Data Mining

UNIT-IV

B.Tech(CSE)-VI SEM

UNIT : IV -Classification

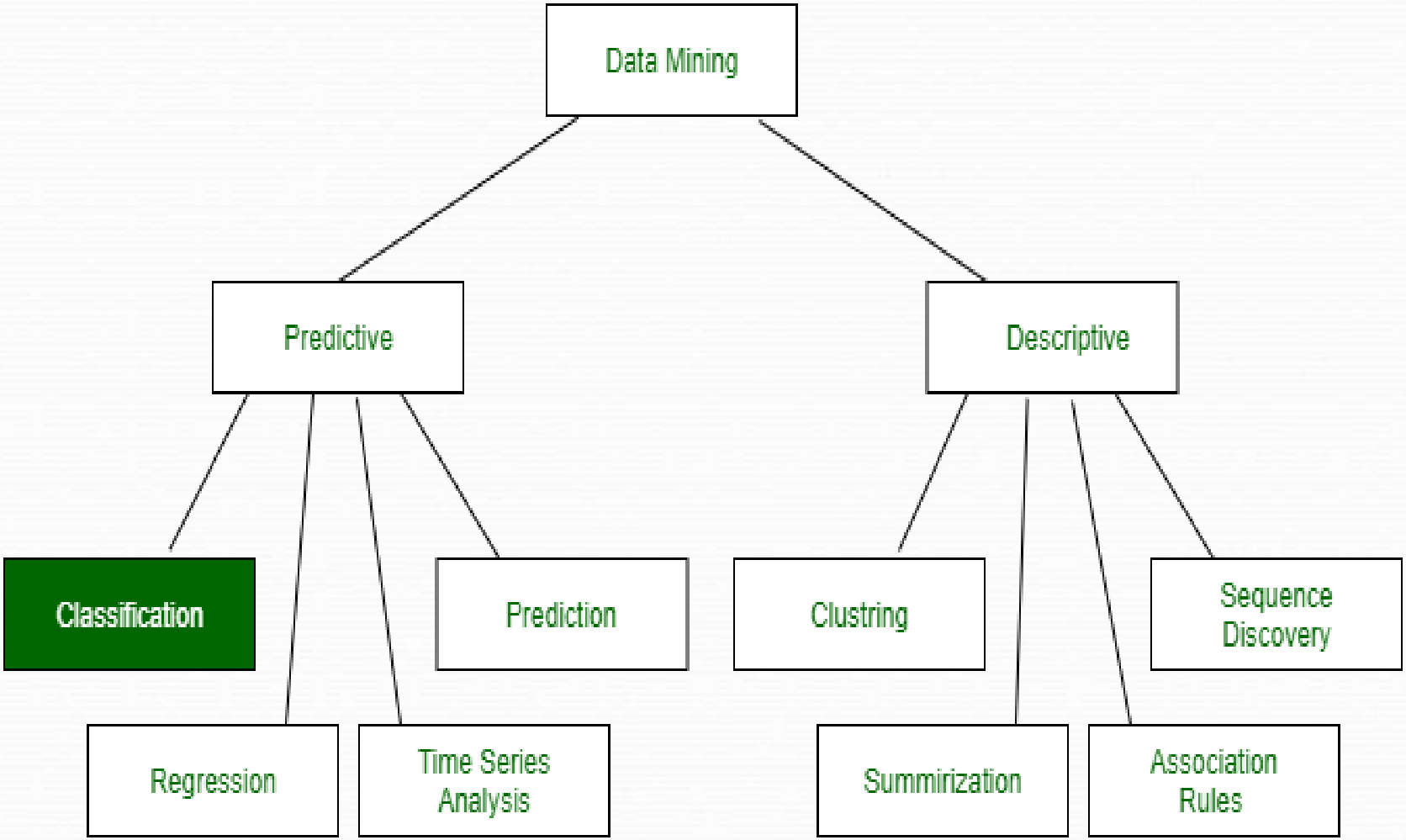
- Basic Concepts
- Decision Tree Induction
- Attribute Selection Measures
- Tree Pruning

Basic Concepts

Definition of Classification

- **Classification is a form** of data analysis that extracts models describing important data classes.
- These models are called classifiers and are used for predict(discrete, unordered) class labels.
- Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statistics. Most algorithms are memory resident, typically assuming a small data size.
- Classification has numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis.

Basic Concepts



Basic Concepts

General Idea behind classification

- The data for a classification task consists of a collection of instances (records).
- Each such instance is characterized by the tuple (x, y), where x is the set of attribute values that describe the instance and y is the class label of the instance.
- The attribute set x can contain attributes of any type, while the class label y must be categorical.
- A classification model is an abstract representation of the relationship between the attribute set and the class label.

Basic Concepts

Task	Attribute set	Class label
Spam filtering	Features extracted from email message header and content	spam or non-spam
Tumor identification	Features extracted from magnetic resonance imaging (MRI) scans	malignant or benign
Galaxy classification	Galaxy classification Features extracted from telescope images	elliptical, spiral, or irregular-shaped

Basic Concepts

Examples of Classification Tasks

In the below examples, the data analysis task is classification , where a model or a classifier is constructed to predict a class(categorical)

- A bank loans officer needs analysis of her data to learn which loan applicants are “safe” and which are “risky” for the bank. Here the class label should be safe or risky

- A marketing manager at *an Electronics company* needs data analysis to help guess whether a customer with a given profile will buy a new computer. Here the class label should be “yes” or “No”.

- A medical researcher wants to analyze breast cancer data to predict which one of three specific treatments a patient should receive. Here the class labels should be “treatment A” , “treatment B” and “treatment C”

Basic Concepts

Classification and numeric prediction are the two major types of **prediction problems**

Classification VS prediction

- **Classification**:
- Predicts categorical class labels (discrete or nominal)
- Classifies data based on training set and values (class labels) in a classifying attribute and uses it in classifying new data.
- **Prediction:**
- Models continuous valued functions i.e., predicts unknown or missing values
- **Applications**:
- Credit approval, Target Marketing, Medical diagnosis etc.

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Basic Concepts

Vertebrate Classification

Vertebrate Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-Nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	No	yes	no	No	No	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	Semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	Yes	No	Yes	yes	Yes	mammal
pigeon	warm-blooded	feather	no	no	yes	yes	no	bird
cat	warm-blooded	s fur	yes	no	no	Yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	Semi	no	Yes	no	reptile
penguin	warm-blooded	feather	no	semi	no	yes	no	bird
porcupine	warm-blooded	s quills	yes	no	No	yes	Yes	mammal
eel	cold-blooded	scales	No	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Basic Concepts

Example:

The above table shows a sample data set for classifying vertebrates into mammals, reptiles, birds, fishes, and amphibians.

The attribute set includes characteristics of the vertebrate such as its body temperature, skin cover, and ability to fly.

The data set can also be used for a binary classification task such as mammal classification, by grouping the reptiles, birds, fishes, and amphibians into a single category called non- mammals.

Vertebrate Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-Nates	Class Label
gila monster	cold-blooded	Scales	No	NO	No	Yes	Yes	?

Basic Concepts

General Framework for Classification

- Classification is the task of assigning to unlabeled data instances and a classifier is used to perform such a task.
- The model is created using a given a set of instances, known as the training set, which contains attribute values as well as class labels for each instance.
- The systematic approach for learning a classification model given a training set is known as a learning algorithm.
- The process of using a learning algorithm to build a classification model from the training data is known as induction. This process is also often described as "***learning a model" or "building a model.***"

Basic Concepts

General Framework for Classification

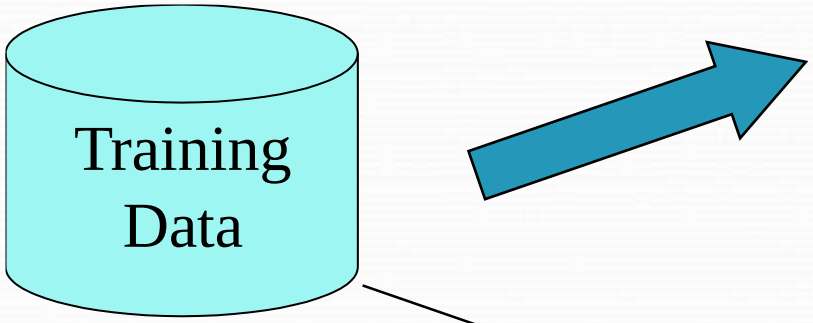
- This process of applying a classification model on unseen test instances to predict their class labels is known as deduction.

