Why we need Data Mining?

Volume of information is increasing everyday that we can handle from business transactions, scientific data, sensor data, Pictures, videos, etc. So, we need a system that will be capable of extracting essence of information available and that can automatically generate report, views or summary of data for better decision-making.

Why Data Mining is used in Business?

Data mining is used in business to make better managerial decisions by:
- Automatic summarization of data
- Extracting essence of information stored.
- Discovering patterns in raw data.
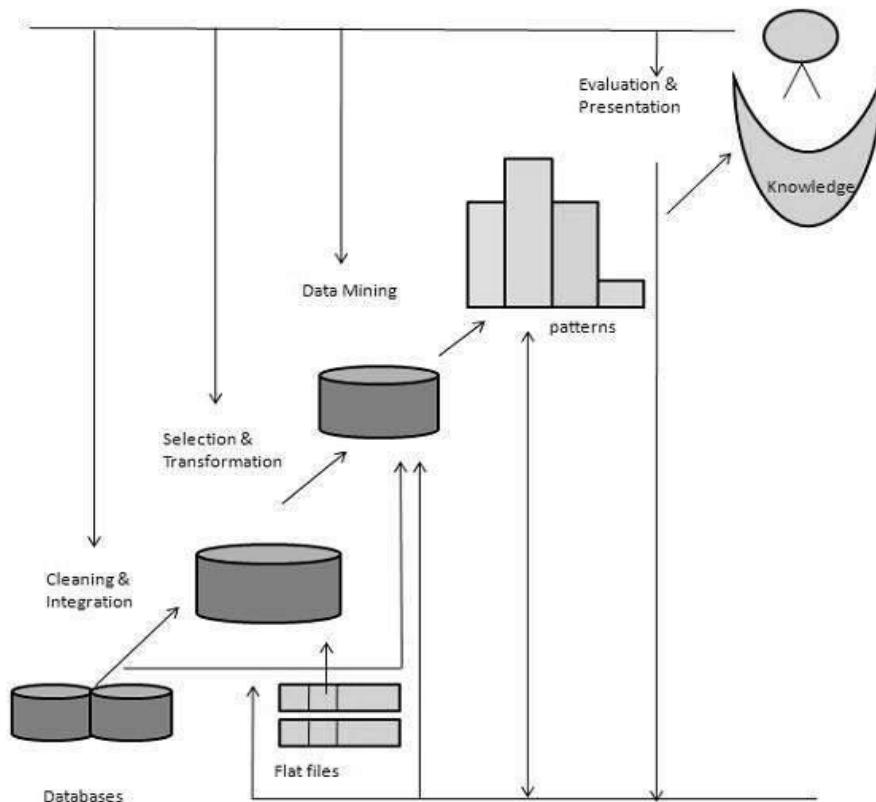
What is Data Mining?

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications −
- Market Analysis
- Fraud Detection
- Customer Retention
- Production Control
- Science Exploration

Knowledge discovery from Data (KDD) is essential for data mining. While others view data mining as an essential step in the process of knowledge discovery. Here is the list of steps involved in the knowledge discovery process −

- **Data Cleaning** − In this step, the noise and inconsistent data is removed.
- **Data Integration** − In this step, multiple data sources are combined.
- **Data Selection** − In this step, data relevant to the analysis task are retrieved from the database.
- **Data Transformation** − In this step, data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
- **Data Mining** − In this step, intelligent methods are applied in order to extract data patterns.
- **Pattern Evaluation** − In this step, data patterns are evaluated.
- **Knowledge Presentation** − In this step, knowledge is represented.

What kinds of data can be mined?

1. Flat Files
2. Relational Databases
3. DataWarehouse
4. Transactional Databases
5. Multimedia Databases
6. Spatial Databases
7. Time Series Databases
8. World Wide Web(WWW)

1. *Flat Files*
- Flat files are defined as data files in text form or binary form with a structure that can be easily extracted by data mining algorithms.
- Data stored in flat files have no relationship or path among themselves, like if a relational database is stored on flat file, and then there will be no relations between the tables.
- Flat files are represented by data dictionary. Eg: CSV file.
- **Application**: Used in Data Warehousing to store data, Used in carrying data to and from server, etc.

2. *Relational Databases*
- A Relational database is defined as the collection of data organized in tables with rows and columns.
- Physical schema in Relational databases is a schema which defines the structure of tables.
- Logical schema in Relational databases is a schema which defines the relationship among tables.
- Standard API of relational database is SQL.
- **Application**: Data Mining, ROLAP model, etc.

3. *DataWarehouse*
   - A datawarehouse is defined as the collection of data integrated from multiple sources that will query and decision making.
   - There are three types of datawarehouse: **Enterprise** datawarehouse, **Data Mart** and **Virtual** Warehouse.
   - Two approaches can be used to update data in DataWarehouse: **Query-driven** Approach and **Update-driven** Approach.
   - **Application**: Business decision making, Data mining, etc.

4. *Transactional Databases*
   - Transactional databases are a collection of data organized by time stamps, date, etc to represent transaction in databases.
   - This type of database has the capability to roll back or undo its operation when a transaction is not completed or committed.
   - Highly flexible system where users can modify information without changing any sensitive information.
   - Follows ACID property of DBMS.
   - **Application**: Banking, Distributed systems, Object databases, etc.

5. *Multimedia Databases*
   - Multimedia databases consists audio, video, images and text media.
   - They can be stored on Object-Oriented Databases.
   - They are used to store complex information in pre-specified formats.
   - **Application**: Digital libraries, video-on demand, news-on demand, musical database, etc.

6. *Spatial Database*
   - Store geographical information.
   - Stores data in the form of coordinates, topology, lines, polygons, etc.
   - **Application**: Maps, Global positioning, etc.

7. *Time-series Databases*
   - Time series databases contain stock exchange data and user logged activities.
   - Handles array of numbers indexed by time, date, etc.
   - It requires real-time analysis.
   - **Application**: eXtremeDB, Graphite, InfluxDB, etc.

8. *WWW*
   - WWW refers to World wide web is a collection of documents and resources like audio, video, text, etc which are identified by Uniform Resource Locators (URLs) through web browsers, linked by HTML pages, and accessible via the Internet network.
   - It is the most heterogeneous repository as it collects data from multiple resources.
   - It is dynamic in nature as Volume of data is continuously increasing and changing.
   - **Application**: Online shopping, Job search, Research, studying, etc.

What kinds of Patterns can be mined?

On the basis of the kind of data to be mined, there are two categories of functions involved in Data Mining −
   a) Descriptive
   b) Classification and Prediction

a) Descriptive Function

The descriptive function deals with the general properties of data in the database. Here is the list of descriptive functions −

1. Class/Concept Description
2. Mining of Frequent Patterns
3. Mining of Associations
4. Mining of Correlations
5. Mining of Clusters

1. Class/Concept Description

Class/Concept refers to the data to be associated with the classes or concepts. For example, in a company, the classes of items for sales include computer and printers, and concepts of customers include big spenders and budget spenders. Such descriptions of a class or a concept are called class/concept descriptions. These descriptions can be derived by the following two ways −

- **Data Characterization** − This refers to summarizing data of class under study. This class under study is called as Target Class.
- **Data Discrimination** − It refers to the mapping or classification of a class with some predefined group or class.

2. Mining of Frequent Patterns

Frequent patterns are those patterns that occur frequently in transactional data. Here is the list of kind of frequent patterns −

- **Frequent Item Set** − It refers to a set of items that frequently appear together, for example, milk and bread.
- **Frequent Subsequence** − A sequence of patterns that occur frequently such as purchasing a camera is followed by memory card.
- **Frequent Sub Structure** − Substructure refers to different structural forms, such as graphs, trees, or lattices, which may be combined with item-sets or subsequences.

3. Mining of Association

Associations are used in retail sales to identify patterns that are frequently purchased together. This process refers to the process of uncovering the relationship among data and determining association rules.

For example, a retailer generates an association rule that shows that 70% of time milk is sold with bread and only 30% of times biscuits are sold with bread.

4. Mining of Correlations

It is a kind of additional analysis performed to uncover interesting statistical correlations between associated-attribute-value pairs or between two item sets to analyze that if they have positive, negative or no effect on each other.

5. Mining of Clusters

Cluster refers to a group of similar kind of objects. Cluster analysis refers to forming group of objects that are very similar to each other but are highly different from the objects in other clusters.

b) Classification and Prediction

Classification is the process of finding a model that describes the data classes or concepts. The purpose is to be able to use this model to predict the class of objects whose class label is unknown. This derived model is based on the analysis of sets of training data. The derived model can be presented in the following forms −

1. Classification (IF-THEN) Rules
2. Prediction
3. Decision Trees
4. Mathematical Formulae
5. Neural Networks
6. Outlier Analysis
7. Evolution Analysis

The list of functions involved in these processes is as follows −

1. **Classification** − It predicts the class of objects whose class label is unknown. Its objective is to find a derived model that describes and distinguishes data classes or concepts. The Derived Model is based on the analysis set of training data i.e. the data object whose class label is well known.

2. **Prediction** − It is used to predict missing or unavailable numerical data values rather than class labels. Regression Analysis is generally used for prediction. Prediction can also be used for identification of distribution trends based on available data.

3. **Decision Trees** − A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label.

4. **Mathematical Formulae** – Data can be mined by using some mathematical formulas.

5. **Neural Networks** − Neural networks represent a brain metaphor for information processing. These models are biologically inspired rather than an exact replica of how the brain actually functions. Neural networks have been shown to be very promising systems in many forecasting applications and business classification applications due to their ability to "**learn**" from the data.

6. **Outlier Analysis** − Outliers may be defined as the data objects that do not comply with the general behavior or model of the data available.

7. **Evolution Analysis** − Evolution analysis refers to the description and model regularities or trends for objects whose behavior changes over time.
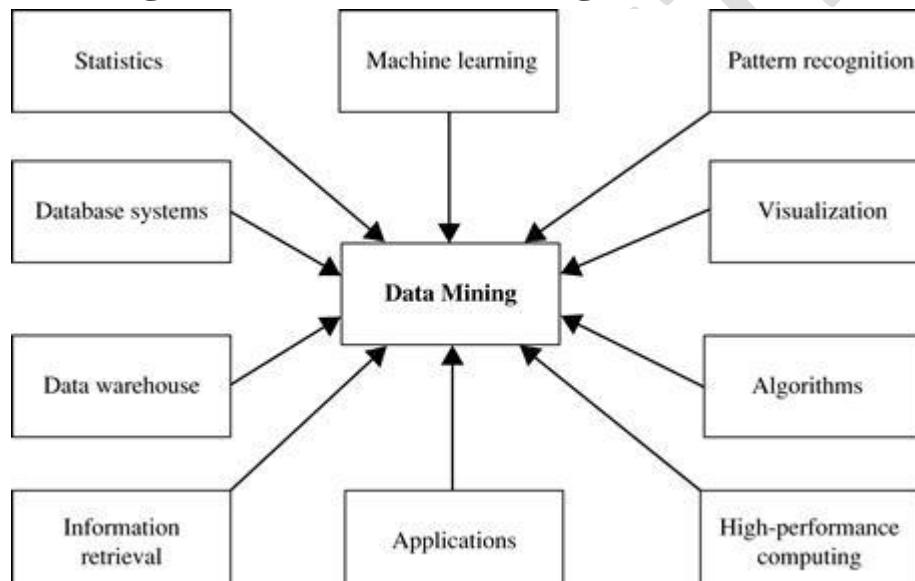
Data Mining Task Primitives

- We can specify a data mining task in the form of a data mining query.
- This query is input to the system.
- A data mining query is defined in terms of data mining task primitives.

Note − These primitives allow us to communicate in an interactive manner with the data mining system. Here is the list of Data Mining Task Primitives −

- Set of task relevant data to be mined.
- Kind of knowledge to be mined.
- Background knowledge to be used in discovery process.
- Interestingness measures and thresholds for pattern evaluation.
- Representation for visualizing the discovered patterns.

Which Technologies are used in data mining?

1. Statistics:
- It uses the mathematical analysis to express representations, model and summarize empirical data or real world observations.
- Statistical analysis involves the collection of methods, applicable to large amount of data to conclude and report the trend.

2. Machine learning
- **Arthur Samuel** defined machine learning as a field of study that gives computers the ability to learn without being programmed.
- When the new data is entered in the computer, algorithms help the data to grow or change due to machine learning.
- In machine learning, an algorithm is constructed to predict the data from the available database **(Predictive analysis).**
- It is related to computational statistics.

The four types of machine learning are:

a. Supervised learning
- It is based on the classification.
- It is also called as **inductive learning**. In this method, the desired outputs are included in the training dataset.

b. Unsupervised learning
- Unsupervised learning is based on clustering. Clusters are formed on the basis of similarity measures and desired outputs are not included in the training dataset.

c. Semi-supervised learning
- Semi-supervised learning includes some desired outputs to the training dataset to generate the appropriate functions. This method generally avoids the large number of labeled examples (i.e. desired outputs).

d. Active learning
- Active learning is a powerful approach in analyzing the data efficiently.
- The algorithm is designed in such a way that, the desired output should be decided by the algorithm itself (the user plays important role in this type).

3. Information retrieval
- Information deals with uncertain representations of the semantics of objects (text, images).

For example: Finding relevant information from a large document.

4. Database systems and data warehouse
- Databases are used for the purpose of recording the data as well as data warehousing.
- Online Transactional Processing (OLTP) uses databases for day to day transaction purpose.
- Data warehouses are used to store historical data which helps to take strategically decision for business.
- It is used for online analytical processing (OALP), which helps to analyze the data.

5. Pattern Recognition:
Pattern recognition is the automated recognition of patterns and regularities in data. Pattern recognition is closely related to artificial intelligence and machine learning, together with applications such as data mining and knowledge discovery in databases (KDD), and is often used interchangeably with these terms.

6. Visualization:
It is the process of extracting and visualizing the data in a very clear and understandable way without any form of reading or writing by displaying the results in the form of pie charts, bar graphs, statistical representation and through graphical forms as well.

7. Algorithms:
To perform data mining techniques we have to design best algorithms.

8. High Performance Computing:
High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

Are all patterns interesting?

- Typically the answer is No − only small fraction of the patterns potentially generated would actually be of interest to a given user.
- What makes patterns interesting?
  - ➢ The answer is if it is (1) easily understood by humans, (2) valid on new or test data with some degree of certainty, (3) potentially useful, (4) novel.
- A Pattern is also interesting if it is validates a hypothesis that the user sought to confirm.

Data Mining Applications

Here is the list of areas where data mining is widely used −
  - ➢ Financial Data Analysis
  - ➢ Retail Industry
  - ➢ Telecommunication Industry
  - ➢ Biological Data Analysis
  - ➢ Other Scientific Applications
  - ➢ Intrusion Detection

Financial Data Analysis

The financial data in banking and financial industry is generally reliable and of high quality which facilitates systematic data analysis and data mining. Like,

- Loan payment prediction and customer credit policy analysis.
- Detection of money laundering and other financial crimes.

Retail Industry

Data Mining has its great application in Retail Industry because it collects large amount of data from on sales, customer purchasing history, goods transportation, consumption and services. It is natural that the quantity of data collected will continue to expand rapidly because of the increasing ease, availability and popularity of the web.

