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 ofdata
- Extracting essence of informationstored.
- Discovering patterns in rawdata.

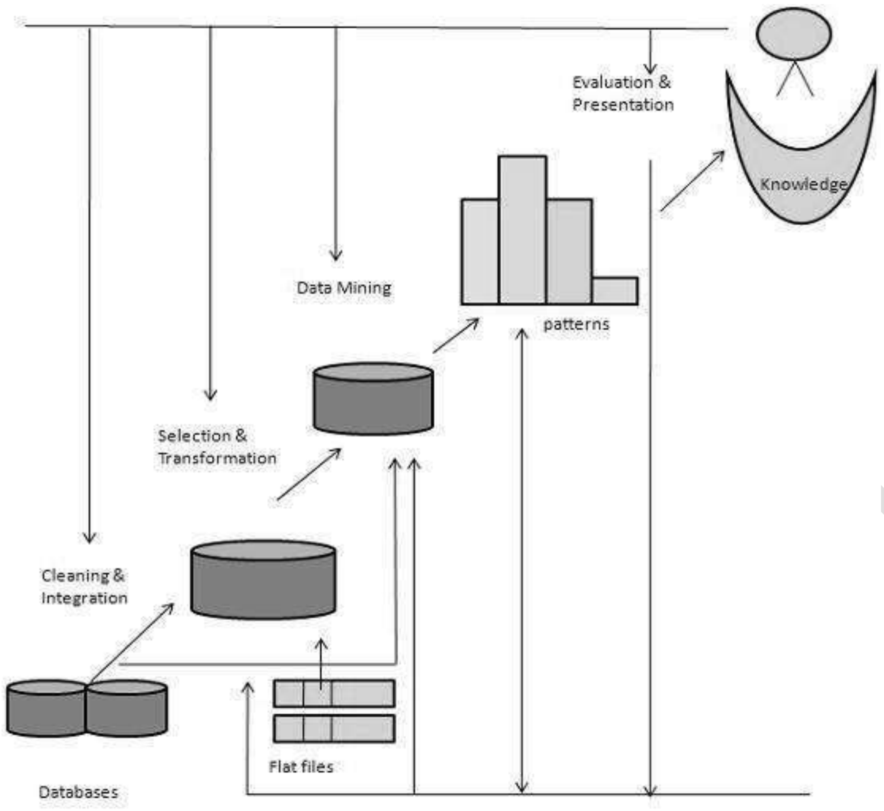
What is Data Mining?

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 −

- MarketAnalysis
- FraudDetection
- CustomerRetention
- Production Control
- ScienceExploration

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 isremoved.
- **Data Integration** − In this step, multiple data sources arecombined.
- **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 aggregationoperations.
- **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 isrepresented.



What kinds of data can be mined?

1. FlatFiles
2. RelationalDatabases
3. DataWarehouse
4. Transactional Databases
5. MultimediaDatabases
6. Spatial Databases
7. Time SeriesDatabases
8. World WideWeb(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: CSVfile.
   - **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. ***Data Warehouse***
    - A data warehouse is defined as the collection of data integrated from multiple sources that will query and decision making.
    - There are three types of data ware house: **Enterprise** data ware house, **Data Mart** and **Virtual** Warehouse.
    - Two approaches can be used to update data in Data Ware house: **Query-driven** Approach and **Update-driven** Approach.
    - **Application**: Business decision making, Data mining ,etc.
4. ***Transactional Data bases***
    - 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 data bases, etc.
5. ***Multimedia Databases***
    - Multimedia databases consists audio, video, images and text media.
    - They can be stored on Object-Oriented Data bases.
    - They are used to store complex information in pre-specified formats.
    - **Application**: Digital libraries, video-on demand, news-on demand, musical data base, etc.
6. ***Spatial Database***
    - Store geo graphical 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 overtime.

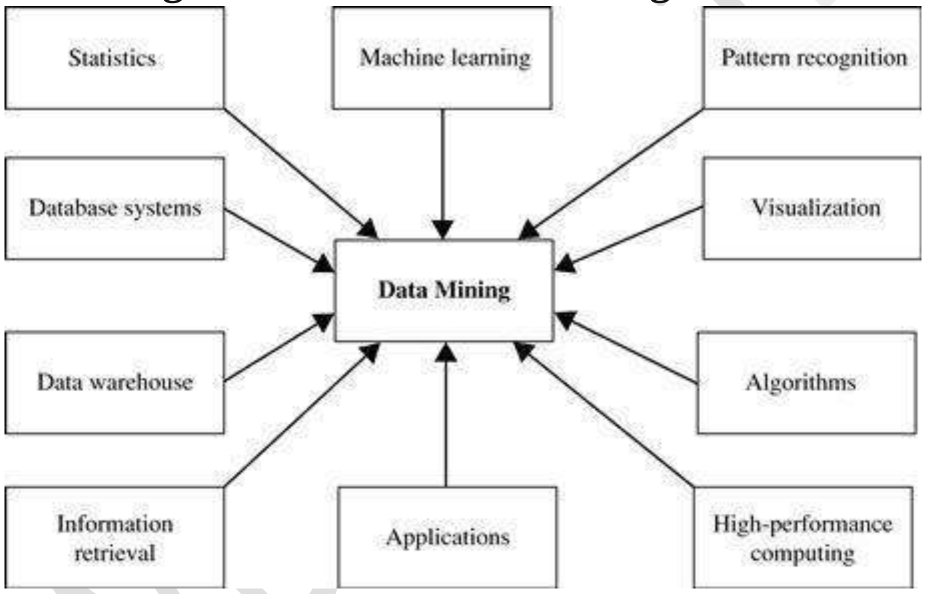
Data Mining Task Primitives

- 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 ware house

- Databases are used for the purpose of recording the data as well as data ware housing.
- 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 forbusiness.
- 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.
- APatternisalsointerestingifitisvalidatesahypothesisthattheusersoughttoconfirm.

Data Mining Applications

Here is the list of areas where data mining is widely used −

- Financial Data Analysis
- Retail Industry
- Tele communication 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 multi dimensional 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 DataTypes 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 itis.

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 2types.
   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 countable 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 and3.
   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 be a set of N observed values or observations ofX.

