: customers who agree to a mailing campaign are placed on the right child node, those who do not agree on the left child node.
  - Example: customers residing in areas  $\{1,2\}$  are on the right child node,  $\{3,4\}$  on the left.
  - Example: customers who are 45 years old or younger are on the right, others on the left.
- **Multisplit classification trees:** each node has an arbitrary number of branches.
  - It is easier to handle multi-valued categorical variables.
  - For numerical attributes, it is necessary to group together adjacent values. This can be achieved by discretization.
- Empirical evidence suggests no significant difference in performance of classification trees with regards to the number of children nodes.

# Introduction

A decision tree example for the concept PlayTennis:



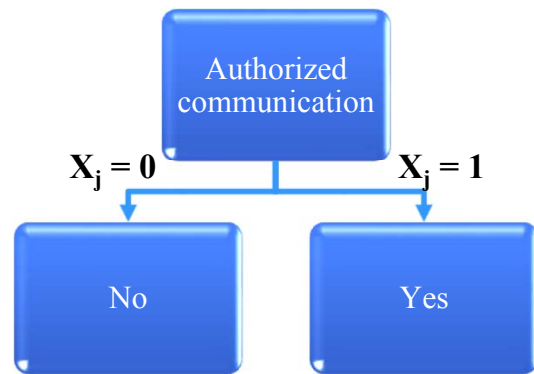


# Introduction

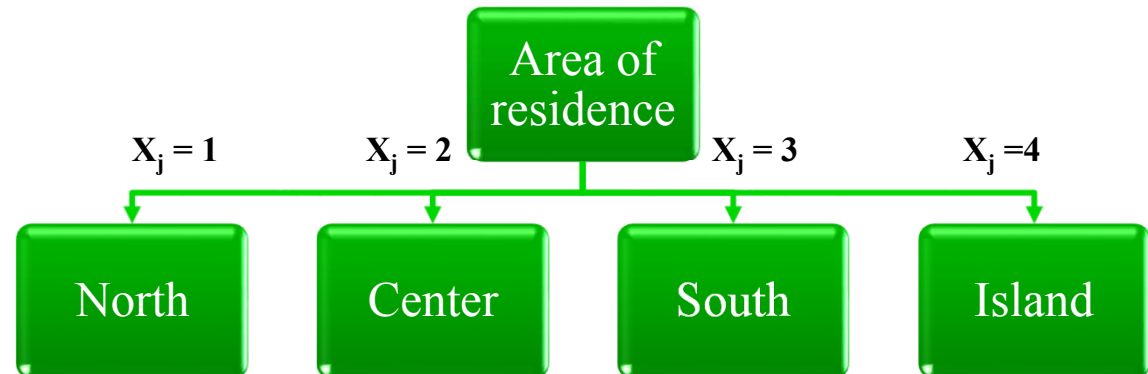
Two splitting rules according to the number of attributes selected

1. **Univariate**: based on the value of a single explanatory variable  $X_j$ .

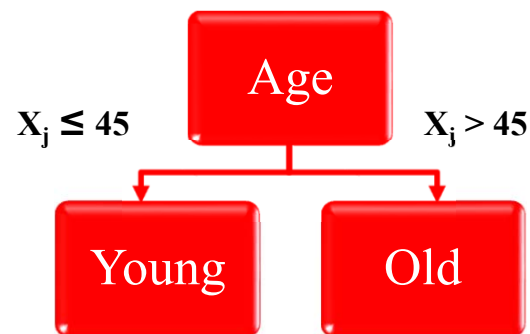
Examples:



Univariate split for a binary attribute



Univariate split for a nominal attribute



Univariate split for a numerical attribute

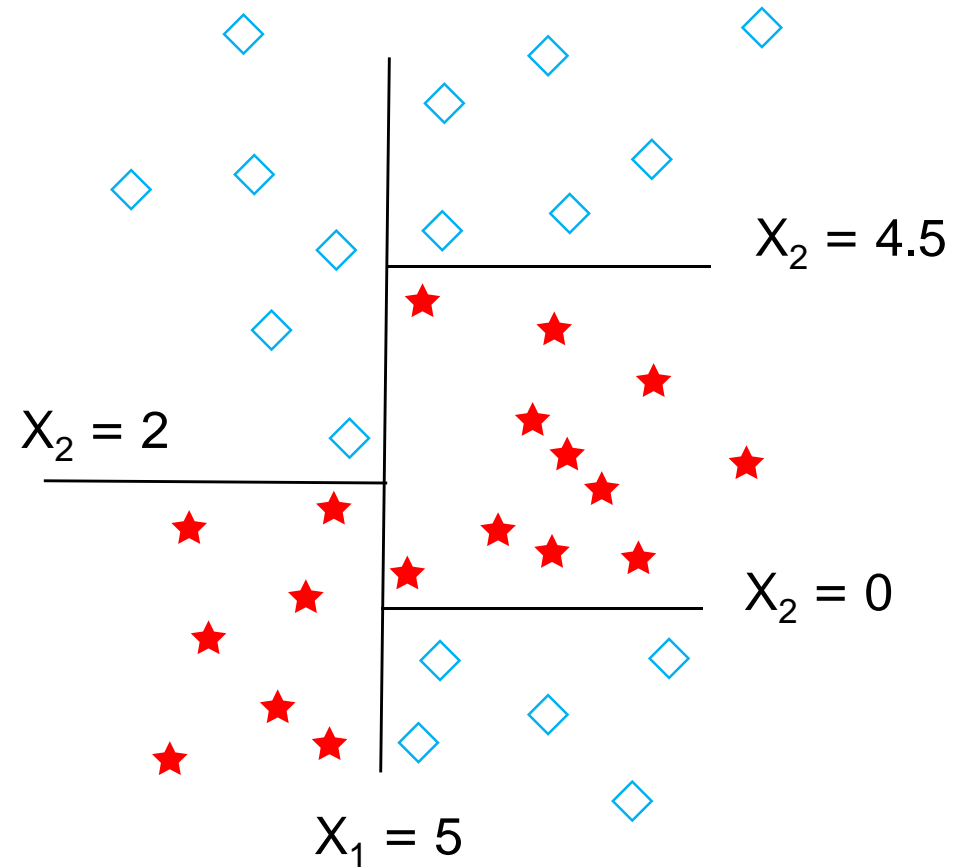
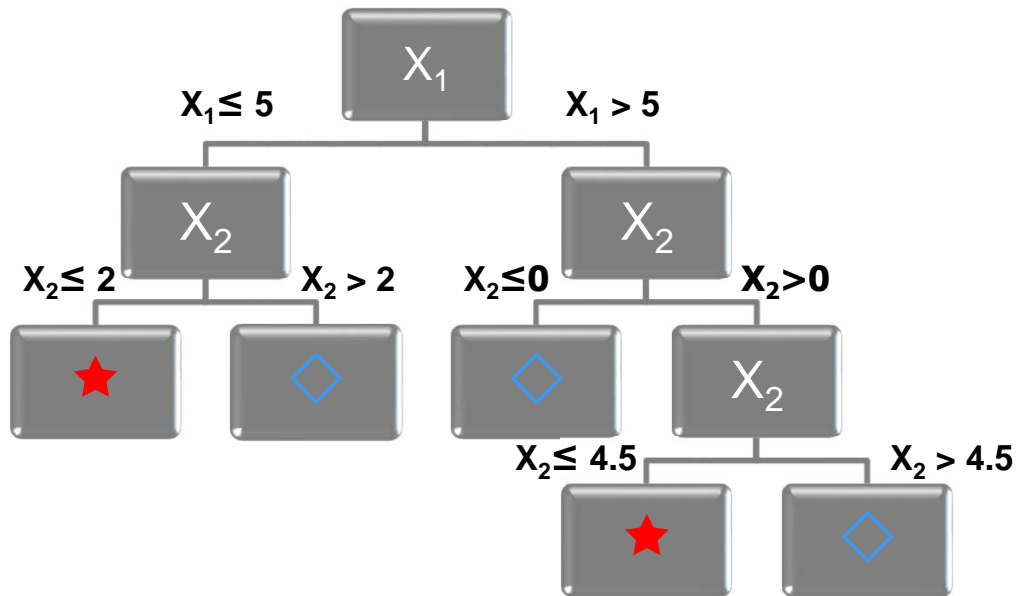