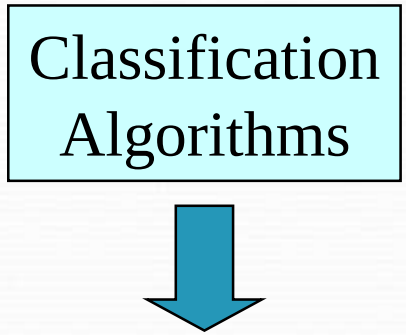
- Therefore, the process of classification involves two steps:
  - applying a learning algorithm to training data to learn a model
  - then applying the model to assign labels to unlabeled instances.

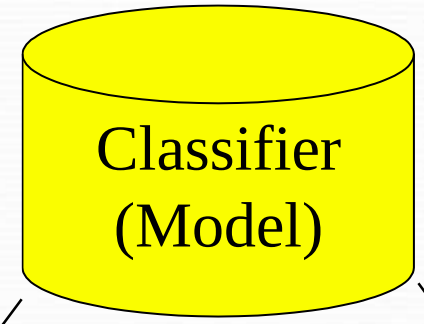
Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction



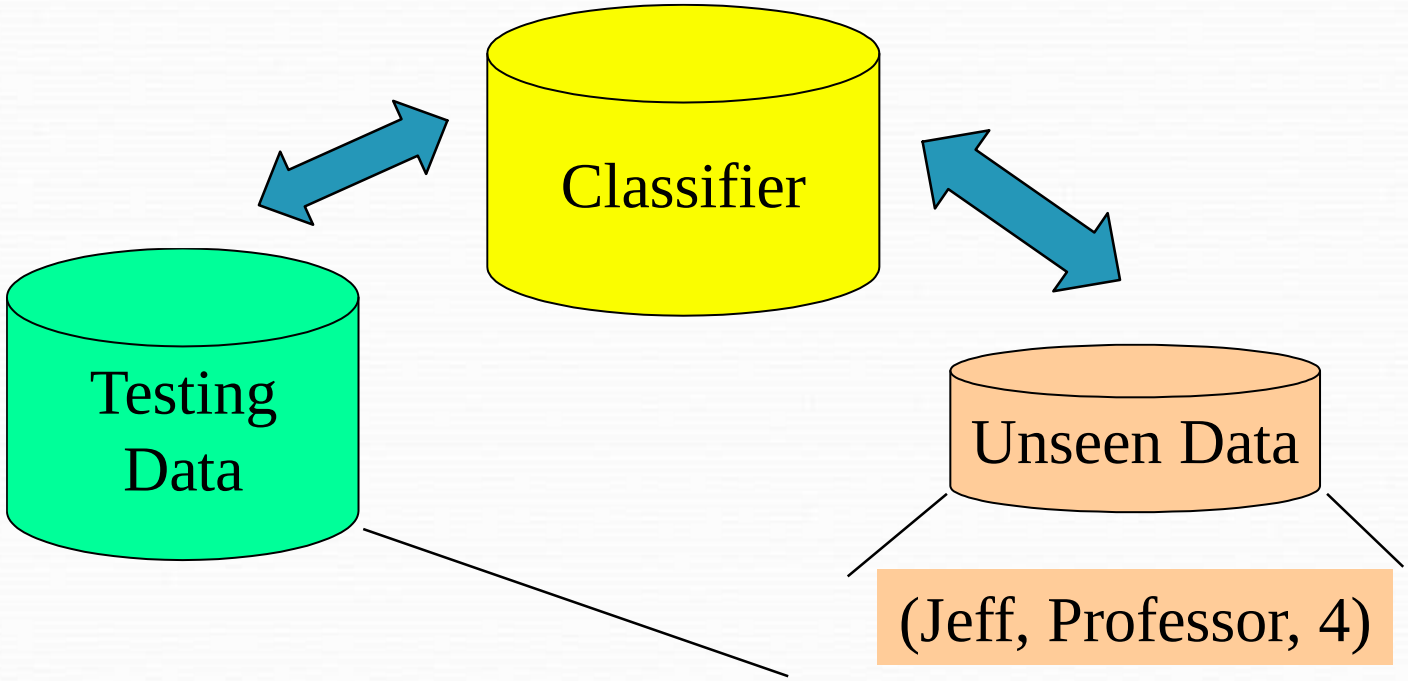




NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

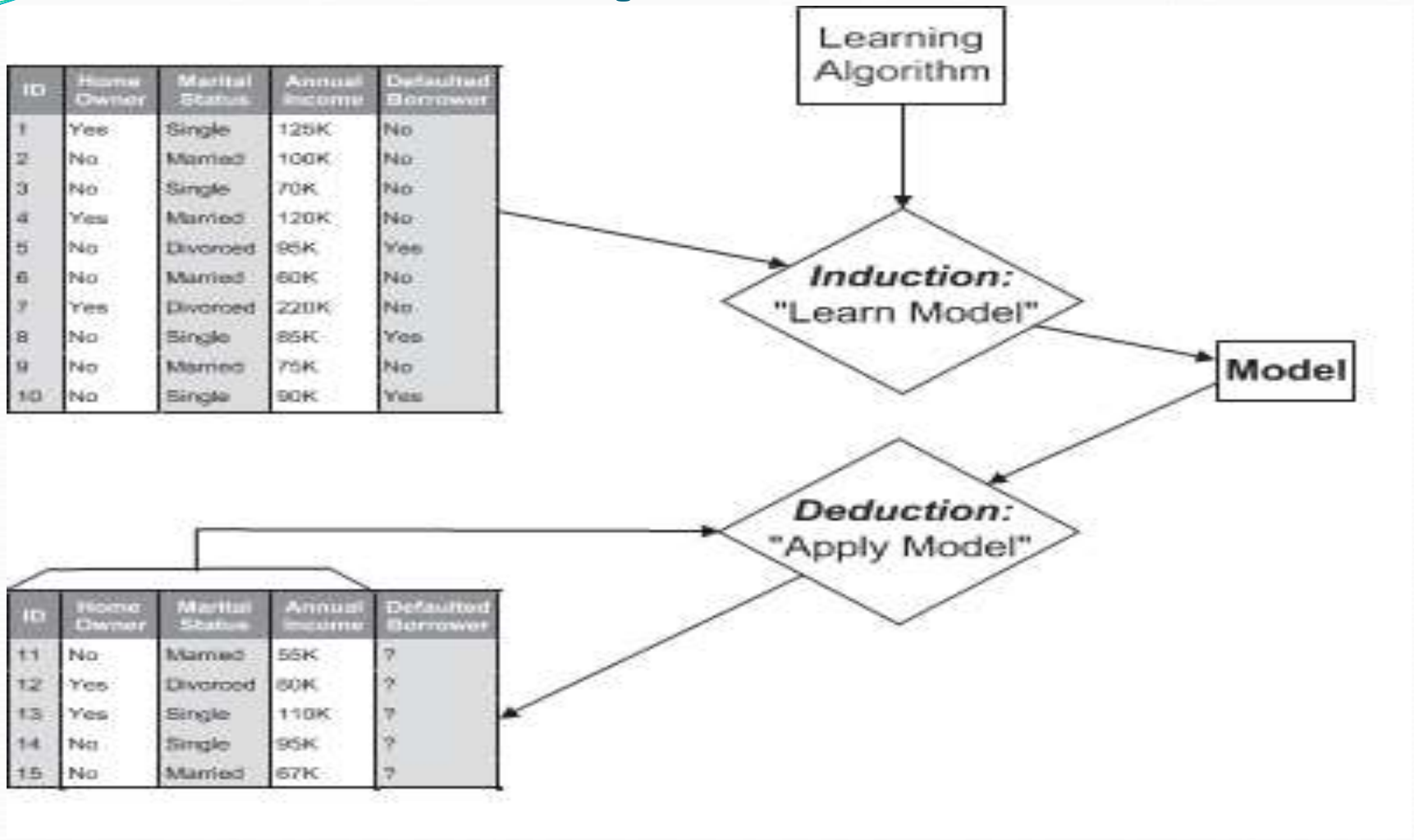
IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Basic Concepts



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
11	No	Married	55K	?
12	Yes	Divorced	80K	?
13	Yes	Single	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

General Framework for Classification

Basic Concepts

Performance of the Model

- The training and test sets should be independent of each other to ensure that the induced model can accurately predict the class labels of instances it has never encountered before.
- Models that deliver such predictive insights are said to have good **generalization performance**.
- The performance of a model (classifier) can be evaluated by comparing the predicted labels against the true labels of instances. This information can be summarized in a table called a **confusion matrix**.

