

Module 3

Classification

Classification vs. Prediction

- **Classification**
 - A process of finding a model that describes and distinguishes the data classes.
 - predicts categorical class labels (discrete or nominal)
- Typical applications
 - Credit approval
 - Target marketing
 - Medical diagnosis
 - Fraud detection

Classification vs. Prediction

- **Classification**
 - A process of finding a model that describes and distinguishes the data classes.
 - Predicts categorical class labels (discrete or nominal)

Classification vs. Prediction

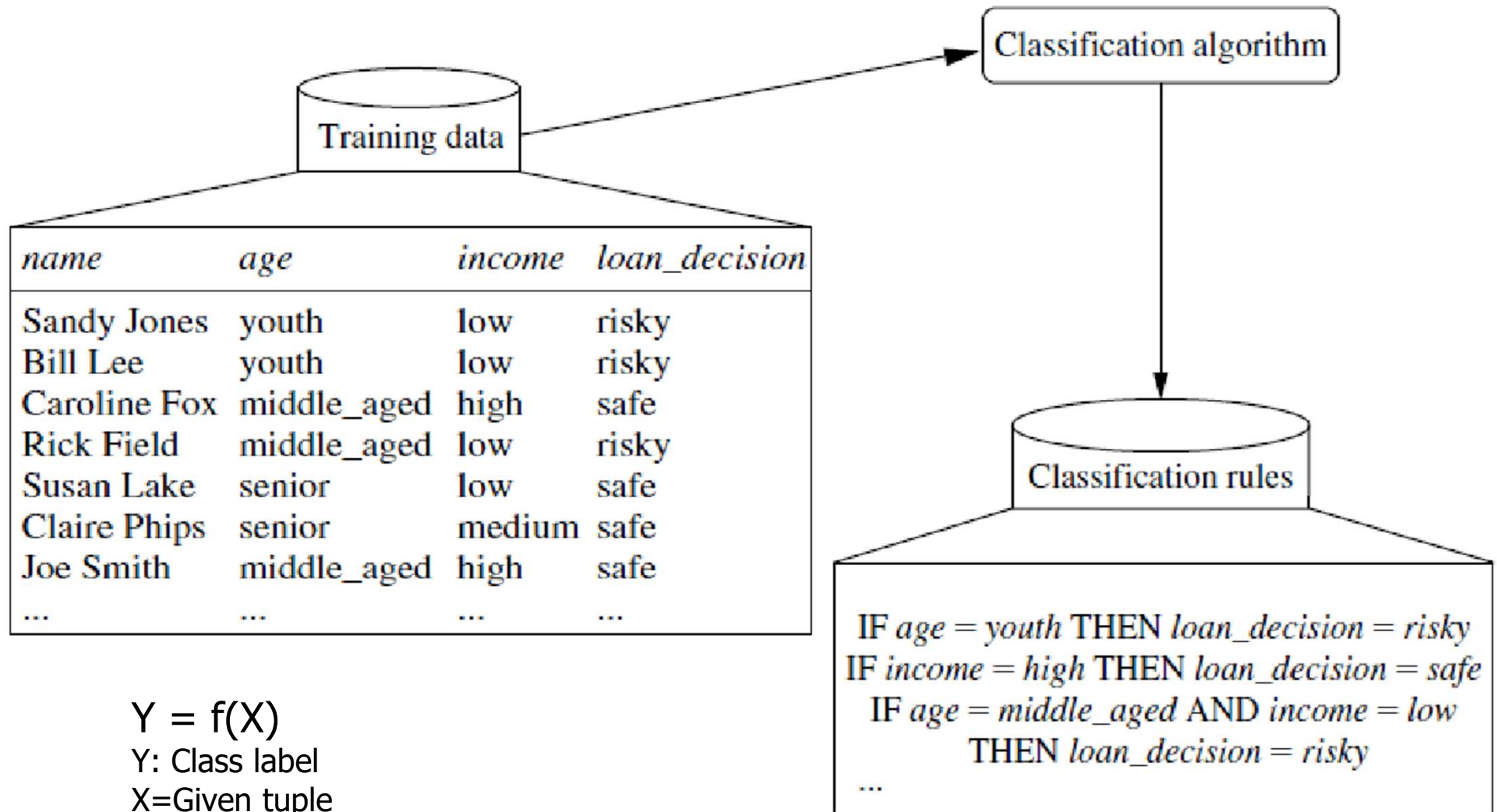
- **Prediction**
 - Models continuous-valued functions, i.e., predicts unknown or missing values
 - Regression analysis is a statistical methodology that is most often used for numeric prediction.
- Classification and numeric prediction are the two major types of prediction problems.

Classification—A Two-Step Process (Learning step+Classification step)

- In the first step, we build a classification model by learning from training dataset — Supervised Learning
- In the second step, we determine if the model's accuracy is acceptable, and if so, we use the model to classify new data.