# Introduction

## Two splitting rules according to the number of attributes selected

Univariate trees are also called axis-parallel trees.

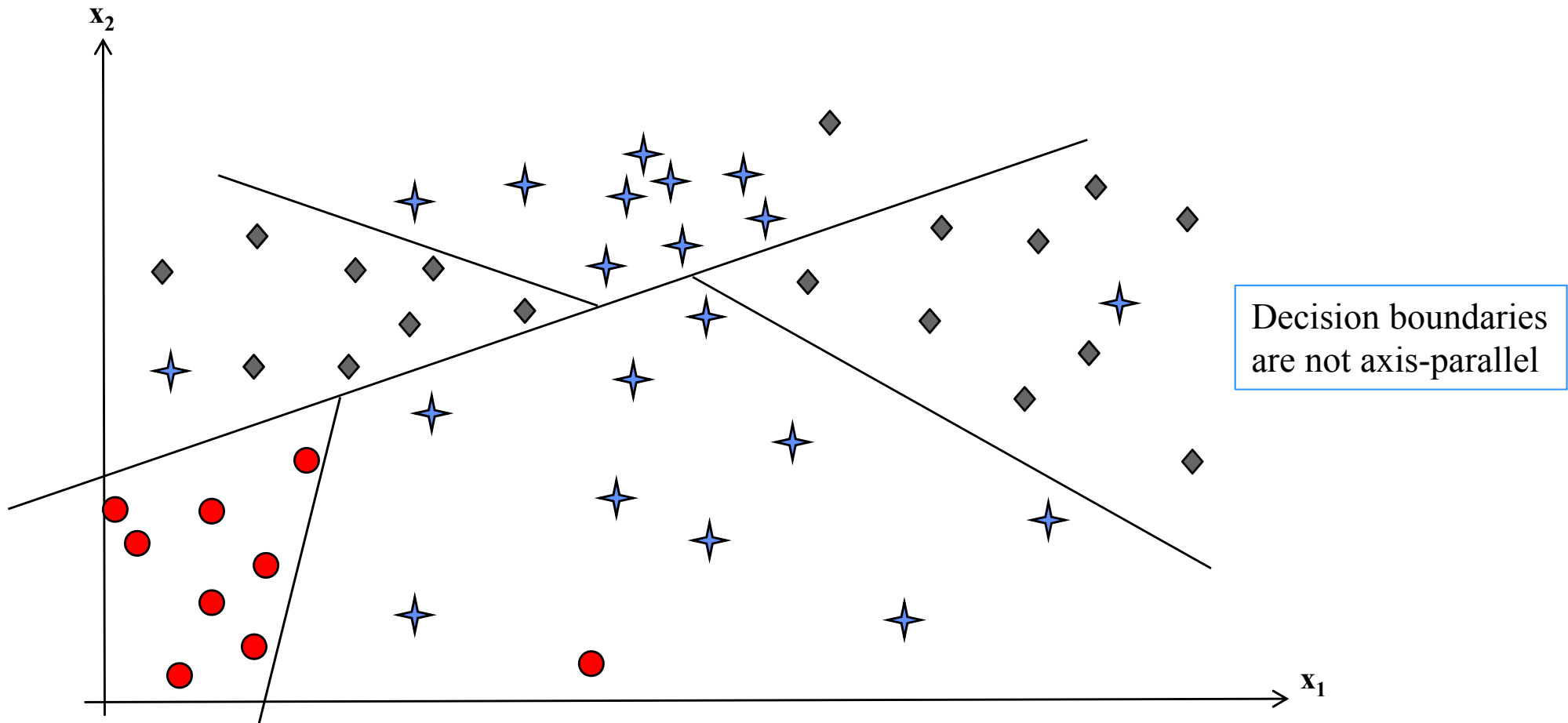
Examples:



# Introduction

2. **Multivariate trees** are also called oblique decision trees.

Observations are separated based on the expression:  $\sum_{j=1}^n w_j x_j \leq b$



## Decision making using classification tree:

- Samples are classified by sorting them down the tree from the root to the leaf node.
- The leaf node provides classification of the sample.
- Each non-leaf node in the tree specifies a test of one or more attributes of the sample.
- Each branch descending from a node corresponds to one of the possible values for these attributes.
- A sample is classified by starting at the root node of the tree, testing the attribute specified by this node, and moving down the tree branch corresponding to the value of the attribute in the given sample. This is repeated for the subtree rooted at the new node.