Telecommunication Industry

Today the telecommunication industry is one of the most emerging industries providing various services such as fax, pager, cellular phone, internet messenger, images, e-mail, web data transmission, etc. Due to the development of new computer and communication technologies, the telecommunication industry is rapidly expanding. This is the reason why data mining is become very important to help and understand the business.

Biological Data Analysis

In recent times, we have seen a tremendous growth in the field of biology such as genomics, proteomics, functional Genomics and biomedical research. Biological data mining is a very important part of Bioinformatics.

Other Scientific Applications

The applications discussed above tend to handle relatively small and homogeneous data sets for which the statistical techniques are appropriate. Huge amount of data have been collected from scientific domains such as geosciences, astronomy, etc.

A large amount of data sets is being generated because of the fast numerical simulations in various fields such as climate and ecosystem modeling, chemical engineering, fluid dynamics, etc.

Intrusion Detection

Intrusion refers to any kind of action that threatens integrity, confidentiality, or the availability of network resources. In this world of connectivity, security has become the major issue. With increased usage of internet and availability of the tools and tricks for intruding and attacking network prompted intrusion detection to become a critical component of network administration.

Major Issues in data mining:

Data mining is a dynamic and fast-expanding field with great strengths. The major issues can divided into five groups:

a) Mining Methodology
b) User Interaction
c) Efficiency and scalability
d) Diverse Data Types Issues
e) Data mining society

a) Mining Methodology:

It refers to the following kinds of issues –

- **Mining different kinds of knowledge in databases** – Different users may be interested in different kinds of knowledge. Therefore it is necessary for data mining to cover a broad range of knowledge discovery task.

- **Mining knowledge in multidimensional space** – when searching for knowledge in large datasets, we can explore the data in multidimensional space.

- **Handling noisy or incomplete data** – the data cleaning methods are required to handle the noise and incomplete objects while mining the data regularities. If the data cleaning methods are not there then the accuracy of the discovered patterns will be poor.

- **Pattern evaluation** – the patterns discovered should be interesting because either they represent common knowledge or lack novelty.

b) User Interaction:

- **Interactive mining of knowledge at multiple levels of abstraction** – The data mining process needs to be interactive because it allows users to focus the search for patterns, providing and refining data mining requests based on the returned results.

- **Incorporation of background knowledge** – To guide discovery process and to express the discovered patterns, the background knowledge can be used. Background knowledge may be used to express the discovered patterns not only in concise terms but at multiple levels of abstraction.

- **Data mining query languages and ad hoc data mining** – Data Mining Query language that allows the user to describe ad hoc mining tasks, should be integrated with a data warehouse query language and optimized for efficient and flexible data mining.

- **Presentation and visualization of data mining results** – Once the patterns are discovered it needs to be expressed in high level languages, and visual representations. These representations should be easily understandable.

c) Efficiency and scalability

There can be performance-related issues such as follows −

- **Efficiency and scalability of data mining algorithms** − In order to effectively extract the information from huge amount of data in databases, data mining algorithm must be efficient and scalable.
- **Parallel, distributed, and incremental mining algorithms** − The factors such as huge size of databases, wide distribution of data, and complexity of data mining methods motivate the development of parallel and distributed data mining algorithms. These algorithms divide the data into partitions which is further processed in a parallel fashion. Then the results from the partitions is merged. The incremental algorithms, update databases without mining the data again from scratch.

d) Diverse Data Types Issues

- **Handling of relational and complex types of data** − The database may contain complex data objects, multimedia data objects, spatial data, temporal data etc. It is not possible for one system to mine all these kind of data.
- **Mining information from heterogeneous databases and global information systems** − The data is available at different data sources on LAN or WAN. These data source may be structured, semi structured or unstructured. Therefore mining the knowledge from them adds challenges to data mining.

e) Data Mining and Society

- **Social impacts of data mining –** With data mining penetrating our everyday lives, it is important to study the impact of data mining on society.
- **Privacy-preserving data mining –** data mining will help scientific discovery, business management, economy recovery, and security protection.
- **Invisible data mining –** we cannot expect everyone in society to learn and master data mining techniques. More and more systems should have data mining functions built within so that people can perform data mining or use data mining results simply by mouse clicking, without any knowledge of data mining algorithms.

Data Objects and Attribute Types:

Data Object:

An Object is real time entity.

Attribute:

It can be seen as a data field that represents characteristics or features of a data object. For a customer object attributes can be customer Id, address etc. The attribute types can represented as follows—

1. **Nominal Attributes – related to names:** The values of a Nominal attribute are name of things, some kind of symbols. Values of Nominal attributes represent some category or state and that's why nominal attribute also referred as categorical attributes.

    Example:

Attribute	Values
Colors	Black, Green, Brown, red

2. **Binary Attributes:** Binary data has only 2 values/states. For Example yes or no, affected or unaffected, true or false.

   i) Symmetric: Both values are equally important (Gender).

   ii) Asymmetric: Both values are not equally important (Result).

Attribute	Values
Gender	Male, Female

Attribute	Values
Result	Pass, Fail

3. **Ordinal Attributes:** The Ordinal Attributes contains values that have a meaningful sequence or ranking (order) between them, but the magnitude between values is not actually known, the order of values that shows what is important but don't indicate how important it is.

Attribute	Values
Grade	O, S, A, B, C, D, F

4. **Numeric:** A numeric attribute is quantitative because, it is a measurable quantity, represented in integer or real values. Numerical attributes are of 2 types.

   i. An **interval-scaled** attribute has values, whose differences are interpretable, but the numerical attributes do not have the correct reference point or we can call zero point. Data can be added and subtracted at interval scale but cannot be multiplied or divided. Consider an example of temperature in degrees Centigrade. If a day's temperature of one day is twice than the other day we cannot say that one day is twice as hot as another day.

   ii. A **ratio-scaled** attribute is a numeric attribute with a fix zero-point. If a measurement is ratio-scaled, we can say of a value as being a multiple (or ratio) of another value. The values are ordered, and we can also compute the difference between values, and the mean, median, mode, Quantile-range and five number summaries can be given.

5. **Discrete:** Discrete data have finite values it can be numerical and can also be in categorical form. These attributes has finite or countably infinite set of values.

   Example

Attribute	Values
Profession	Teacher, Business man, Peon
ZIP Code	521157, 521301

6. **Continuous:** Continuous data have infinite no of states. Continuous data is of float type. There can be many values between 2 and 3.

   Example:

Attribute	Values
Height	5.4, 5.7, 6.2, etc.,
Weight	50, 65, 70, 73, etc.,

Basic Statistical Descriptions of Data:

Basic Statistical descriptions of data can be used to identify properties of the data and highlight which data values should be treated as noise or outliers.

1. Measuring the central Tendency:

There are many ways to measure the central tendency.

a) Mean: Let us consider the values are $x_1, x_2, x_3, \ldots\ldots, x_N$ be a set of N observed values or observations of X.

The Most common numeric measure of the "center" of a set of data is the *mean.*

$$\text{Mean} = \bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + x_3 + ----- + x_N}{N}$$

Sometimes, each values x_i in a set maybe associated with weight w_i for i=1, 2, …., N. then the mean is as follows

$$\text{Mean} = \bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{N} = \frac{w_1 x_1 + w_2 x_2 + w_3 x_3 + ----- + w_N x_N}{N}$$

This is called **weighted arithmetic mean** or **weighted average**.
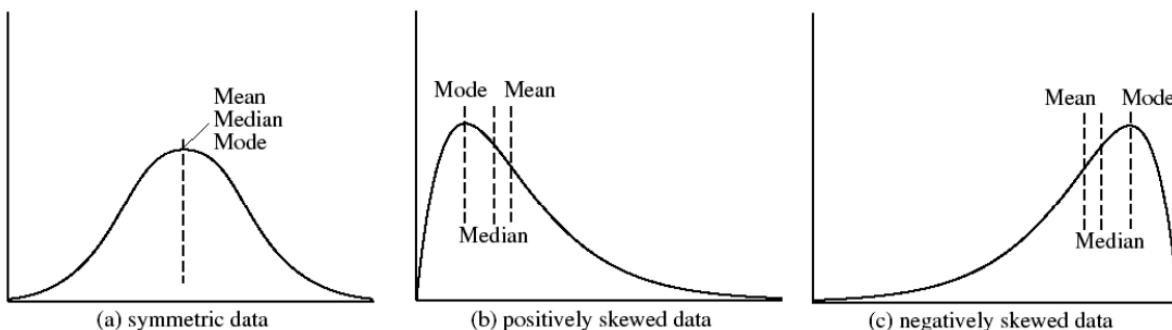
b) Median: Median is middle value among all values.

For N number of odd list median is $\frac{N+1}{2}$ th value.

For N number of even list median is $\frac{N+(N+1)}{2}$ th value.

c) Mode:

 ➢ The *mode* is another measure of central tendency.
 ➢ Datasets with one, two, or three modes are respectively called unimodal, bimodal, and trimodal.
 ➢ A dataset with two or more modes is multimodal. If each data occurs only once, then there is no mode.

(a) symmetric data   (b) positively skewed data   (c) negatively skewed data
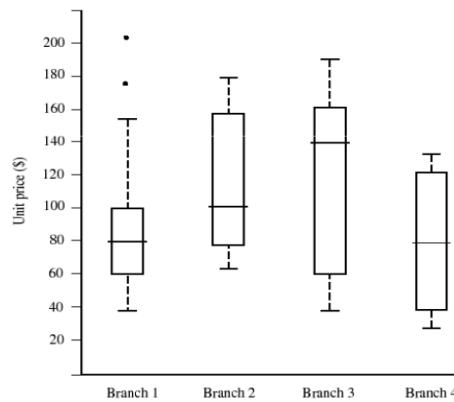
2. Measuring the Dispersion of data:

 ➢ The data can be measured as range, quantiles, quartiles, percentiles, and interquartile range.
 ➢ The kth percentile of a set of data in numerical order is the value x_i having the property that k percent of the data entries lay at or below x_i.
 ➢ The measures of data dispersion: Range, Five-number summary, Interquartile range (IQR), Variance and Standard deviation

Five Number Summary:

This contains five values Minimum, Q1 (25% value), Median, Q3 (75% value), and Maximum. These Five numbers are represented as Boxplot in graphical format.

➢ Boxplot is Data is represented with a box.
➢ The ends of the box are at the first and third quartiles, i.e., the height of the box is IRQ.
➢ The median is marked by a line within the box.
➢ Whiskers: two lines outside the box extend to Minimum and Maximum.
➢ To show outliers, the whiskers are extended to the extreme low and high observations only if these values are less than 1.5 * IQR beyond the quartiles.

Variance and Standard Deviation:

Let us consider the values are $x_1, x_2, x_3, \ldots\ldots., x_N$ be a set of N observed values or observations of X. The variance formula is

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

➢ Standard Deviation is the square root of variance σ^2.
➢ σ measures spread about the mean and should be used only when the mean is chosen as the measure of center.
➢ σ =0 only when there is no spread, that is, when all observations have the same value.

3. Graphical Displays of Basic Statistical data:

There are many types of graphs for the display of data summaries and distributions, such as:
➢ Bar charts
➢ Pie charts
➢ Line graphs
➢ Boxplot
➢ Histograms
➢ Quantile plots, Quantile - Quantile plots
➢ Scatter plots
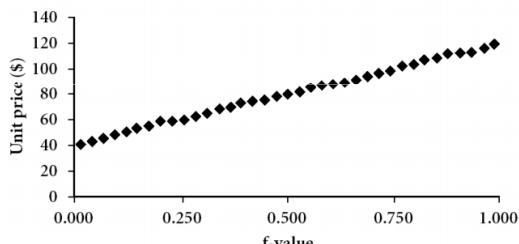
The data values can represent as Bar charts, pie charts, Line graphs, etc.

Quantile plots:
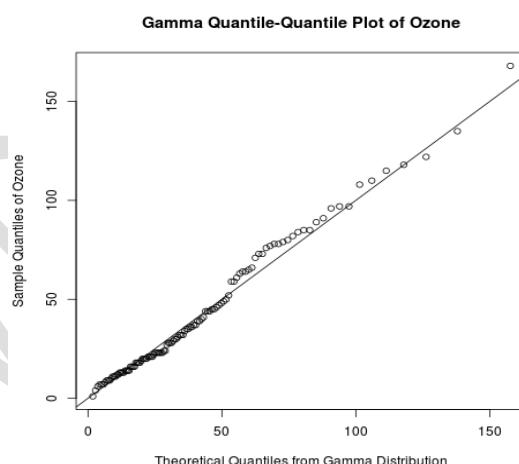
➢ A quantile plot is a simple and effective way to have a first look at a univariate data distribution.

➢ Plots quantile information

• For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i% of the data are below or equal to the value x_i

➢ Note that

• the 0.25 quantile corresponds to quartile Q1,

• the 0.50 quantile is the median, and

• the 0.75 quantile is Q3.

● A quantile plot for the unit price data of AllElectronics.

Quantile - Quantile plots:

In statistics, a Q-Q plot is a portability plot, which is a graphical method for comparing two portability distributions by plotting their quantiles against each other.


Gamma Quantile-Quantile Plot of Ozone
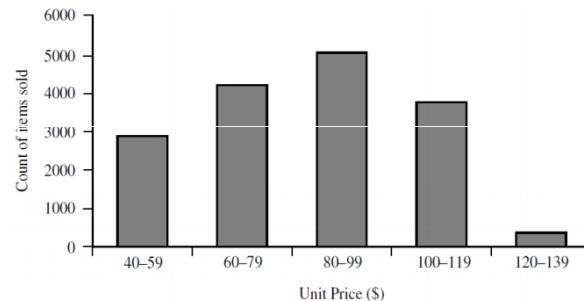
Histograms *or* **frequency histograms:**

➢ A univariate graphical method

➢ Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data

➢ If the attribute is categorical, then one rectangle is drawn for each known value of A, and the resulting graph is more commonly referred to as a bar chart.

➢ If the attribute is numeric, the term histogram is preferred.

● **Example:** A set of unit price data for items sold at a branch of *AllElectronics*
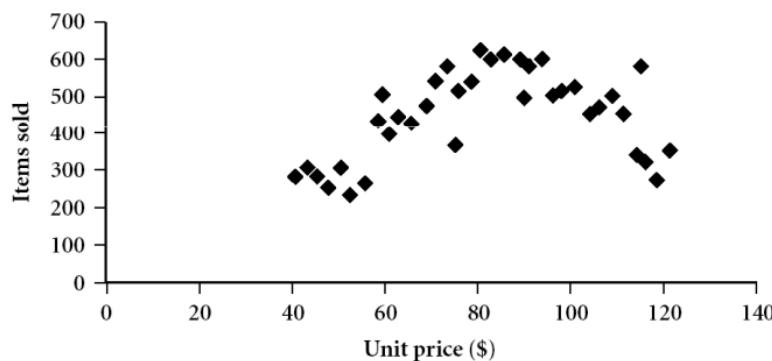
| Unit price ($)	Count of items sold
40	275
43	300
47	250
..	..
74	360
75	515
78	540
..	..
115	320
117	270
120	350

● **Example: A histogram**

Scatter Plot:
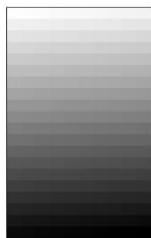
➤ Scatter plot
- Is one of the most effective graphical methods for determining if there appears to be a relationship, clusters of points, or outliers between two numerical attributes.