Basic Concepts

Performance of the model

		Predicted Class	
		Class =1	Class = 0
Actual class	Class =1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

- Table depicts the confusion matrix for a binary classification problem. Each entry f_{ij} denotes the number of instances from class i predicted to be of class j.
- f_{01} is the number of instances from class 0 incorrectly predicted as class 1. The number of correct predictions made by the model is $(f_{11} + f_{00})$ and the number of incorrect predictions is $(f_{10} + f_{01})$

Basic Concepts

Performance of the Model

- Although a confusion matrix provides the information needed to determine how well a classification model performs, summarizing this information into a single number makes it more convenient to compare the relative performance of different models.

- This can be done using an **evaluation metric** such as **accuracy and error rate.**

Basic Concepts

Performance of the Model

- Accuracy = (No of correct predictions / Total No of predictions)

- For Binary classification problem , the accuracy is given by

  Accuracy = $((f_{11} + f_{00})/ (f_{11} + f_{10} + f_{01}+ f_{00}))$

- **Error rate** is another related metric which is defined as for binary classification

  Error Rate = $((f_{10} + f_{01})/ (f_{11} + f_{10} + f_{01}+ f_{00}))$

Basic Concepts

Summary

classification of a class with some predefined group or class is known as Data Discrimination

Classification may be defined as the data objects that do not comply with the general behavior or model of the data available.

Data Discrimination can be considered as the classification or mapping of a set or class with some predefined group or classes

Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure, where each **internal node** (nonleaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or *terminal node*) holds a class label.
- The tree has three types of nodes:
  - A root node, with no incoming links and zero or more outgoing links.
  - Internal nodes, each of which has exactly one incoming link and two or more outgoing links.
  - Leaf or terminal nodes, each of which has exactly one incoming link and no outgoing links.

Decision Tree Induction

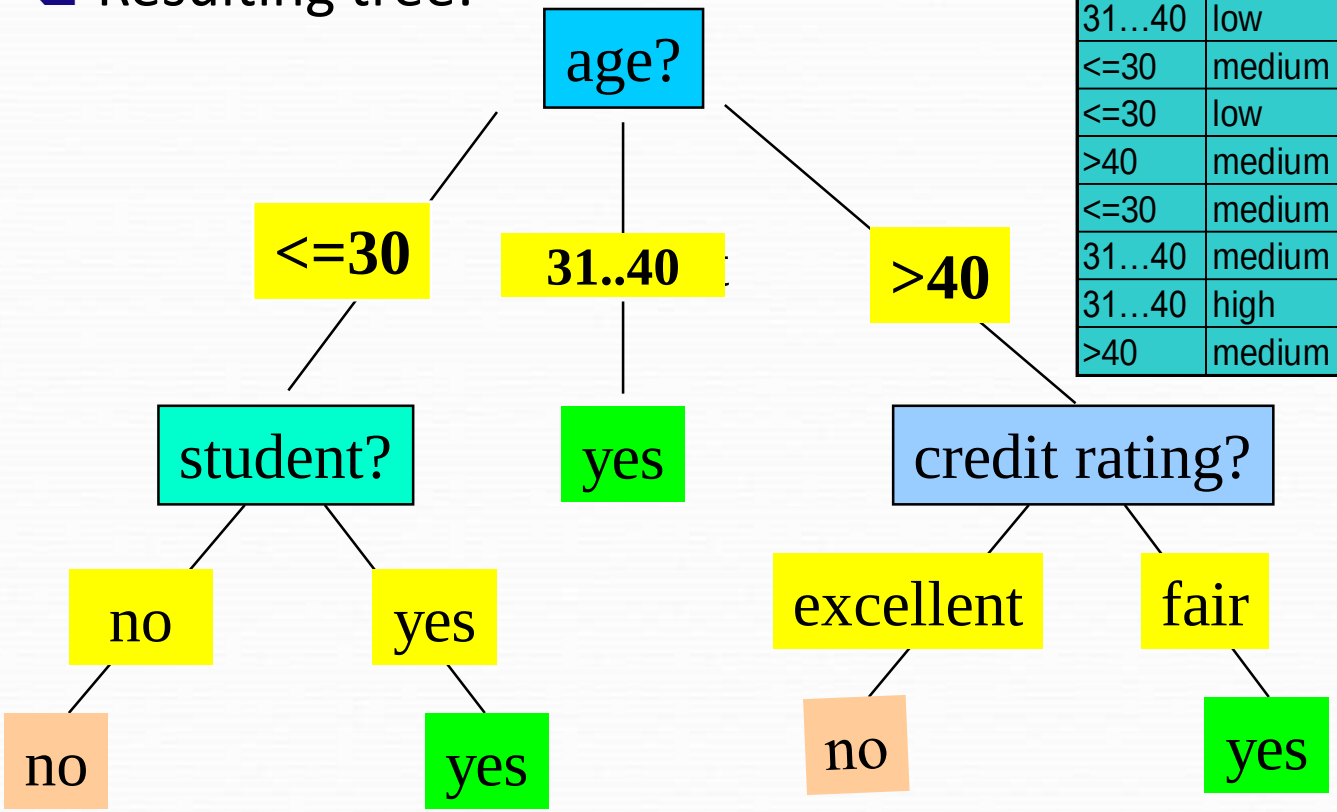
- Every leaf node in the decision tree is associated with a class label. The **non-terminal** nodes, which include the root and internal nodes, contain **attribute test conditions** that are typically defined using a single attribute.
- Each possible outcome of the attribute test condition is associated with exactly one child of this node.

Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals. Some decision tree algorithms produce only *binary* trees (where each internal node branches to exactly two other nodes), whereas others can produce non-binary trees.

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Decision Tree Induction Algorithm

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*.

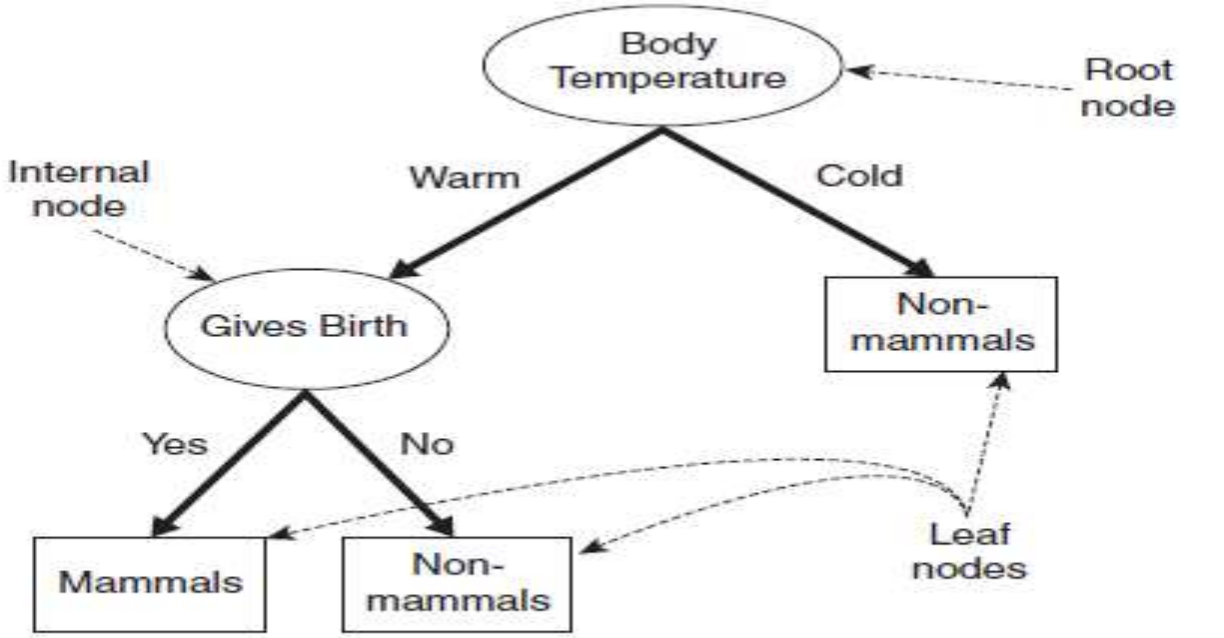
Output: A decision tree.