# Introduction

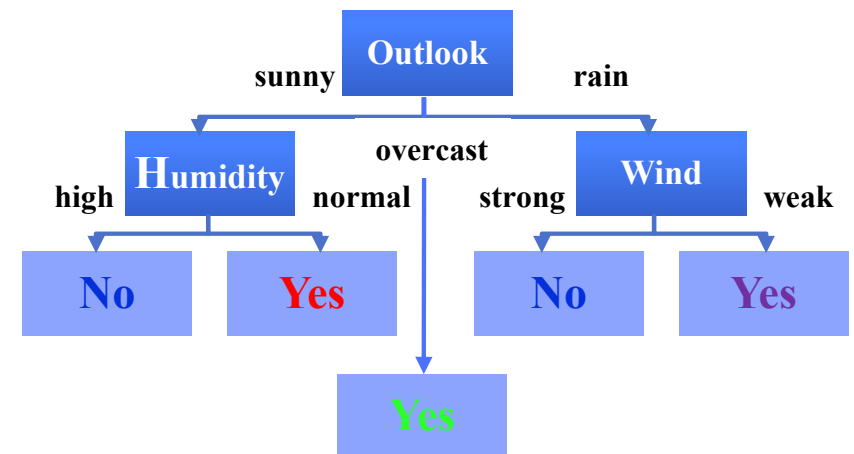
- A decision tree represents a **disjunction** (OR) of **conjunctions** (AND) of constraints on the attribute values of the samples.
- A path from the tree root to a leaf node corresponds to a conjunction of attribute tests.
- The tree is a disjunction of these conjunctions.
- The tree shown correspond to the expression:

$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$

$\vee (\text{Outlook} = \text{Overcast})$

$\vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$

then play tennis = yes



# Introduction

**Various decision tree generating methods differ in:**

- the selection of splits
- when to stop node splitting
- the assignment of a class to a non-split node

How a node is split:

- **ID3/C4.5/C5.0:** information gains are used, nodes are split using the value of a single feature.
- **CART** (Classification and Regression Trees): nodes are split to reduce impurity. A pure node has samples from only one class.
- **OC1 (Oblique Classifier 1):** randomization algorithms are used and nodes are split according to the values of all features.

## 2. ID3 (Inductive Dichotomizer 3)

- ID3 and its successors (C4.5 and C5.0) are the most widely used decision tree algorithms.
- They are developed by Ross Quinlan, University of Sydney.
- ID3 uses a single best attribute to test at each node of the tree, it selects the most useful attribute for classifying the samples.
- How can the "goodness" or “usefulness” of an attribute be measured?
- A statistical property called **information gain** is used.
- It measures how well a given attribute separates the training examples according to their target classification.

# ID3 (Inductive Dichotomizer 3)

## Entropy

- Entropy is computed to measure the amount of information in a message:
  - The key ideas are the amount of information is quantified by looking not at the message, but rather at the receiver end of the message.
  - the greater the surprise experienced by the receiver, the greater the amount of information, and vice versa.
- Shannon (1949) used the base 2 logarithm of the assigned probability:

$$I(M) = -\log_2 P(M)$$

- Large probability of receiving a message  $P(M)$  leads to small  $I(M)$  and vice versa.



# ID3 (Inductive Dichotomizer 3)

- The overall measure of expected surprise is given by the entropy formula

$$\text{Entropy} = - \sum_i P_i \log_2 P_i, \text{ where}$$

$$\sum_i P_i = 1 \text{ and } P_i \text{ is the probability of receiving the } i\text{-th message.}$$

- Why use logarithm?
  - Information can be summed up. Example: M1 and M2 are independent.

$$P(M1) = 0.2, P(M2) = 0.3, \text{ then}$$

$$P(M1 \wedge M2) = P(M1) \times P(M2) = 0.06 \text{ and}$$

$$-\log_2(0.06) = -\log_2(0.2) - \log_2(0.3)$$

- It suggests an intuitive link between the level of uncertainty in a given case and the amount of work we have to do to dispel the uncertainty.

For example: there are 128 books on the shelf, we need  $\log_2(128) = 7$  bits of information content of a message which tells us the exact location of the book.

# ID3 (Inductive Dichotomizer 3)

- Suppose the collection S consists of positive examples and negative examples of same target concept, the entropy S relative to this Boolean classification is

$$\text{Entropy} = - p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

where  $p_{\oplus}$  and  $p_{\ominus}$  are proportions of positive and negative examples, respectively.

- Example: S is a collection of 14 examples, 9 of which are positive and the remaining 5 are negative. The entropy S relative to this Boolean classification is

$$\text{Entropy}([9+,5-]) = - (9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) = 0.940$$

- If all examples belong to one class, the entropy is 0.**
- If 50% of the examples belong to one class, and the other 50% the other class, the entropy is equal to 1.**

# ID3 (Inductive Dichotomizer 3)

- Entropy specifies the number of bits of information needed to encode classification of a member of  $S$  that has been randomly selected with uniform probability.
- If  $p_{\oplus} = 1$ , then Entropy = 0. The receiver knows that a random example will be positive so that no message need be sent.
- If  $p_{\oplus} = 1/2$ , then Entropy = 1, that is one bit is needed to indicate whether the example drawn is positive or negative.
- If  $p_{\oplus} = 0.8$ , then a collection of messages can be encoded using on average less than one bit per message by assigning shorter codes to the collections of positive examples and longer codes to the less likely negative examples.
- If the target attribute can take on  $c$  possible values, the entropy can be as large as  $\log_2 c$ .

# ID3 (Inductive Dichotomizer 3)

- If  $p_{\oplus} = 0.8$ , then a collection of messages can be encoded using on average less than one bit per message by assigning shorter codes to the collections of positive examples and longer codes to the less likely negative examples.

**Consider a string consisting of just two symbols  $\oplus$  and  $\emptyset$ :**

- If we assign 1 bit for each symbol (say  $0 = \oplus$  and  $1 = \emptyset$ ), then we (a) have not made use of the fact that  $p_{\oplus} = 0.8$  and hence (b) need as many bits as the length of the string.
- Suppose we consider the symbols in “blocks” of 2 and we find:

$$p(\oplus, \oplus) = 0.65, \quad p(\oplus, \emptyset) = 0.15, \quad p(\emptyset, \oplus) = 0.15, \quad p(\emptyset, \emptyset) = 0.05$$

- Note:  $p(\oplus) = 0.65 + 0.075 + 0.075 + 0 = 0.80$ ,  $p(\emptyset) = 0 + 0.075 + 0.075 + 0.05 = 0.20$

Encoding:

Substring	Bit string	Probability
$\oplus, \oplus$	0	0.65
$\oplus, \emptyset$	10	0.15
$\emptyset, \oplus$	110	0.15
$\emptyset, \emptyset$	111	0.05

- **Average number of bits:**

$$(1)(0.65) + (2)(0.15) + (3)(0.15) + 3(0.05) = 1.55$$

- **Average number of bits per original symbol =  $0.775 = 1.55/2$**

- **Entropy([80%  $\oplus$ , 20%  $\emptyset$ ]) =**

$$- (4/5) \log_2 (4/5) - (1/5) \log_2 (1/5) = 0.722$$

# ID3 (Inductive Dichotomizer 3)

## Information gain

- **Information gain** measures the effectiveness of an attribute in classifying the training data.
- $\text{Gain}(S,A)$  of an attribute  $A$  relative to the collection of examples  $S$  is defined as

$$\text{Gain}(S,A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- $\text{Values}(A)$  is the set of all possible values for attribute  $A$ .
- $S_v$  is the subset of  $S$  for which attribute  $A$  has value  $v$ :
$$S_v = \{s \in S \mid A(s) = v\}$$
- $\text{Gain}(S,A)$  is the expected reduction in entropy by knowing the value of attribute  $A$ .
- The value of  $\text{Gain}(S,A)$  is the number of bits saved when encoding the target value of a member of  $S$ , by knowing the value of attribute  $A$ .