➤ Each pair of values is treated as a pair of coordinates and plotted as points in the plane

● A scatter plot for the data set of AllElectronics.
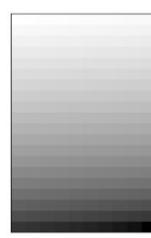
Data Visualization:

Visualization is the use of computer graphics to create visual images which aid in the understanding of complex, often massive representations of data.
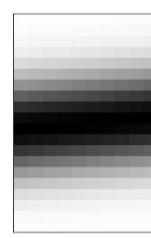
⊙ Categorization of visualization methods:
   - a) Pixel-oriented visualization techniques
   - b) Geometric projection visualization techniques
   - c) Icon-based visualization techniques
   - d) Hierarchical visualization techniques
   - e) Visualizing complex data and relations

a) Pixel-oriented visualization techniques

➤ For a data set of m dimensions, create m windows on the screen, one for each dimension

➤ The m dimension values of a record are mapped to m pixels at the corresponding positions in the windows

➤ The colors of the pixels reflect the corresponding values
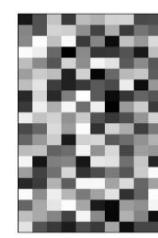
(a) Income   (b) Credit Limit   (c) transaction volume   (d) age
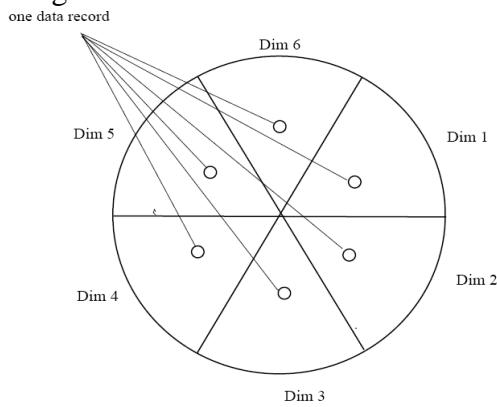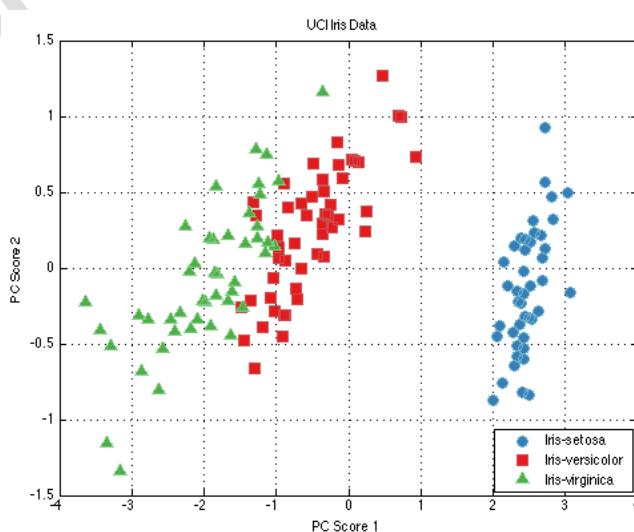
> To save space and show the connections among multiple dimensions, space filling is often done in a circle segment
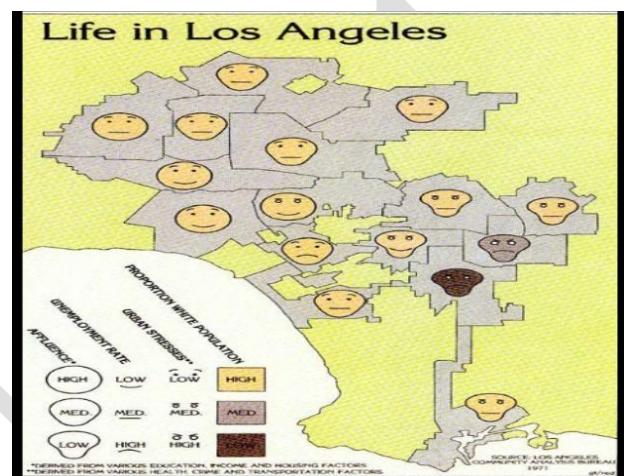

b) **Geometric projection visualization techniques**

 ⬥ Visualization of geometric transformations and projections of the data
  ⬥ Methods
    ➢ Direct visualization
    ➢ Scatterplot and scatterplot matrices
    ➢ Landscapes
    ➢ Projection pursuit technique: Help users find meaningful projections of multidimensional data
   ➢ Prosection views
    ➢ Hyperslice
    ➢ Parallel coordinates
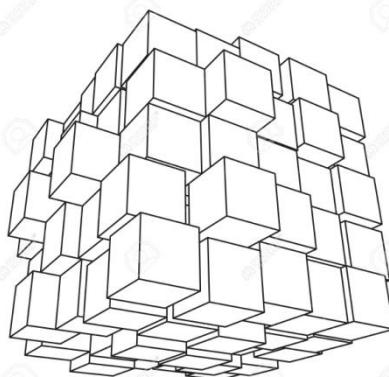
c) Icon-based visualization techniques
- Visualization of the data values as features of icons
- Typical visualization methods
  - Chernoff Faces
  - Stick Figures
- General techniques
  - Shape coding: Use shape to represent certain information encoding
  - Color icons: Use color icons to encode more information
  - Tile bars: Use small icons to represent the relevant feature vectors in document retrieval
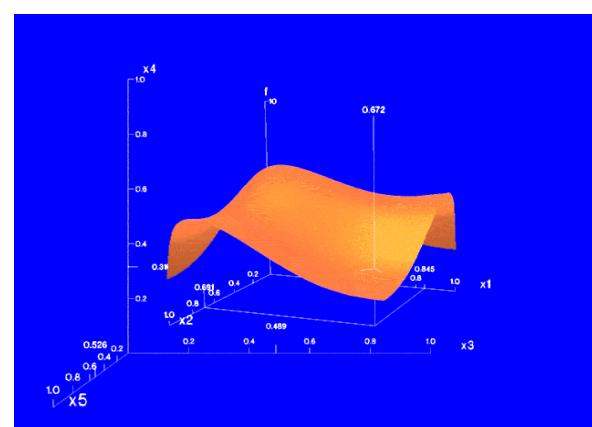
**d) Hierarchical visualization techniques**
- Visualization of the data using a hierarchical partitioning into subspaces
- Methods
  - Dimensional Stacking
  - Worlds-within-Worlds
  - Tree-Map
  - Cone Trees
  - InfoCube

Info Cube   **Worlds-within-worlds**

e) **Visualizing complex data and relations**
   + Visualizing non-numerical data: text and social networks
   + **Tag cloud:** visualizing user-generated tags
     ➢ The importance of tag is represented by font size/color
   + Besides text data, there are also methods to visualize relationships, such as visualizing

Similarity and Dissimilarity

Distance or similarity measures are essential to solve many pattern recognition problems such as classification and clustering. Various distance/similarity measures are available in literature to compare two data distributions. As the names suggest, a similarity measures how close two distributions are. For multivariate data complex summary methods are developed to answer this question.

Similarity Measure
   ▪ Numerical measure of how alike two data objects are.
   ▪ Often falls between 0 (no similarity) and 1 (complete similarity).

Dissimilarity Measure
   ▪ Numerical measure of how different two data objects are.
   ▪ Range from 0 (objects are alike) to ∞ (objects are different).

Proximity refers to a similarity or dissimilarity.

Similarity/Dissimilarity for Simple Attributes

Here, *p* and *q* are the attribute values for two data objects.

Attribute Type	Similarity	Dissimilarity
Nominal	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$
Ordinal	$s = 1 - \frac{\|p-q\|}{n-1}$ (values mapped to integer 0 to n-1, where n is the number of values)	$d = \frac{\|p-q\|}{n-1}$
Interval or Ratio	$s = 1 - \|p - q\|, s = \frac{1}{1+\|p-q\|}$	$d = \|p - q\|$

Common Properties of Dissimilarity Measures

Distance, such as the Euclidean distance, is a dissimilarity measure and has some well known properties:

1. $d(p, q) \geq 0$ for all p and q, and $d(p, q) = 0$ if and only if $p = q$,
2. $d(p, q) = d(q,p)$ for all p and q,
3. $d(p, r) \leq d(p, q) + d(q, r)$ for all p, q, and r, where $d(p, q)$ is the distance (dissimilarity) between points (data objects), p and q.

A distance that satisfies these properties are called a **metric**. Following is a list of several common distance measures to compare multivariate data. We will assume that the attributes are all continuous.

a) Euclidean Distance

Assume that we have measurements x_{ik}, $i = 1, \ldots, N$, on variables $k = 1, \ldots, p$ (also called attributes).

The Euclidean distance between the *i*th and *j*th objects is

$$d_E(i, j) = \left(\sum_{k=1}^{p} (x_{ik} - x_{jk})^2 \right)^{\frac{1}{2}}$$

for every pair (i, j) of observations.

The weighted Euclidean distance is

$$d_{WE}(i, j) = \left(\sum_{k=1}^{p} W_k (x_{ik} - x_{jk})^2 \right)^{\frac{1}{2}}$$

If scales of the attributes differ substantially, standardization is necessary.

b) Minkowski Distance

The Minkowski distance is a generalization of the Euclidean distance.

With the measurement, x_{ik}, $i = 1, \ldots, N$, $k = 1, \ldots, p$, the Minkowski distance is

$$d_M(i, j) = \left(\sum_{k=1}^{p} |x_{ik} - x_{jk}|^{\lambda} \right)^{\frac{1}{\lambda}},$$

where $\lambda \geq 1$. It is also called the L_λ metric.

+ $\lambda = 1$: L_1 metric, Manhattan or City-block distance.
+ $\lambda = 2$: L_2 metric, Euclidean distance.
+ $\lambda \to \infty$: L_∞ metric, Supremum distance.

$$\lim \lambda \to \infty = \left(\sum_{k=1}^{p} |x_{ik} - x_{jk}|^{\lambda} \right)^{\frac{1}{\lambda}} = \max \left(|x_{i1} - x_{j1}|, \ldots, |x_{ip} - x_{jp}| \right)$$

Note that λ and p are two different parameters. Dimension of the data matrix remains finite.

c) Mahalanobis Distance

Let **X** be a N × p matrix. Then the ith row of **X** is

$$x_i^T = (x_{i1}, x_{i2}, \ldots, x_{ip})$$

The Mahalanobis distance is

$$d_{MH}(i,j) = \left((x_i - x_j)^T \Sigma^{-1} (x_i - x_j) \right)^{\frac{1}{2}}$$

where $\sum$ is the p×p sample covariance matrix.

Common Properties of Similarity Measures

Similarities have some well known properties:

1. $s(p, q) = 1$ (or maximum similarity) only if $p = q$,
2. $s(p, q) = s(q, p)$ for all p and q, where $s(p, q)$ is the similarity between data objects, p and q.

Similarity Between Two Binary Variables

The above similarity or distance measures are appropriate for continuous variables. However, for binary variables a different approach is necessary.

	q=1	q=0
p=1	$n_{1,1}$	$n_{1,0}$
p=0	$n_{0,1}$	$n_{0,0}$

Simple Matching and Jaccard Coefficients

- Simple matching coefficient = $(n_{1,1} + n_{0,0}) / (n_{1,1} + n_{1,0} + n_{0,1} + n_{0,0})$.
- Jaccard coefficient = $n_{1,1} / (n_{1,1} + n_{1,0} + n_{0,1})$.

DATA PREPROCESSING

1. Preprocessing
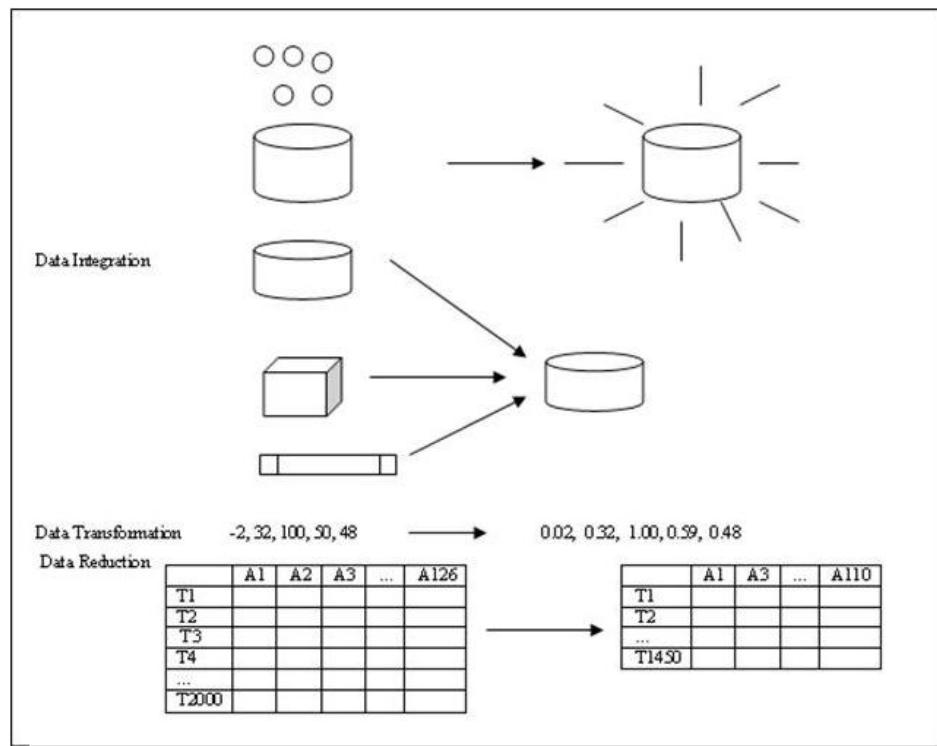
Real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results, so we prefer a preprocessing concepts.

Data Preprocessing Techniques

* **Data cleaning** can be applied to remove noise and correct inconsistencies in the data.
* **Data integration** merges data from multiple sources into coherent data store, such as a data warehouse.
* **Data reduction** can reduce the data size by aggregating, eliminating redundant features, orclustering, for instance. These techniques are not mutually exclusive; they may worktogether.
* **Data transformations**, such as normalization, may be applied.

Need for preprocessing

➢ Incomplete, noisy and inconsistent data are common place properties of large real world databases and data warehouses.

➢ Incomplete data can occur for a number of reasons:

- Attributes of interest may not always be available
- Relevant data may not be recorded due to misunderstanding, or because of equipment malfunctions.
- Data that were inconsistent with other recorded data may have been deleted.
- Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.
- The data collection instruments used may be faulty.
- There may have been human or computer errors occurring at data entry.
- Errors in data transmission can also occur.
- There may be technology limitations, such as limited buffer size for coordinating synchronized data transfer and consumption.
- Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- Data integration is the process of integrating multiple databases cubes or files. Yet some attributes representing a given may have different names in different databases, causing inconsistencies and redundancies.
- Data transformation is a kind of operations, such as normalization and aggregation, are additional data preprocessing procedures that would contribute toward the success of the mining process.
- Data reduction obtains a reduced representation of data set that is much smaller in volume, yet produces the same(or almost the same) analytical results.