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 ← 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 Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj 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(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

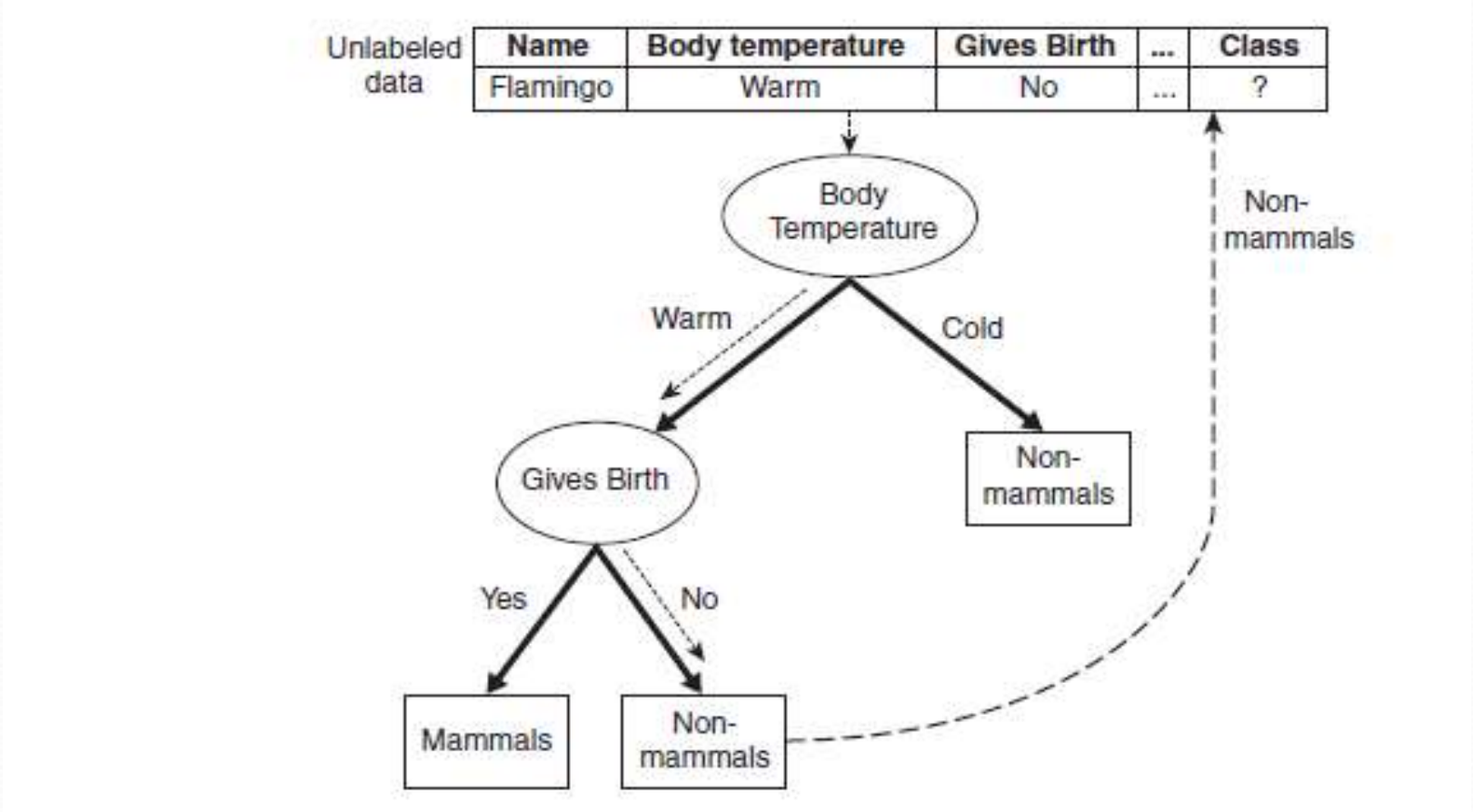
Decision Tree Induction

- Example 2:
- Suppose a new species is discovered by scientists. How can we tell whether it is a mammal or a non-mammal?
- The first question we may ask is whether the species is cold- or warm-blooded.



A decision tree for the mammal classification problem.

Decision Tree Induction



Classifying unlabeled vertebrate

Decision Tree Induction

- **Strategy for Decision Tree Induction:**
- The algorithm is called with three parameters: *D*, *attribute list*, and *Attribute selection method*.
- Treat *D* 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 describing the tuples.
- *Attribute selection method* specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class.
- This procedure employs an attribute selection measure such as information gain or the Gini index.

Decision Tree Induction

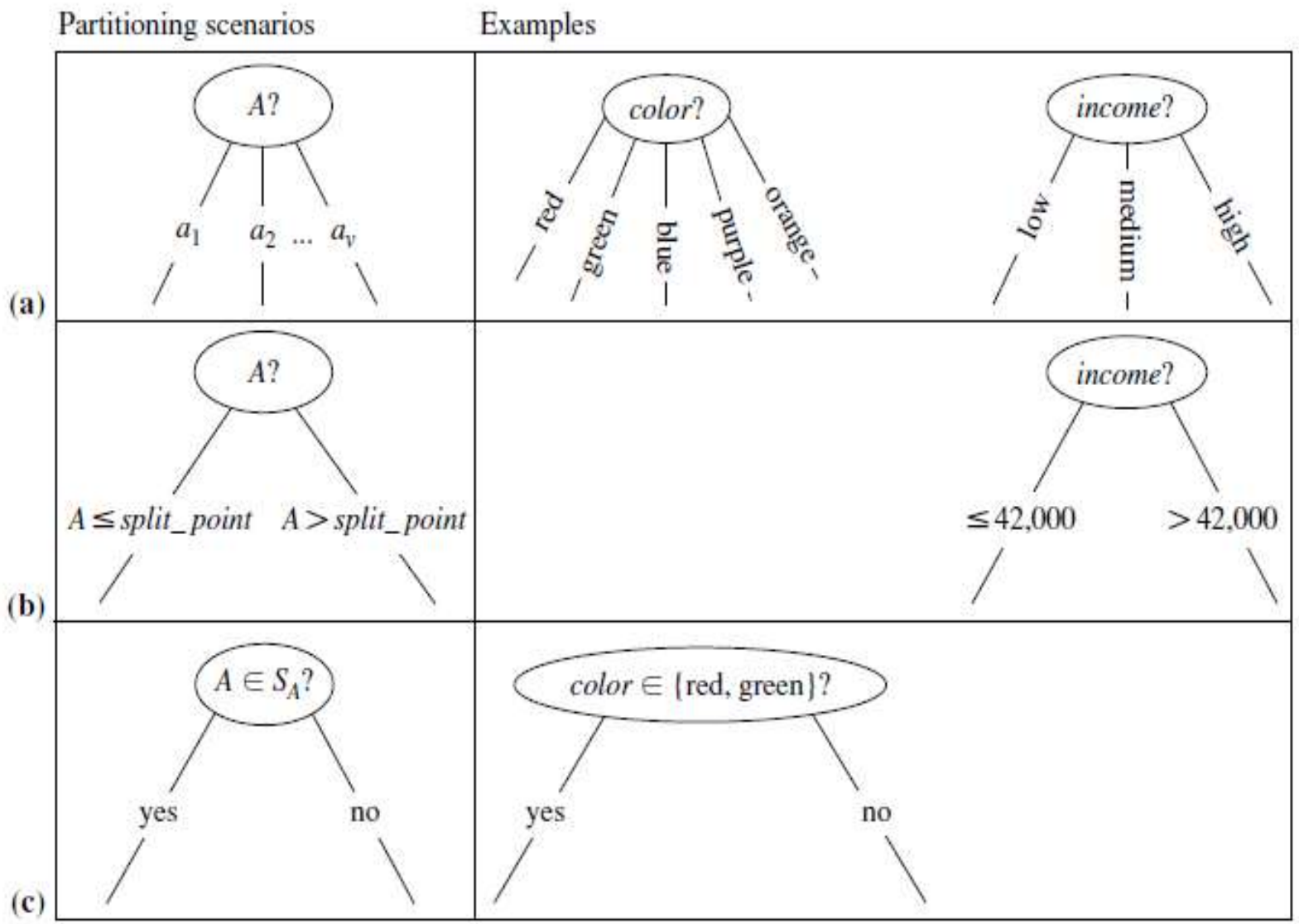
- Whether the tree is strictly binary is generally driven by the attribute selection measure.
- Some attribute selection measures, such as the Gini index, enforce the resulting tree to be binary.
- Some other attribute selection measures will also be there like information gain etc.