# ID3 (Inductive Dichotomizer 3)

Data set:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

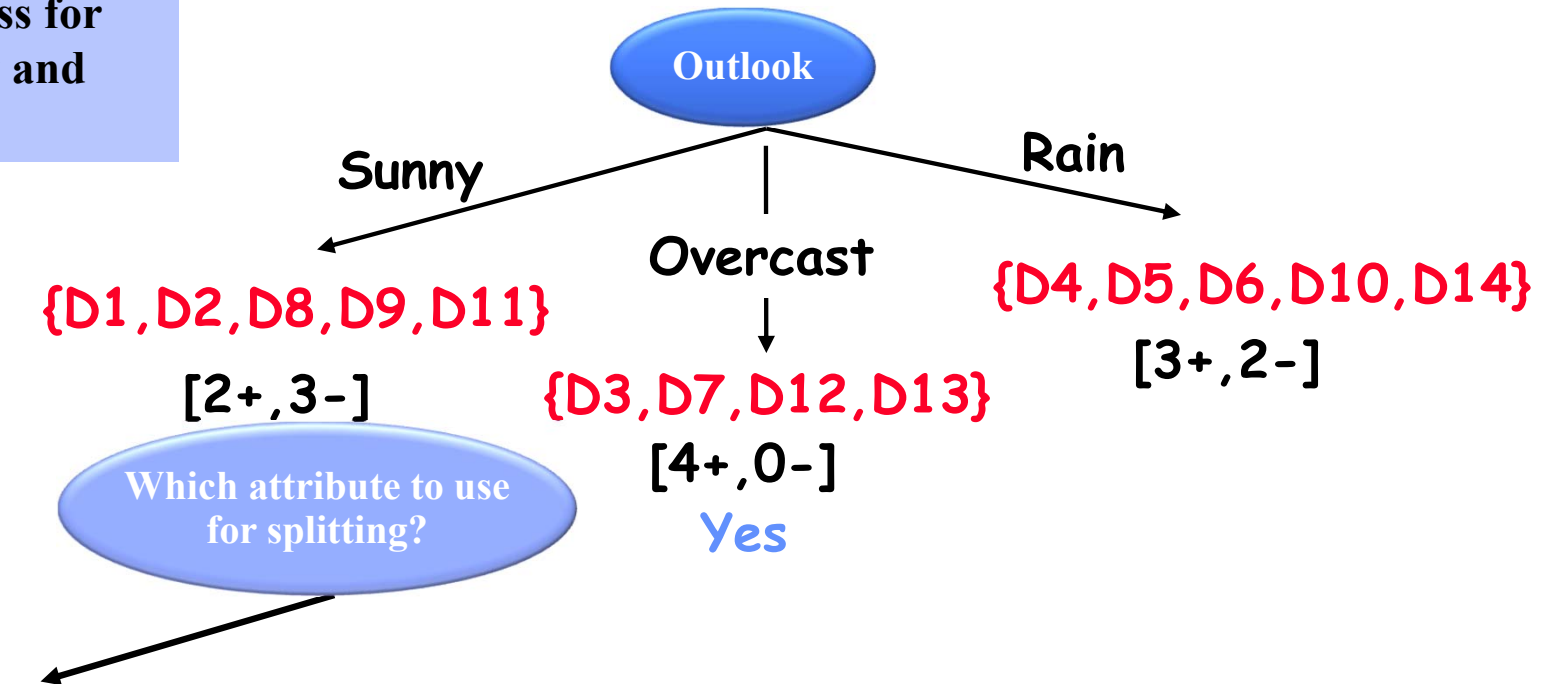
# ID3 (Inductive Dichotomizer 3)

- There are 9 positive (Yes) examples and 5 negative examples (No), so  $S = [9+, 5-]$
- $\text{Entropy}(S) = 0.940$ .
- Let us consider the attribute Wind which has 2 possible values, Weak and Strong.
- $S_{\text{weak}} \leftarrow [6+, 2-]$  and  $S_{\text{strong}} \leftarrow [3+, 3-]$
- $$\begin{aligned}\text{Gain}(S, \text{Wind}) &= \text{Entropy}(S) - \sum_{v \in \{\text{Weak}, \text{Strong}\}} |S_v|/|S| \text{Entropy}(S_v) \\ &= \text{Entropy}(S) - (8/14) \text{Entropy}(S_{\text{Weak}}) - (6/14) \text{Entropy}(S_{\text{Strong}}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 = 0.048\end{aligned}$$
- The gain is similarly computed for:
  - Humidity:  $\text{Gain}(S, \text{Humidity}) = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151$ .
  - Outlook:  **$\text{Gain}(S, \text{Outlook}) = 0.246$**  and
  - Temperature:  $\text{Gain}(S, \text{Temperature}) = 0.029$



# ID3 (Inductive Dichotomizer 3)

Repeat the process for  
Outlook = Sunny and  
Outlook = Rain



- $S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$ , Entropy = 0.970
- $\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$
- $\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.570$
- $\text{Gain}(S_{\text{sunny}}, \text{Wind}) = 0.970 - (2/5)1.0 - (3/5)0.918 = 0.019$
- Use Humidity as the branching variable.

# ID3 (Inductive Dichotomizer 3)

## Stopping criteria

- The process of selecting a new attribute and partitioning the training examples continues until one of the two conditions is satisfied:
  - Every attribute has already been included along this path through the tree.
  - The training examples associated with this leaf node all have the same target attribute value.
- Any given attribute can appear at most once along any path through the tree.
- Note: This data set has only discrete attributes.
  - Test for node splitting: if **Value of Attribute<sub>i</sub> = x**
  - The value of Attribute<sub>i</sub> for all data samples in the sub-tree below this node is **x**: not useful to check on this Attribute<sub>i</sub> again.

# ID3 (Inductive Dichotomizer 3)

## **ID3: capabilities and limitations**

- ID3 maintains only a single current hypothesis as it searches through the space of the decision trees. It loses the ability to determine how many alternative trees are consistent with the samples.
- ID3 does not perform backtracking in its search. Once an attribute is selected at a particular level in the tree, it never reconsiders this choice. It may get a local solution, not global solution.
- ID3 uses all training samples at each step in the search to refine the current hypothesis. It is less sensitive to errors in individual training examples. It can handle noisy data easily.