2. DATA CLEANING

Real-world data tend to be incomplete, noisy, and inconsistent. Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers and correct inconsistencies in the data.

2.1 Missing Values

Many tuples have no recorded value for several attributes, such as customer income.so we can fill the missing values for this attributes.

The following methods are useful for performing missing values over several attributes:

1. **Ignore the tuple**: This is usually done when the class label missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of the missing values per attribute varies considerably.

2. **Fill in the missing values manually**: This approach is time –consuming and may not be feasible given a large data set with many missing values.

3. **Use a global constant to fill in the missing value**: Replace all missing attribute value by the same constant, such as a label like "unknown" or $-\infty$.

4. **Use the attribute mean to fill in the missing value**: For example, suppose that the average income of customers is \$56,000. Use this value to replace the missing value for income.

5. **Use the most probable value to fill in the missing value**: This may be determined with regression, inference-based tools using a Bayesian formalism or decision tree induction. For example, using the other customer attributes in the sets decision tree is constructed to predict the missing value for income.

2.2 *Noisy Data*

Noise is a random error or variance in a measured variable. Noise is removed using data smoothing techniques.

Binning: Binning methods smooth a sorted data value by consulting its "neighborhood," that is the value around it. The sorted values are distributed into a number of "buckets" or "bins". Because binning methods consult the neighborhood of values, they perform local smoothing.

Sorted data for price (in dollars): 3,7,14,19,23,24,31,33,38.

Example 1: Partition into (equal-frequency) bins:

Bin 1: 3,7,14

Bin 2: 19,23,24

Bin 3: 31,33,38

In the above method the data for price are first sorted and then partitioned into equal-frequency bins of size 3.

Smoothing by bin means:

Bin 1: 8,8,8

Bin 2: 22,22,22

Bin 3: 34,34,34

In smoothing by bin means method, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 3,7&14 in bin 1 is 8[(3+7+14)/3].

Smoothing by bin boundaries:

Bin 1: 3,3,14

Bin 2: 19,24,24

Bin 3: 31,31,38

In smoothing by bin boundaries, the maximum & minimum values in give bin or identify as the bin boundaries. Each bin value is then replaced by the closest boundary value.

In general, the large the width, the greater the effect of the smoothing. Alternatively, bins may be equal-width, where the interval range of values in each bin is constant Example 2: Remove the noise in the following data using smoothing techniques:

8, 4,9,21,25,24,29,26,28,15

Sorted data for price (in dollars):4,8,9,15,21,21,24,25,26,28,29,34

Partition into equal-frequency (equi-depth) bins:

Bin 1: 4, 8,9,15

Bin 2: 21,21,24,25

Bin 3: 26,28,29,34

Smoothing by bin means:

Bin 1: 9,9,9,9

Bin 2: 23,23,23,23

Bin 3: 29,29,29,29

Smoothing by bin boundaries:
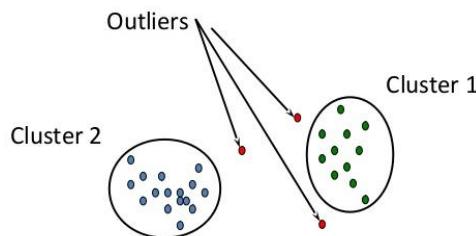
Bin 1: 4, 4,4,15

Bin 2: 21,21,25,25

Bin3: 26,26,26,34

Regression: Data can be smoothed by fitting the data to function, such as with regression. Linear regression involves finding the "best" line to fit two attributes (or variables), so that one attribute can be used to predict the other. Multiple linear regressions is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

Clustering: Outliers may be detected by clustering, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers.

2.3 Inconsistent Data

Inconsistencies exist in the data stored in the transaction. Inconsistencies occur due to occur during data entry, functional dependencies between attributes and missing values. The inconsistencies can be detected and corrected either by manually or by knowledge engineering tools.

Data cleaning as **a** *process*

   a) Discrepancy detection
   b) Data transformations

a) *Discrepancy detection*

The first step in data cleaning is discrepancy detection. It considers the knowledge ofmeta data and examines the following rules for detecting the discrepancy.

Unique rules- each value of the given attribute must be different from all other values for that attribute.

Consecutive rules – Implies no missing values between the lowest and highest values for the attribute and that all values must also be unique.

Null rules - specifies the use of blanks, question marks, special characters, or other strings that may indicates the null condition

Discrepancy detection Tools:

   ❖ Data scrubbing tools - use simple domain knowledge (e.g., knowledge of postal addresses, and spell-checking) to detect errors and make corrections in the data
   ❖ Data auditing tools – analyzes the data to discover rules and relationship, and detecting data that violate such conditions.

b) *Data transformations*

This is the second step in data cleaning as a process. After detecting discrepancies, we need to define and apply (a series of) transformations to correct them.

Data Transformations Tools:

   ❖ Data migration tools – allows simple transformation to be specified, such to replaced the string "gender" by "sex".
   ❖ ETL (Extraction/Transformation/Loading) tools – allows users to specific transforms through a graphical user interface(GUI)

3. Data Integration

Data mining often requires data integration - the merging of data from stores into a coherent data store, as in data warehousing. These sources may include multiple data bases, data cubes, or flat files.

Issues in Data Integration

a) Schema integration & object matching.
b) Redundancy.
c) Detection & Resolution of data value conflict

a) Schema Integration & Object Matching

Schema integration & object matching can be tricky because same entity can be represented in different forms in different tables. This is referred to as the entity identification problem. Metadata can be used to help avoid errors in schema integration. The meta data may also be used to help transform the data.

b) Redundancy:

Redundancy is another important issue an attribute (such as *annual revenue*, for instance) may be redundant if it can be "derived" from another attribute are set of attributes. Inconsistencies in attribute of dimension naming can also cause redundancies in the resulting data set. Some redundancies can be detected by correlation analysis and covariance analysis.

For Nominal data, we use the χ^2 (Chi-Square) test.

For Numeric attributes we can use the correlation coefficient and covariance.

χ^2 Correlation analysis for numerical data:

For nominal data, a correlation relationship between two attributes, A and B, can be discovered by a χ^2 (Chi-Square) test. Suppose A has c distinct values, namely a_1, a_2, a_3, ..., a_c. B has r distinct values, namely b_1, b_2, b_3, ..., b_r. The data tuples are described by table.

The χ^2 value is computed as

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

Where o_{ij} is the observed frequency of the joint event (A_i, B_j) and e_{ij} is the expected frequency of (A_i, B_j), which can computed as

$$e_{ij} = \frac{count(A=a_i) X count(B=b_j)}{n}$$

For Example,

	Male	Female	Total
Fiction	250	200	450
Non_Fiction	50	1000	1050
Total	300	1200	1500

$$e_{11} = \frac{count\ (male\ ) X count\ (fiction\ )}{n} = \frac{300 X 450}{1500} = 90$$

$$e_{12} = \frac{count\ (male\ ) X count\ (non\_fiction\ )}{n} = \frac{300 X 1050}{1500} = 210$$

$$e_{21} = \frac{count\ (female\ ) X count\ (fiction\ )}{n} = \frac{1200 X 450}{1500} = 360$$

$$e_{22} = \frac{count\ (female\ ) X count\ (non\_fiction\ )}{n} = \frac{1200 X 1050}{1500} = 840$$

	Male	**Female**	**Total**
Fiction	250 (90)	200 (360)	450
Non_Fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

For χ2 computation, we get

$$\chi 2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840}$$
$$= 284.44 + 121.90 + 71.11 + 30.48 = 507.93$$

For this 2 X 2 table, the degrees of freedom are (2-1(2-1)=1. For 1 degree of freedom, the x2 value needed to reject the hypothesis at the 0.001 significance level is 10.828(from statistics table). Since our computed value is greater than this, we can conclude that two attributes are strongly correlated for the given group of people.

Correlation Coefficient for Numeric data:

For Numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the correlation coefficient. This is

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \overline{A})(b_i - \overline{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_ib_i) - n\overline{A}\overline{B}}{n\sigma_A\sigma_B}$$

For Covariance between A and B defined as

$$Cov(A,B) = \frac{\sum_{i=1}^{n}(a_i - \overline{A})(b_i - \overline{B})}{n}$$

c) Detection and Resolution of Data Value Conflicts.

A third important issue in data integration is the *detection and resolution of data valueconflicts*. For example, for the same real–world entity, attribute value from differentsources may differ. This may be due to difference in representation, scaling, or encoding.

For instance, a weight attribute may be stored in metric units in one system and British imperial units in another. For a hotel chain, the *price* of rooms in different cities may involve not only different currencies but also different services (such as free breakfast) and taxes. An attribute in one system may be recorded at a lower level of abstraction than the "same" attribute in another.

Careful integration of the data from multiple sources can help to reduce and avoid redundancies and inconsistencies in the resulting data set. This can help to improve the accuracy and speed of the subsequent of mining process.

4. Data Reduction:

Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results.
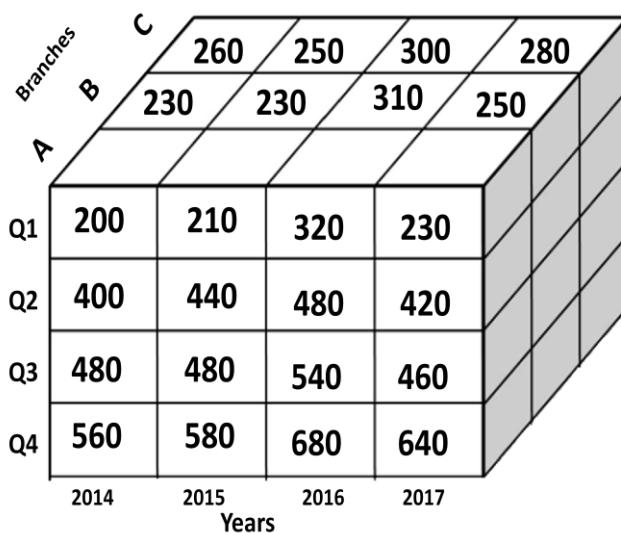
Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.

Data reduction strategies
    4.1.Data cube aggregation
    4.2.Attribute Subset Selection
    4.3.Numerosity reduction — e.g., fit data into models
    4.4.Dimensionality reduction - Data Compression
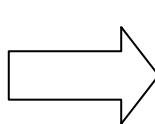
4.1 Data cube aggregation:

    For example, the data consists of AllElectronics sales per quarter for the years 2014 to 2017.You are, however, interested in the annual sales, rather than the total per quarter. Thus, the data can be *aggregated*so that the resulting data summarize the total sales per year instead of per quarter.

Year/Quarter	2014	2015	2016	2017
Quarter 1	200	210	320	230
Quarter 2	400	440	480	420
Quarter 3	480	480	540	460
Quarter 4	560	580	680	640

Year	Sales
2014	1640
2015	1710
2016	2020
2017	1750

4.2 Attribute Subset Selection

    Attribute subset selection reduces the data set size by removing irrelevant or redundant attributes (or dimensions). The goal of attribute subset selection is to find a minimum set of attributes. It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.

    For *n* attributes, there are 2*n* possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as *n* and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching to attribute space, they always make what looks to be the best choice at that time. Their strategy to make a locally optimal choice in the hope that this will lead to a

globally optimal solution. Many other attributes evaluation measure can be used, such as the information gain measure used in building decision trees for classification.

Initial attribute set: {A1, A2, A3, A4, A5, A6} Initial Reduced Set: ➢ { } ➢ {A1 } ➢ {A1, A4 } ➢ {A1, A4, A6 } Reduced Attribute Set: { A1, A4, A6 }	Initial attribute set: {A1, A2, A3, A4, A5, A6} {A1, A2, A3, A4, A5, A6} {A1, A3, A4, A5, A6} {A1, A4, A5, A6} {A1, A4, A6} Reduced Attribute Set: { A1, A4, A6 }	Initial attribute set: {A1, A2, A3, A4, A5, A6} A4? A1? A6? Class 1 Class 2 Class 1 Class 2 Reduced Attribute Set: { A1, A4, A6 }

Techniques for heuristic methods of attribute sub set selection
   ➢ Stepwise forward selection
   ➢ Stepwise backward elimination
   ➢ Combination of forward selection and backward elimination
   ➢ Decision tree induction

1. *Stepwise forward selection*: The procedure starts with an empty set of attributes as the reduced set. The best of original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.
2. *Stepwise backward elimination*: The procedure starts with full set of attributes. At each step, it removes the worst attribute remaining in the set.
3. *Combination of forward selection and backward elimination*: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
4. *Decision tree induction*: Decision tree induction constructs a flowchart like structure where each internal node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each leaf node denotes a class prediction. At each node, the algorithm choices the "best" attribute to partition the data into individual classes. A tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree from the reduced subset of attributes. Threshold measure is used as stopping criteria.

4.3Numerosity Reduction:

Numerosity reduction is used to reduce the data volume by choosing alternative, smaller forms of the data representation
Techniques for Numerosity reduction:
   ➢ Parametric - In this model only the data parameters need to be stored, instead of the actual data. (e.g.,) Log-linear models, Regression

> Nonparametric – This method stores reduced representations of data include histograms, clustering, and sampling

Parametric model
1. Regression
- **Linear regression**
  - In linear regression, the data are model to fit a straight line. For example, a random variable, Y called a response variable), can be modeled as a linear function of another random variable, X called a predictor variable), with the equation $Y = \alpha X + \beta$
  - Where the variance of Y is assumed to be constant. The coefficients, α and β (called regression coefficients), specify the slope of the line and the Y- intercept, respectively.
- **Multiple- linear regression**
  - Multiple linear regression is an extension of (simple) linear regression, allowing a response variable Y, to be modeled as a linear function of two or more predictor variables.

2. Log-Linear Models
  - Log-Linear Models can be used to estimate the probability of each point in a multidimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations.

Nonparametric Model
1. Histograms

A histogram for an attribute A partitions the data distribution of A into disjoint subsets, or buckets. If each bucket represents only a single attribute-value/frequency pair, the buckets are called singleton buckets.