Data classification Process

Step 1: Learning Step

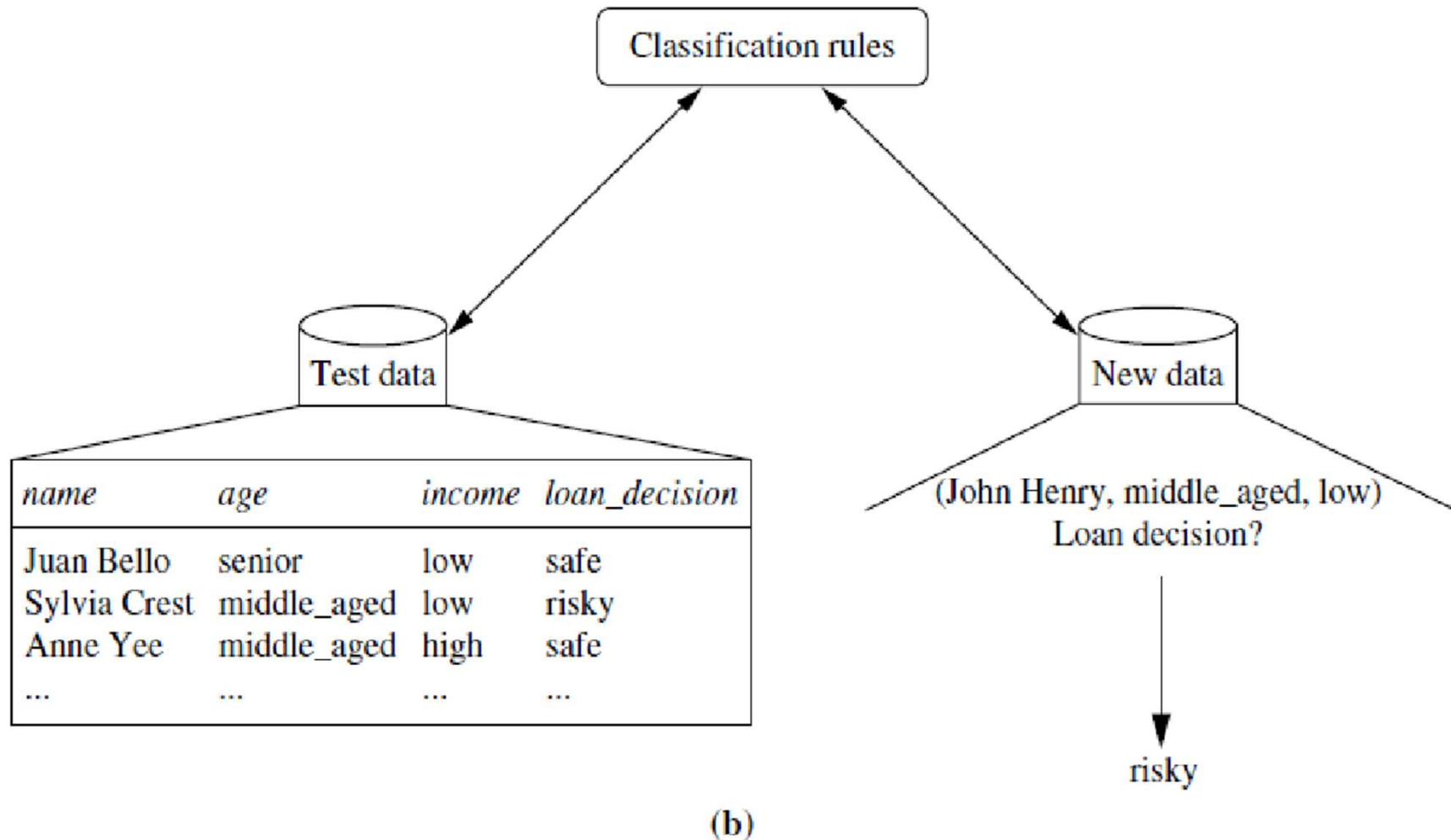


(a)

The data classification process: (a) Learning step: Training data are analyzed by a classification algorithm. Here, the class label attribute is loan decision, and the learned model or classifier is represented in the form of classification rules.

Data Classification Process

Step 2: Classification Step

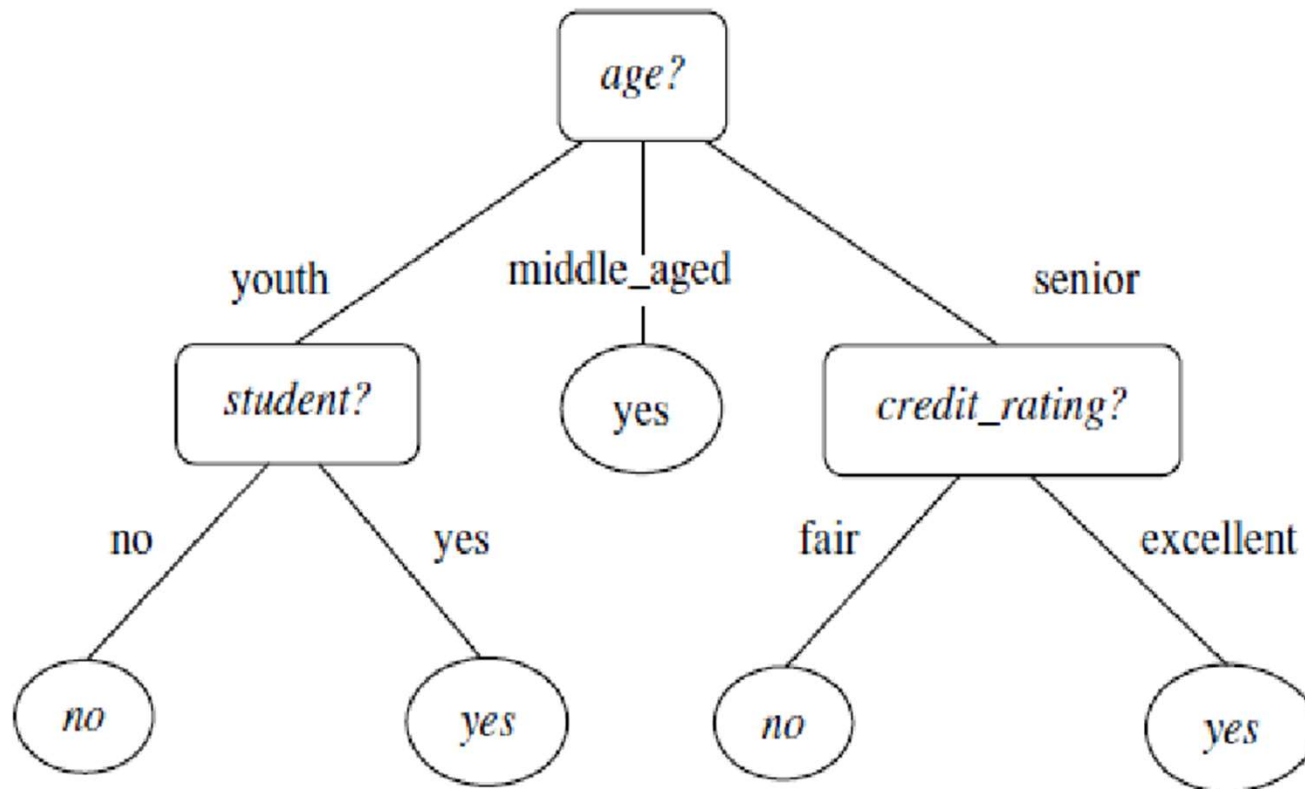


(b) Classification step : Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

- The accuracy of classifier on a given test set is % of test set tuples that are correctly classified by classifier.