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

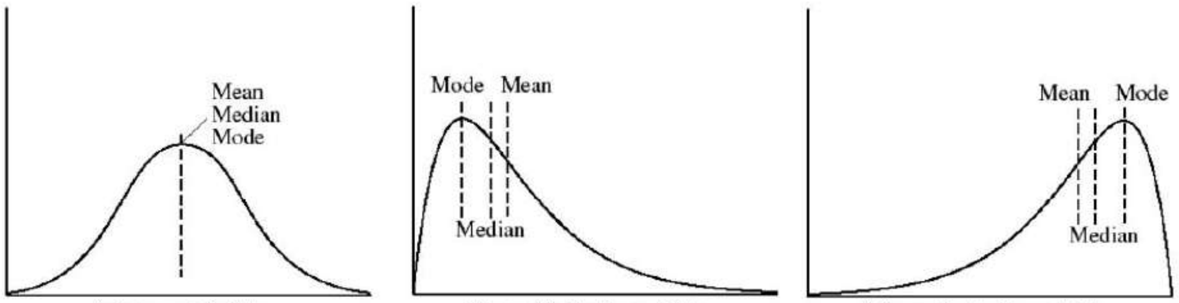
b) ***Median:*** Median is middle value among all values.

For N number of odd list median is ~~th v~~alue.

For N number of even list median is ~~th value~~.

c) Mode:

- The *mode* is another measure of central tendency.
- Datasets with one, two, or three modes are respectively called uni modal, 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

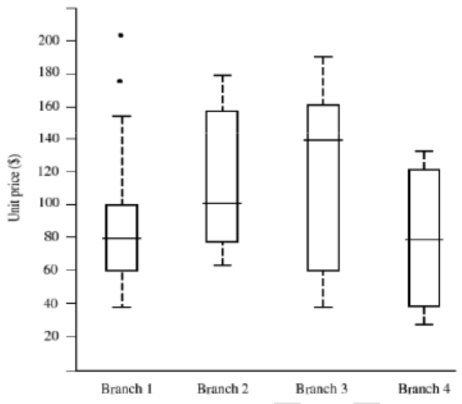
2. Measuring the Dispersion of data:

- The data can be measured as range, quantiles, quartiles, percentiles, and inter quartile 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, Inter quartile 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 Box plot in graphical format.

- Box plot 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 isIRQ.
- 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 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 varianceσ^2.
- σ measures spread about the mean and should be used only when the mean is chosen as the measure ofcenter.
- 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:

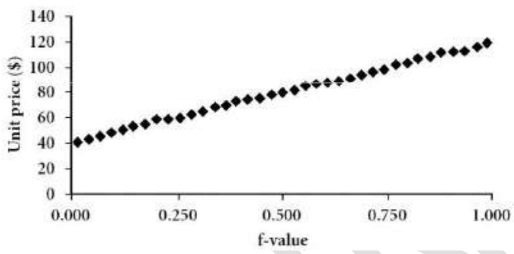
- Barcharts
- Piecharts
- Line graphs
- Boxplot
- Histograms
- Quantile plots, Quantile - Quantileplots
- Scatterplots

The data values can represent as Bar charts, pie charts, Line graphs, etc.

Quantile plots:

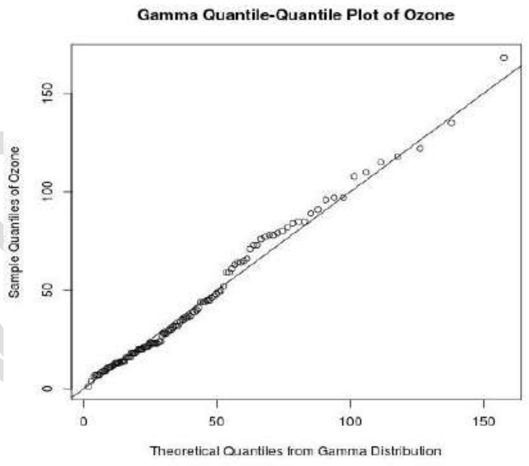
- A quantile plot is a simple and effective way to have a first look at a univariate data distribution.
- Plots quantile information
    - For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i% of the data are below or equal to the value x_i
- Note that
    - the 0.25 quantile corresponds to quartileQ1,
    - the 0.50 quantile is the median, and
    - the 0.75 quantile isQ3.

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



Quantile - Quantile plots:

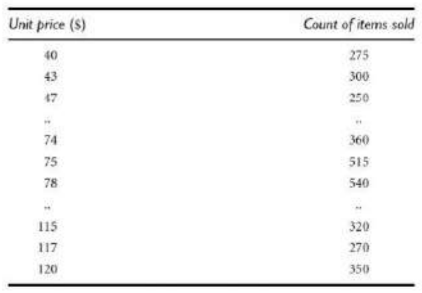
In statistics, a Q-Q plot is a portability plot, which is a graphical method for comparing two portability distributions by plotting their quantiles against each other.



Histograms *or* **frequency histograms:**

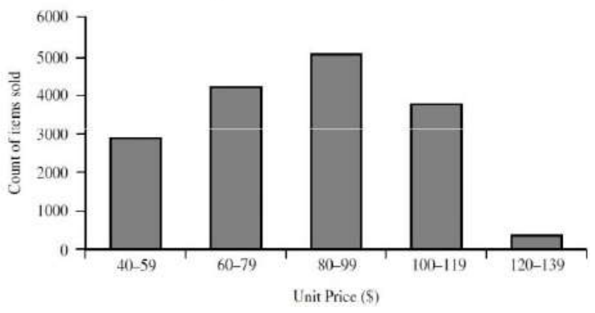
- A univariate graphical method
- Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data
- If the attribute is categorical, then one rectangle is drawn for each known value of A, and the resulting graph is more commonly referred to as a barchart.
- If the attribute is numeric, the term histogram ispreferred.

- **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

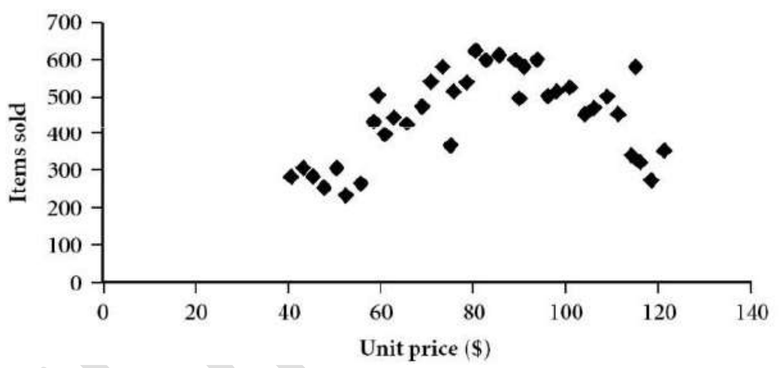
- **Example: A histogram**



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.



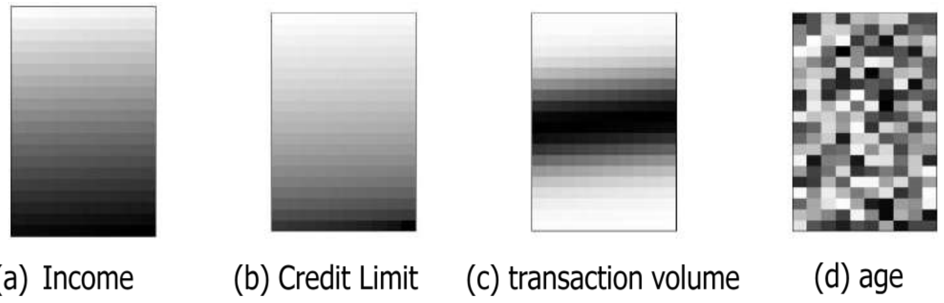
Data Visualization:

Visualization is the use of computer graphics to create visual images which aid in the understanding of complex, often massive representations of data.