- **Procedure:**
- The tree starts as a single node, *N*, representing the training tuples in *D*
- If the tuples in *D* are all of the same class, then node *N* becomes a leaf and is labeled with that class

Decision Tree Induction

- 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**.
- A partition is **pure** if all the tuples in it belong to the same class.

Decision Tree Induction



Decision Tree Induction

- (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* <= *split point* and *A* > *split point*.
- (c) If *A* is discrete-valued and a binary tree must be produced, then the test is of the form $A \in SA$, where *SA* is the splitting subset for *A*.

- The computational complexity of the algorithm given training set *D* is $O(n \times |D|)$, where *n* is the number of attributes describing the tuples in *D* and $|D|$ is the number of training tuples in *D*.
- This means that the computational cost of growing a tree grows at most $n \times |D| \times \log(|D|)$ with $|D|$ tuples.

Decision Tree Induction

- **Decision Tree Algorithm**
- A decision tree is grown in a recursive fashion.
- The tree initially contains a single root node that is associated with all the training instances.
- . If a node is associated with instances from more than one class, it is expanded using an attribute test condition that is determined using a splitting criterion.
- A child leaf node is created for each outcome of the attribute test condition and the instances associated with the parent node are distributed to the children based on the test outcomes.
- This node expansion step can then be recursively applied to each child node, as long as it has labels of

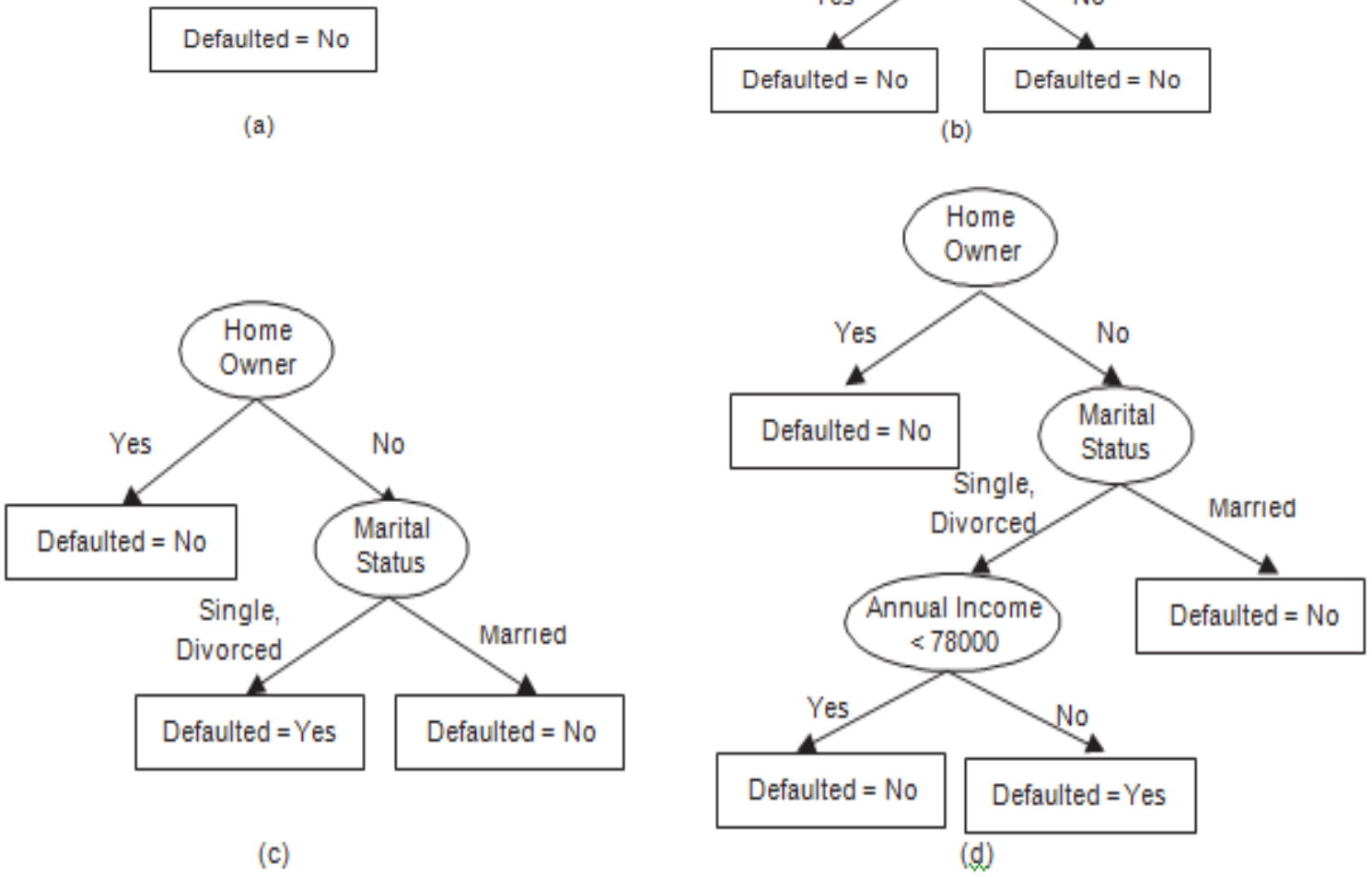
Decision Tree Induction

- This node expansion step can then be recursively applied to each child node, as long as it has labels of more than one class.

- If all the instances associated with a leaf node have identical class labels, then the node is not expanded any further.

- Each leaf node is assigned a class label that occurs most frequently in the training instances associated with the node.

Decision Tree Induction



Decision Tree Induction

- In the Algorithm, it makes some simplifying assumptions that are often not true in practice.
- The possible ways to handle the assumptions are:
  - Some of the child nodes created in Hunt's algorithm can be empty if none of the training instances have the particular attribute values. One way to handle this is by declaring each of them as a leaf node with a class label that occurs most frequently among the training instances associated with their parent nodes.
  - If all training instances associated with a node have identical attribute values but different class labels, it is not possible to expand this node any further. One way to handle this case is to declare it a leaf node and assign it the class label that occurs most frequently in the training instances associated with this node

Decision Tree Induction

Design Issues of Decision Tree Induction

- To implement the algorithm, there are two key design issues that must be addressed.

- **What is the splitting criterion?** At each recursive step, an attribute must be selected to partition the training instances associated with a node into smaller subsets associated with its child nodes. The splitting criterion determines which attribute is chosen as the test condition and how the training instances should be distributed to the child nodes.