Decision Tree Induction

(Learning of a decision tree from class labeled training tuples)



A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

Decision Tree Induction

- ID3 (Iterative Dichotomiser), C4.5, CART(Classification And Regression Trees) Algorithms.
- These algorithms adopt a greedy (i.e., non backtracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.
- It starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built.

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D .

Input:

- Data partition, D , which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate 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*.

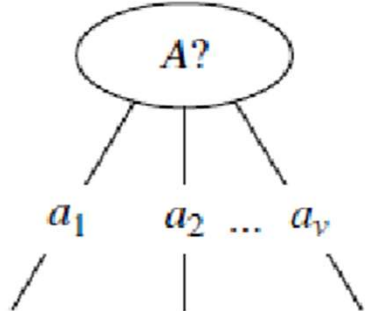
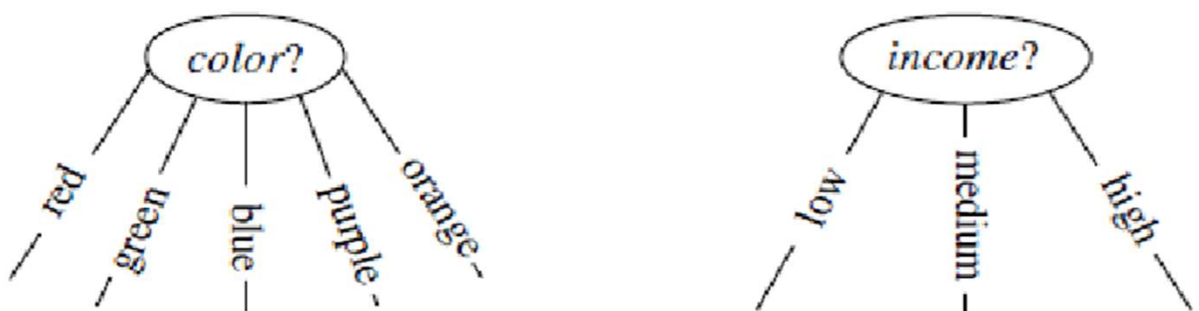
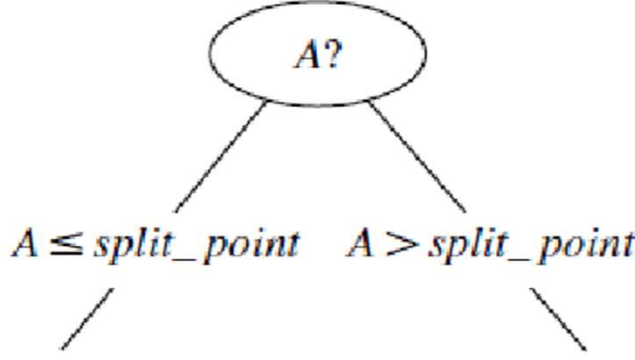
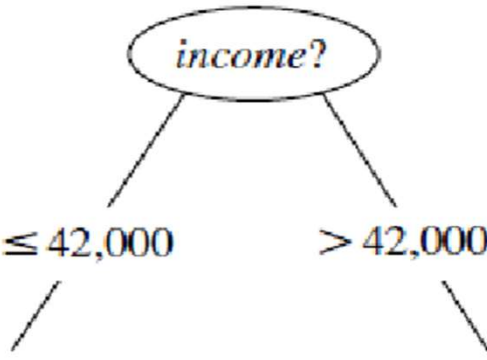
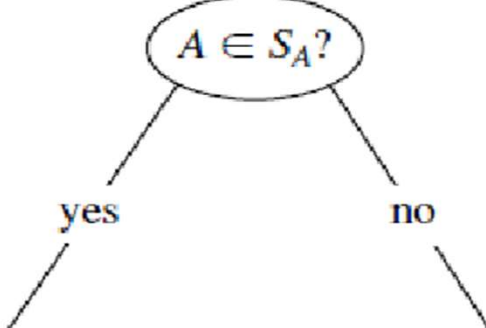
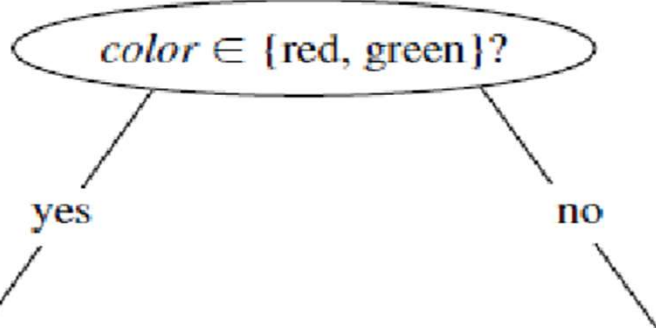
Output: A decision tree.

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* \leftarrow *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 ;

Partitioning scenarios

Examples

(a)		
(b)		
(c)		

Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A . (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq \text{split point}$ and $A > \text{split point}$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A?$, where S_A is the splitting subset for A .

Algorithm for Decision Tree Induction

1. The tree starts as a single node, N , representing the training tuples in D
2. If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class.
3. Otherwise, the algorithm calls *Attribute selection method* to determine the **splitting criterion**.
 - The splitting criterion 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.
 - The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test.
 - More specifically, 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 the tuples in it belong to the same class.