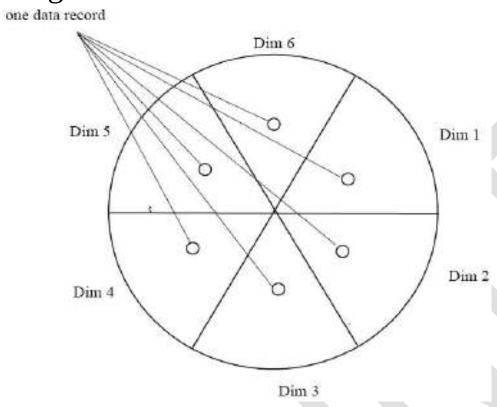
- ⊙ Categorization of visualization methods:
  - a) Pixel-oriented visualization techniques
  - b) Geometric projection visualization techniques
  - c) Icon-based visualization techniques
  - d) Hierarchical visualization techniques
  - e) Visualizing complex data and relations

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



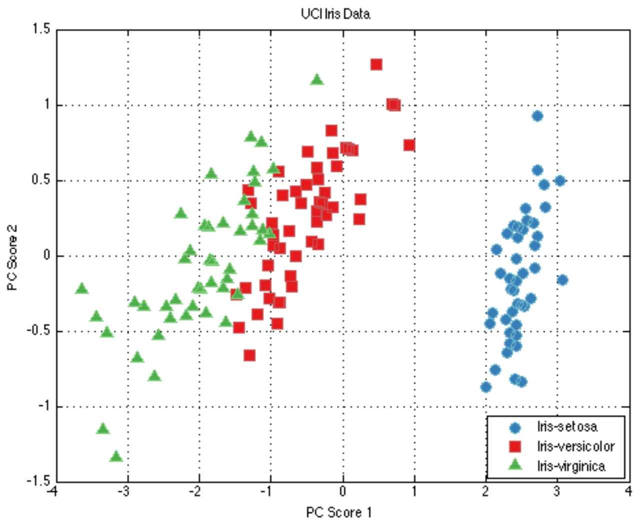
- To save space and show the connections among multiple dimensions, space filling is often done in a circle segment



b) Geometric projection visualization techniques

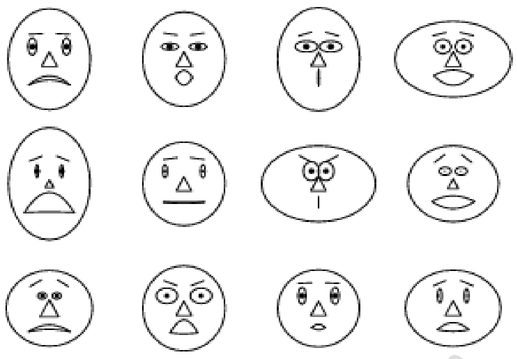
Visualization of geometric transformations and projections of the data Methods

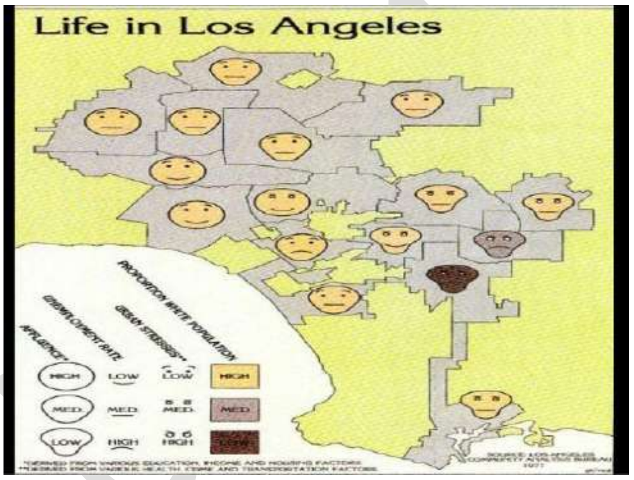
- Direct visualization
- Scatter plot and scatter plot matrices
- Landscapes
- Projection pursuit technique: Help users find meaningful projections of multi dimensional data
- Prosection views
- Hyper slice
- Parallel coordinates



c) Icon-based visualization techniques

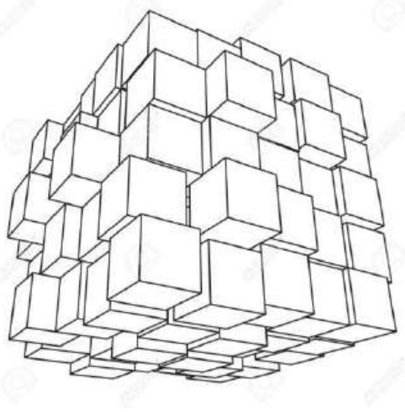
- Visualization of the data values as features of icons
- Typical visualization methods
    - Chern off Faces
    - Stick Figures
- General techniques
    - Shape coding: Use shape to represent certain information encoding
    - Color icons: Use color icons to encode more information
    - Tile bars: Use small icons to represent the relevant feature vectors in document retrieval



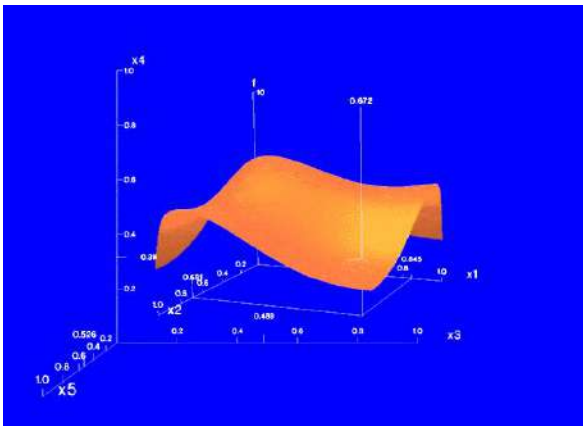


d) Hierarchical visualization techniques

- Visualization of the data using a hierarchical partitioning into subspaces
- Methods
    - DimensionalStacking
    - Worlds-within-Worlds
    - Tree-Map
    - ConeTrees
    - InfoCube



