IIT ROORKEE

NPTEL ONLINE CERTIFICATION COURSE

Classification and Regression Trees (CART - I)

Dr. A. Ramesh

DEPARTMENT OF MANAGEMENT STUDIES

Agenda

- Introduction to Classification and Regression Trees
- Attribute selection measures – Introduction

Introduction

Distinction between classification and regression analysis in the context of data analysis.

Classification

Classification is a method used in data analysis to develop models that describe distinct data classes or predict categorical labels. It's particularly useful when dealing with discrete and unordered categories. For instance, a classification model could be employed to categorize bank loan applications as either safe or risky based on various features and historical data.

Regression Analysis

Regression analysis, on the other hand, is a statistical technique mainly utilized for numeric prediction. It's commonly applied when the goal is to predict continuous variables. An example could be predicting the amount of money potential customers might spend on computer equipment based on factors like their income and occupation.

In summary, classification is employed when the outcome is categorical, such as labeling something as safe or risky, while regression analysis is used when the outcome is numerical, like predicting dollar amounts. Both methodologies serve distinct purposes within data analysis.

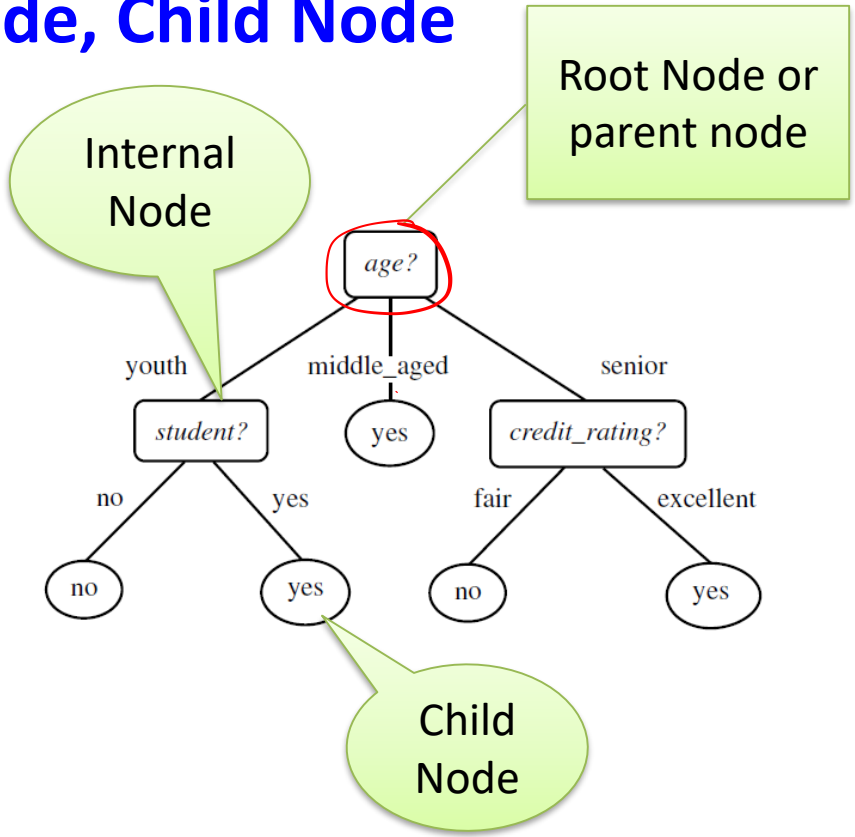
Problem Description for Illustration

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Han, J., Pei, J. and Kamber, M., 2011. *Data mining: concepts and techniques*. Elsevier.

Root Node, Internal Node, Child Node

- A decision tree uses a tree structure to represent a number of possible *decision paths* and an outcome for each path
- A decision tree consists of root node, internal node and leaf node
- The topmost node in a tree is the **root node** or **parent node**
- It represents entire sample population
- **Internal node** (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test
- **Leaf node** (or **terminal** node or **child** node) holds a class label
- It can not be further split



Decision Tree Introduction

- A decision tree for the concept *buys_computer*, indicating whether a customer at All Electronics is likely to purchase a computer

- Each internal (non-leaf) node represents a test on an attribute
- Each leaf node represents a class (either *buys_computer = yes* or *buys computer = no*).