Ex: The following data are bast of prices of commonly sold items at All Electronics. The numbers have been sorted:

1,1,5,5,5,5,5,8,8,10,10,10,10,12,14,14,14,15,15,15,15,15,18,18,18,18,18,18,18,18,20,20,20,20,20,20,21,21,21,21,21,25,25,25,25,25,28,28,30,30,30
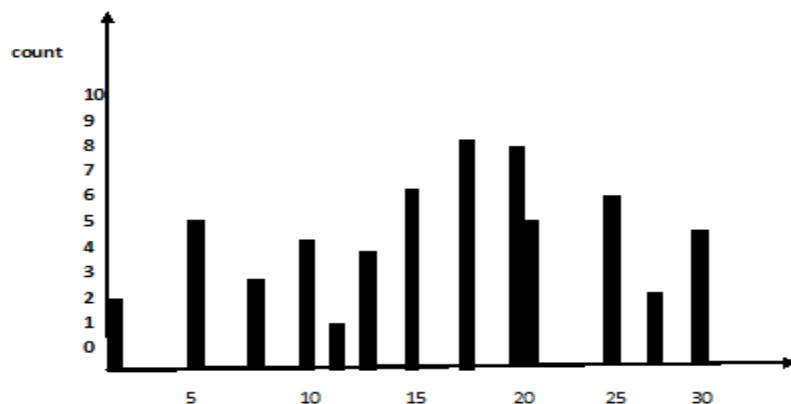

Figure 3.7 A Histogram for price using Singleton Buckets

There are several partitioning rules including the following:
Equal-width: The width of each bucket range is uniform

- (Equal-frequency (or equi-depth): the frequency of each bucket is constant
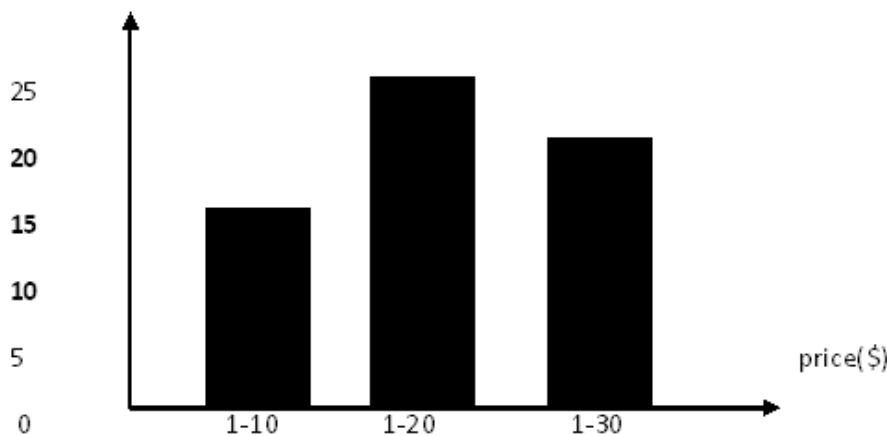
Figure.2.8 An equal-width histogram for price, where values are aggregated so **that each bucket has a uniform width of $10.**

2. Clustering

Clustering technique consider data tuples as objects. They partition the objects into groups or clusters, so that objects within a cluster are similar to one another and dissimilar to objects in other clusters. Similarity is defined in terms of how close the objects are in space, based on a distance function. The quality of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster. Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid.

3. Sampling:

Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random sample (or subset) of the data. Suppose that a large data set D, contains N tuples, then the possible samples are Simple Random sample without Replacement (SRS WOR) of size n: This is created by drawing „n" of the „N" tuples from D (n<N), where the probability of drawing any tuple in D is 1/N, i.e., all tuples are equally likely to be sampled.

Figure 2.9. Sampling can be used for data reduction.

4.4 Dimensionality Reduction:

In dimensionality reduction, data encoding or transformations are applied so as to obtained reduced or "compressed" representation of the oriental data.

Dimension Reduction Types

➢ Lossless - If the original data can be *reconstructed* from the compressed data without any loss of information

➢ Lossy - If the original data can be reconstructed from the compressed data with loss of information, then the data reduction is called lossy.

*Effective methods in lossy dimensional reduction*
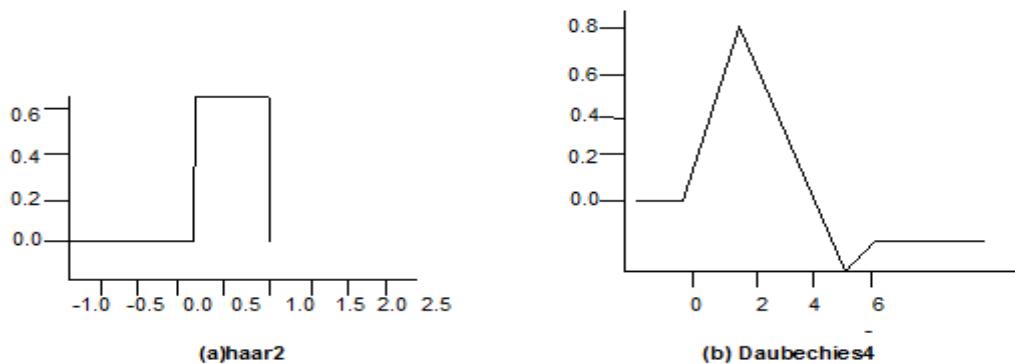
a) Wavelet transforms

b) Principal components analysis.

a) Wavelet transforms:

The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector, transforms it to a numerically different vector, of wavelet coefficients. The two vectors are of the same length. When applying this technique to data reduction, we consider each tuple as an n-dimensional data vector, that is, $X=(x_1, x_2, \ldots \ldots, x_n)$, depicting n measurements made on the tuple from n database attributes.

For example, all wavelet coefficients larger than some user-specified threshold can be retained. All other coefficients are set to 0. The resulting data representation is therefore very sparse, so that can take advantage of data sparsity are computationally very fast if performed in wavelet space.

The numbers next to a wave let name is the number of vanishing moment of the wavelet this is a set of mathematical relationships that the coefficient must satisfy and is related to number of coefficients.

(a)haar2   (b) Daubechies4

1. The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros as necessary (L >=n).

2. Each transform involves applying two functions
    *   The first applies some data smoothing, such as a sum or weighted average.
    *   The second performs a weighted difference, which acts to bring out the detailed features of data.

3. The two functions are applied to pairs of data points in X, that is, to all pairs of measurements (X_{2i}, X_{2i+1}). This results in two sets of data of length *L/2*. In general,

these represent a smoothed or low-frequency version of the input data and high frequency content of it, respectively.
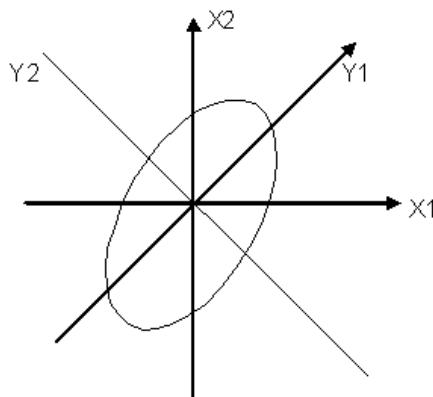
4. The two functions are recursively applied to the sets of data obtained in the previous loop, until the resulting data sets obtained are of length 2.

b) Principal components analysis

Suppose that the data to be reduced, which Karhunen-Loeve, K-L, method consists of tuples or data vectors describe by n attributes or dimensions. Principal components analysis, or PCA (also called the Karhunen-Loeve, or K-L, method), searches for k n-dimensional orthogonal vectors that can best be used to represent the data where $k<=n$. PCA combines the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

The basic procedure is as follows:

- The input data are normalized.
- PCA computes k orthonormal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others.
- The principal components are sorted in order of decreasing significance or strength.

In the above figure, Y1 and Y2, for the given set of data originally mapped to the axes X1 and X2. This information helps identify groups or patterns within the data. The sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on.

- The size of the data can be reduced by eliminating the weaker components.

Advantage of PCA
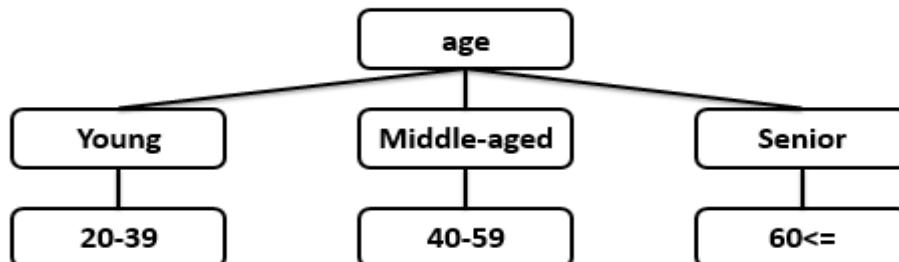
- PCA is computationally inexpensive
- Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions.
- Principal components may be used as inputs to multiple regression and cluster analysis.

5. Data Transformation and Discretization

Data transformation is the process of converting data from one format or structure into another format or structure.

In *data transformation*, the data are transformed or consolidated into forms appropriatefor mining. Strategies for data transformation include the following:

1. Smoothing, which works to remove noise fromthe data. Techniques include binning,regression, and clustering.

2. Attribute construction (or *feature construction*), where new attributes are constructedand added from the given set of attributes to help the mining process.

3. Aggregation, where summary or aggregation operations are applied to the data. Forexample, the daily sales data may be aggregated so as to compute monthly and annualtotal amounts. This step is typically used in constructing a data cube for data analysisat multiple abstraction levels.

4. Normalization, where the attribute data are scaled so as to fall within a smaller range,such as $\Box 1.0$ to 1.0, or 0.0 to 1.0.

5. Discretization, where the raw values of a numeric attribute (e.g., *age*) are replaced byinterval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth, adult, senior*).The labels, in turn, can be recursively organized into higher-level concepts, resultingin a *concept hierarchy* for the numeric attribute. Figure 3.12 shows a concept hierarchyfor the attribute *price*. More than one concept hierarchy can be defined for the sameattribute to accommodate the needs of various users.

6. Concept hierarchy generation for nominal data, where attributes such as *street* canbe generalized to higher-level concepts, like *city* or *country*. Many hierarchies fornominal attributes are implicit within the database schema and can be automaticallydefined at the schema definition level.

5.1 Data Transformation by Normalization:

The measurement unit used can affect the data analysis. For example, changing measurement units frommeters to inches for *height*, or fromkilograms to pounds for *weight*, may lead to very different results.

For distance-based methods, normalization helps prevent attributes with initially large ranges (e.g., *income*) from outweighing attributes with initially smaller ranges (e.g., binary attributes). It is also useful when given no prior knowledge of the data.

There are many methods for data normalization. We study *min-max normalization,z-score normalization,* and *normalization by decimal scaling.* For our discussion, let *A* bea numeric attribute with *n* observed values, $v_1, v_2, \ldots, v_n$.

a) **Min-max normalization** performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute, A. Min-max normalization maps a value, v_i, of A to v_i' in the range $[new\_min_A, new\_max_A]$ by computing

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original data range for A.

Example:-Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are $12,000 and $98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of $73,600 for *income* is transformed to

$$\frac{73,600 - 12,000}{98,000 - 12,000}(1.0 - 0) + 0 = 0.716.$$

b) **Z-Score Normalization**

The values for an attribute, A, are normalized based on the mean (i.e., average) and standard deviation of A. A value, v_i, of A is normalized to v_i' by computing

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A.

Example z-score normalization. Suppose that the mean and standard deviation of the values for the attribute income are $54,000 and $16,000, respectively. With z-score normalization, a value of $73,600 for income is transformed to

$$\frac{73,600 - 54,000}{16,000} = 1.225.$$

c) **Normalization by Decimal Scaling:**

Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A. A value, vi, of A is normalized to vi' by computing

$$v_i' = \frac{v_i}{10^j}$$

where j is the smallest integer such that $max(|v_i'|) < 1$.

Example Decimal scaling. Suppose that the recorded values of A range from -986 to 917. The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1000 (i.e., j = 3) so that -986 normalizes to -0.986 and 917 normalizes to 0.917.

5.2. Data Discretization

a) Discretization by binning:

Binning is a top-down splitting technique based on a specified number of bins.

Forexample, attribute values can be discretized by applying equal-width or equal-frequencybinning, and then replacing each bin value by the bin mean or median, as in *smoothingby bin means* or *smoothing by bin medians*, respectively. These techniques can be appliedrecursively to the resulting partitions to generate concept hierarchies.

b) Discretization by Histogram Analysis:

Like binning, histogram analysis is an unsupervised discretization technique because itdoes not use class information. A histogrampartitions the values of an attribute, A, into disjoint ranges called bucketsor bins.

In anequal-width histogram, for example, the values are partitioned into equal-size partitionsor ranges (e.g., for price, where each bucket has a width of $10).With an equal-frequency histogram, the values are partitioned so that, ideally, each partitioncontains the same number of data tuples.

c) Discretization by Cluster, Decision Tree,and Correlation Analyses

Cluster analysis is a popular data discretization method. A clustering algorithm canbe applied to discretize a numeric attribute, A, by partitioning the values of A into clustersor groups.

Techniques to generate decision trees for classification can be applied todiscretization. Such techniques employ a top-down splitting approach. Unlike the othermethods mentioned so far, decision tree approaches to discretization are supervised,that is, they make use of class label information.

5.3Concept Hierarchy Generation for Nominal Data

Nominal attributes have a finite (but possiblylarge) number of distinct values, with no ordering among the values. Examplesinclude geographic location, job category, and item type.

Manual definition of concept hierarchies can be a tedious and time-consuming taskfor a user or a domain expert. Fortunately, many hierarchies are implicit within thedatabase schema and can be automatically defined at the schema definition level. Theconcept hierarchies can be used to transform the data into multiple levels of granularity.
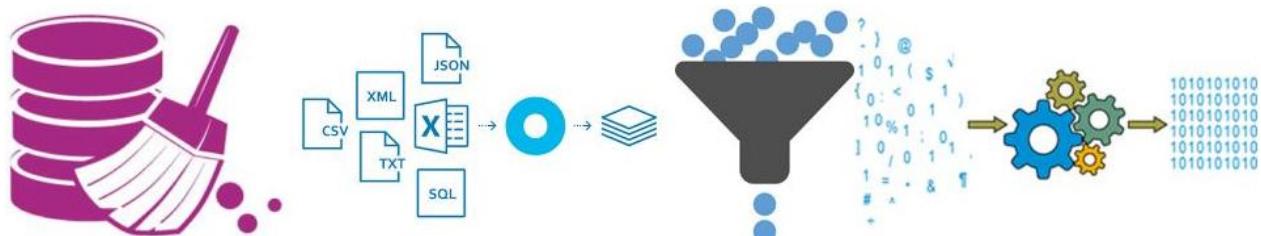
1. **Specification of a partial ordering of attributes explicitly at the schema level byusers or experts:**A user or expert can easily define a concept hierarchy byspecifying a partial or total ordering of the attributes at the schema level.
2. **Specification of a portion of a hierarchy by explicit data grouping:**In a large database,it is unrealistic to define an entire concept hierarchy by explicit value enumeration.For example, after specifying that province and countryform a hierarchy at the schema level, a user could define some intermediate levelsmanually.
3. **Specification of a set of attributes, but not of their partial ordering:** A user mayspecify a set of attributes forming a concept hierarchy, but omit to explicitly

statetheir partial ordering. The systemcan then try to automatically generate the attributeordering so as to construct a meaningful concept hierarchy.

4. **Specification of only a partial set of attributes:** Sometimes a user can be carelesswhen defining a hierarchy, or have only a vague idea about what should be includedin a hierarchy. Consequently, the user may have included only a small subset of therelevant attributes in the hierarchy specification.

✓ **Data cleaning** routines attempt to fill in missing values, smooth out noise whileidentifying outliers, and correct inconsistencies in the data.

✓ **Data integration** combines data from multiple sources to form a coherent datastore. The resolution of semantic heterogeneity, metadata, correlation analysis,tuple duplication detection, and data conflict detection contribute to smooth dataintegration.

✓ **Data reduction** techniques obtain a reduced representation of the data while minimizingthe loss of information content. These include methods of *dimensionalityreduction*, *numerosity reduction*, and *data compression*.

✓ **Data transformation** routines convert the data into appropriate forms for mining.For example, in **normalization**, attribute data are scaled so as to fall within asmall range such as 0.0 to 1.0. Other examples are **data discretization** and **concepthierarchy generation**.