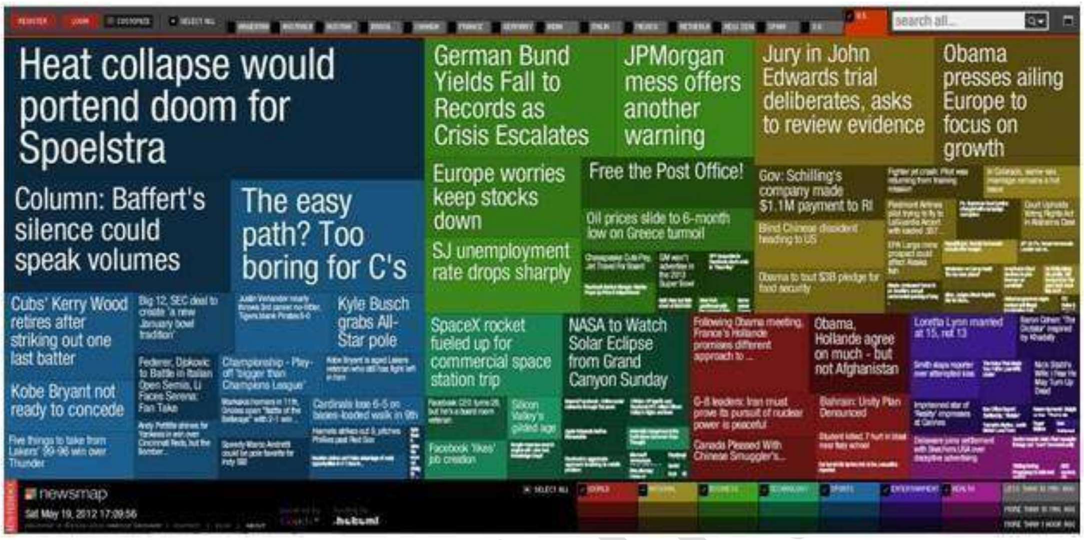
InfoCube



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 fontsize/color
- Besides text data, there are also methods to visualize relationships, such as visualizing



Heat collapse would portend doom for Spoelstra
Column: Baffert's silence could speak volumes
The easy path? Too boring for C's
German Bund Yields Fall to Records as Crisis Escalates
JPMorgan mess offers another warning
Jury in John Edwards trial deliberates, asks to review evidence
Obama presses ailing Europe to focus on growth
Europe worries keep stocks down
Free the Post Office!
SJ unemployment rate drops sharply
Oil prices slide to 6-month low on Greece turmoil
Cubs' Kerry Wood retires after striking out one last batter
Kobe Bryant not ready to concede
Kyle Busch grabs All-Star pole
SpaceX rocket fueled up for commercial space station trip
NASA to Watch Solar Eclipse from Grand Canyon Sunday
Obama, Hollande agree on much - but not Afghanistan
newsmap

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 object sare.
- Often falls between 0 (no similarity) and 1 (complete similarity).

Dissimilarity Measure

- Numerical measure of how different two data object sare.
- 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	$s = 1 - \frac{\lVert p-q \rVert}{n-1}$ (values mapped to integer 0 to n-1, where n is the number of values)	$d = \frac{\lVert p-q \rVert}{n-1}$
Interval or Ratio	$s = 1 - \lVert p - q \rVert$, $s = \frac{1}{1+\lVert p-q \rVert}$	$d = \lVert p - q \rVert$

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 ith and jth 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.
- $\lambda \rightarrow \infty$: L_∞ metric, Supremum distance.

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

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

c) Mahalano bis Distance

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

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 ∑ is the p×p sample covariance matrix.

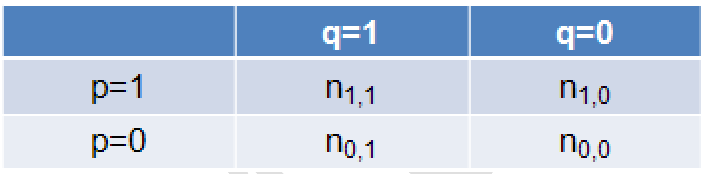
Common Properties of Similarity Measures

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

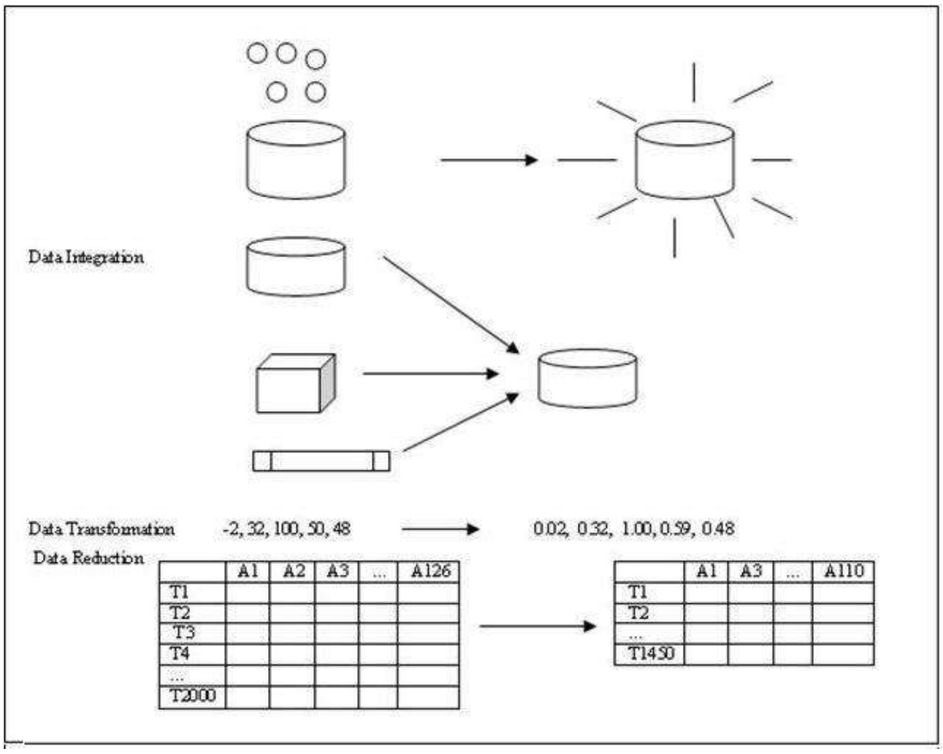
Real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results, so we prefer a preprocessing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

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 -∞.
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.

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 ofthe 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:

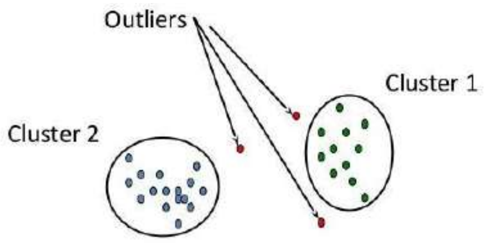
Bin 1: 9,9,9,9
Bin 2: 23,23,23,23
Bin 3: 29,29,29,29

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.



