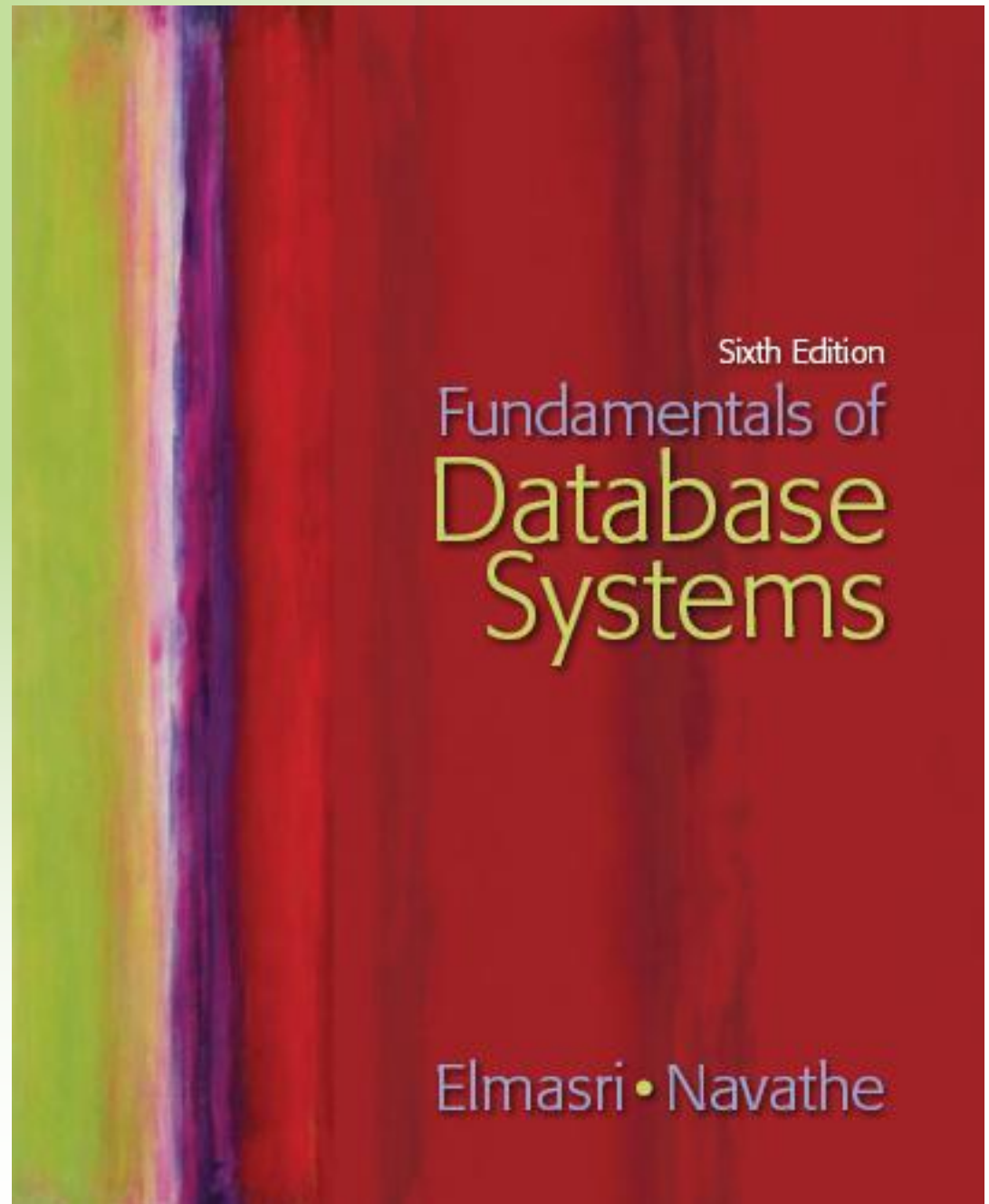


Chapter 28

Data Mining Concepts



Sixth Edition

Fundamentals of Database Systems

Elmasri • Navathe

Addison-Wesley
is an imprint of

PEARSON

Definitions of Data Mining

- The discovery of new information in terms of patterns or rules from vast amounts of data.
- The process of finding interesting structure in data.
- The process of employing one or more computer learning techniques to automatically analyze and extract knowledge from data.

Data Warehousing

- The data warehouse is a historical database designed for decision support.
- Data mining can be applied to the data in a warehouse to help with certain types of decisions.
- Proper construction of a data warehouse is fundamental to the successful use of data mining.

Knowledge Discovery in Databases (KDD)

- Data mining is actually one step of a larger process known as **knowledge discovery in databases (KDD)**.
- The KDD process model comprises six phases
 - Data selection
 - Data cleansing
 - Enrichment
 - Data transformation or encoding
 - Data mining
 - Reporting and displaying discovered knowledge

Goals of Data Mining and Knowledge Discovery (PICO)

■ Prediction:

- Determine how certain attributes will behave in the future.

■ Identification:

- Identify the existence of an item, event, or activity.

■ Classification:

- Partition data into classes or categories.

■ Optimization:

- Optimize the use of limited resources.