Algorithm for Decision Tree Induction

4. The node N is labeled with the splitting criterion, which serves as a test at the node. A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly.
5. The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition, D_j , of D .
6. The recursive partitioning stops only when any one of the following terminating conditions is true:
 1. All the tuples in partition D (represented at node N) belong to the same class.
 2. There are no remaining attributes on which the tuples may be further partitioned.
 - In this case, **majority voting** is employed. This involves converting node N into a leaf and labeling 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 D_j is empty.
 - In this case, a leaf is created with the majority class in D .
7. The resulting decision tree is returned.

Attribute Selection Measures

(Also known as Splitting Rules)

- An **attribute selection measure** is a heuristic 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 three popular attribute selection measures—
 1. *information gain*
 2. *gain ratio*, and
 3. *Gini index*

DOUBT

Attribute Selection Measures: Information Gain (ID3 algorithm)

- This measure is based on pioneering work by Claude Shannon on information theory, which studied the **value or "information content"** of messages.
- **The attribute with the highest information gain is chosen as the splitting attribute for node N .**
- This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or "impurity" in these partitions.
- Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.

Attribute Selection Measure: Information Gain (ID3)

- Select the attribute with the highest information gain.
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

- **Information** needed (after using A to split D into v partitions) to classify D :

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

- **Information gained** by branching on attribute A

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

**Highest
Information
gain**

Information Gain (ID3)

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 $i = 1, \dots, m$). 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.

Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by

$$p_i = |C_{i,D}| / |D|$$

Expected information (entropy) needed to classify a tuple in D :

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

Information needed (after using A to split D into v partitions) to classify D :

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

Information gained by branching on attribute A

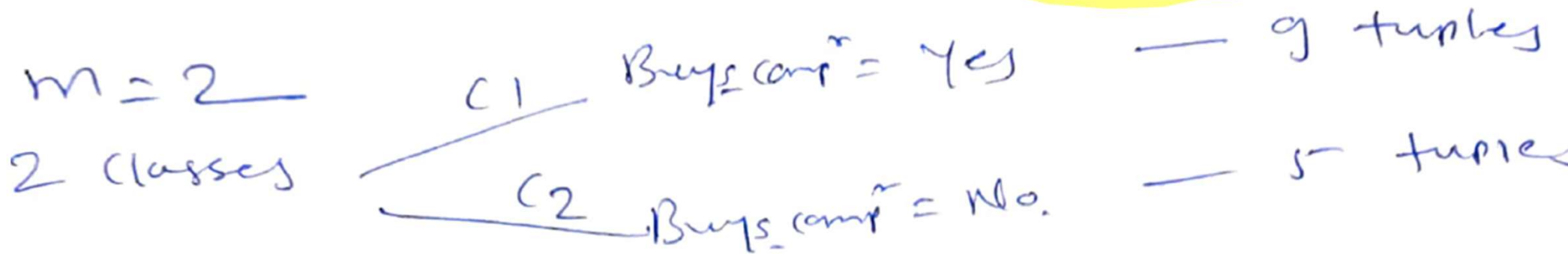
$$Gain(A) = Info(D) - Info_A(D)$$

- Select the attribute with the highest information gain.

Example: Information Gain

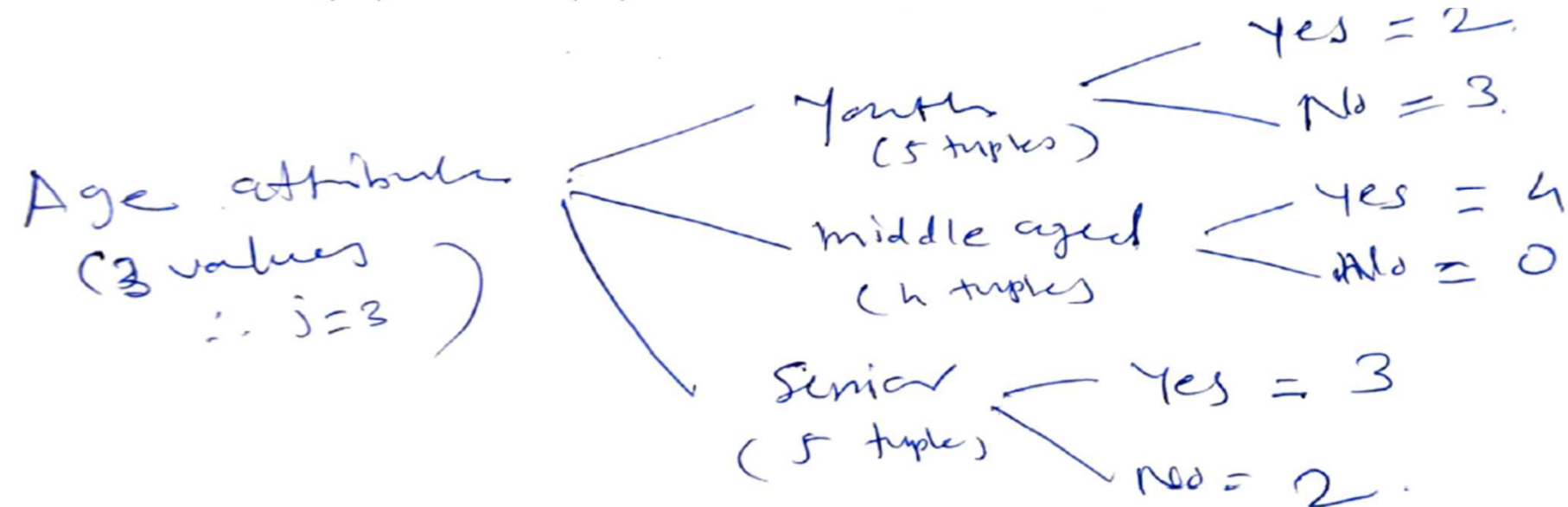
<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
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