Decision Tree Induction

- **What is the stopping criterion?** The basic algorithm stops expanding a node only when all the training instances associated with the node have the same class labels or have identical attribute values. Although these conditions are sufficient, there are reasons to stop expanding a node much earlier even if the leaf node contains training instances from more than one class. This process is called early termination and the condition used to determine when a node should be stopped from expanding is called a stopping criterion.

Attribute Selection Measures

- 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.
- If we were to split *D* into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure
- Attribute selection measures are also known as **splitting rules** because they determine how the tuples at a given node are to be split.
- 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.
- The tree node created for partition *D* is labeled with the splitting criterion, branches are grown for each outcome of the criterion, and the tuples are partitioned accordingly.

- Three popular attribute selection measures—
  - *information gain*
  - *gain ratio*
  - *Gini index*.

Attribute Selection Measures (Information Gain)

- Let node *N* represent or hold the tuples of partition *D*. The attribute with the highest information gain is chosen as the splitting attribute for node *N*.

  - 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:

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

  - Information gained by branching on attribute A

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

Attribute Selection Measures (Information Gain)

Example

- The following Table presents a training set, D, of class-labeled tuples randomly selected from the AllElectronics customer database

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

Attribute Selection Measures (Information Gain)

- In this example, each attribute is discrete-valued
- Continuous-valued attributes have been generalized
- The class label attribute, buys computer, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is, m = 2)
- Let class C_1 correspond to 'yes' and class C_2 correspond to 'no'.
- There are nine tuples of class 'yes' and five tuples of class 'no'.
- A (root) node N is created for the tuples in D

Attribute Selection Measures (Information Gain)

Expected information needed to classify a tuple in *D*

- To find the splitting criterion for these tuples, we must compute the information gain of each attribute
- Let us consider Class: buys computer as decision criteria D
- Calculate information:
- $= -p_y \log_2 (p_y) - p_n \log_2 (p_n)$
- Where p_y is probability of 'yes' and p_n is probability of '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 (Information Gain)

Calculation of entropy for ' Youth'

- Age can be:
  - youth
  - Middle_aged
  - Senior
- Youth

Youth	Class: buys computer
Yes	2
No	3

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

Attribute Selection Measures (Information Gain)

Calculation of entropy for ' Youth'

- Calculate Entropy for youth:
- Entropy youth = $-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}$
- Middle_aged

middle	Class: buys computer
Yes	4
No	0

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculation of entropy for ' Middle Age'

- Calculate Entropy for middle_aged
- $= -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$
- $= 0$

- For Senior

Senior	Class: buys computer
Yes	3
No	2

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for senior

Calculate Entropy for senior

$$= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$$

The expected information needed to classify a tuple in D according to age

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

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14}\times\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+ \frac{4}{14}\times\left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ &+ \frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.694 \text{ bits.} \end{aligned}$$

Attribute Selection Measures (Information Gain)

Calculation information Gain of Age

- Gain of Age:

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

Attribute Selection Measures (Information Gain)

Calculation information Gain of Income

- Calculation of gain for income:
- Income cane be:
  - High
  - Medium
  - Low

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for high

- High :

High	Class: buys computer
Yes	2
No	2

- Calculate Entropy for high:

$= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'medium'

- Medium:

Medium	Class: buys computer
Yes	4
No	2

- Calculate Entropy for Medium:

$= -(4/6)\log_2(4/6) - (2/6)\log_2(2/6)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'low'

- Low :

Low	Class: buys computer
No	1
Yes	3

- Calculate Entropy for Low:

  $= -(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Gain of income

- The expected information needed to classify a tuple in D if the tuples are partitioned according to income is:
- $Info_{income}(D) = (4/14)\,(-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)) +$
  $(6/14)\,(-(4/6)\log_2(4/6) - (2/6)\log_2(2/6)) +$
  $(4/14)\,-(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$
  $= 0.911$

Gain of income : $Info(D) - Info_{income}(D)$

$= 0.94 - 0.911 = 0.029$

Attribute Selection Measures (Information Gain)

Calculation of gain for student

- Calculation of gain for student
- Student can be:
  - Yes
  - No

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for No

- No :

No	Class: buys computer
Yes	3
No	4

- Calculate Entropy for No:

$= -(3/7)\log_2(3/7) - (4/7)\log_2(4/7)$

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'Yes'

- Yes :

Yes	Class: buys computer
Yes	6
No	1

- Calculate Entropy for Yes:

$= -(6/7)\log_2(6/7) - (1/7)\log_2(1/7)$

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

Attribute Selection Measures (Information Gain)

Gain of student

- The expected information needed to classify a tuple in D if the tuples are partitioned according to student is:
- $Info_{Student}(D) = (7/14)\,(-(3/7)\log_2(3/7) - (4/7)\log_2(4/7)) +$
  $(7/14)\,(-(6/7)\log_2(6/7) - (1/7)\log_2(1/7))$
  $=0.789$
- Gain(student) :
  $Info(D) - Info_{student}(D)$
  $= 0.94 - 0.789 = 0.151$

Attribute Selection Measures (Information Gain)

Calculation of gain for credit rating

- Calculation of gain for credit rating
- Credit rating can be:
  - Fair
  - Excellent

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for Fair

- Fair :

Fair	Class: buys computer
Yes	6
No	2

- Calculate Entropy for Fair:

$= -(6/8)\log_2(6/8) - (2/8)\log_2(2/8)$

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for Excellent

- Excellent :

Yes	Class: buys computer
Yes	3
No	3

- Calculate Entropy for Excellent:

$= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$

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

Attribute Selection Measures (Information Gain)

Gain for credit rating

- The expected information needed to classify a tuple in D if the tuples are partitioned according to Credit rating is:
- $\text{Info}_{\text{Credit rating}}(D) = (8/14)\,(-(6/8)\log_2(6/8) - (2/8)\log_2(2/8)) +$
  $(6/14)\,(-(3/6)\log_2(3/6) - (3/6)\log_2(3/6))$
  $=0.892$
- Gain for credit rating :
  $\text{Info}(D) - \text{Info}_{\text{Credit rating}}(D)$
  $= 0.94 - 0.892 = 0.048$

Attribute Selection Measures (Information Gain)

Independent variable	Information gain
Age	0.246
Income	0.029
Student	0.151
Credit_rating	0.048

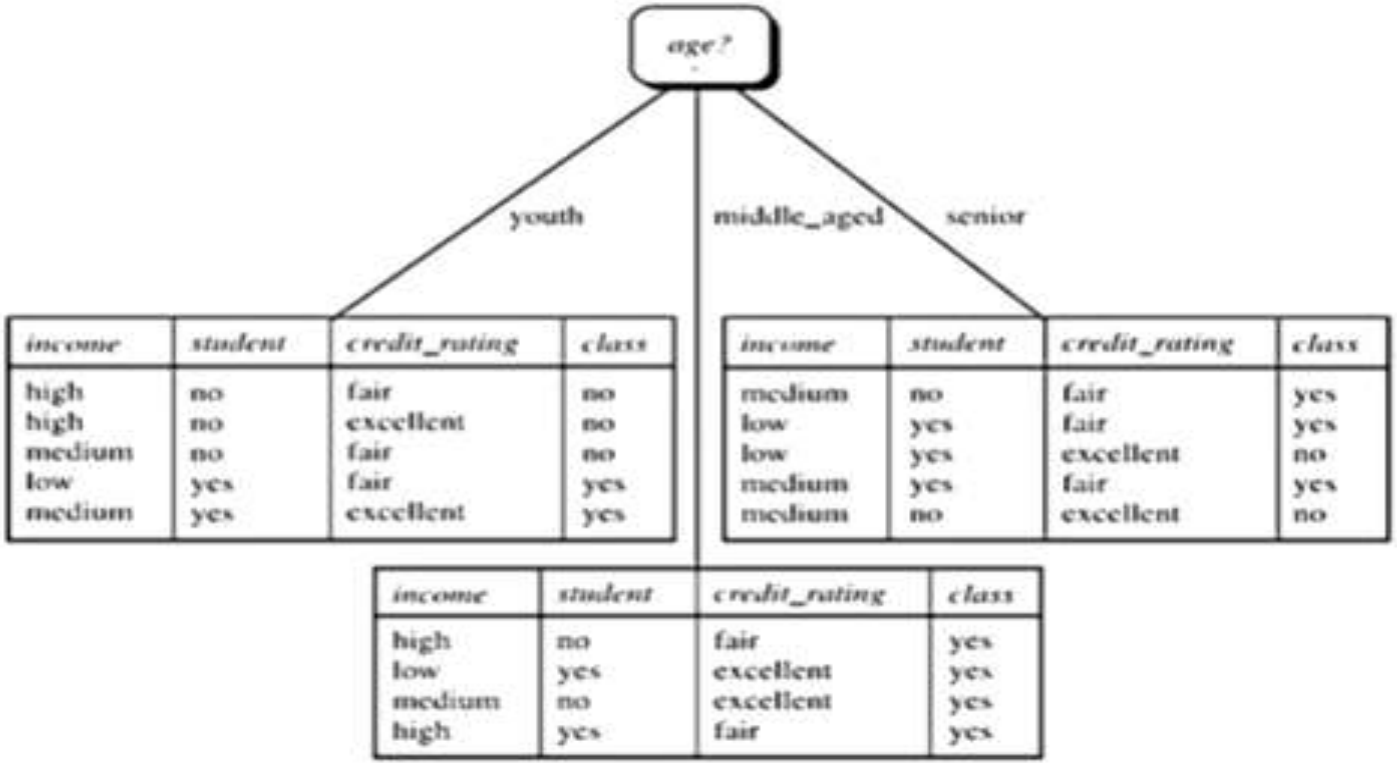
Attribute Selection Measures (Information Gain)