# ID3 (Inductive Dichotomizer 3)

## ID3: capabilities and limitations

- ID3 evaluates the contribution of a single attribute in each node, the feature space is divided into rectangular subregions. Large trees may be generated for more ‘difficult’ problems where it would be better to consider the contribution of several attributes at once.
- Consider the well-known Monk test problem. The attributes are:
  - A1. Head shape  $\in \{\text{round, square, octagon}\}$
  - A2. Body shape  $\in \{\text{round, square, octagon}\}$
  - A3. Is smiling  $\in \{\text{yes, no}\}$
  - A4. Holding  $\in \{\text{sword, ballon, flag}\}$
  - A5. Jacket color  $\in \{\text{red, yellow, green, blue}\}$
  - A6. Has tie  $\in \{\text{yes, no}\}$

# ID3 (Inductive Dichotomizer 3)

**Monk2 : Exactly two of the 6 attributes have their first value.**

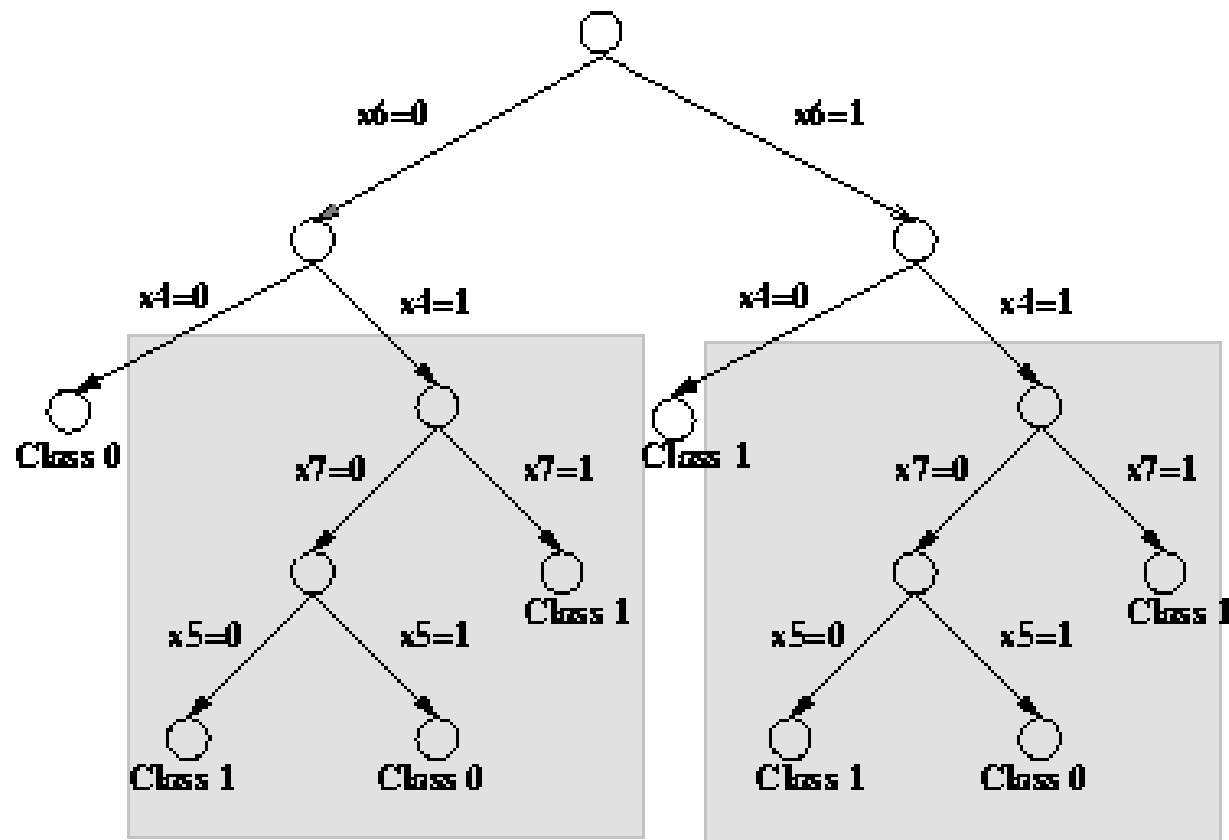
Example: {round, square, yes, flag, blue, no} is a monk

{square, square, no, balloon, green, yes} is not a monk

- Given 169 training samples, the tree generated by C4.5 has 47 leaf nodes and 86.4% accuracy. On the entire sample set consisting of 432 samples, the accuracy is 70.4%.
- The two problems of decision tree methods that test on a single attribute at each node are:
  - **fragmentation problem**: partitioning of the sample space reduces the support or evidential credibility of each rule.
  - **replication problem**: replication of subtrees along the output tree.

# ID3 (Inductive Dichotomizer 3)

- Example: if  $(x_4 \vee x_6) \wedge (\neg x_4 \vee \neg x_5 \vee x_7)$ , then class 1,  
otherwise class 0.
- Decision tree generated:

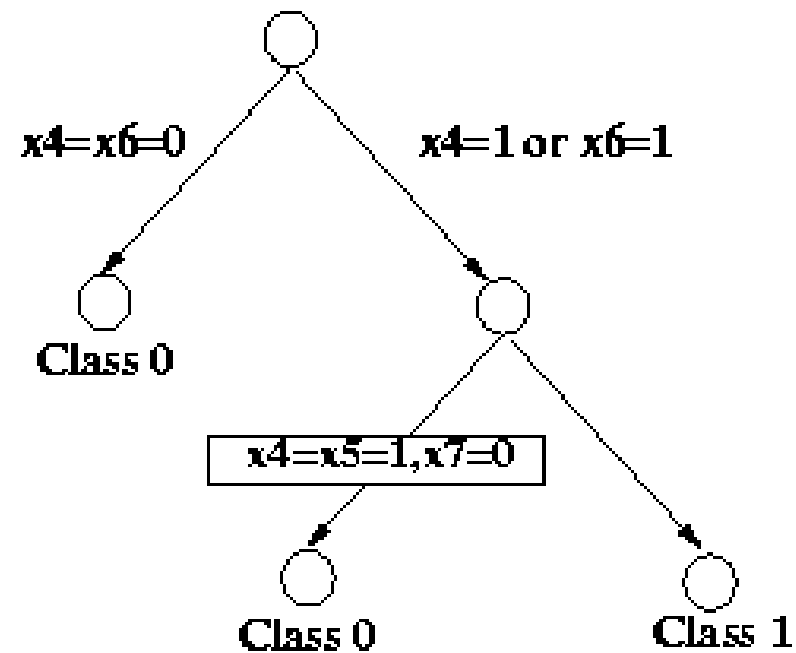


# ID3 (Inductive Dichotomizer 3)

if  $(x_4 \vee x_6) \wedge (\neg x_4 \vee \neg x_5 \vee x_7)$ , then class 1,

otherwise class 0.

- If we can test more than one attribute at each node with Feature construction:  
constructing different compound features at every tree node.





# ID3 (Inductive Dichotomizer 3)

## Avoiding overfitting the data:

- **Definition:** Given a hypothesis space  $\mathcal{H}$ , a hypothesis  $k \in \mathcal{H}$  is said to **overfit** the training data if there exists some alternative hypothesis  $k' \in \mathcal{H}$ , such that  $k$  has smaller training error than  $k'$  over the training samples, but  $k'$  has a smaller error than  $k$  over the entire distribution of instances.
- Overfitting may occur when the training examples contain random error or noise.
- Two types of approaches used to avoid/reduce overfitting:
  - stop growing the tree earlier
  - allow the tree to overfit the data, and then post-prune the tree.