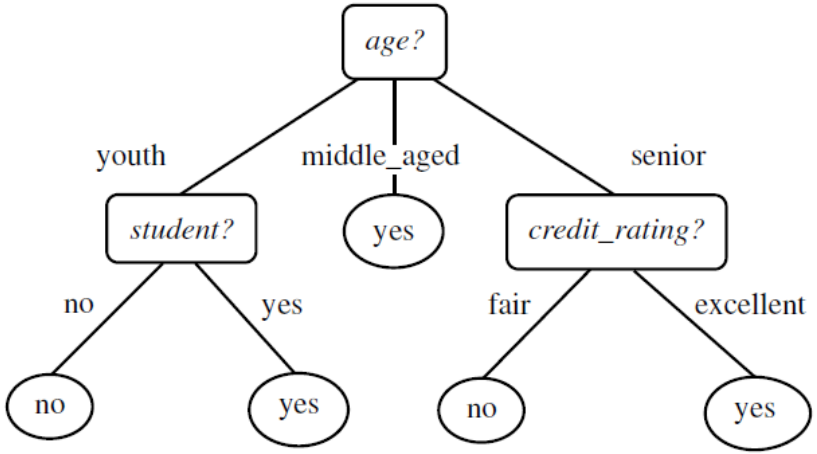


Figure 1.1 : Decision Tree

CART Introduction

- CART comes under supervised learning technique
- CART adopt a greedy(i.e., non backtracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner
- It is very interpretable model

Decision Tree Algorithm

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;
- Attribute list, the set of candidate's attributes;
- Attribute selection method, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split point or splitting subset.

Output: **A decision tree**

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Decision Tree Algorithm

- The algorithm is called with three parameters: Data partition D, attribute list, and Attribute selection method
- D is defined as a data partition. Initially, it is the complete set of training tuples and their associated class labels
- The parameter attribute list is a list of attributes or independent variables which are describing the tuples
- Attribute selection method specifies a heuristic procedure for selecting the attribute that “best” discriminates the given tuples according to class

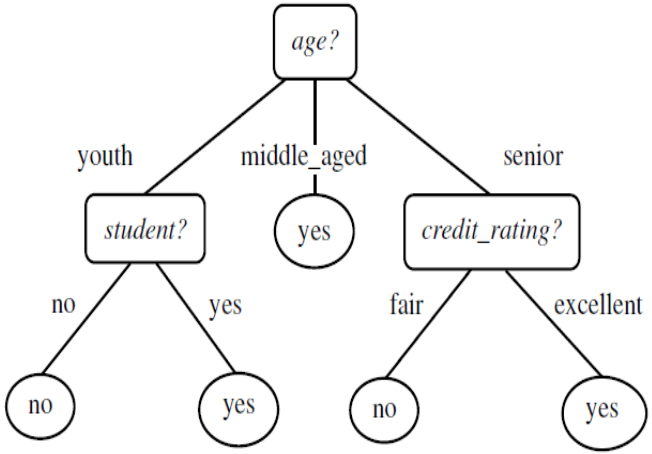
Decision Tree Algorithm

- This procedure employs an attribute selection measure, such as information gain, gain ratio or the Gini index.
- Whether the tree is strictly binary or not is generally driven by the attribute selection measure
- Some attribute selection measures, such as the Gini index, enforce the resulting tree to be binary. Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).