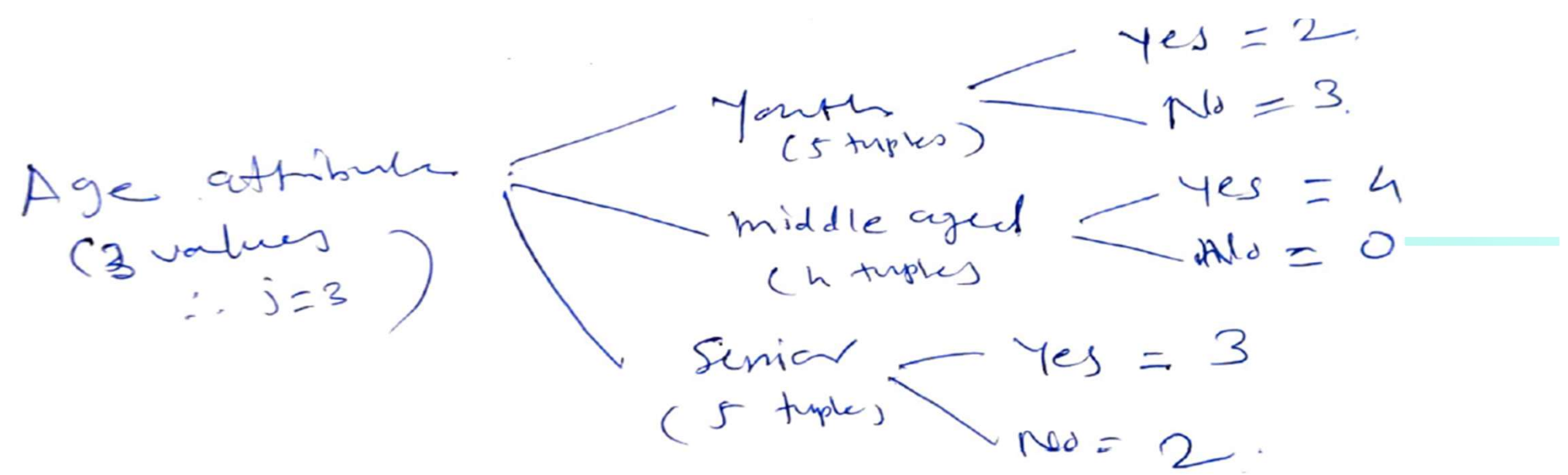
Example: Information Gain



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

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





- The expected information needed to classify a tuple in D if the tuples are partitioned according to *age* is:

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

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

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \times \left(-\frac{4}{4} \log_2 \frac{4}{4} \right) + \frac{5}{14} \times \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.694 \text{ bits.}$$

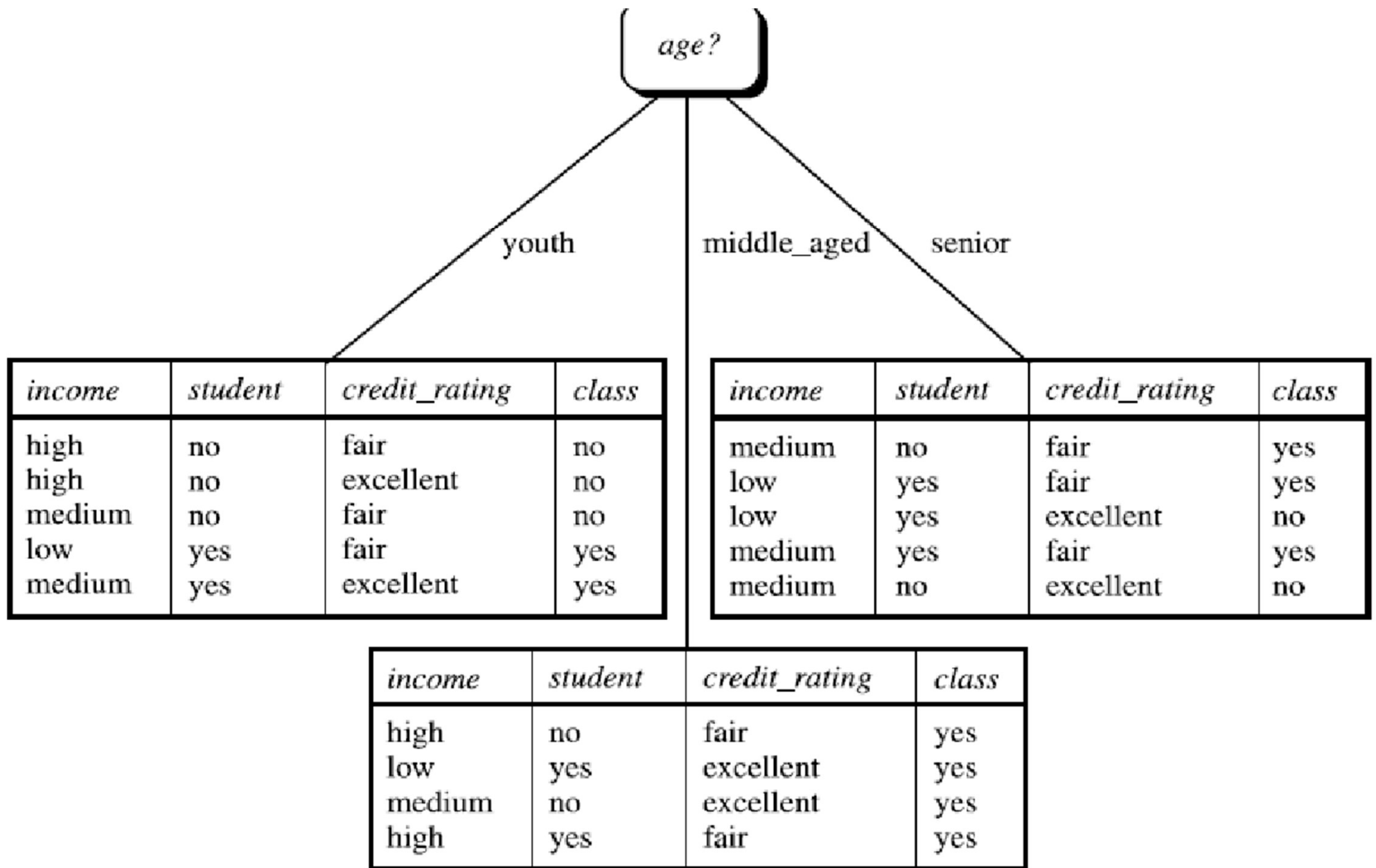
Hence, the gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