# ID3 (Inductive Dichotomizer 3)

**What is the criterion that can be used to determine the final tree size?**

- apply the training and validation set approach.

A validation set consists of examples that are distinct from those in the training set.

- use all the available data for training but apply statistical test such as the chi-square test to estimate if further expanding a node is likely to improve the performance over the entire instance distribution or only on the current sample data.

# ID3 (Inductive Dichotomizer 3)

## Reduced error pruning:

- Remove the sub-tree rooted at a node, making it a leaf node and assigning it the most common classification of the training examples in that node.
- Nodes are removed only if the resulting pruned tree performs no worse than the original over the cross-validation set.
- A major drawback of this approach is when the available data is limited.

# ID3 (Inductive Dichotomizer 3)

## The steps in rule post-pruning:

- infer the decision tree from the training set, grow the tree until the training data fit as well as possible and allow overfitting to occur.
- convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
- prune each rule by removing any preconditions that result in improving its estimated accuracy.
- sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying new examples.
- Example:

(Outlook = Sunny  $\wedge$  Humidity = High) then play tennis = no

one of the two conditions in the above rule will be considered for possible removal.

# ID3 (Inductive Dichotomizer 3)

## Handling continuous attributes

Suppose that the attribute Temperature has continuous values.

- The following table shows the training examples associated with a particular node in the decision tree:

<i>Temperature</i>	<i>40</i>	<i>48</i>	<i>60</i>	<i>72</i>	<i>80</i>	<i>90</i>
<b>Play tennis</b>	No	No	Yes	Yes	Yes	No

- The value of **c** which maximises the information gain lies midway between the values that differ in their target classification.
- For the above data, there are 2 candidate thresholds:

$$(48+60)/2 = 54 \text{ and } (80 + 90)/2 = 85.$$

- The information gain is computed for each of the candidate attributes

Temperature<sub>>54</sub> and Temperature<sub>>85</sub>.

- The best is selected, it is Temperature<sub>>54</sub>

# ID3 (Inductive Dichotomizer 3)

## Split Information value and Gain Ratio

- Consider the attributes date for learning the play-tennis concept or student identification number for predicting the student's CAP.
- Such attributes have the highest information gain but are not useful in learning the concepts.
- One way to avoid this difficulty is to have a new measure of information gain.
- Define

$$\text{SplitInformation}(S,A) \equiv - \sum_{i=1}^c |S_i|/|S| \log_2 |S_i|/|S|$$

where  $S_1, S_2, \dots, S_c$  are the  $c$  subsets of examples split according to the values of attribute  $A$ .

# ID3 (Inductive Dichotomizer 3)

- Define

$$\text{GainRatio}(S,A) \equiv \text{Gain}(S,A)/\text{SplitInformation}(S,A)$$

- SplitInformation discourages the selection of attributes with many values: A set of  $n$  examples that are completely separated by an attribute like **date**:

$$\text{SplitInformation}(S,A) = \log_2 n.$$

- In contrast, for a Boolean attribute  $B$  that splits the examples in half:

$$\text{SplitInformation}(S,B) = 1.$$

- The denominator in  $\text{GainRatio}(S,A)$  can be very small or zero when  $|S_i| \approx |S|$  for one of the  $i$  subsets. The heuristic approach is to apply the GainRatio test only to attributes that have above average Gain.



# ID3 (Inductive Dichotomizer 3)

## Missing attribute values

- Suppose  $\text{Gain}(S,A)$  is to be computed at node  $n$  to determine if attribute  $A$  is the best to test at this node.
- How can  $\text{Gain}(S,A)$  be computed when for one sample  $(x,c(x))$  the value of  $A(x)$  is not known?
  - Strategy 1: Assign  $A(x) = \text{most common value of attribute } A \text{ among the training samples in node } n$ .
  - Strategy 2: Assign the probability to each possible value of  $A$ .

For example, node  $n$  contains 6 samples with  $A = 1$  and 4 samples with  $A = 0$ , then  $p(A(x) = 1) = 0.6$  and  $p(A(x) = 0) = 0.4$ . Sixty percent of the samples are distributed down the branch for  $A = 1$  and the rest down the other tree branch.

### 3. CART (Classification And Regression Tree)

- CART is a binary decision tree algorithm: non leaf nodes have two branches.
- It differs from ID3 in how it evaluates the attributes for branching.
- The **impurity measure** is used and computed as the GINI diversity index of node impurity:

$$I(T) = \sum_{i \neq j} (p_i|T) (p_j|T) = 1 - \sum_i (p_i|T)^2$$

where  $(p_i|T)$  is the probability of class  $i$  in node  $T$ .

- For a binary classification problem the GINI index becomes

$$I(T) = 2(p_1|T) (p_2|T) = 1 - (p_1|T)^2 - (p_2|T)^2$$

- For each possible split the impurity of the subgroups is summed and the split with the maximum reduction in impurity is chosen.

# CART (Classification And Regression Tree)

- For ordered and numeric attributes, all possible splits are considered in sequence.

When there are  $n$  values, there are  $n-1$  possible splits.

- Example: {Small, Medium, Large}
- Possible 2 splits: {S} vs {M,L} and {S,M} vs {L}

- For categorical attributes, all possible binary splits are considered. When there are  $n$  values of the attribute, there are  $2^{(n-1)} - 1$  splits.

- Example: {Red, Yellow, Blue, Green}
- Possible  $2^{(4-1)} - 1 = 7$  splits:

{R} vs {Y,B,G}, {Y} vs {R,B,G}, {B} vs {R,Y,G} and {G} vs {R,Y,B}

{R,Y} vs {B,G}, {R,B} vs {Y,G} and {R,G} vs {Y,B}

# CART (Classification And Regression Tree)

## Example 1:

- At node n1, the subset contains 5 samples from class 1, 10 samples of class 2, 25 samples of class 3:

$$\begin{aligned} I(n1) &= 2 (p_1|n1) (p_2|n1) + 2 (p_2|n1) (p_3|n1) + 2 (p_3|n1) (p_1|n1) \\ &= 2 \times (5/40) \times (10/40) + 2 \times (10/40) \times (25/40) + 2 \times (25/40) \times (5/40) = 0.53125 \end{aligned}$$

- At node n2, the subset contains 2 samples from class 1, 3 samples of class 2, 35 samples of class 3:


$$\begin{aligned} I(n2) &= 2 \times (2/40) \times (3/40) + 2 \times (3/40) \times (35/40) + 2 \times (35/40) \times (2/40) = 0.22625 \\ &= 1 - (2/40)^2 - (3/40)^2 - (35/40)^2 \end{aligned}$$