Decision Tree Method

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;



N-Node
C- Class
D- tuples in training data set

Decision Tree Method step 1 to 6

- The tree starts as a single node, N, representing the training tuples in D (step 1).
- If the tuples in D are all of the same class, then node N becomes a leaf and is labelled with that class (steps 2 and 3)
- Steps 4 and 5 are terminating conditions
- Otherwise, the algorithm calls Attribute selection method to determine the splitting criterion
- The splitting criterion (like Gini) tells us which attribute to test at node N by determining the "best" way to separate or partition the tuples in D into individual classes (step 6)

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;

Decision Tree Method - Step 7 - 11

- The splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset
- The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as "pure" as possible. A partition is pure if all of the tuples in it belong to the same class.
- The node N is labelled with the splitting criterion, which serves as a test at the node (step 7).
- A branch is grown from node N for each of the outcomes of the splitting criterion.
- The tuples in D are partitioned accordingly (steps 10 to 11)

Method:

(1) create a node N;
(2) **if** tuples in D are all of the same class, C **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;

Three possibilities for partitioning tuples based on the splitting criterion

- There are three possible scenarios, as illustrated in Figure (a), (b) and (c).
- Let A be the splitting attribute. A has ‘v’ distinct values,{a1,a2,...,av}, based on the training data
- If A is discrete-valued in figure (a), then one branch is grown for each known value of A.

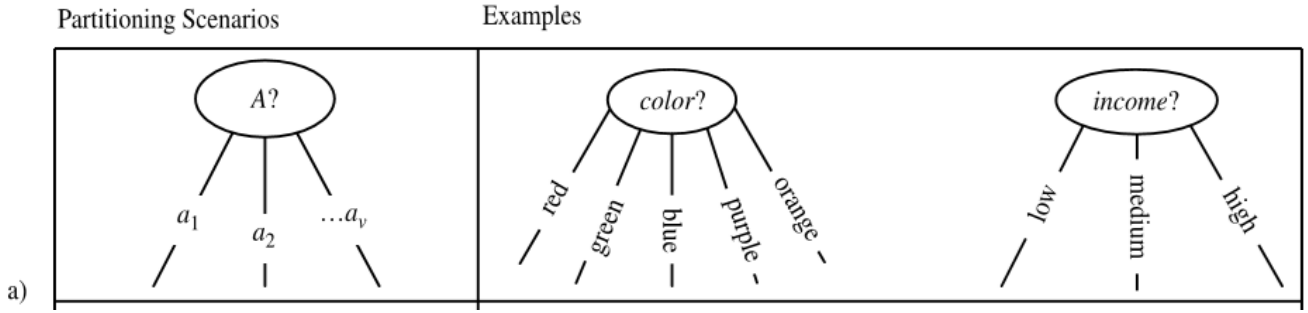


Figure (a)

Three possibilities for partitioning tuples based on the splitting criterion

- If A is continuous-valued in figure (b), then two branches are grown, corresponding to A ≤ split point and A > split point.
- Where split point is the split-point returned by Attribute selection method as part of the splitting criterion.

Figure (b)

Three possibilities for partitioning tuples based on the splitting criterion

- If A is discrete-valued and a binary tree must be produced, then the test is of the form A ∈ S_A, where S_A is the splitting subset for A.

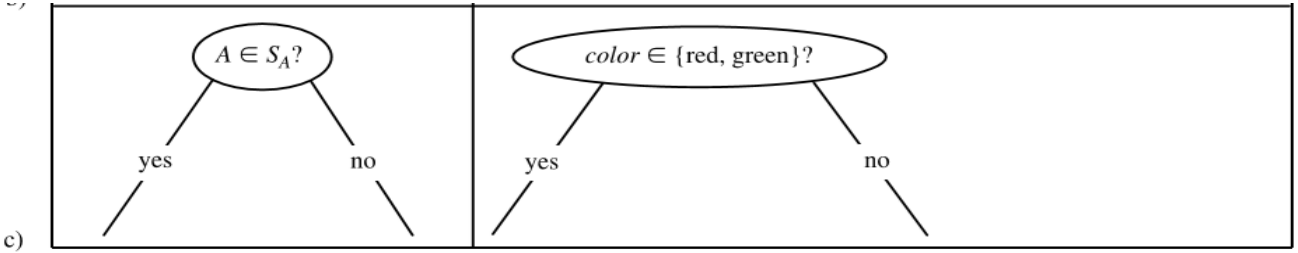


Figure (c)

Decision Tree Method – termination condition

- The algorithm uses the same process recursively to form a decision tree for the tuples D_j at each resulting partition, j (step 14).