- Similarly, compute $Gain(income)$, $Gain(student)$, and $Gain(credit\ rating)$

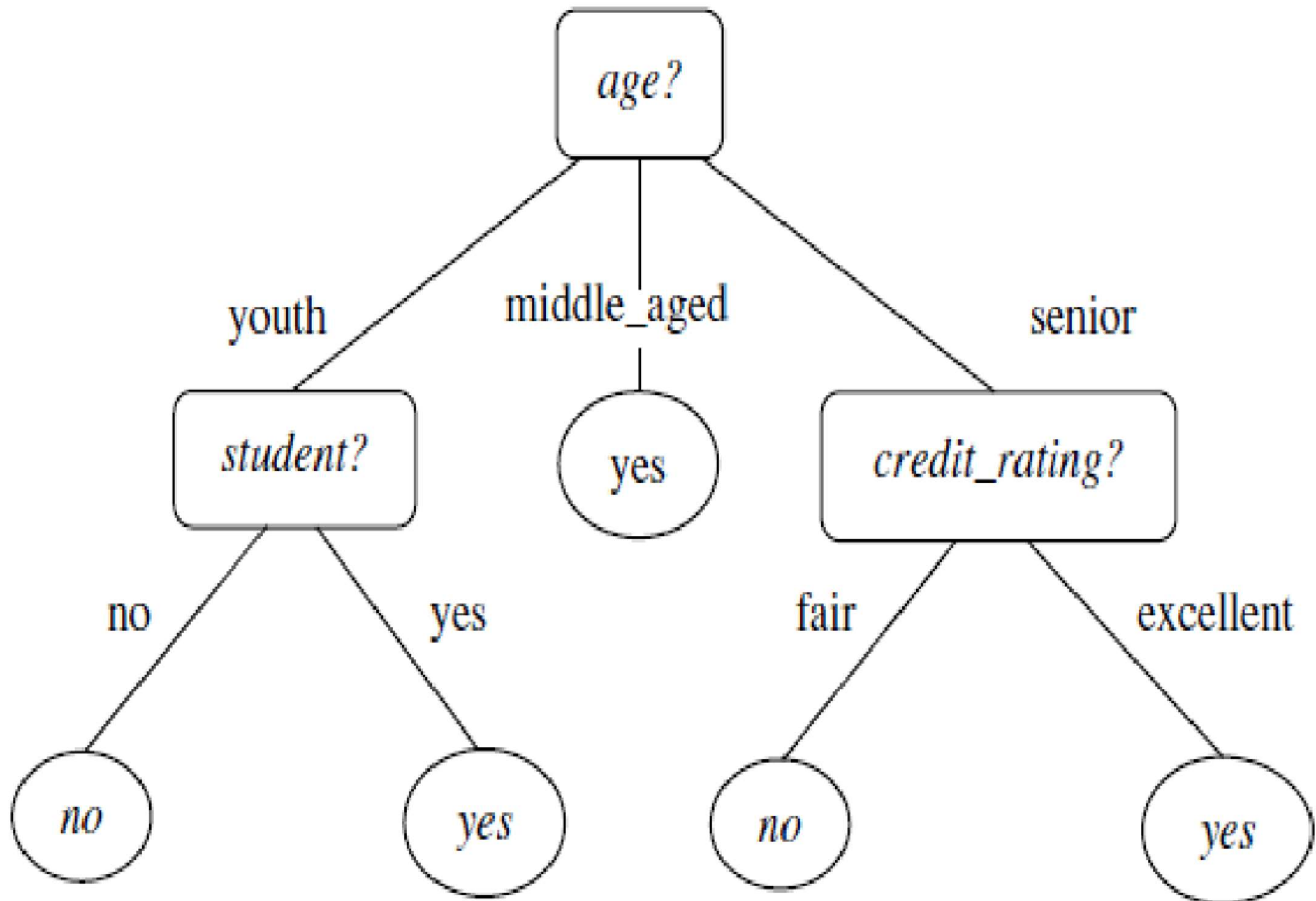
Example: Information Gain(Cntd...)

- we get $Gain(income) = 0.029$ bits, $Gain(student) = 0.151$ bits, and $Gain(credit\ rating) = 0.048$ bits.
- Because *age* has the highest information gain among the attributes, it is selected as the splitting attribute.
- Node N is labeled with *age*, and branches are grown for each of the attribute's values.



- Notice that the tuples falling into the partition for *age* = *middle_aged* all belong to the same class.
- Because they all belong to class “yes,” a leaf should therefore be created at the end of this branch and labeled “yes.”

The final decision tree returned by the algorithm



Using following training data set ,create decision tree classification model.

	Income	Age	Own House?
1	Very High	Young	Yes
2	High	Medium	Yes
3	Low	Young	Rented
4	High	Medium	Yes
5	Very High	Medium	Yes
6	Medium	Young	Yes
7	High	Old	Yes
8	Medium	Medium	Rented
9	Low	Medium	Rented
10	Low	Old	Rented
11	High	Young	Yes
12	Medium	Old	Rented

Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the ***best split point*** for A
 - Sort the values of A in increasing order.
 - Typically, the midpoint between each pair of adjacent values where there is a change in classification is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
 - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
 - D1 is the set of tuples in D satisfying $A \leq \text{split-point}$, and D2 is the set of tuples in D satisfying $A > \text{split-point}$

Example: Computing Information-Gain for Continuous-Value Attributes

Step 1: Sort the values of A in increasing order.