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^2Correlation analysis for numerical data:

For nominal data, a correlation relationship between two attributes, A and B, can be discovered by a $\chi 2$ (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.

c) Detection and Resolution of Data Value Conflicts.

A third important issue in data integration is the *detection and resolution of data value conflicts*. For example, for the same real–world entity, attribute value from different sources 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.

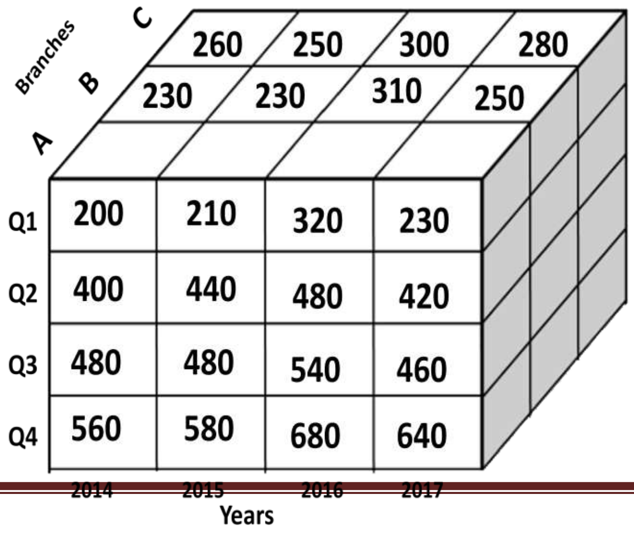
Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.

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

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.



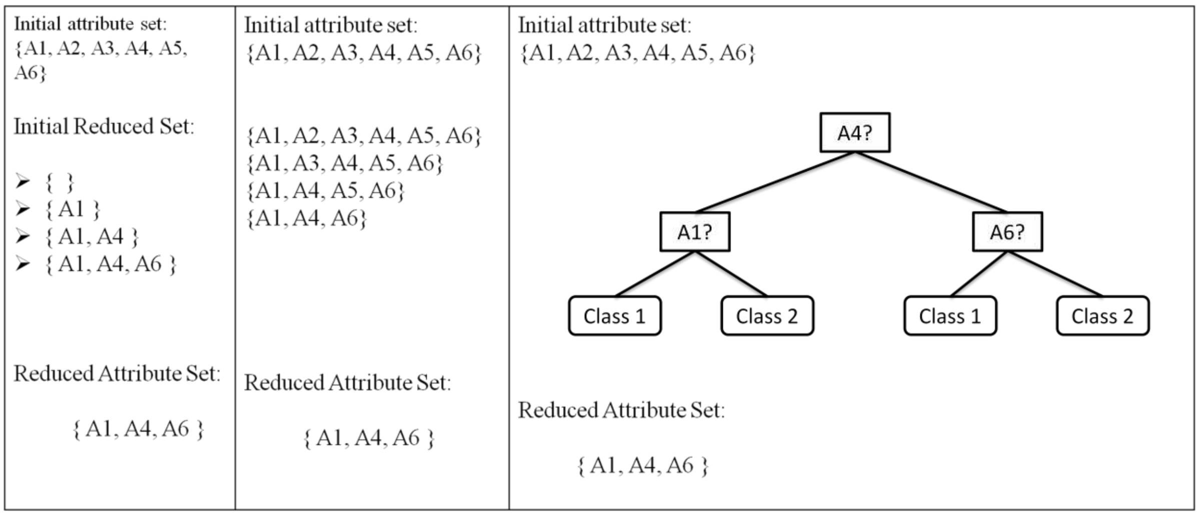
Year/Quarter	2014	2015	2016	2017
Quarter 1	200	210	320	230
Quarter 2	400	440	480	420
Quarter 3	480	480	540	460
Quarter 4	560	580	680	640

Year	Sales
2014	1640
2015	1710
2016	2020
2017	1750

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 attribute set: {A1, A2, A3, A4, A5, A6}	Initial attribute set: {A1, A2, A3, A4, A5, A6}
Initial Reduced Set: ➢ { } ➢ { A1 } ➢ { A1, A4 } ➢ { A1, A4, 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 }	Reduced Attribute Set: { A1, A4, A6 }	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.

Numerosity 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=αX+β
  - 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