- The recursive partitioning stops only when anyone of the following terminating conditions is true:

1. All of the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3), or

Decision Tree Method – termination condition

2. There are no remaining attributes on which the tuples may be further partitioned (step4).
   - In this case, majority voting is employed(step 5).
   - This involves converting node N into a leaf and labelling it with the most common class in D.
   - Alternatively, the class distribution of the node tuples may be stored.
3. There are no tuples for a given branch, that is, a partition Dj is empty (step 12).
   - In this case, a leaf is created with the majority class in D (step 13).
   - The resulting decision tree is returned (step 15).

Attribute Selection Measures

- Attribute selection measures are also known as splitting rules because they determine how the tuples at a given node are to be split
- It is a heuristic approach for selecting the splitting criterion that “best” separates a given data partition, D, of class-labeled training tuples into individual classes
- The attribute selection measure provides a ranking for each attribute describing the given training tuples
- The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples

Attribute Selection Measures

- If the splitting attribute is continuous-valued or if we are restricted to binary trees then, respectively, either a ‘split point’ or a ‘splitting subset’ must also be determined as part of the splitting criterion

- There are three popular attribute selection measures
  - information gain,
  - gain ratio, and
  - Gini index

- CART algorithm uses information gain and Gini index measure for attribute selection

Attribute Selection Measures

The notation used herein is as follows. Let D, the data partition, be a training set of class-labeled tuples. Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for $i = 1, \ldots, m$). Let $C_{i,D}$ be the set of tuples of class C_i in D. Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$m = 2$

Information Gain

- Information gain evaluates the value or "information content" of attributes in a dataset. It assesses how much a particular attribute contributes to the decision-making process in classifying tuples.
- The attribute with the highest information gain is chosen as the splitting attribute for node.
- Information gain quantifies how much the attribute helps in reducing uncertainty or randomness in the resulting partitions.
- The attribute selected based on information gain minimizes the information needed to classify the tuples in the resulting partitions. This means that it helps in organizing the data in a way that reduces uncertainty and facilitates accurate classification.
- By choosing the attribute with the highest information gain, the approach aims to minimize the expected number of tests needed to classify a given tuple.
- This leads to a more efficient decision-making process, as the algorithm prioritizes attributes that provide the most useful information for classification. In summary, information gain is a measure used to evaluate attributes in decision tree algorithms, guiding the selection of the most informative attributes for partitioning data and minimizing the complexity of classification tasks.

Information Gain-Entropy Measure

- The expected information needed to classify a tuple in *D* is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Where p_i is the probability that an arbitrary tuple in *D* belongs to class C_i and is estimated by $|C_{i,D}|/|D|$.
- A log function to the base 2 is used, because the information is encoded in bits
- Info(D) (or **Entropy** of D)is just the average amount of information needed to identify the class label of a tuple in *D*

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Attribute Selection Measures

- It is quite likely that the partitions will be impure (e.g., where a partition may contain a collection of tuples from different classes rather than from a single class).
- How much more information would we still need (after the partitioning) in order to arrive at an exact classification?
- This amount is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- The term $|D_j|$ / $|D|$ acts as the weight of the j_{th} partition. $Info_A(D)$ is the expected information required to classify a tuple from *D* based on the partitioning by *A*.

Information Gain

- The smaller the expected information (still) required, the greater the purity of the partitions
- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on *A*). That is,

$$Gain(A) = Info(D) - Info_A(D)$$

- The attribute A with the highest information gain, (*Gain*(*A*)), is chosen as the splitting attribute at node N.

Gini Index

- Gini index is used to measures the impurity of D, a data partition or set of training tuples, as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- Where p_i is the probability that a tuple in Dbelongs to class C_i and is estimated by $|C_{i,D}|/|D|$.
- The sum is computed over 'm'classes.
- The Gini index considers a binary split for each attribute

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Gini Index

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition
- For example, if a binary split on Apartitions Dinto D_1 and D_2 , the Gini index of Dgiven that partitioning is-

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- For each attribute, each of the possible binary splits is considered
- For a discrete-valued attribute, the subset that gives the minimum Gini index for that attribute is selected as its splitting subset

Gini Index

- For continuous-valued attributes, each possible split-point must be considered
- The strategy is similar where the midpoint between each pair of (sorted) adjacent values is taken as a possible split-point.
- For a possible split-point of A, D_1 is the set of tuples in D satisfying A ≤ split point, and D_2 is the set of tuples in D satisfying A > split point.
- The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute A is

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute

Which attribute selection measure is the best?