Types of Discovered Knowledge

- Association Rules
- Classification Hierarchies
- Sequential Patterns
- Patterns Within Time Series
- Clustering

Association Rules

- Association rules are frequently used to generate rules from **market-basket data**.
 - A market basket corresponds to the sets of items a consumer purchases during one visit to a supermarket.
- The set of items purchased by customers is known as an **itemset**.
- An **association rule** is of the form $X \Rightarrow Y$, where $X = \{x_1, x_2, \dots, x_n\}$, and $Y = \{y_1, y_2, \dots, y_n\}$ are sets of items, with x_i and y_j being distinct items for all i and all j .
 - For an association rule to be of interest, it must satisfy a minimum support and confidence.

What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?
 - What are the subsequent purchases after buying a PC?
 - What kinds of DNA are sensitive to this new drug?
 - Can we automatically classify web documents?
- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

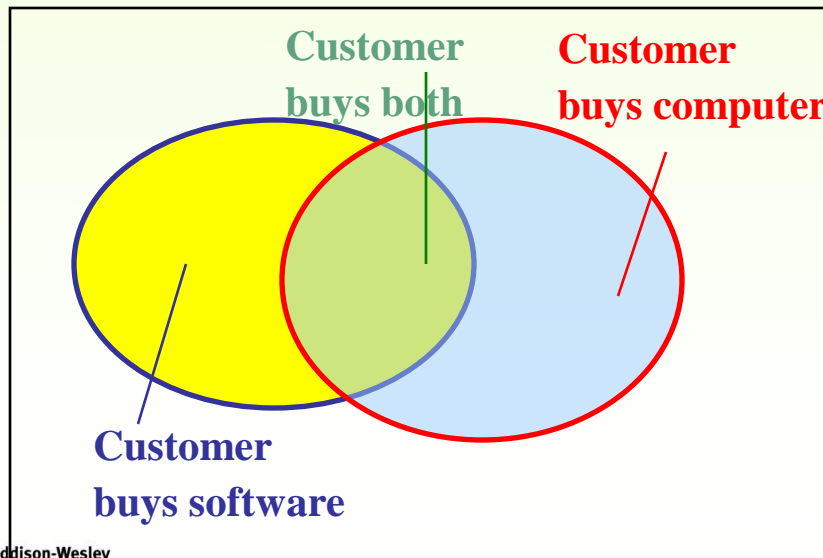
Basic Concepts: Frequent Patterns and Association Rules

- Market basket analysis: Which groups or sets of items are customers likely to purchase on a given trip to the store?
- These patterns can be represented in the form of *association rules*.
 - For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in Association Rule:
 - computer \Rightarrow antivirus software [support = 2%, confidence = 60%]
 - A **support** of 2% means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
 - A **confidence** of 60% means that 60% of the customers who purchased a computer also bought the software.
- Association rules are considered interesting if they satisfy both a **minimum support threshold** and a **minimum confidence threshold**.

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Itemset $X = \{x_1, \dots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - support**, s , $P(X \cup Y)$ probability that a transaction contains $X \cup Y$.
 - confidence**, c , $P(Y/x)$ conditional probability that a transaction having X also contains $Y = \text{count}(X \cup Y) / \text{count}(X)$.



Let $sup_{min} = 50\%$, $conf_{min} = 50\%$
 Freq. Pat.: $\{A:3, B:3, D:4, E:3, AD:3\}$
 Association rules:
 $A \rightarrow D$ (60%, 100%)
 $D \rightarrow A$ (60%, 75%)

Closed Patterns and Max-Patterns

- In general, association rule mining can be viewed as a two-step process:
 - **Find all frequent itemsets:** By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
 - **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.
- Because if an itemset is frequent, then each of its subsets is frequent as well, a major challenge in mining frequent itemsets from a large data set is that it generates a huge number of itemsets satisfying the minimum support (min sup) threshold, especially when min sup is set low.