✓ **Data discretization** transforms numeric data by mapping values to interval or conceptlabels. Such methods can be used to automatically generate *concept hierarchies*for the data, which allows for mining at multiple levels of granularity.

DATA CLASSIFICATION

Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statistics.

3.1 Why Classification?

A bank loans officer needs analysis of her data to learn which loan applicants are "safe" and which are "risky" for the bank. A marketing manager at AllElectronics needs data analysis to help guess whether a customer with a given profile will buy a new computer.

A medical researcher wants to analyze breast cancer data to predict which one of three specific treatments a patient should receive. In each of these examples, the data analysis task is classification, where a model or classifier is constructed to predict class (categorical) labels, such as "safe" or "risky" for the loan application data; "yes" or "no" for the marketing data; or "treatment A," "treatment B," or "treatment C" for the medical data.

Suppose that the marketing manager wants to predict how much a given customer will spend during a sale at AllElectronics. This data analysis task is an example of **numeric prediction**, where the model constructed predicts a continuous-valued function, or ordered value, as opposed to a class label. This model is a **predictor**.

Regression analysis is a statistical methodology that is most often used for numeric prediction; hence the two terms tend to be used synonymously, although other methods for numeric predictionexist. Classification and numeric prediction are the two major types of **prediction problems**.

3.2 General Approach for Classification:

Data classification is a two-step process, consisting of a*learning step* (where a classification model is constructed) and a *classification step* (wherethe model is used to predict class labels for given data).

- In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the **learning step** (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels.
- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- In the second step, the model is used for **classification**. First, the predictive accuracy of the classifier is estimated. If we were to use the training set to measure the classifier's accuracy, this estimate would likely be optimistic, because the classifier tends to overfit the data.
- Accuracy rate is the percentage of test set samples that are correctly classified by the model
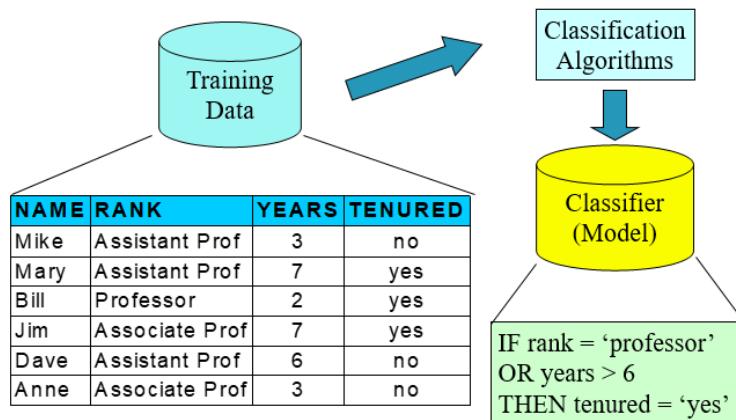
Fig: Learning Step
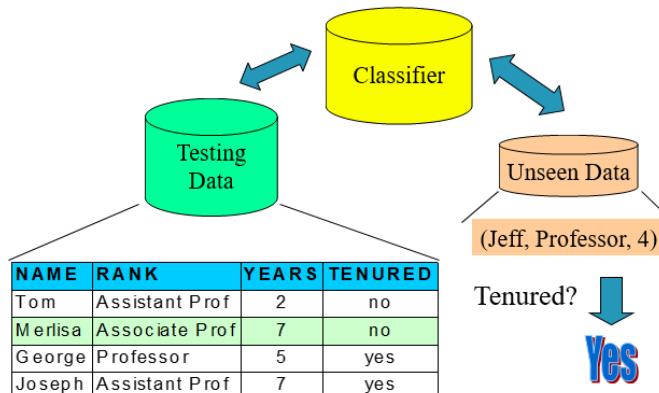
Fig: Classification Step

3.3 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** (non leaf 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 topmost node in a tree is the **root** node. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals.

"How are decision trees used for classification?" Given a tuple, *X*, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

"Why are decision tree classifiers so popular?" The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle multidimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast.

Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems.

During tree construction, *attribute selection measures* are used to select the attribute that best partitions the tuples into distinct classes. When decision trees are built, many of the branches may reflect noise or outliers in the training data. *Tree pruning* attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.

- ❖ During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as **ID3** (Iterative Dichotomiser).
- ❖ This work expanded on earlier work on *concept learning systems*, described by E. B. Hunt, J. Marin,and P. T. Stone. Quinlan later presented **C4.5 (a successor of ID3)**, which became a benchmark to which newer supervised learning algorithms are often compared.
- ❖ In 1984,a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) publishedthe book *Classification and Regression Trees* (**CART**), which described the generation ofbinary decision trees.

3.4 Decision Tree 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.
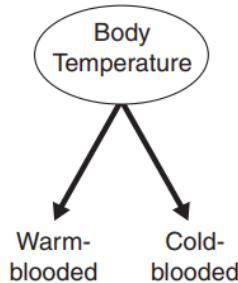
Method:
1) create a node *N*;
2) **if** tuples in *D* are all of the same class, *C*, **then**
3) return *N* as a leaf node labeled with the class *C*;
4) **if** *attribute list* is empty **then**
5) return *N* as a leaf node labeled with the majority class in *D*; // majority voting
6) apply **Attribute 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*;
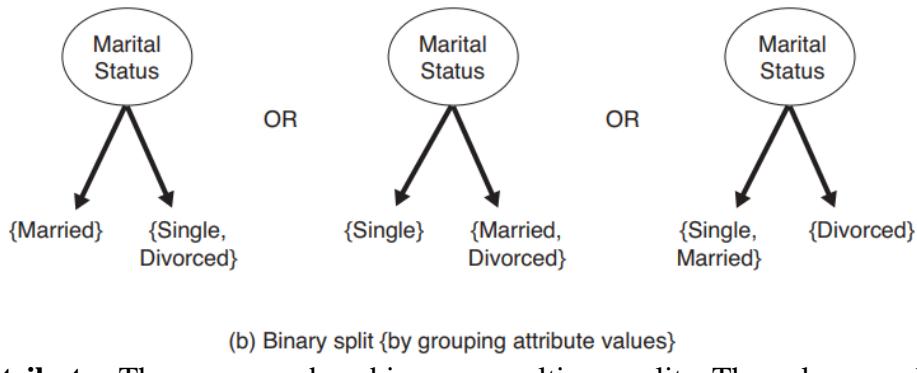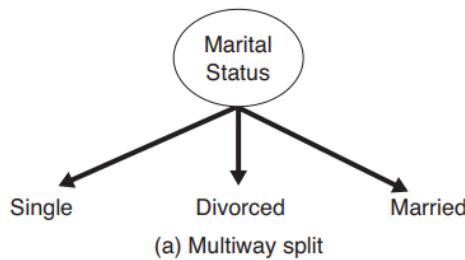
3.5Methods for selecting best test conditions

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.
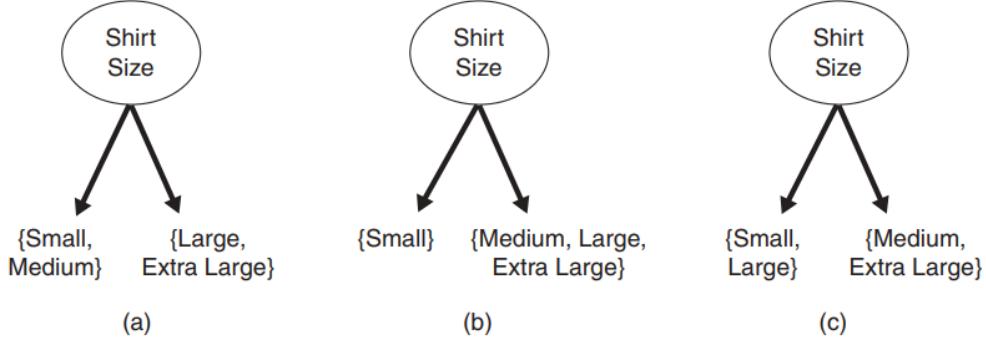
Binary Attributes: The test condition for a binary attribute generates two potential outcomes.

Nominal Attributes: These can have many values. These can be represented in two ways.

(a) Multiway split

(b) Binary split {by grouping attribute values}

Ordinal attributes: These can produce binary or multiway splits. The values can be grouped as long as the grouping does not violate the order property of attribute values.

3.6 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 (i.e., all the tuples that fall into a given partition would belong to the same class).

> Conceptually, the "best" splitting criterion is the one that most closely results in such a scenario. 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 measure4 is chosen as the splitting attribute for the given tuples.

> 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.

> There are three popular attribute selection measures—*information gain, gain ratio*, and *Gini index*.

3.6.1 Information Gain

ID3 uses **information gain** as its attribute selection measure. 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*. 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.

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

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

Where p_i is the nonzero probability that an arbitrary tuple in *D* belongs to class C_i and is estimated by $|C_i,D|/|D|$. A log function to the base 2 is used, because the information is encoded in bits. *Info(D)* is also known as the **entropy** of *D*.

Information needed after using A to split D into V partitions.

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

Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,

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

The attribute *A* with the highest information gain, *Gain(A)*, is chosen as the splittingattribute at node*N*. This is equivalent to saying that we want to partition on the attribute*A* that would do the "best classification," so that the amount of information still requiredto finish classifying the tuples is minimal.

3.6.2 Gain Ratio

C4.5, a successor of ID3, 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 trainingdata set, *D*, into *v* partitions, corresponding to the *v* outcomes of a test on attribute *A*. Note that, for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in *D*. It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning. The gain ratio is defined as

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

3.6.3 Gini Index

The Gini index is used in CART. Using the notation previously described, the Gini indexmeasures 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 nonzero probability that an arbitrary tuple in *D* belongs to class $C_i$and is estimated by $|C_i,D|/|D|$ over *m* classes.

Note: The Gini index considers a binary split for each attribute.

When considering a binary split, we compute a weighted sum of the impurity of eachresulting 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.

➢ For continuous-valued attributes, each possible split-point must be considered. The strategy is similar to that described earlier for information gain, where the midpoint between each pair of (sorted) adjacent values is taken as a possible split-point.

➢ The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute *A* is

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

3.7 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.

➢ 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.

"How does tree pruning work?" There are two common approaches to tree pruning: *prepruning* and *postpruning*.

➢ In the **prepruning** approach, a tree is "pruned" by halting its construction early. 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.

➢ 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.

➢ In the postpruning, 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.
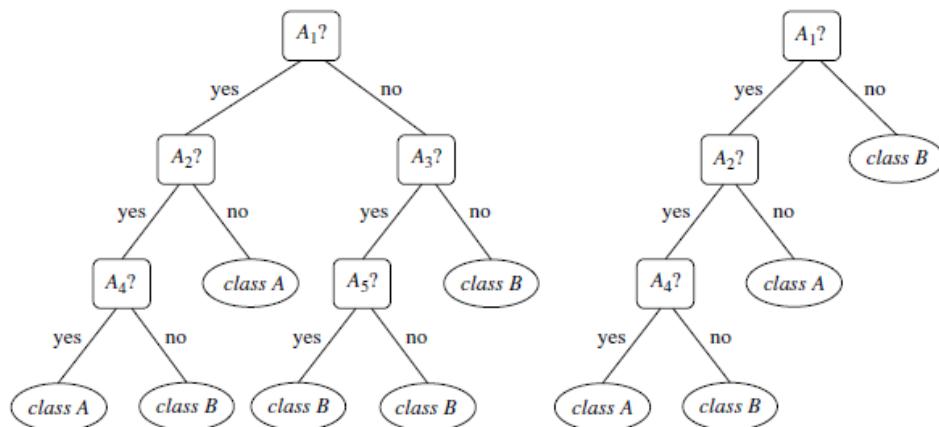
Fig: Unpruned and Pruned Trees

➢ The **cost complexity** pruning algorithm used in CART is an example of the postpruning approach.

➢ 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.

➢ This set isindependent of the training set used to build the unpruned tree and of any test set usedfor accuracy estimation.

➢ The algorithm generates a set of progressively pruned trees. Ingeneral, the smallest decision tree that minimizes the cost complexity is preferred.

➢ C4.5 uses a method called **pessimistic pruning**, which is similar to the cost complexitymethod in that it also uses error rate estimates to make decisions regarding subtreepruning.

3.8 Scalability of Decision Tree Induction:

"What if D, the disk-resident training set of class-labeled tuples, does not fit in memory? Inother words, how scalable is decision tree induction?" The efficiency of existing decisiontree algorithms, such as ID3, C4.5, and CART, has been well established for relativelysmall data sets. Efficiency becomes an issue of concern when these algorithms are appliedto the mining of very large real-world databases. The pioneering decision tree algorithmsthat we have discussed so far have the restriction that the training tuples should reside*in memory*.

In data mining applications, very large training sets of millions of tuples are common.Most often, the training data will not fit in memory! Therefore, decision tree construction becomes inefficient due to swapping of the training tuples in and out of main and cache memories. More scalable approaches, capable of handling training data that are too large to fit in memory, are required. Earlier strategies to "save space" included discretizing continuous-valued attributes and sampling data at each node. These techniques, however, still assume that the training set can fit in memory.

Several scalable decision tree induction methods have been introduced in recent studies.RainForest, for example, adapts to the amount of main memory available and appliesto any decision tree induction algorithm. The method maintains an **AVC-set** (where"AVC" stands for "*Attribute-Value*, *Classlabel*") for each attribute, at each tree node,describing the training tuples at the node. The AVC-set of an attribute *A* at node *N*gives the class label counts for each value of *A* for the tuples at *N*. The set of all AVC-sets at a node *N* is the **AVC-group**of *N*. The size of an AVC-set for attribute *A* at node *N* depends only on the number ofdistinct values of *A* and the number of classes in the set of tuples at *N*. Typically, this sizeshould fit in memory, even for real-world data. RainForest also has techniques, however,for handling the case where the AVC-group does not fit in memory. Therefore, themethod has high scalability for decision tree induction in very large data sets.

age	buys_computer	
	yes	no
youth	2	3
middle_aged	4	0
senior	3	2

income	buys_computer	
	yes	no
low	3	1
medium	4	2
high	2	2

student	buys_computer	
	yes	no
yes	6	1
no	3	4

credit_ratting	buys_computer	
	yes	no
fair	6	2
excellent	3	3

Fig: AVC Sets for dataset

Example for Decision Tree construction and Classification Rules:

Construct Decision Tree for following dataset,

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Solution:

Here the target class is buys_computer and values are yes, no. By using ID3 algorithm, we are constructing decision tree.

For ID3 Algorithm we have calculate Information gain attribute selection measure.