Selection of root classifier

- Because age has the highest information gain among the attributes, it is selected as the splitting attribute
- Node N is labelled with age, and branches are grown for each of the attribute's values
- The tuples are then partitioned accordingly
- 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 labelled with "yes."

Attribute Selection Measures (Information Gain)

Decision tree



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

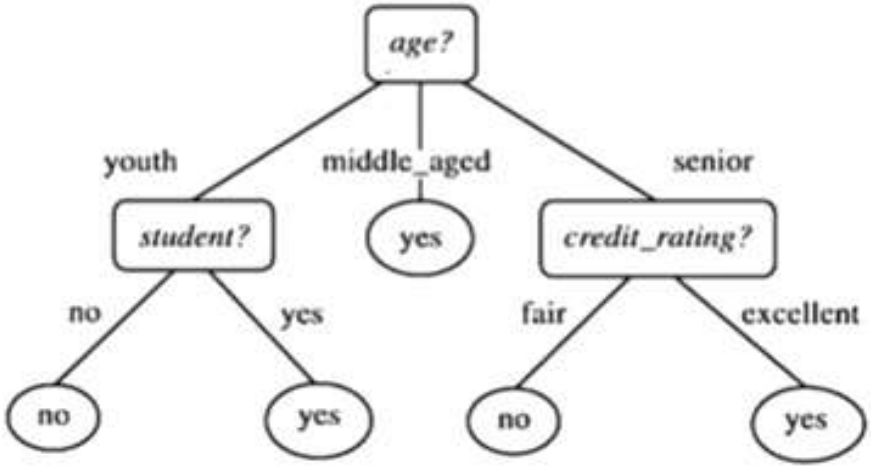
income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Attribute Selection Measures (Information Gain)

Decision tree

- The final decision tree returned by the algorithm is shown in Figure



Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values 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 ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Attribute Selection Measures (Gain Ratio)

- **Disadvantages of Inforamtion Gain**
  - Need to calculate entroy for every attribute.
  - If any attribute have unique values, then the entropy value will be 0, In this case, the partition will be useless.

- The information gain measure is biased toward tests with many outcomes.
- It prefers to select attributes having a large number of values
- C4.5, a successor of ID3, uses an extension to information gain known as *gain ratio*, which attempts to overcome this bias.

Attribute Selection Measures (Gain Ratio)

- It applies a kind of normalization to information gain using a "split information" value defined 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*.

Attribute Selection Measures (Gain Ratio)

- The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}.$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.
- If the split information approaches 0, the ratio becomes unstable.

Attribute Selection Measures (Gain Ratio)

- Example:
- A test on *income* splits the data of data set into three partitions, namely *low*, *medium*, and *high*, containing four, six, and four tuples, respectively.
- To compute the gain ratio of *income*

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

$$= 1.557.$$

- $Gain(income) = 0.029$
- $GainRatio(income) = 0.029/1.557 = 0.019$.
- The attribute with the maximum gain ratio is selected as the splitting attribute.

Attribute Selection Measures (Gini Index)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

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

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

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

$$= 0.443$$

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

  $Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is [illegible], so split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

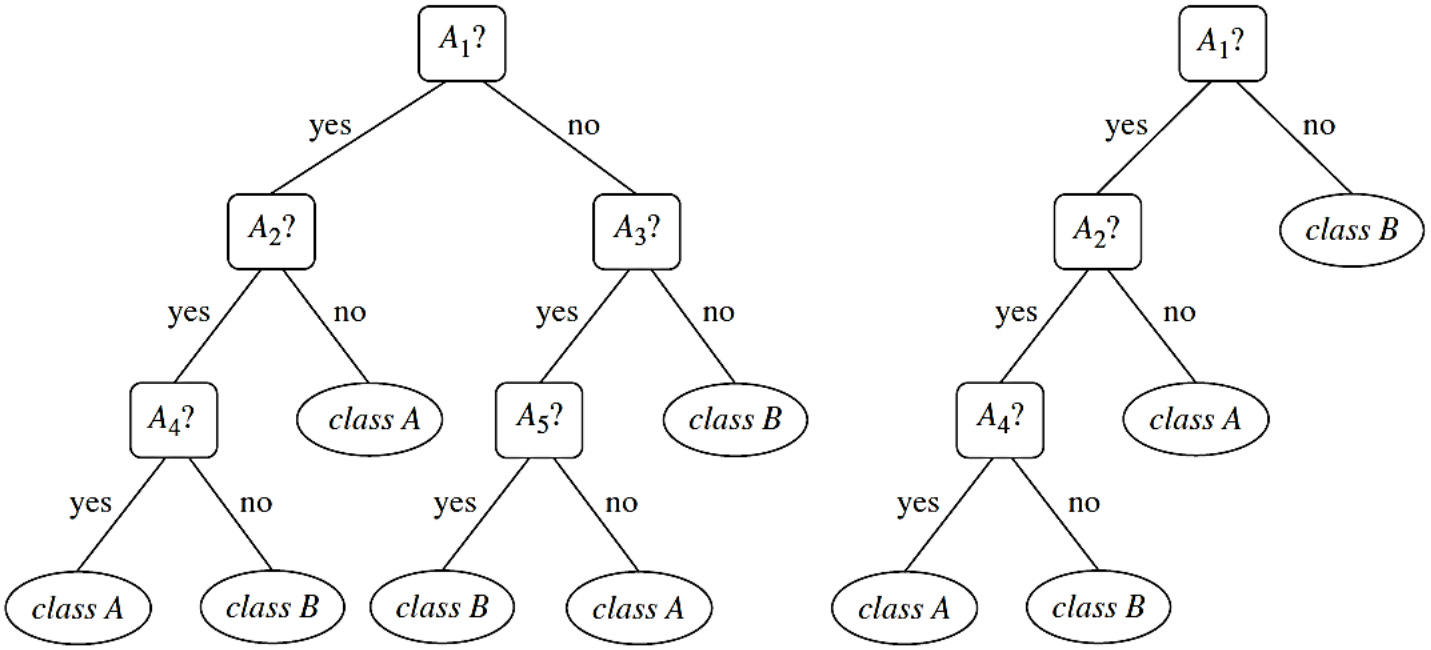
- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes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

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than 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 methods address this problem of overfitting the data. Such methods typically use statistical measures to remove the least-reliable branches. An unpruned tree and a pruned version of it are shown below:



Tree Pruning

- 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 (i.e., of previously unseen tuples) than unpruned trees.
- There are two common approaches to tree pruning:
  1) Prepruning
  2) Postpruning

1) Prepruning

- In the prepruning 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).
- 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
- When constructing a tree, measures such as statistical significance, information gain, Gini index, and so on, can be used to assess the goodness of a split

Tree Pruning

1) Prepruning

- If partitioning the tuples at a node would result in a split that falls below a prespecified threshold, then further partitioning of the given subset is halted
- There are difficulties, however, in choosing an appropriate threshold
- High thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification

2) Postpruning

- It is a more common approach which removes subtrees 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
- The cost complexity pruning algorithm used in CART is an example of the postpruning approach

Tree Pruning

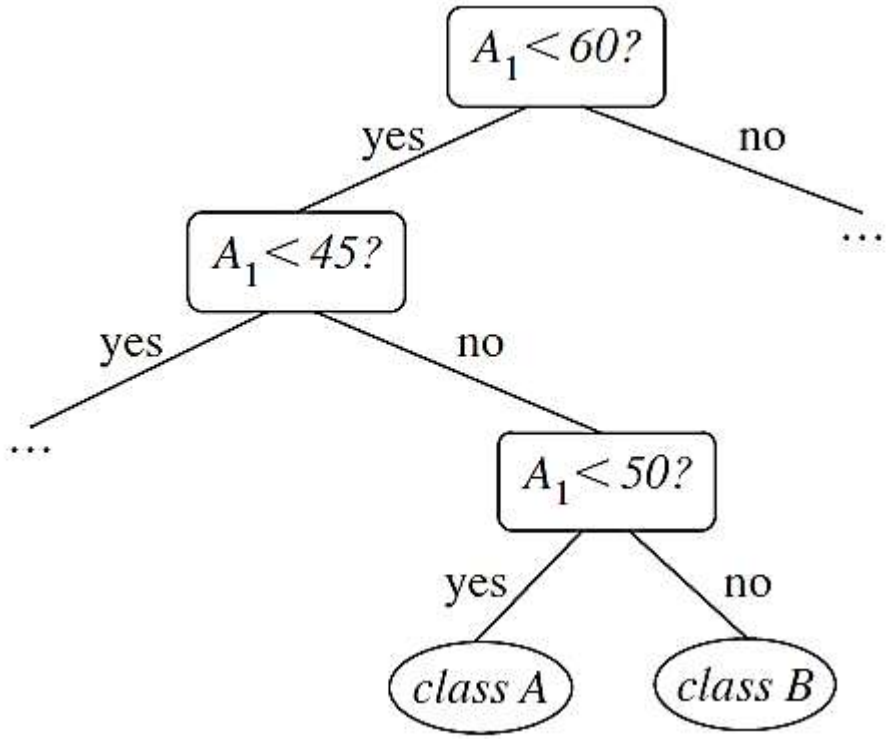
2) Postpruning

- This approach considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).
- It starts from the bottom of the tree
- For each internal node, N, it computes the cost complexity of the subtree at N, and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node). The two values are compared.
- If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept
- A pruning set of class-labeled tuples is used to estimate cost complexity
- C4.5 uses a method called pessimistic pruning, which is similar to the cost complexity method in that it also uses error rate estimates to make decisions regarding subtree pruning
- Pessimistic pruning, however, does not require the use of a prune set. Instead, it uses the training set to estimate error rates

Tree Pruning

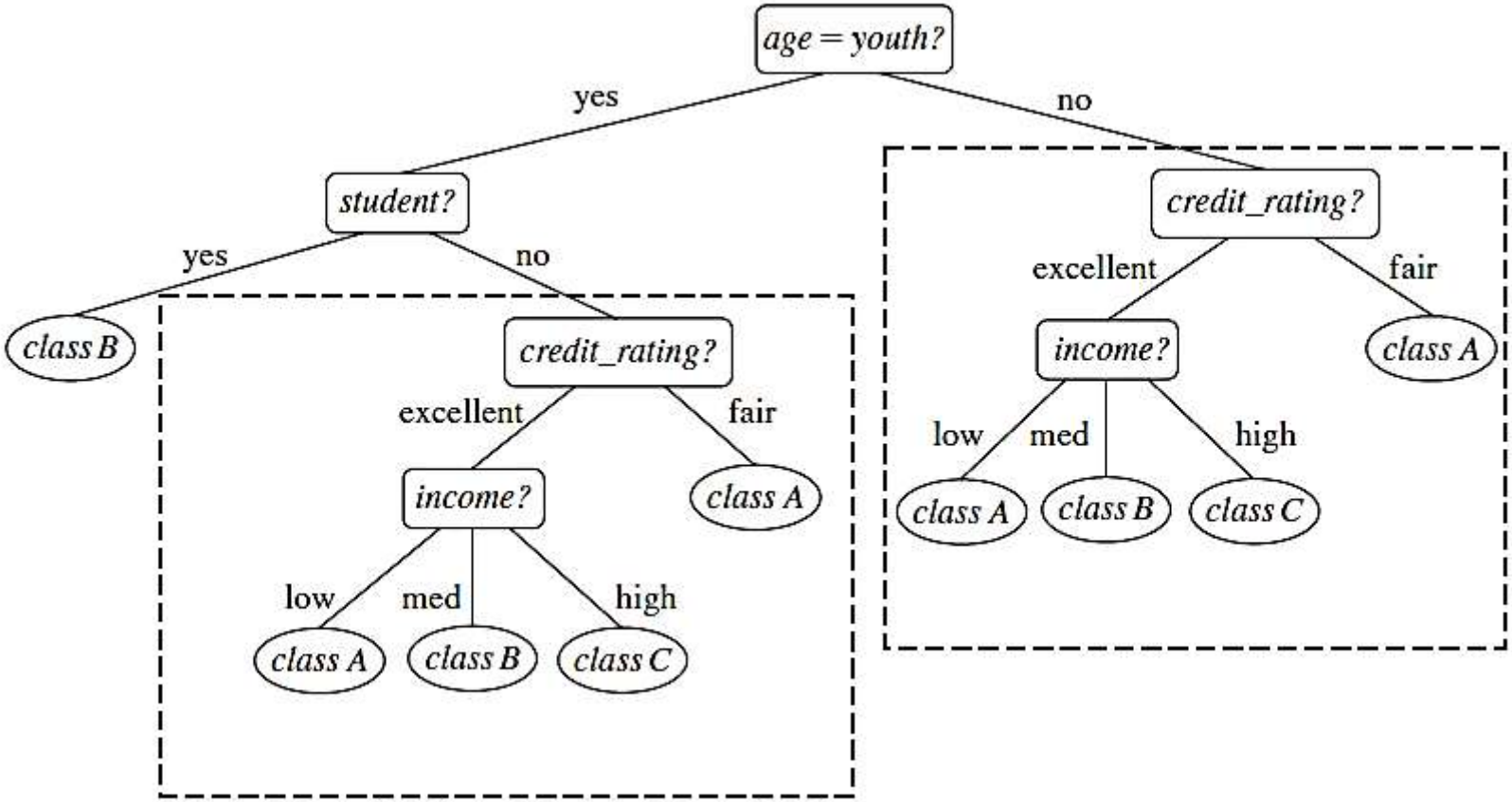
- Prepruning and postpruning may be interleaved for a combined approach.
- Postpruning requires more computation than prepruning, yet generally leads to a more reliable tree
- No single pruning method has been found to be superior over all others
- Although pruned trees tend to be more compact than their unpruned counterparts, they may still be rather large and complex.
- Decision trees can suffer from repetition and replication, making them overwhelming to interpret

Tree Pruning



Subtree repetition, where an attribute is repeatedly tested along a given branch of the tree (e.g. age)

Tree Pruning



Subtree replication, where duplicate subtrees exist within a tree (e.g. the subtree headed by the node "credit rating?")

Tree Pruning

- These situations can impede the accuracy and comprehensibility of a decision tree
- The use of multivariate splits (splits based on a combination of attributes) can prevent these problems
- Another approach is to use a different form of knowledge representation, such as rules, instead of decision trees