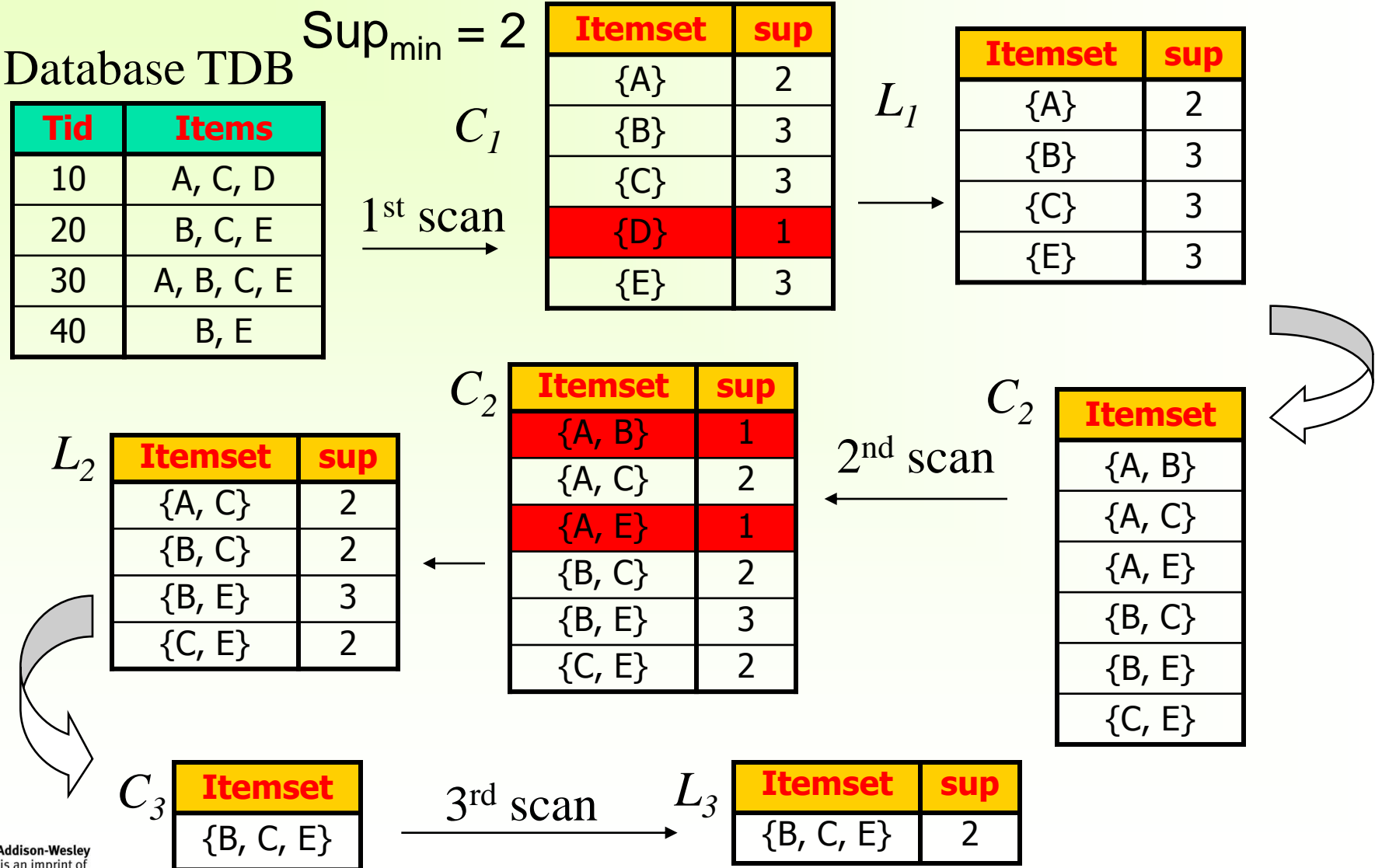
Scalable Methods for Mining Frequent Patterns

- The ***Apriori property*** (downward closure) of frequent patterns
 - Any subset of a frequent itemset must be frequent
 - If {milk, diaper, nuts} is frequent, so is {milk, diaper}
 - i.e., every transaction having {milk, diaper, nuts} also contains {milk, diaper}
 - If a set cannot pass the test, all of its supersets will fail the same test as well. (*antimonotone*)
- Scalable mining methods: *Three major approaches*
 - Apriori (Agrawal & Srikant@VLDB'94)
 - Freq. pattern growth (FP-growth—Han, Pei & Yin @SIGMOD'00)
 - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Method:
 - **Initially**, scan DB once to get frequent 1-itemset
 - **Generate** length $(k+1)$ candidate itemsets from length k frequent itemsets, C_k .
 - **Prune** C_k using the Apriori property.
 - **Test** the candidates against DB to generate the list of frequent $(k+1)$ itemsets satisfying the min sup, L_k .
 - **Terminate** when no frequent itemset, L_k , or candidate set, C_k , can be generated (Apriori pruning principle)

The Apriori Algorithm - Example 1



The Apriori Algorithm - Example 2

Transactional data for an *AMElectronics* branch.

TID	List of Item IDs
T100	11, 12, 15
T200	12, 14
T300	12, 13
T400	11, 12, 14
T500	11, 13
T600	12, 13
T700	11, 13
T800	11, 12, 13, 15
T900	11, 12, 13