Temp	Play Tennis?
40	NO
48	NO
60	YES
72	YES
80	YES
90	NO

Step 2: Determine the split points by averaging consecutive values where there is a change in classification.

So split points here are :

point 1 = $(48+60)/2=54$ and

point 2 = $(80+90)/2=85$

Step 3: Evaluate the information gain of the candidate split points 54 and 85 and select the split point.

Example: Computing Information-Gain for Continuous-Value Attributes

$$\text{Entropy}(D) = \text{Info}(D) = -(3/6) (\log 3/6) - (3/6)(\log (3/6)) = 1$$

Temp	Play Tennis?
40	NO
48	NO
60	YES
72	YES
80	YES
90	NO

Consider split point=54.

Temp ≤ 54 : 2 tuples(Yes=0, No=2)

Temp > 54 : 4 tuples(Yes=3, No=1)

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

$$\text{Info}_{\text{Temp}}(D) = 0.54$$

$$\text{Info Gain}(\text{Temp}) = 1 - 0.54 = 0.46$$

Similarly for split point=85, calculate $\text{Info Gain}(\text{Temp}) = 0.1908$

As information gain for split point=54 is high, it will be selected.

Gain Ratio for Attribute Selection (C4.5)

- 'Information gain' measure is biased towards attributes with a large number of values(e.g. Unique ID attribute like Product_ID)
- C4.5 uses an extension to information gain known as **gain ratio**, which attempts to overcome this bias.
- It applies a kind of **normalization to information gain** using a “**split information**” value defined analogously with $Info(D)$ as

$$SplitInfo_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- This value represents the potential information generated by splitting the training data set, D , into v partitions, corresponding to the v outcomes of a test on attribute A .
- The gain ratio is defined as
 - **GainRatio(A) = Gain(A)/SplitInfo_A(D)**
- The attribute with the maximum gain ratio is selected as the splitting attribute

Example: Gain Ratio

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
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

Calculate Gain Ratio(Income)

Example: Gain Ratio for income attribute

- From previous example, $\text{Gain}(\text{income}) = 0.029$.
- A test on **income** attribute splits the data of Table into three partitions, namely *low*, *medium*, and *high*, containing four, six, and four tuples, respectively.

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$\text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right)$$

$$= 1.557.$$

- $\text{GainRatio}(\text{income}) = \text{Gain}(\text{income}) / \text{SplitInfo}_{\text{income}}(D)$
- Therefore, $\text{GainRatio}(\text{income}) = 0.029 / 1.557 = 0.019$.
- The attribute with the **maximum gain ratio** is selected as the splitting attribute
- Drawback: tends to prefer unbalanced splits in which one partition is much smaller than the others.

Gini index (CART)

- Gini index 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 D belongs 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.
- **Case 1: A is discrete-valued:** To determine the best binary split on discrete valued attribute A , we examine all the possible subsets of the known values of A .
- Each subset, S_A , can be considered as a binary test for attribute A of the form “ $A \in S_A$?”
- If A has v possible values, then there are 2^v possible subsets.
- Out of these, there are $2^v - 2$ possible ways to form two partitions of the data, D , based on a binary split on A .

Gini index (CART)

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D_1 and D_2 , the Gini index of D given 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.
- 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. This attribute and its splitting subset (for a discrete-valued splitting attribute) together form the splitting criterion.

Gini index (Continuous-valued attribute)

- For continuous-valued attributes, each possible split-point must be considered.
- The strategy is to take the midpoint between each pair of (sorted) adjacent values as a possible split-point.
- The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.
- Recall that for a possible split-point of A , D_1 is the set of tuples in D satisfying $A \leq \text{split point}$, and D_2 is the set of tuples in D satisfying $A > \text{split point}$
- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute. This attribute and its split-point (for a continuous-valued splitting attribute) together form the splitting criterion.

Gini index (Continuous-valued attribute)

- For continuous-valued attributes, each possible split-point must be considered.
- The strategy is to take the midpoint between each pair of (sorted) adjacent values as a possible split-point.
- The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.
- Recall that for a possible split-point of A , D_1 is the set of tuples in D satisfying $A \leq \text{split point}$, and D_2 is the set of tuples in D satisfying $A > \text{split point}$
- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute. This attribute and its split-point (for a continuous-valued splitting attribute) together form the splitting criterion.

Example: Gini Index

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
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

Example: Gini index

- Ex: Let D be the training data where there are nine tuples belonging to the class *buys computer = yes* and the remaining five tuples belong to the class *buys computer = no*.
- A (root) node N is created for the tuples in D . We first compute the impurity of D as:

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$
- To find the splitting criterion for the tuples in D , we need to compute the Gini index for each attribute.
- Let's start with the attribute *income* and Consider the subset $\{low, medium\}$. This would result in 10 tuples in partition D_1 satisfying the condition "*income* $\in \{low, medium\}$." The remaining four tuples of D would be assigned to partition D_2 .
- The Gini index value computed based on this partitioning is

$$Gini_{income \in \{low, medium\}}(D)$$

$$= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$

$$= \frac{10}{14} \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 \right) + \frac{4}{14} \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 \right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$$Gini_{income \in \{low, high\}}(D) 0.458$$

$$Gini_{income \in \{medium, high\}}(D) 0.450$$

- Therefore, the best binary split for attribute *income* is on $\{low, medium\}$ or $\{high\}$ because it minimizes the Gini index.

Example: Gini index (Cntd...)