CLASS	P	buys_computer (yes)	9
	N	buys_computer (no)	5
		TOTAL	14

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

Age	P	N	TOTAL	I(P,N)
youth	2	3	5	I(2,3
middle_aged	4	0	4	I(4,0)
senior	3	2	5	I(3,2)

$$I(2,3) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.970$$

$$I(4,0) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$$

$$I(3,2) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.970$$

Age	P	N	TOTAL	I(P,N)	
youth	2	3	5	I(2,3	0.970
middle_aged	4	0	4	I(4,0)	0
senior	3	2	5	I(3,2)	0.970

$$\textbf{Info}_{\textbf{Age}}(\textbf{D}) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = \textbf{0.693}$$

$$\textbf{Gain(Age)} = \textbf{Info(D)} - \textbf{Info}_{\textbf{Age}}\textbf{(D)}$$
$$= \textbf{0.940} - \textbf{0693} = \textbf{0.247}$$
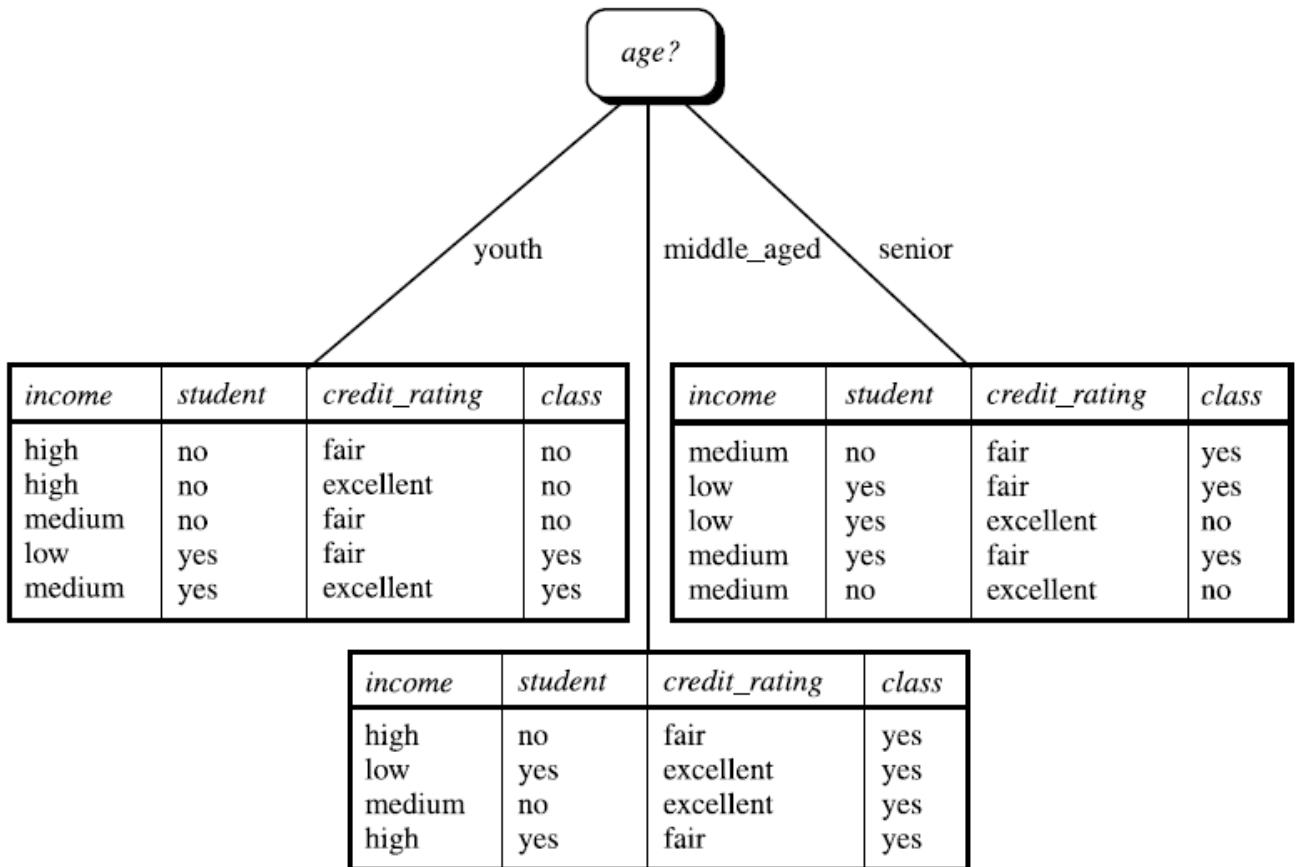
Similarly,
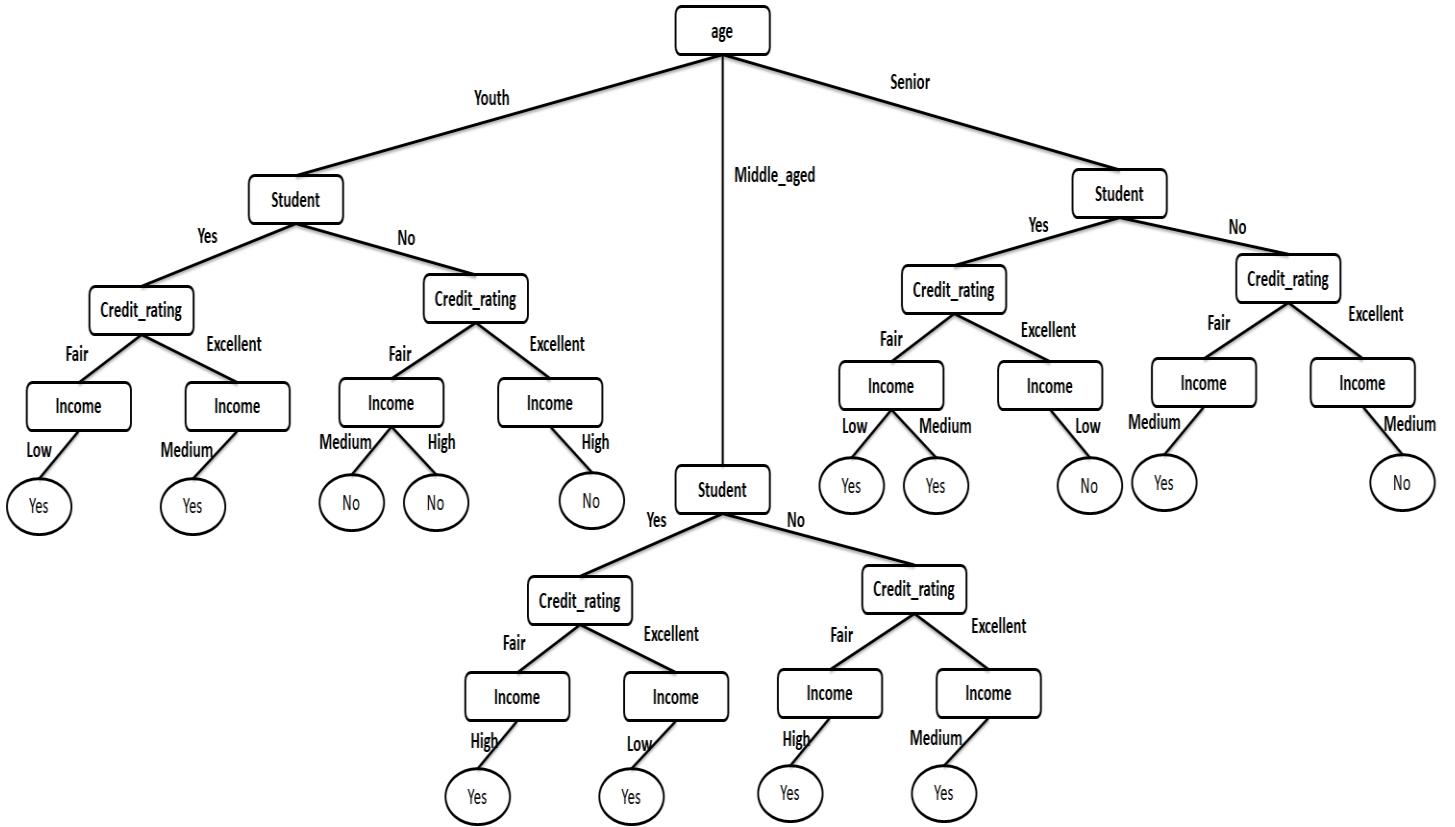
Gain(Income) = 0.029
Gain (Student) = 0.151
Gain (credit_rating) = 0.048
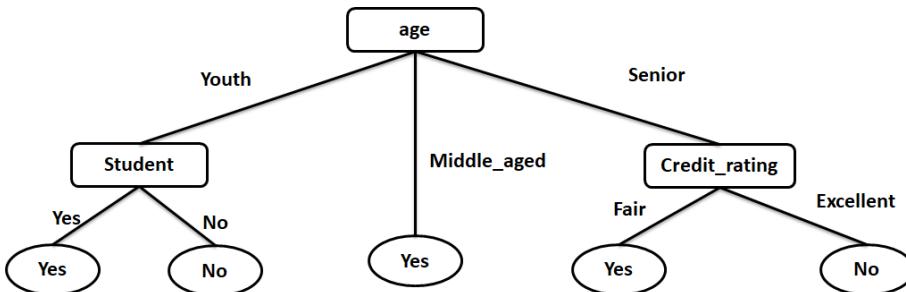
Finally, *age* has the highest information gain among the attributes, it is selected as the splitting attribute. Node *N* is labeled with *age*, and branches are grown for each of the attribute's values. The tuples are then partitioned accordingly, as

The Tree after splitting branches is

The Tree after Tree Pruning,

Finally, The Classification Rules are,
  ➢ **IF** age=Youth **AND** Student=Yes **THEN** buys_computer=Yes
  ➢ **IF** age=Middle_aged **THEN** buys_computer=Yes
  ➢ **IF** age=Senior **AND** Credit_rating=Fair **THEN** buys_computer=Yes

DATA CLASSIFICATION (Alternative Techniques)

Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statistics.

Classification: Alternative Techniques:

Bayesian Classification:
- Bayesian classifiers are statistical classifiers.
- They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' theorem.

Bayes' Theorem:
- Let X be a data tuple. In Bayesian terms, X is considered ― "evidence" and it is described by measurements made on a set of n attributes.
- Let H be some hypothesis, such as that the data tuple X belongs to a specified class C.
- For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the ―evidence‖ or observed data tuple X.
- P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.
- Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X).

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Naïve Bayesian Classification:

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let=be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2, …,xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, …, An.

2. Suppose that there are m classes, C1, C2, …, Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(C_i|X) > P(C_j|X) \ \ for \ \ 1 \leq j \leq m, j \neq i.$$

Thus we maximize $P(C_j|X)$. The class Ci for which $P(C_j|X)$. is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

3. As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = …= P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci).

4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). In order to reduce computation in evaluating P(X|Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple. Thus,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$
$$= P(x_1|C_1) \times P(x_2|C_2) \times \dots \dots \times P(x_n|C_i)$$

5. We can easily estimate the probabilities P(x1|Ci), P(x2|Ci), : : : , P(xn|Ci) from the training tuples.

6. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute P(X|Ci), we consider the following:

➢ If A_k is categorical, then P(x_k|Ci) is the number of tuples of class Ci in=havingthe value x_k for A_k, divided by |Ci ,D| the number of tuples of class Ci in D.

➢ If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward.

Example:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

We wish to predict the class label of a tuple using naïve Bayesian classification, given the same training data above. The training data were shown above in Table. The data tuples are described by the attributes *age*, *income*, *student*, and *credit rating*. The class label

attribute, *buys computer*, has two distinct values (namely, {yes, no}). Let *C*1 correspond to the class *buys computer=yes* and *C*2 correspond to b*uys computer=no.* The tuple we wish to classify is

X={age= "youth", income= "medium", student= "yes", credit_rating= "fair"}

We need to maximize P(X|Ci)P(Ci), for i=1,2. P(Ci), the prior probability of each class, can be computed based on the training tuples:

P(*buys computer = yes*) = 9/14 = 0.643
P(*buys computer = no*) = 5/14 = 0.357

To compute P(X|Ci), for *i* = 1, 2, we compute the following conditional probabilities:

P(*age = youth | buys computer = yes*)   = 2/9   = 0.222

P(*income=medium | buys computer=yes*)   = 4/9   = 0.444

P(*student=yes | buys computer=yes*)   = 6/9   = 0.667

P(*credit rating=fair | buys computer=yes*)   = 6/9   = 0.667

P(*age=youth | buys computer=no*)   = 3/5   = 0.600

P(*income=medium | buys computer=no*)   = 2/5   = 0.400

P(*student=yes | buys computer=no*)   = 1/5   = 0.200

P(*credit rating=fair | buys computer=no*)   = 2/5   = 0.400

Using these probabilities, we obtain

P(**X** | *buys computer=yes*) = P(*age=youth | buys computer=yes*)

$\times$ P(*income=medium | buys computer=yes*)

$\times$ P(*student=yes | buys computer=yes*)

$\times$ P(*credit rating=fair | buys computer=yes*)

= 0.222 $\times$ 0.444 $\times$ 0.667 $\times$ 0.667 = 0.044.

Similarly,

P(**X** | *buys computer=no*) = 0.600 $\times$ 0.400 $\times$ 0.200 $\times$ 0.400 = 0.019.

To find the class, C_i, that P(X|Ci)P(Ci), we compute
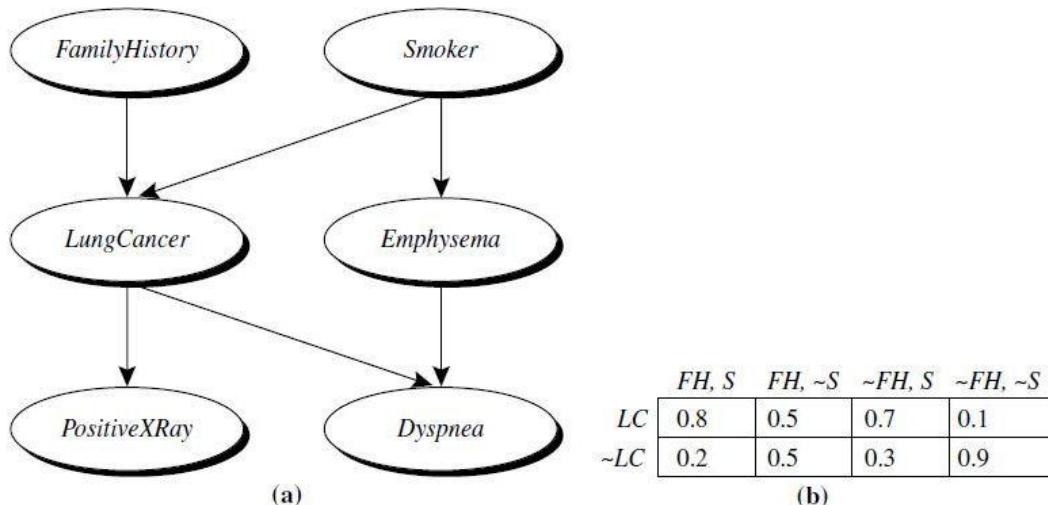
P(**X** | *buys computer=yes*) P(*buys computer=yes*) = 0.044 $\times$ 0.643 = 0.028

P(**X** | *buys computer=no*) P(*buys computer=no*) = 0.019 $\times$ 0.357 = 0.007

Therefore, the naïve Bayesian classifier predicts *buys computer = yes* for tuple **X**.