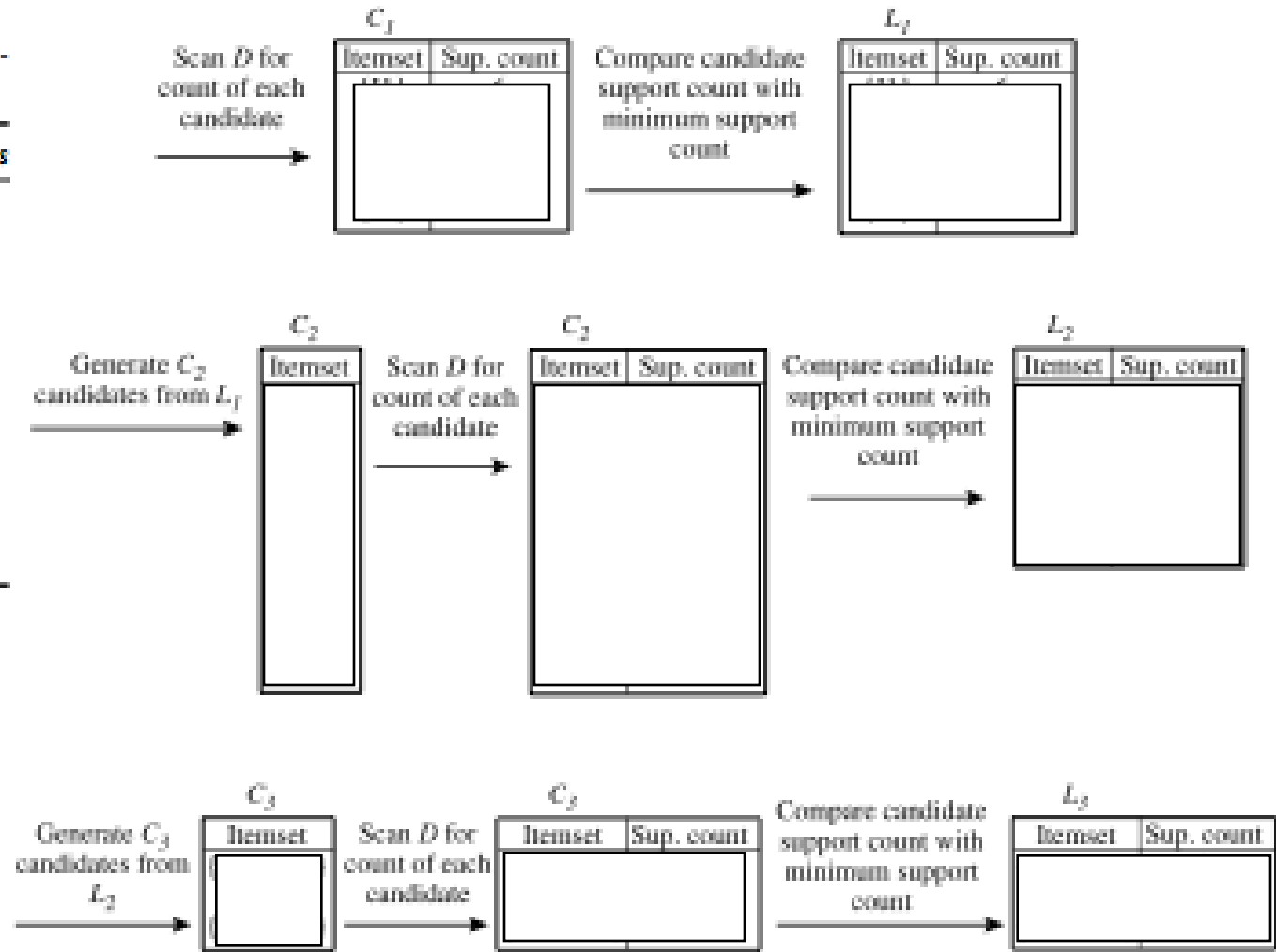


Figure 5.2 Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

The Apriori Algorithm - Example 2

Transactional data for an *AMElectronics* branch.

TID	List of Item IDs
T100	11, 12, 15
T200	12, 14
T300	12, 13
T400	11, 12, 14
T500	11, 13
T600	12, 13
T700	11, 13
T800	11, 12, 13, 15
T900	11, 12, 13