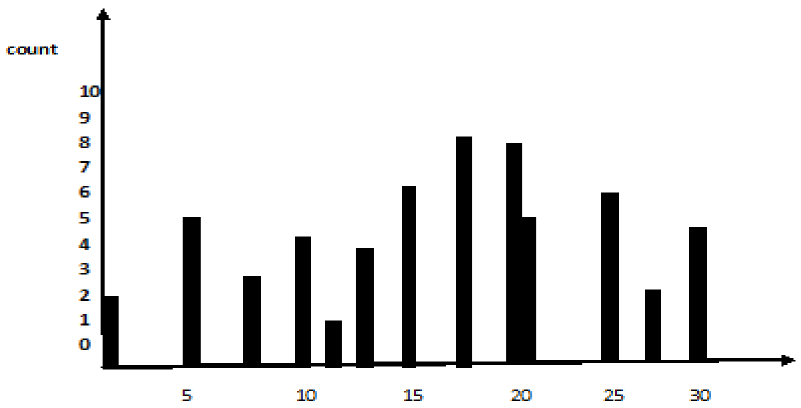


Figure 3.7 A Histogram for price using Singleton Buckets

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

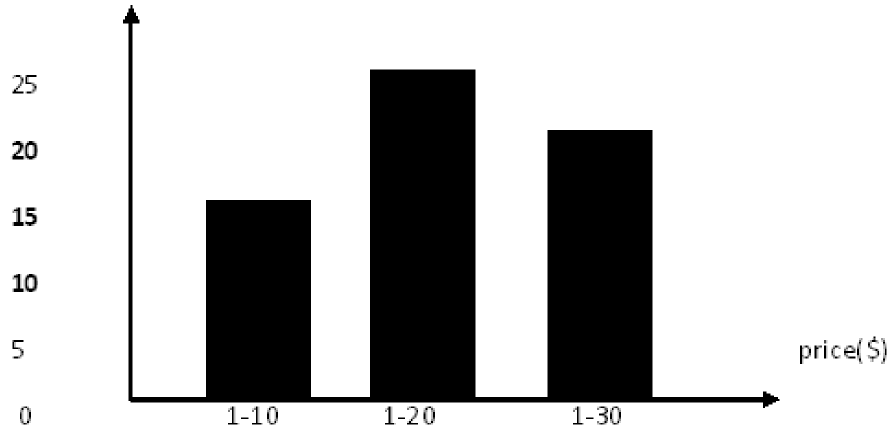


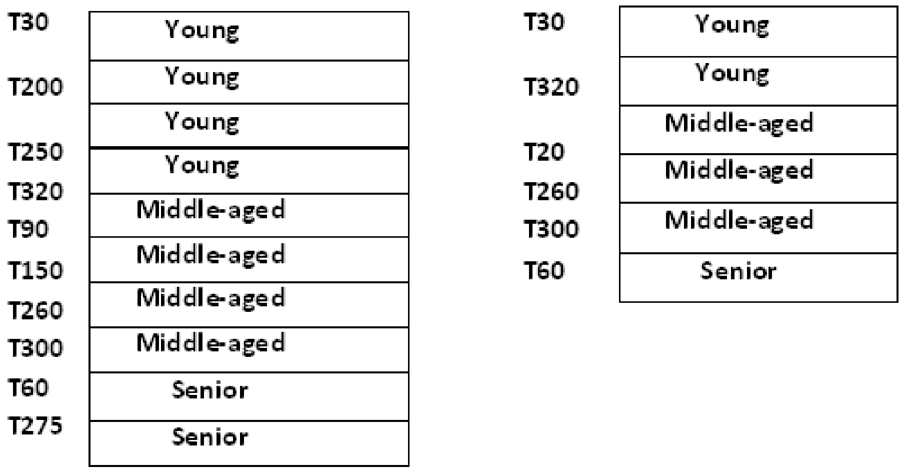
Figure.2.8 An equal-width histogram for price, where values are aggregated so ***that each bucket has a uniform width of $10.***

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.



T30	Young
T200	Young
T250	Young
T320	Young
T90	Middle-aged
T150	Middle-aged
T260	Middle-aged
T300	Middle-aged
T60	Senior
T275	Senior

T30	Young
T320	Young
T20	Middle-aged
T260	Middle-aged
T300	Middle-aged
T60	Senior

Figure 2.9. Sampling can be used for data reduction.

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

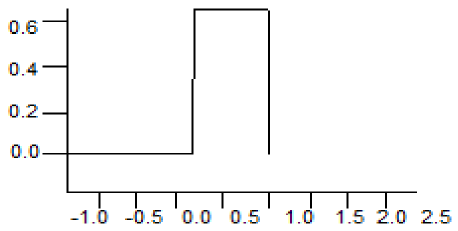
a) **Wavelet transforms**
b) **Principal components analysis.**

a) Wavelet transforms:

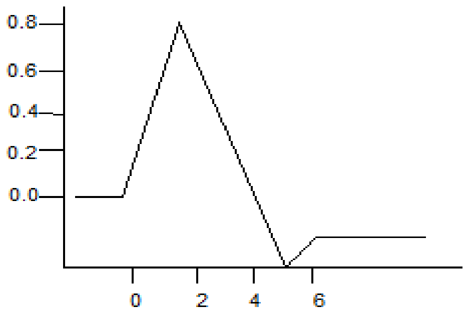
The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector, transforms it to a numerically different vector, of wavelet coefficients. The two vectors are of the same length. When applying this technique to data reduction, we consider each tuple as an n-dimensional data vector, that is, X=(x1,x2,…………,xn), 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 (X2i , X2i+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.

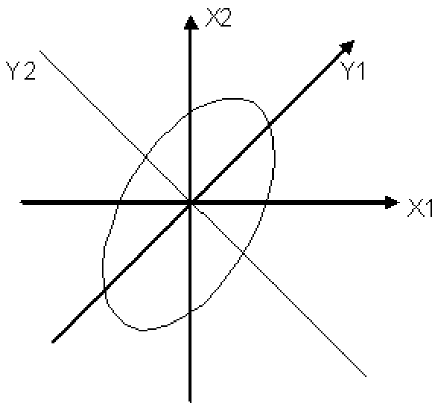
4. The two functions are recursively applied to the sets of data obtained in the previous loop, until the resulting data sets obtained are of length 2.

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* ortho normal 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 appropriate for mining. Strategies for data transformation include the following:

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

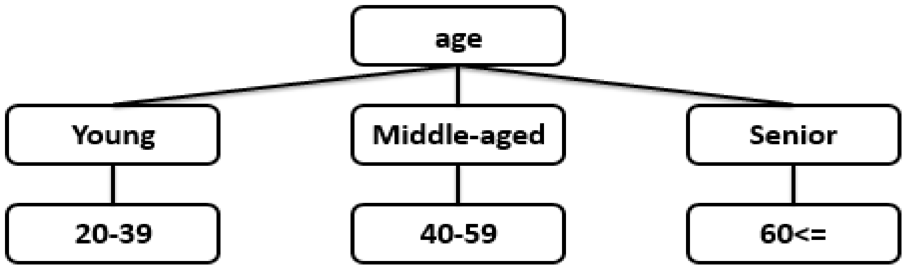
2. Attribute construction (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.

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

4. Normalization, where the attribute data are scaled so as to fall within a smaller range, such as 1.0 to 2.0, or 0.0 to 1.0.

5. Discretization, where the raw values of a numeric attribute (e.g., *age*) are replaced by interval 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, resulting in a *concept hierarchy* for the numeric attribute. Figure 3.12 shows a concept hierarchy for the attribute *price*. More than one concept hierarchy can be defined for the same attribute to accommodate the needs of various users.

6. Concept hierarchy generation for nominal data, where attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*. Many hierarchies for nominal attributes are implicit within the database schema and can be automatically defined 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 from meters to inches for *height*, or from kilograms 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* be a numeric attribute with *n* observed values, v_1, v_2, …., 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. Itwill encounter an –out-of-bounds‖ error if a future input case for normalization fallsoutside of the original data range for *A*.

Example:-Min-max normalization. Suppose that the minimum and maximum values fortheattribute *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* istransformed 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*A* and σA 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 917normalizes 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. For example, attribute values can be discretized by applying equal-width or equal-Frequency binning, and then replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions to generate concept hierarchies.

b) Discretization by Histogram Analysis:

Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information. A histogram partitions the values of an attribute, A, into disjoint ranges called buckets or bins.

In an equal-width histogram, for example, the values are partitioned into equal-size partitions or 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 partition contains 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 can be applied to discretize a numeric attribute, A, by partitioning the values of A into clusters or groups.

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

Concept Hierarchy Generation for Nominal Data

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

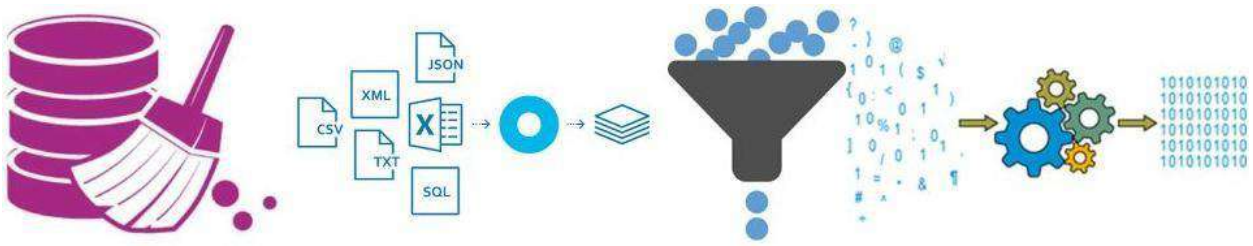
Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, many hierarchies are implicit within the database schema and can be automatically defined at the schema definition level. The concept hierarchies can be used to transform the data into multiple levels of granularity.