Bayesian Belief Networks

- *Bayesian belief networks*—probabilistic graphical models, which unlike naïve Bayesian classifiers allow the representation of dependencies among subsets of attributes.
- The naïve Bayesian classifier makes the assumption of class conditional independence, that is, given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another.
- When the assumption holds true, then the naïve Bayesian classifier is the most accurate in comparison with all other classifiers.
- They provide a graphical model of causal relationships, on which learning can be performed.
- A belief network is defined by two components—a *directed acyclic graph* and a set of *conditional probability tables* (See Figure).
- Each node in the directed acyclic graph represents a random variable. The variables may be discrete- or continuous-valued.
- They may correspond to actual attributes given in the data or to "hidden variables" believed to form a relationship.
- Each arc represents a probabilistic dependence. If an arc is drawn from a node *Y* to a node *Z*, then *Y* is a **parent** or **immediate predecessor** of *Z*, and *Z* is a **descendant** of *Y*.
- *Each variable is conditionally independent of its nondescendants in the graph, given its parents.*

	FH, S	FH, ~S	~FH, S	~FH, ~S
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

Simple Bayesian belief network. (a) A proposed causal model, represented by a directed acyclic graph. (b) The conditional probability table for the values of the variable *LungCancer* (*LC*) showing each possible combination of the values of its parent nodes, *FamilyHistory* (*FH*) and *Smoker* (*S*). *Source:* Adapted from Russell, Binder, Koller, and Kanazawa [RBKK95].

For example, having lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker. Note that the variable *PositiveXRay* is independent of whether the patient has a family history of lung cancer or is a smoker, given that we know the patient has lung cancer.

In other words, once we know the outcome of the variable *LungCancer*, then the variables *FamilyHistory* and *Smoker* do not provide any additional information regarding *PositiveXRay*. The arcs also show that the variable *LungCancer* is conditionally independent of *Emphysema*, given its parents, *FamilyHistory* and *Smoker*.

A belief network has one **conditional probability table (CPT)** for each variable. The CPT for a variable *Y* specifies the conditional distribution *P(Y|Parents(Y))*, where *Parents*(*Y*) are the parents of *Y*. Figure (b) shows a CPT for the variable *LungCancer*. The conditional probability for each known value of *LungCancer* is given for each possible combination of the values of its parents. For instance, from the upper leftmost and bottom rightmost entries, respectively.

Association Analysis

Association:

      Association mining aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items or objects in transaction databases, relational database or other data repositories. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

Examples:

      Rule Form: Body→Head [Support, confidence]

      Buys (X, "Computer") → Buys (X, "Software") [40%, 50%]

Association rule: basic concepts:

- ➢ Given: (1) database of transaction, (2) each transaction is a list of items (purchased by a customer in visit)
- ➢ Find: all rules that correlate the presence of one set of items with that of another set of items.
  - E.g., 98% of people who purchase tires and auto accessories also get automotive— services done.
  - E.g., Market Basket Analysis — This process analyzes customer buying habits by finding associations between the different items that customers place in their "Shopping Baskets". The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customer.

Applications:

- ➢ Maintenance agreement (what the store should do to boost maintenance agreement sales)
- ➢ Home Electronics (what other products should the store stocks up?)
- ➢ Attached mailing in direct marketing

Association Rule:

      An association rule is an implication expression of the form X→Y, where X and Y are disjoint itemsets, i.e., $X \cap Y = \emptyset$. The strength of an association rule can be measured in terms of its support and confidence. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in Y appear in transactions that contain X. The formal definition of these metrics are

$$\text{Support, } s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$
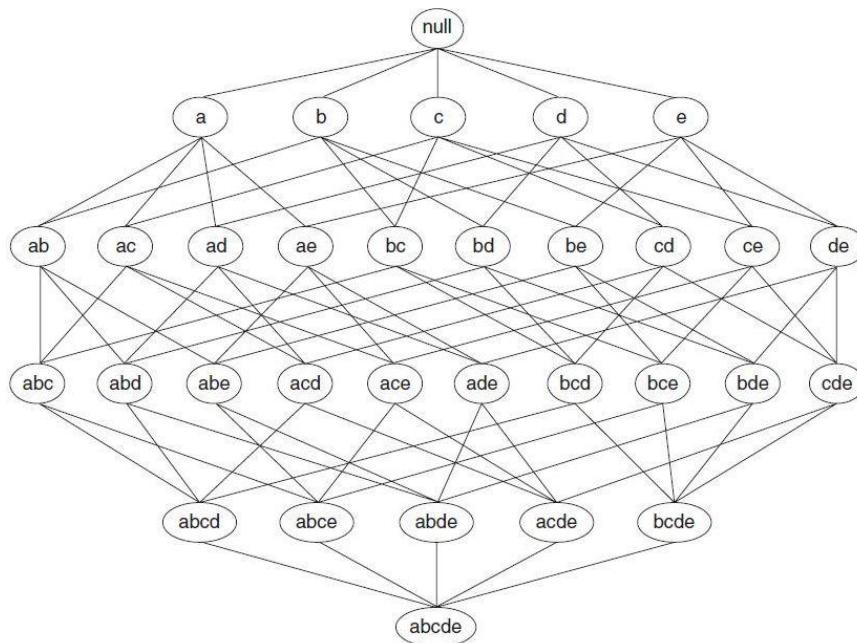
$$\text{Confidence, } c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Why Use Support and Confidence? Support is an important measure because a rule that has very low support may occur simply by chance. A low support rule is also likely to be uninteresting from a business perspective because it may not be profitable to promote items that customers seldom buy together. For these reasons, support is often used to eliminate uninteresting rules.

Confidence, on the other hand, measures the reliability of the inference made by a rule. For a given rule $X\rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Confidence also provides an estimate of the conditional probability of Y given X.

Therefore, a common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

1. Frequent Itemset Generation, whose objective is to find all the item- sets that satisfy the *minsup* threshold. These itemsets are called frequent itemsets.

2. Rule Generation, whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules.

Frequent Itemset Generation:

A lattice structure can be used to enumerate the list of all possible itemsets. Above Figure shows an itemset lattice for $I = \{a, b, c, d, e\}$. In general, a data set that contains k items can potentially generate up to $2^k - 1$ frequent itemsets, excluding the null set. Because k can be very large in many practical applications, the search space of itemsets that need to be explored is exponentially large.

To find frequent itemsets we have two algorithms,
   a) Apriori Algorithm
   b) FP-Growth

a) Apriori Algorithm:

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties, as we shall see later. Apriori employs an iterative approach known as a *level-wise* search, where k-itemsets are used to explore $(k+1)$-itemsets.

First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted by L1. Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space.

Apriori property: *All nonempty subsets of a frequent itemset must also be frequent.*

The Apriori property is based on the following observation. By definition, if an itemset *I* does not satisfy the minimum support threshold, *min sup*, then *I* is not frequent, that is, $P(I)$< *min sup*. If an item *A* is added to the itemset *I*, then the resulting itemset (i.e.,IUA) cannot occur more frequently than *I*. Therefore, IUA is not frequent either, that is, $P(\text{IUA}) <$ *min sup*.

This property belongs to a special category of properties called **antimonotonicity** in the sense that *if a set cannot pass a test, all of its supersets will fail the same test as well*. It is called *antimonotonicity* because the property is monotonic in the context of failing a test.
A two-step process is followed, consisting of **join** and **prune** actions.

1. The join step: To find L_k, a set of **candidate** k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k.
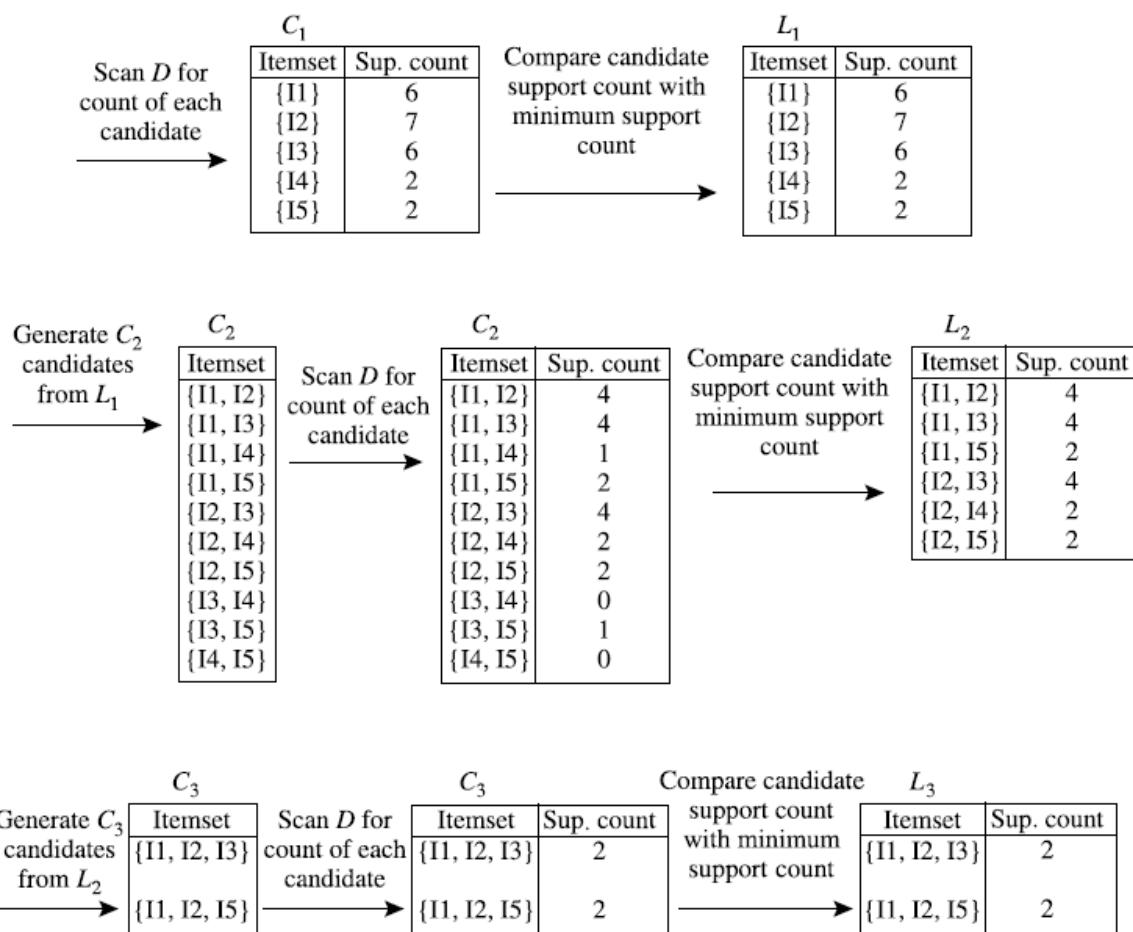
2. The prune step: C_k is a superset of L_k, that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in Ck. A database scan to determine the count of each candidate in C_k would result in the determination of L_k.

Example:

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

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions to count the number of occurrences of each item.

2. Suppose that the minimum support count required is 2, that is, *min sup* = 2. (Here, we are referring to *absolute* support because we are using a support count. The corresponding relative support is 2/9 = 22%.) The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. In our example, all of the candidates in C_1 satisfy minimum support.

3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2. C_2 consists of 2-itemsets. Note that no candidates are removed from C_2 during the prune step because each subset of the candidates is also frequent.

4. Next, the transactions in *D* are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure

5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

Scan *D* for count of each candidate →

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count →

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan *D* for count of each candidate →

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count →

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 →

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan *D* for count of each candidate →

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count →

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

6. The generation of the set of the candidate 3-itemsets, C_3, is detailed in Figure From the join step, we first get $C_3 = L_2 \bowtie L_2 =$ {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}} Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3.

7. The transactions in D are scanned to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support.

8. The algorithm uses $L_3 \bowtie L3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in {I1, I2, I3, I5}, itemset {I1, I2, I3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 \neq \emptyset$, and the algorithm terminates, having found all of the frequent itemsets.

b) FP-Growth:
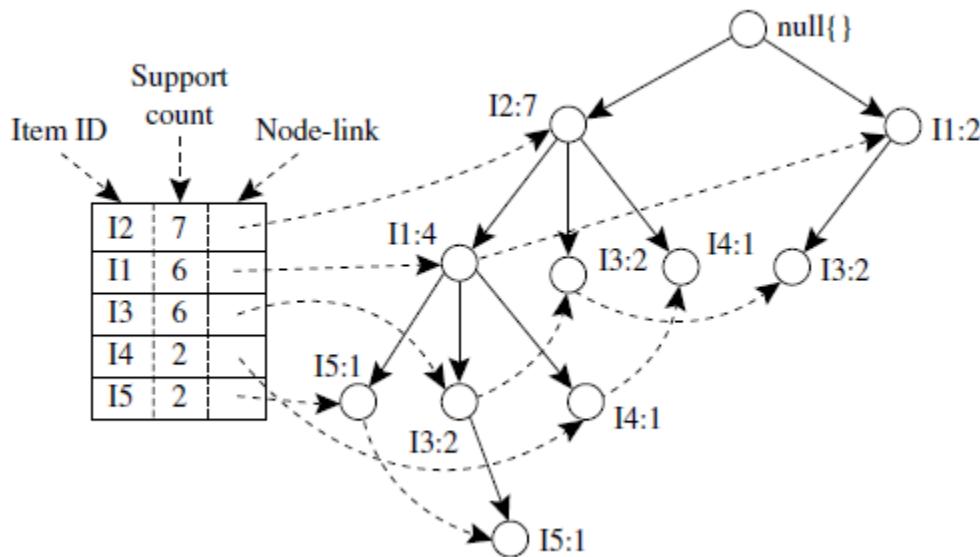
FP-growth (finding frequent itemsets without candidate generation). We reexamine the mining of transaction database, *D*, of Table in previous Example using the frequent pattern growth approach.

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

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or *list* is denoted by *L*. Thus, we have L = {{I2:7}, {I1:6}, {I3:6}, {I4:2}, {I5:2}}

An FP-tree is then constructed as follows. First, create the root of the tree, labeled with "null." Scan database D a second time. The items in each transaction are processed in L order (i.e., sorted according to descending support count), and a branch is created for each transaction.

The FP-tree is mined as follows. Start from each frequent length-1 pattern (as an initial **suffix pattern**), construct its **conditional pattern base** (a "sub-database," which consists of the set of *prefix paths* in the FP-tree co-occurring with the suffix pattern), then construct its (*conditional*) FP-tree, and perform mining recursively on the tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

Finally, we can conclude that frequent itemsets are {I2, I1, I5} and {I2, I1, I3}.

Generating Association Rules from Frequent Itemsets

Once the frequent itemsets from transactions in a database *D* have been found, it is straightforward to generate strong association rules from them (where *strong* association rules satisfy both minimum support and minimum confidence). This can be done using Eq. for confidence, which we show again here for completeness:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

The conditional probability is expressed in terms of itemset support count, where $support\_count(A \cup B)$ is the number of transactions containing the itemsets $A \cup B$, and $support\_count(A)$ is the number of transactions containing the itemset A. Based on this equation, association rules can be generated as follows:

- For each frequent itemset l, generate all nonempty subsets of l.

- For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where min_conf is the minimum confidence threshold.

Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support. Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.

Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown before in Table 6.1. The data contain frequent itemset $X = \{I1, I2, I5\}$. What are the association rules that can be generated from X? The nonempty subsets of X are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$. The resulting association rules are as shown below, each listed with its confidence:

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

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules are output, because these are the only ones generated that are strong. Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right side of the rule. ∎