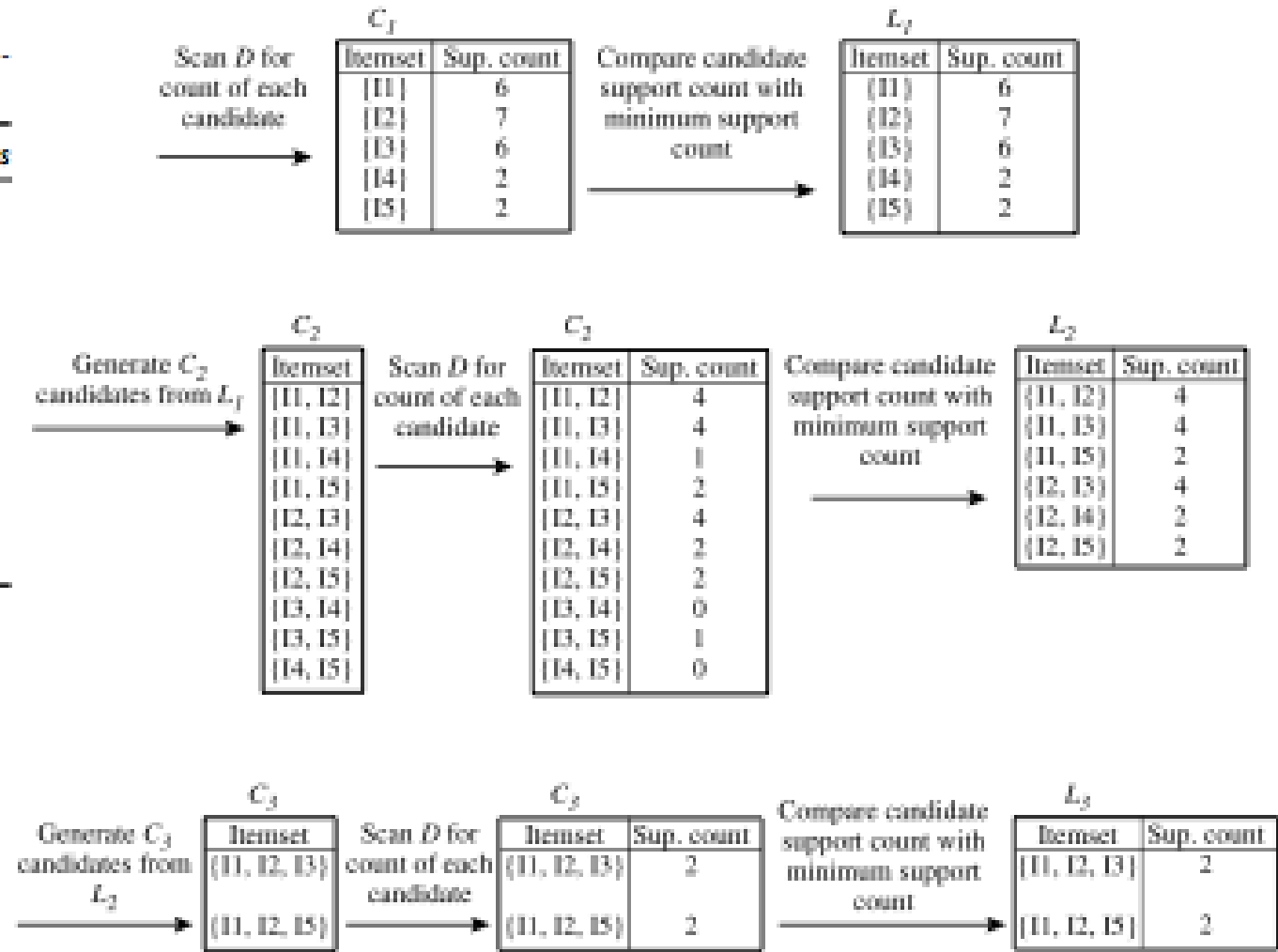


Figure 5.2 Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

Generating Association Rules

- Once the frequent itemsets have been found, it is straightforward to generate strong association rules from them.
- Strong association rules satisfy both minimum support and minimum confidence.

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}$$

- Support-count($A \cup B$) is the number of transactions containing the item sets $A \cup B$, and support-count(A) is the number of transactions containing the itemset A .
- Association rules can be generated as follows:
 - For each frequent itemset I , generate all nonempty subsets of I .
 - For every nonempty subset s of I , output the rule " $s \Rightarrow (I - s)$ ", where $\text{confidence}(s \Rightarrow (I - s)) \geq \text{min_conf}$, where min_conf is the minimum confidence threshold.
- Because the rules are generated from frequent itemsets, each one automatically satisfies minimum support.

Generating Association Rules

- Suppose the data contain the frequent itemset $I = \{I1, I2, I5\}$.
- The nonempty subsets of I are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$.
- The resulting association rules are as shown below, each listed with its confidence:

$I1 \wedge I2 \Rightarrow I5$, confidence = $2/4 = 50\%$
 $I1 \wedge I5 \Rightarrow I2$, confidence = $2/2 = 100\%$
 $I2 \wedge I5 \Rightarrow I1$, confidence = $2/2 = 100\%$
 $I1 \Rightarrow I2 \wedge I5$, confidence = $2/6 = 33\%$
 $I2 \Rightarrow I1 \wedge I5$, confidence = $2/7 = 29\%$
 $I5 \Rightarrow I1 \wedge I2$, confidence = $2/2 = 100\%$

Transactional data for an AllElectronics branch.

TID	List of item IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong.

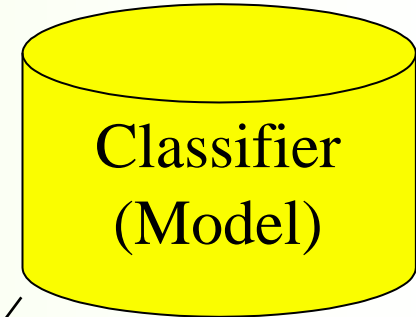
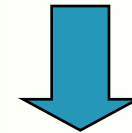
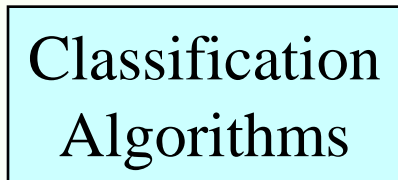
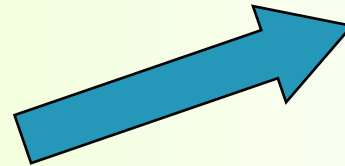
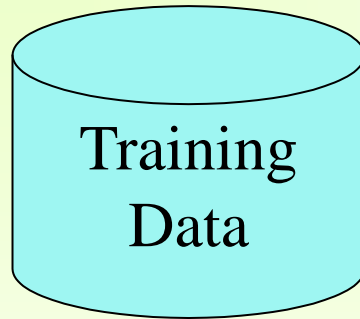
Classification

- **Classification** is the process of learning a model that is able to describe different classes of data.
- Learning is **supervised** as the classes to be learned are predetermined.
- Learning is accomplished by using a training set of pre-classified data.
- The model produced is usually in the form of a decision tree or a set of rules.