- All measures have some bias.
- The time complexity of decision tree generally increases exponentially with tree height
- Hence, measures that tend to produce shallower trees (e.g., with multiway rather than binary splits, and that favour more balanced splits) may be preferred.
- However, some studies have found that shallow trees tend to have a large number of leaves and higher error rates
- Several comparative studies suggests no one attribute selection measure has been found to be significantly superior to others.

Tree Pruning

- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers
- Tree pruning use statistical measures to remove the least reliable branches
- Pruned trees tend to be smaller and less complex and, thus, easier to comprehend
- They are usually faster and better at correctly classifying independent test data than unpruned trees

How does Tree Pruning Work?

- There are two common approaches to tree pruning: **pre-pruning** and **post-pruning**.
- In the **pre-pruning** approach, a tree is "pruned" by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node).
- When constructing a tree, measures such as statistical significance, information gain, Gini index can be used to assess the goodness of a split.

How does Tree Pruning Work?

- The **post-pruning** approach removes sub_trees from a “fully grown” tree
- A subtree at a given node is pruned by removing its branches and replacing it with a leaf
- The leaf is labelled with the most frequent class among the subtree being replaced
- For example, the subtree at node “A3?” in the unpruned tree of Figure 1.2
- The most common class within this subtree is “class B”
- In the pruned version of the tree, the subtree in question is pruned by replacing it with the leaf “class B

How does Tree Pruning Work?

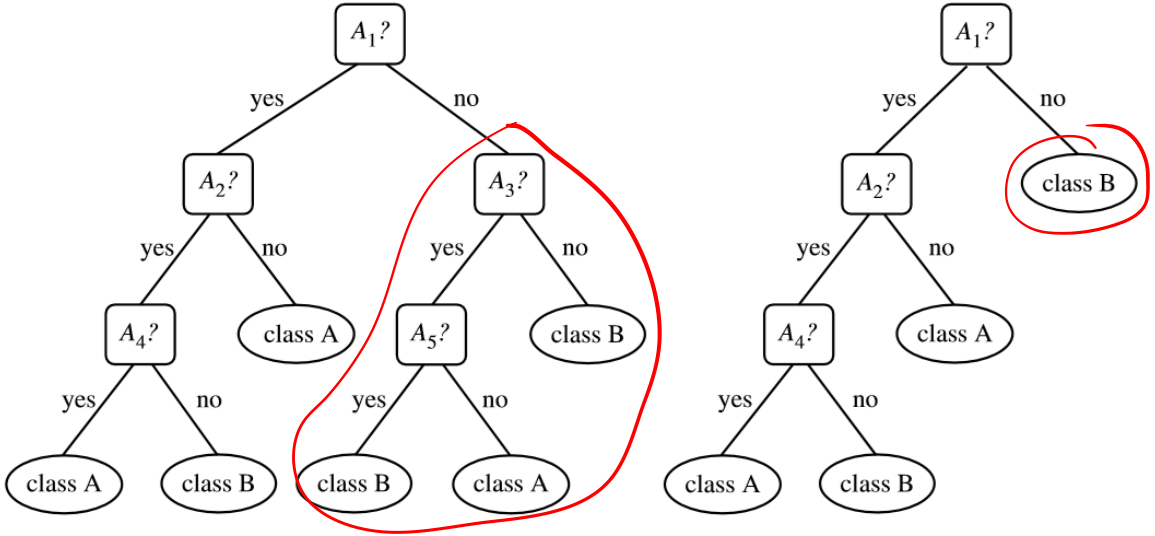


Figure: 1.2 An unpruned decision tree and a post-pruned decision tree

THANK YOU