# CART (Classification And Regression Tree)

## Example 2:

- Should Income be used as the attribute to split the root node?
- Income is an attribute with continuous values.
- Sort the data according to Income values:

	Observation #	Income	Credit Rating	Target variable: Loan Risk
	1	17	Low	High
	5	20	High	High
Split 1	0	23	High	High
Split 2	4	32	Moderate	Low
Split 3	2	43	Low	High
	3	68	High	Low



# CART (Classification And Regression Tree)

## Example 2 (continued):

- We consider 3 possible splits when there are changes in the value of Loan-Risk.
- Case 1: split condition  $\text{Income} \leq 23$  versus  $\text{Income} > 23$

Observation #	Income	Credit Rating	Target variable: Loan Risk
1	17	Low	High
5	20	High	High
0	23	High	High
4	32	Moderate	Low
2	43	Low	High
3	68	High	Low

Predict: Loan Risk = High  
Acc = 66.67%  
Gini(T) = 4/9

Income  $\leq 23$

Income  $> 23$

3 High Loan-Risks  
0 Low Loan Risk  
Gini(T<sub>1</sub>) = 0

1 High-Loan Risk  
2 Low Loan Risks  
Gini(T<sub>2</sub>) = 4/9

$$\begin{aligned}\text{Gini}(T) &= 1 - \left(\frac{2}{3} \times \frac{2}{3}\right) - \left(\frac{1}{3} \times \frac{1}{3}\right) \\ &= \frac{4}{9} = 0.4444\end{aligned}$$

Impurity after split:

$$\begin{aligned}I_G(T_1, T_2) &= \left(\frac{3}{6}\right) \times 0 + \left(\frac{3}{6}\right) \times \left(\frac{4}{9}\right) = \frac{2}{9} \\ &= 0.22222\end{aligned}$$

# CART (Classification And Regression Tree)

## Example 2 (continued):

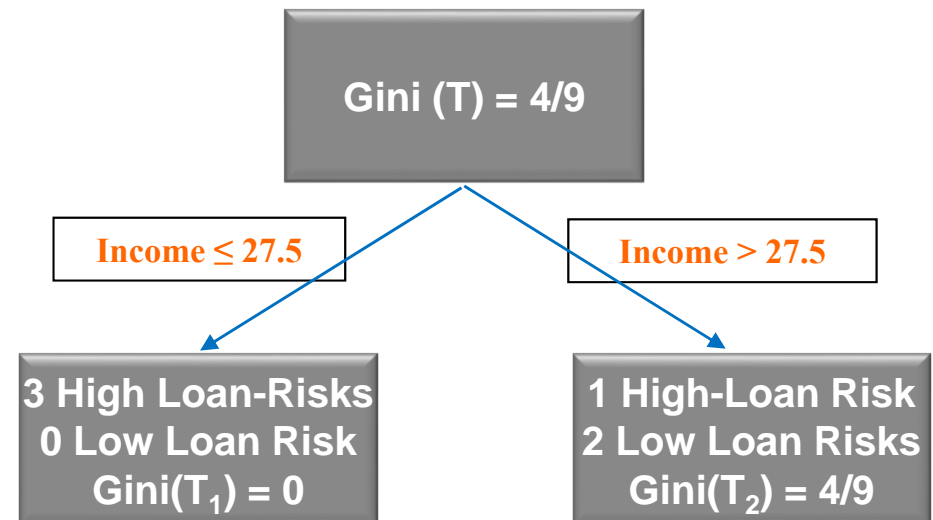
- Case 2: split condition  $\text{Income} \leq 32$  versus  $\text{Income} > 32$

$$I_G(q_1, q_2) = (4/6) \times (3/8) + (2/6) \times (1/2) = 5/12 = 0.41667$$

- Case 3: split condition  $\text{Income} \leq 43$  versus  $\text{Income} > 43$

$$I_G(q_1, q_2) = (5/6) \times (8/25) + (1/6) \times 0 = 4/15 = 0.26667$$

- Case 1 is the **best**.
- Instead of splitting between  $\text{Income} \leq 23$  versus  $\text{Income} > 23$ , the midpoint is selected as actual splitting point:  $(23 + 32)/2$



## 4. OC1 (Oblique classifier)

- OC1 generates oblique decision trees.

- At each node, the test has the form  $\sum_{i=1}^d a_i x_i + a_{d+1} > 0$

where  $a_1, a_2, \dots, a_{d+1}$  are real-valued coefficients and  $d$  is the number of attributes.

- OC1 applies a randomisation algorithm to find the coefficients of the hyperplane to reduce node impurity.
- It first finds the best axis-parallel (single attribute) split and uses the oblique split only if it improves the axis-parallel split.
- Randomisation is done by having a random vector  $(r_1, r_2, \dots, r_{d+1})$  and a parameter  $\alpha$  and computing the new hyperplane:

$$\sum_{i=1}^d (a_i + \alpha r_i) x_i + (a_{d+1} + \alpha r_{d+1})$$



# OC1 (Oblique classifier)

- Impurity measures include information gain, the GINI index, and the **twoing rule**.
- The twoing rule is the default impurity measure:

$$\text{Twoing value} = (|T_L|/n) \times (|T_R|/n) \times \left( \sum_{i=1}^k \left| L_i/|T_L| - R_i/|T_R| \right| \right)^2$$

where  $|T_L|$  is the number of samples on the left subtree

$|T_R|$  is the number of samples on the right subtree

$n$  is the number of samples at node  $T$

$L_i$  is the number of samples of class  $i$  at left subtree

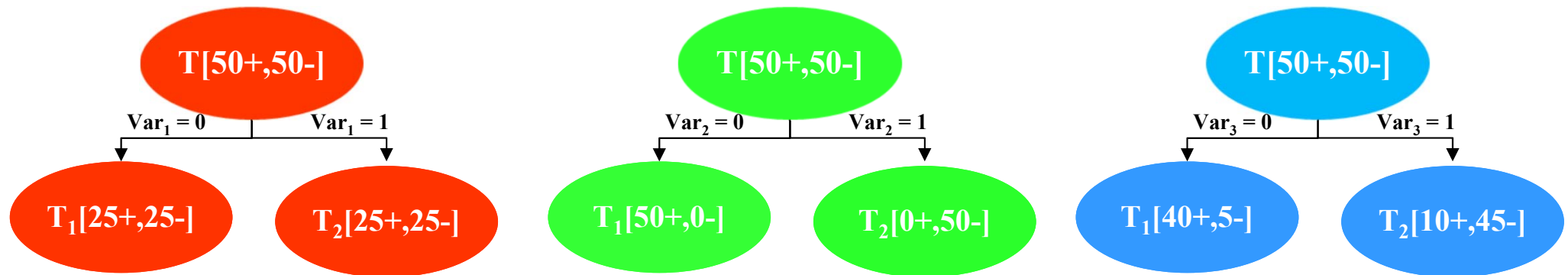
$R_i$  is the number of samples of class  $i$  at right subtree

$k$  is the number of classes

# OC1 (Oblique classifier)

## Example:

- Consider the following 3 possible splits at node T [50+,50-] :



- Red**  $TV = (50/100) \times (50/100) \times [ |25/50 - 25/50| + |25/50 - 25/50| ]^2 = 0$
- Green**  $TV = (50/100) \times (50/100) \times [ |50/50 - 0/50| + |0/50 - 50/50| ]^2 = 1$
- Blue**  $TV = (45/100) \times (55/100) \times [ |40/45 - 10/55| + |5/45 - 45/55| ]^2 = 0.49$
- OC1 attempts to **maximise** the Twoing Value.