Classification—A Two-Step Process

- **Model construction:** Learning step
 - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
 - The set of tuples/labels 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 (Test-set)**
 - 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 over-fitting will occur
 - **Classification step:**
 - If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known

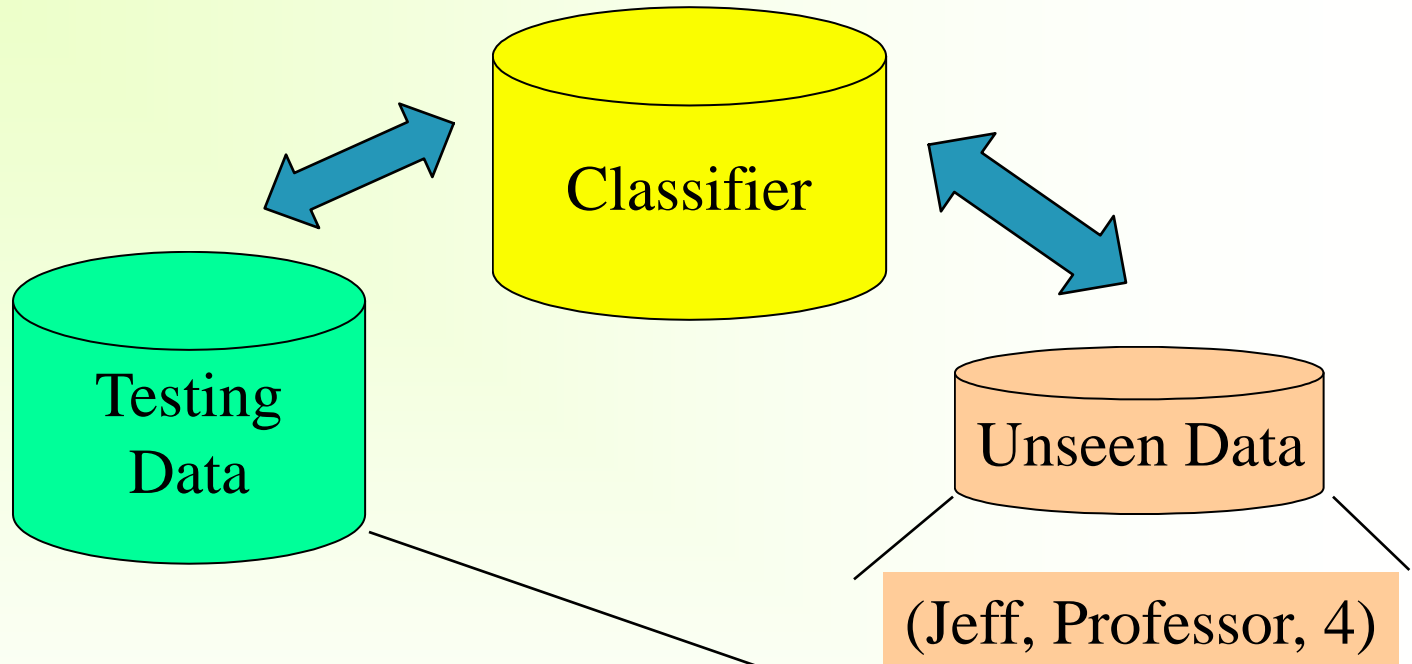
Process (1): Model Construction



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

(Jeff, Professor, 4)