1. **Specification of a partial ordering of attributes explicitly at the schema level by users or experts:** A user or expert can easily define a concept hierarchy by specifying 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 country form a hierarchy at the schema level, a user could define some intermediate levels manually.
3. **Specification of a set of attributes, but not of their partial ordering:** A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly

State their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy.

4. **Specification of only a partial set of attributes:** Sometimes a user can be careless when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the re
5. levant 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.

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.

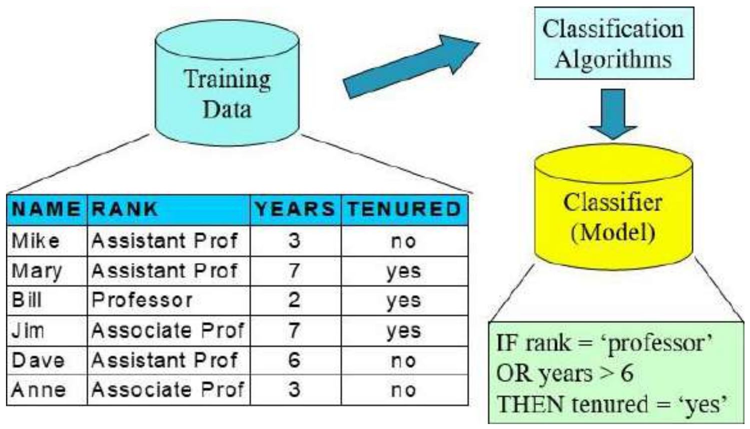
Suppose that the marketing manager wants to predict how much a given customer will spend during a sale at AllElectronics. This data analysis task is an example of **numeric prediction**, where the model constructed predicts a continuous-valued function, or ordered value, as opposed to a class label.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

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* (where the model is used to predict class labels for given data).

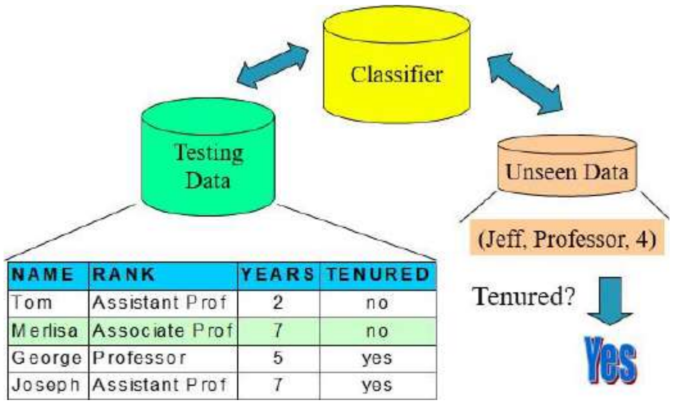
- In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the **learning step** (or training phase), where a classification algorithm builds the classifier by analyzing or “learning from” a training set made up of database tuples and their associated 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



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

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Fig: Learning Step



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

Fig: Classification Step

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 of binary decisiontrees.

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.

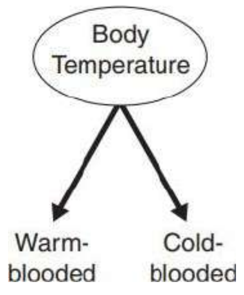
Method:

```
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;
```

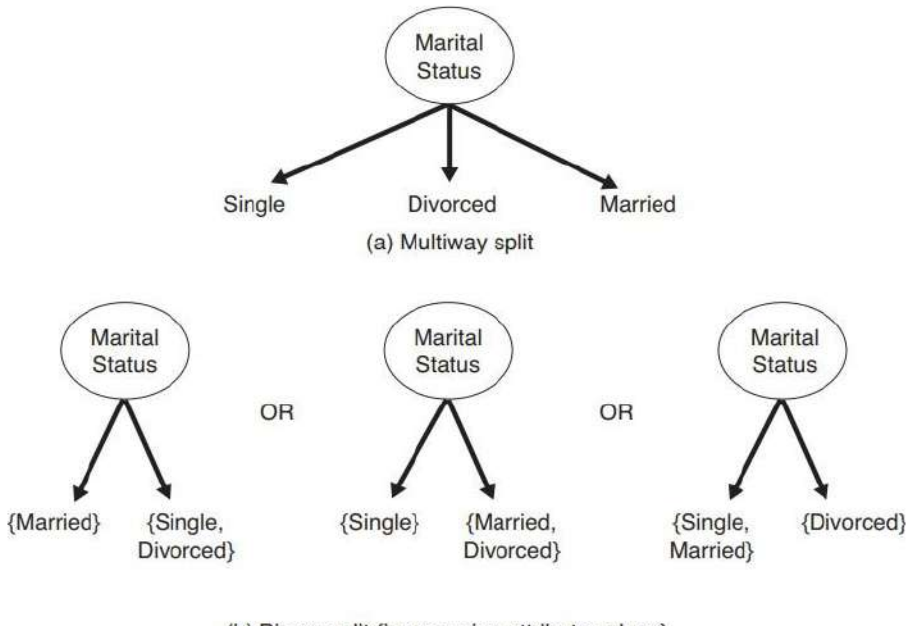
Methods for selecting best test conditions

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.

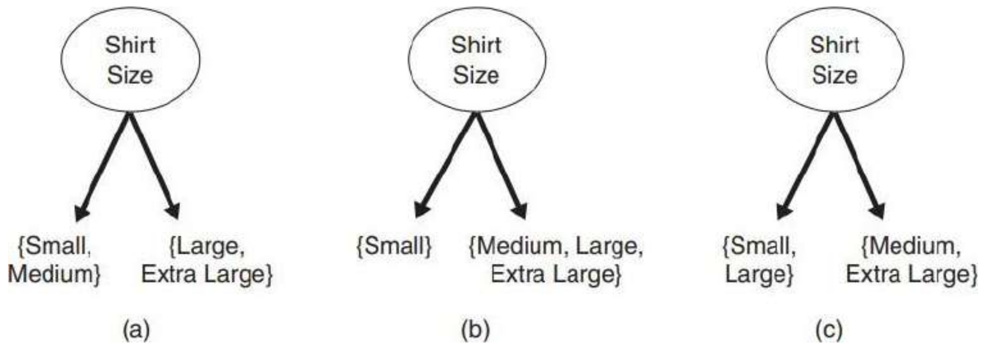


Nominal Attributes: These can have many values. These can be represented in two ways.