## 5. Decision tree applications (1)

- A journal article that describes an application of decision trees (and other machine learning methods):

Detecting financial restatements using data mining techniques

by Ila Dutta, Shantanu Dutta, Bijan Raahemi,

Expert Systems Applications, Vol. 90 (2017), pp. 374-393.

- Problem: differentiate between (intentional/fraudulent + unintentional/erroneous financial restatements) and non-restatements.
- Data set: 3513 restatement cases over a period of 2001 to 2014, 109 intentional + 3404 unintentional from Audit Analytics database + COMPUSTAT financial database.
- 116 input attributes
- Combined data set: 3513 restatement cases (class 1) and 60710 non-restatement category (class 0)

# 5. Decision tree applications (1)

Detecting financial restatements using data mining techniques

**Research methodology:** Cross Industry Standard Process for Data Mining (CRISP – DM)

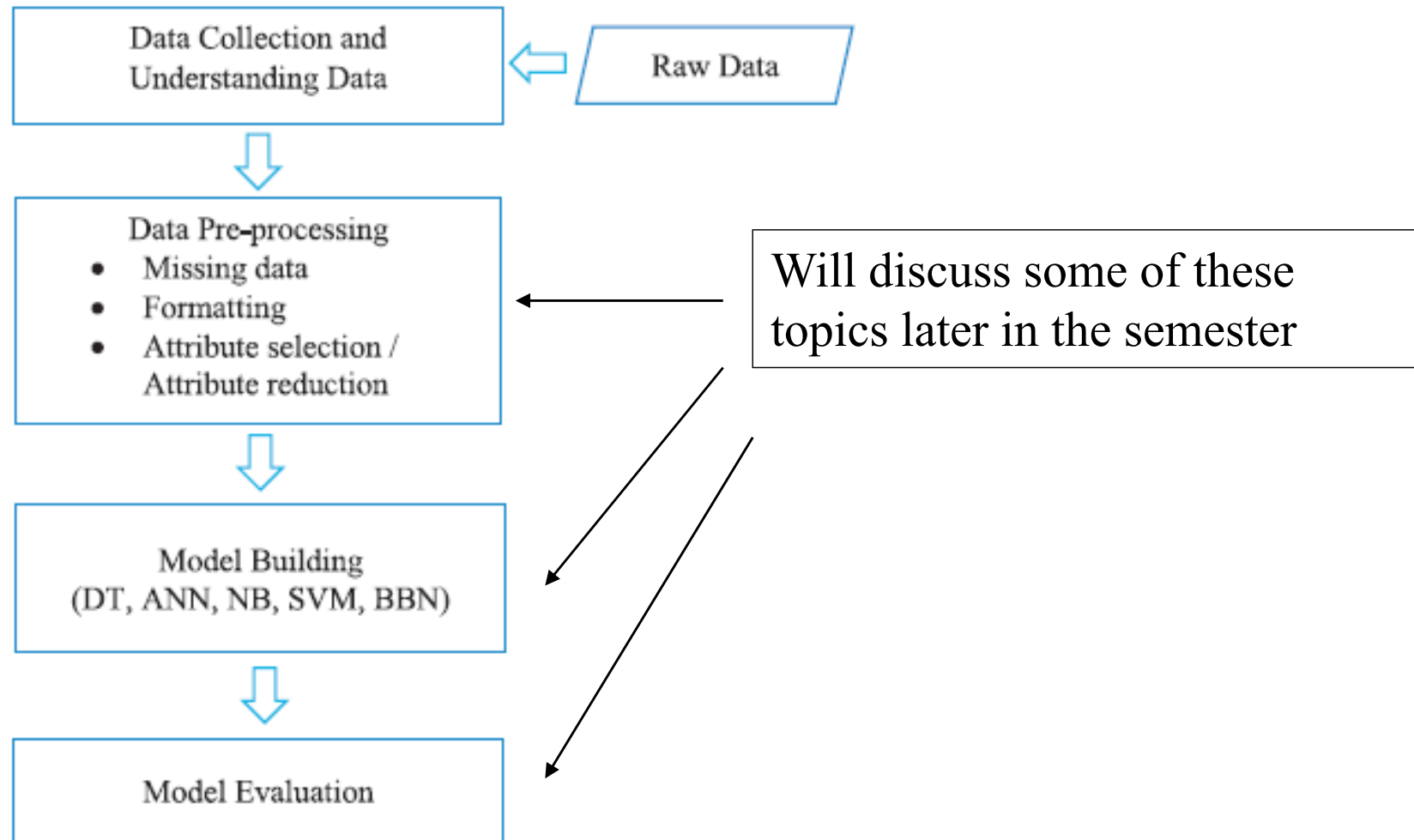


Fig. 1. A schematic diagram of the research methodology: CRISP-DM Approach.

## 5. Decision tree applications (1)

Detecting financial restatements using data mining techniques

**Research methodology:** Cross Industry Standard Process for Data Mining (CRISP – DM)

Results:

**Table 6**

Performance measures for the period 2001 to 2014.

Classifier	Recall	FP	Accuracy	Precision	Specificity	F - Measure	AUC
DT	0.778	0.312	77.756	0.774	0.688	0.775	0.818
ANN	0.78	0.363	77.9846	0.772	0.637	0.769	0.83
NB	0.602	0.311	60.2067	0.701	0.689	0.614	0.693
SVM	0.656	0.333	65.6005	0.704	0.667	0.668	0.662
BBN	0.726	0.311	72.6155	0.741	0.689	0.732	0.786

# Reference

- T. M. Mitchell, Machine Learning, Chapter 3, 1997, McGraw Hill.
- L. Breiman, J.H. Friedman, R.A. Olshen and C.J.Stone, 1984, Classification and Regression Trees, Wadsworth and Brooks.
- D. Michie, D.J. Spiegelhalter and C.C. Taylor, Machine Learning, Neural and Statistical Classification, 1994, Ellis Horwood. Also available at:

<http://www.amsta.leeds.ac.uk/~charles/statlog/>

This book provides a review of different approaches to classification, including ID3, CART, and neural networks. It compares the performance of these methods on a wide range of problems.