Tenured? 
Yes

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

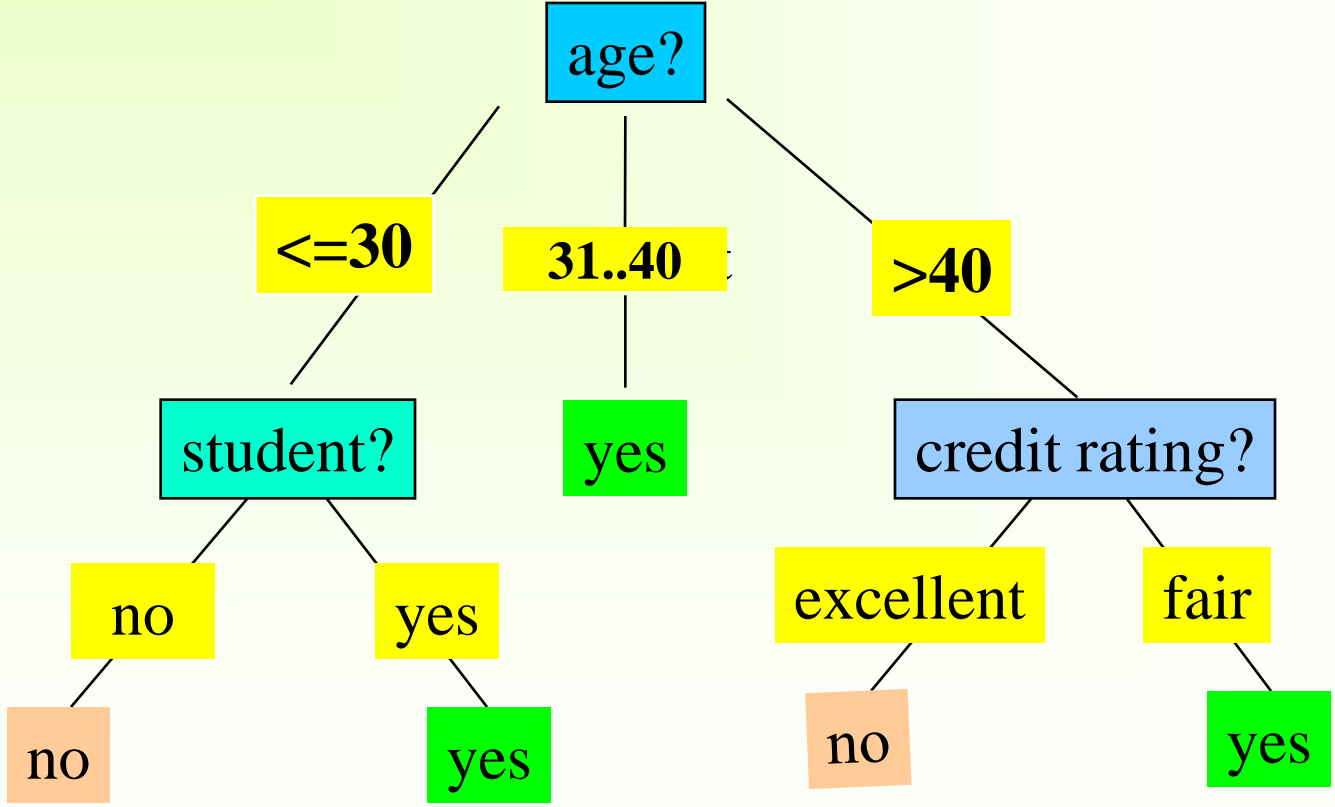
Issues: Evaluating Classification Methods

- Accuracy
 - classifier accuracy: predicting class label
 - predictor accuracy: guessing value of predicted attributes
- Speed
 - time to construct the model (training time)
 - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
 - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Decision Tree Induction: Training Dataset

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

Output: A Decision Tree for “buys_computer”



Decision Tree: Partitioning Scenarios

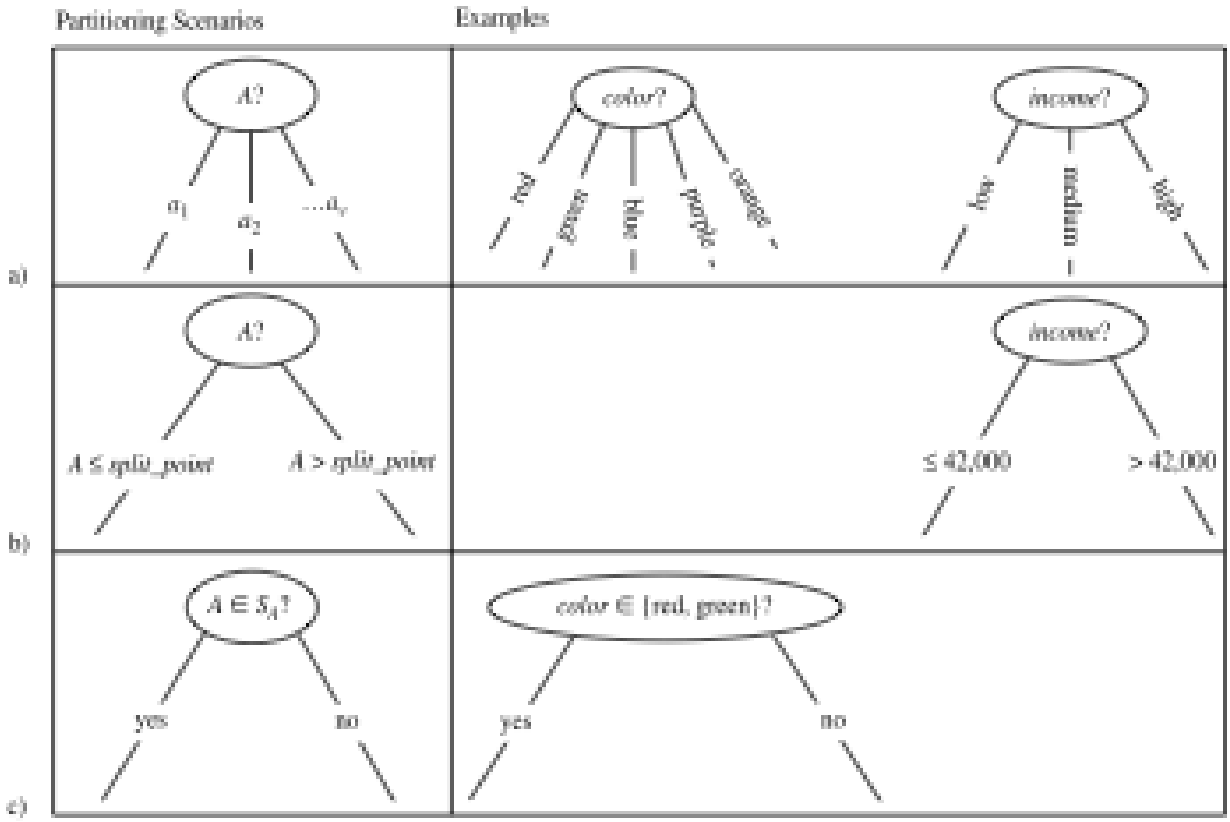


Figure 6.4 Three possibilities for partitioning tuples based on the splitting criterion, shown with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A . (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq \text{split_point}$ 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