- Evaluating *age*, we obtain $\text{Gini}_{\text{age} \in \{\text{youth}, \text{senior}\}}^{(D)} = 0.357 = \text{Gini}_{\text{age} \in \{\text{middle aged}\}}^{(D)}$
- The attributes *student* and *credit rating* are both binary, with Gini index values of 0.367 and 0.429, respectively.
- The attribute *age* and splitting subset $\{\text{youth}, \text{senior}\}$ therefore give the minimum Gini index overall, with a reduction in impurity of $0.459 - 0.357 = 0.102$. The binary split “*age* $\in \{\text{youth}, \text{senior}\}?$ ” results in the maximum reduction in impurity of the tuples in *D* and is returned as the splitting criterion.
- Node *N* is labeled with the criterion, two branches are grown from it, and the tuples are partitioned accordingly.

Comparing Attribute Selection Measures

- Information gain:
 - biased towards multivalued attributes
- Gain ratio:
 - tends to prefer unbalanced splits in which one partition is much smaller than the others
- Gini index:
 - biased to multivalued attributes
 - has difficulty when # of classes is large.
 - tends to favor tests that result in equal-sized partitions and purity in both partitions
- Although biased, these measures give reasonably good results in practice.

Other Attribute selection measures

1. CHAID, a decision tree algorithm that is popular in marketing, uses an attribute selection measure that is based on the statistical χ^2 test for independence.
2. C-SEP (which performs better than information gain and the Gini index in certain cases) and G-statistic (an information theoretic measure that is a close approximation to 2 distribution).
3. Attribute selection measures based on the Minimum Description Length (MDL) principle have the least bias toward multivalued attributes.
4. Other attribute selection measures consider multivariate splits (i.e., where the partitioning of tuples is based on a combination of attributes, rather than on a single attribute)

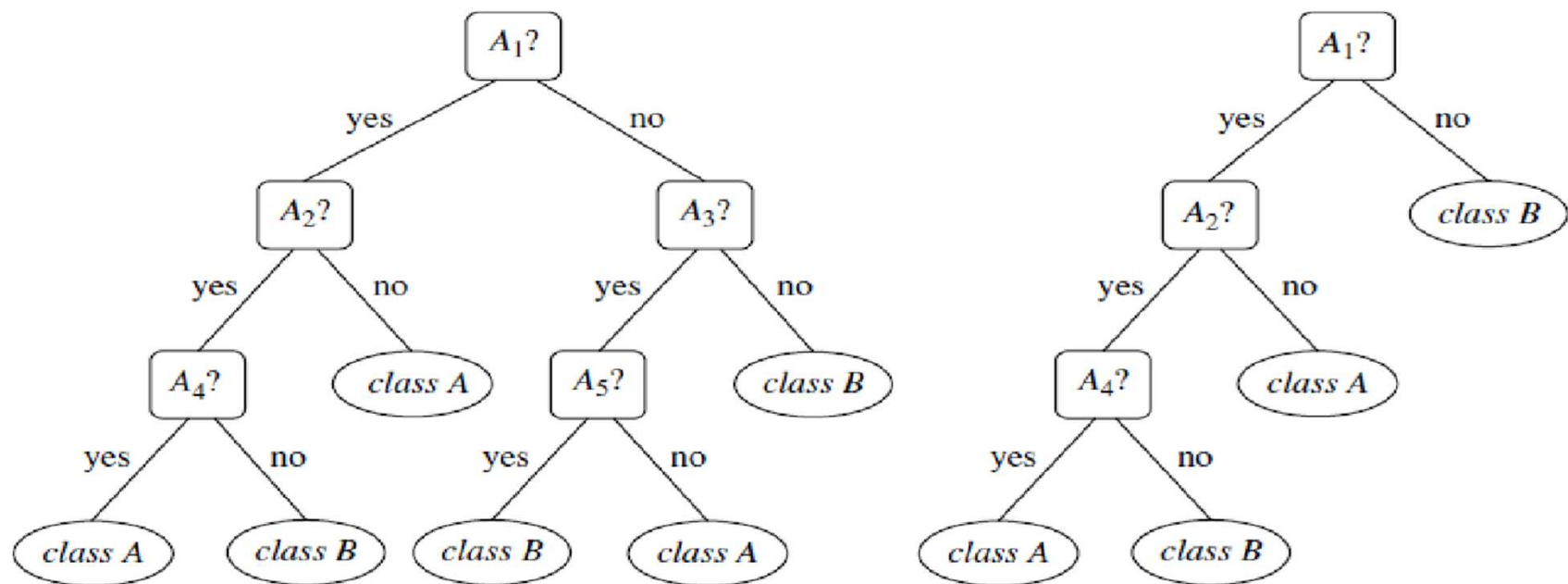
“Which attribute selection measure is the best?”

- It has been shown that the time complexity of decision tree induction generally increases exponentially with tree height. Hence, measures that tend to produce shallower trees (e.g., with multiway rather than binary splits, and that favor 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.
- Despite several comparative studies, no one attribute selection measure has been found to be significantly superior to others. Most measures give quite good results.

Overfitting and Tree Pruning

Overfitting:

- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers.
- Tree pruning methods address this problem of *overfitting* the data.
- Such methods typically use statistical measures to remove the least-reliable branches.



An unpruned decision tree and a pruned version of it.

Overfitting and Tree Pruning

■ Two approaches to avoid overfitting : **Prepruning and Postpruning**

■ **Prepruning:**

- Halt tree construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node).
- Upon halting, the node becomes a leaf.
- The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.
 - Difficult to choose an appropriate threshold

■ **Postpruning:**

- Remove branches 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 labeled with the most frequent class among the subtree being replaced.

Q : Using the following training data set. Create classification model using decision -tree.

	Income	Age	Own House
1.	Very High	Young	Yes
2.	High	Medium	Yes
3.	Low	Young	Rented
4.	High	Medium	Yes
5.	Very high	Medium	Yes
6.	Medium	Young	Yes
7.	High	Old	Yes
8.	Medium	Medium	Rented
9.	Low	Medium	Rented
10.	Low	Old	Rented
11.	High	Young	Yes
12.	medium	Old	Rented