- 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 (convert a node to a leaf that is labeled with the most common class).
 - There are no samples left

Attribute Selection Measure: Information Gain (ID3/C4.5)

- 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 for m classes:

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

- **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: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
31...40	4	0	0
> 40	3	2	0.971

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

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$$\frac{5}{14} I(2,3) = \frac{5}{14} \left(-\frac{2}{5} \log_2\left(\frac{2}{5}\right) - \frac{3}{5} \log_2\left(\frac{3}{5}\right) \right)$$

means "age ≤ 30 " has 5 out of 14 samples, with 2 yes's and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

hing:

30

Attribute Selection: Information Gain

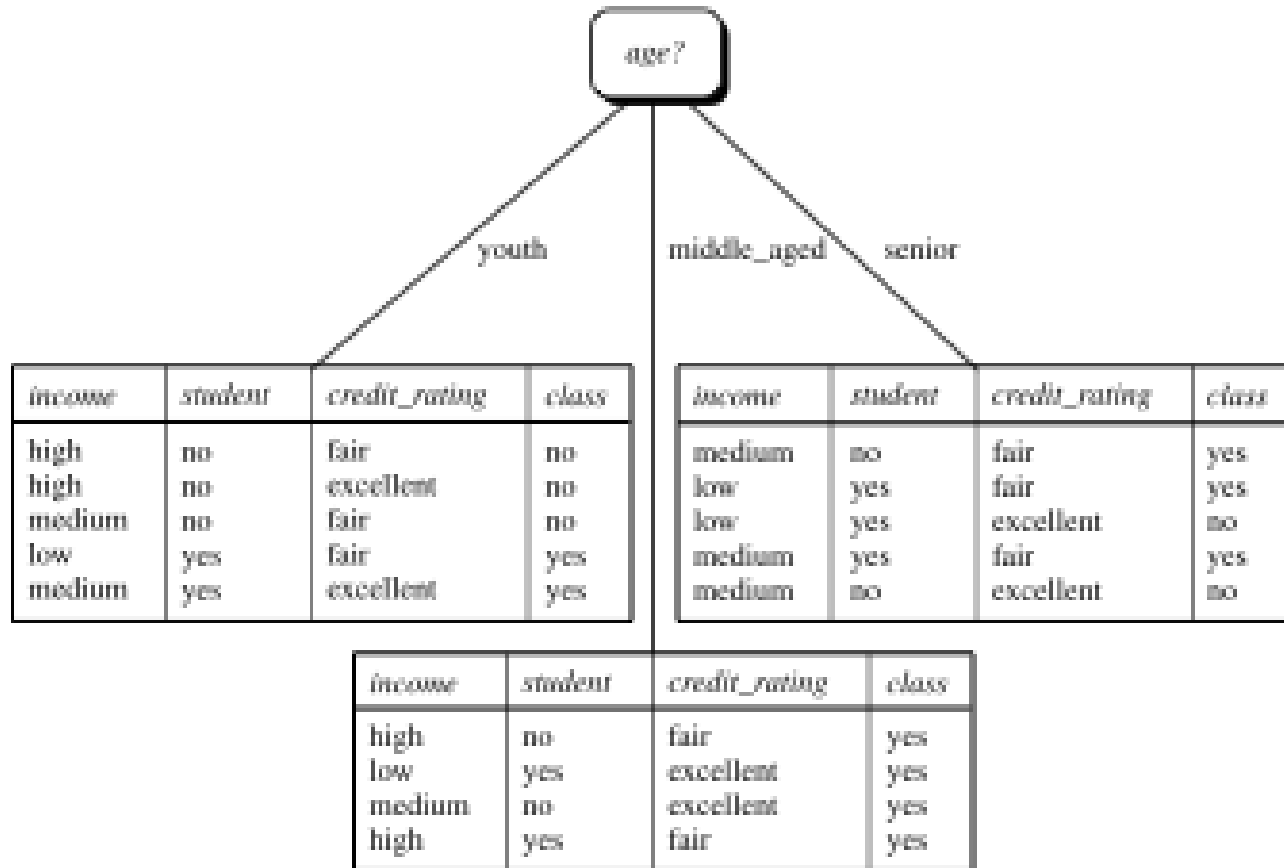


Figure 6.5 The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.

Data Mining Applications

- **Marketing**

- Marketing strategies and consumer behavior

- **Finance**

- Fraud detection, creditworthiness and investment analysis

- **Manufacturing**

- Resource optimization

- **Health**

- Image analysis, side effects of drug, and treatment effectiveness