(b) Binary split {by grouping attribute values}

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



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*.

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 splitting attribute 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.

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 training data 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

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

Gini Index

The Gini index is used in CART. Using the notation previously described, the Gini index measures the impurity of *D*, a data partition or set of training tuples, as

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

Where $p_i$is the 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

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

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 *over fitting* 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 pre specified threshold, then further partitioning of the given subset is halted. There are difficulties, however, in choosing an appropriate threshold.
- In the post pruning, which removes sub trees from a "fully grown" tree. A sub tree 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 sub tree being replaced.

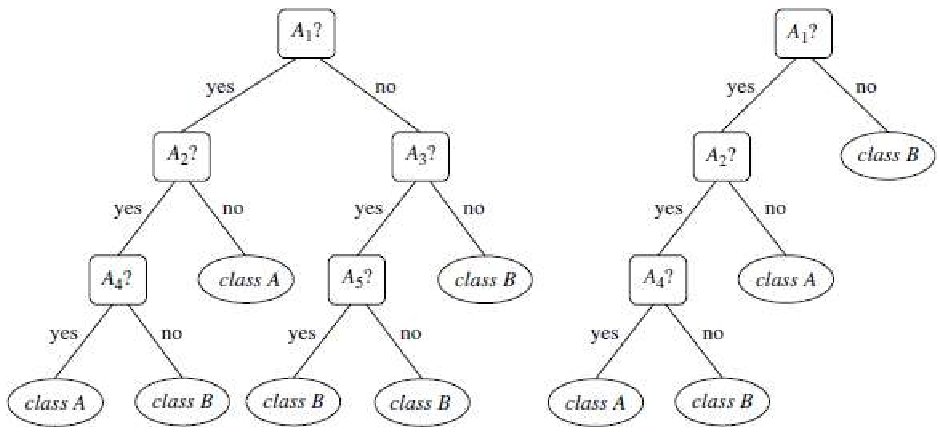


Fig: Unpruned and Pruned Trees

- 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.

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.

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

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

	buys_computer	
student	yes	no
yes	6	1
no	3	4

	buys_computer	
credit_ratting	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 $_2$ —	$_2$14

—

entropy$_{\textbf{buys\_computer}}$=-9/14*log(9/14)-5/14* log(5/14)

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)=-2/5*log$_2$(2/5)-3/5*log$_2$(3/5)=0.969

I(4,0)=-4/4*log$_2$(4/4)-0/4*log$_2$(0/4)=0

I(3,2)=-3/5*log$_2$(3/5)-2/5*log$_2$(2/5)=0.969

Entropy$_{age}$=5/14*0.969+0+5/14*0.969

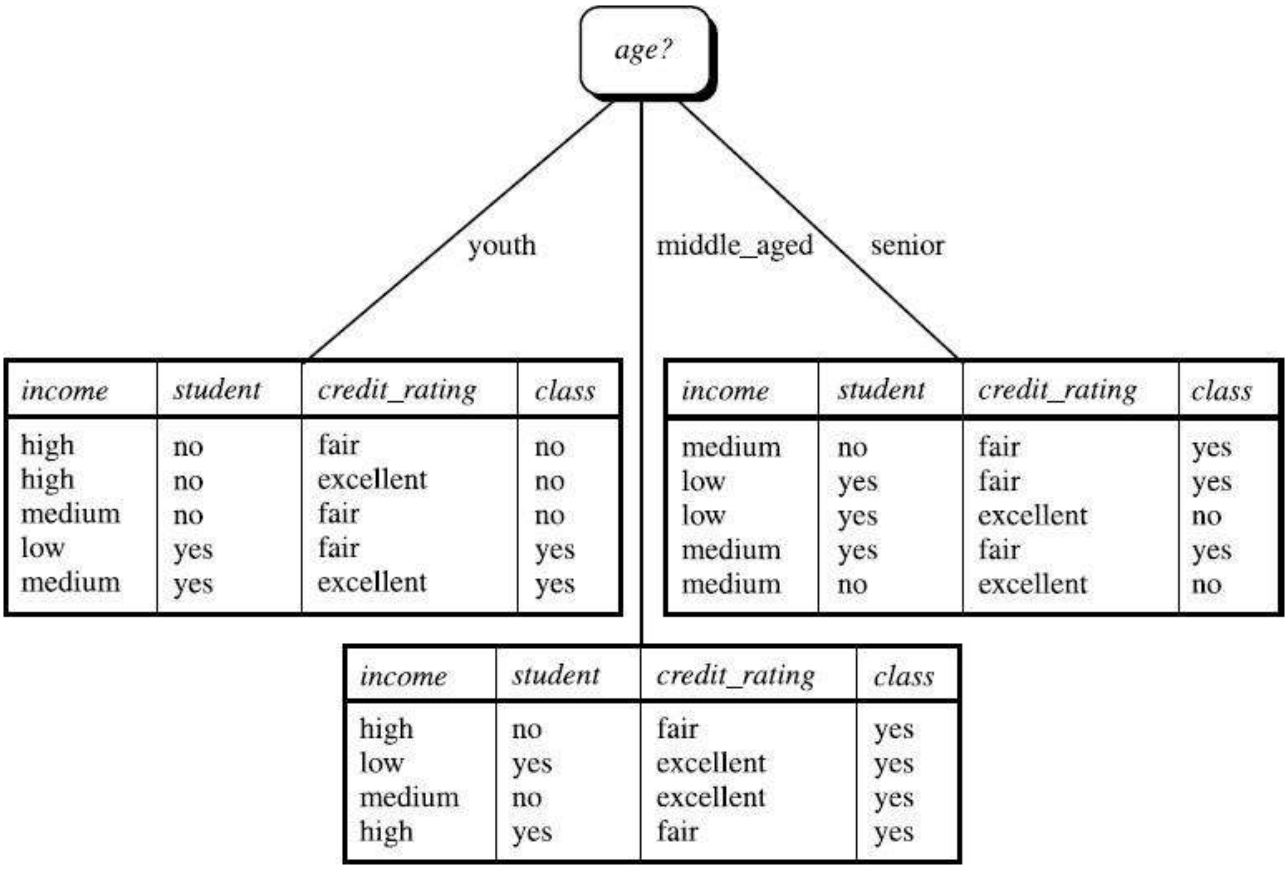
Age	P	N	TOTAL	I(P,N)	
youth	2	3	5	I(2,3	0.970
middle_aged	4	0	4	I(4,0)	0
senior	3	2	5	I(3,2)	0.970

Gain(Age) = Info(D) – Info$_{Age}$(D)
= 0.940 – 0693 = 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

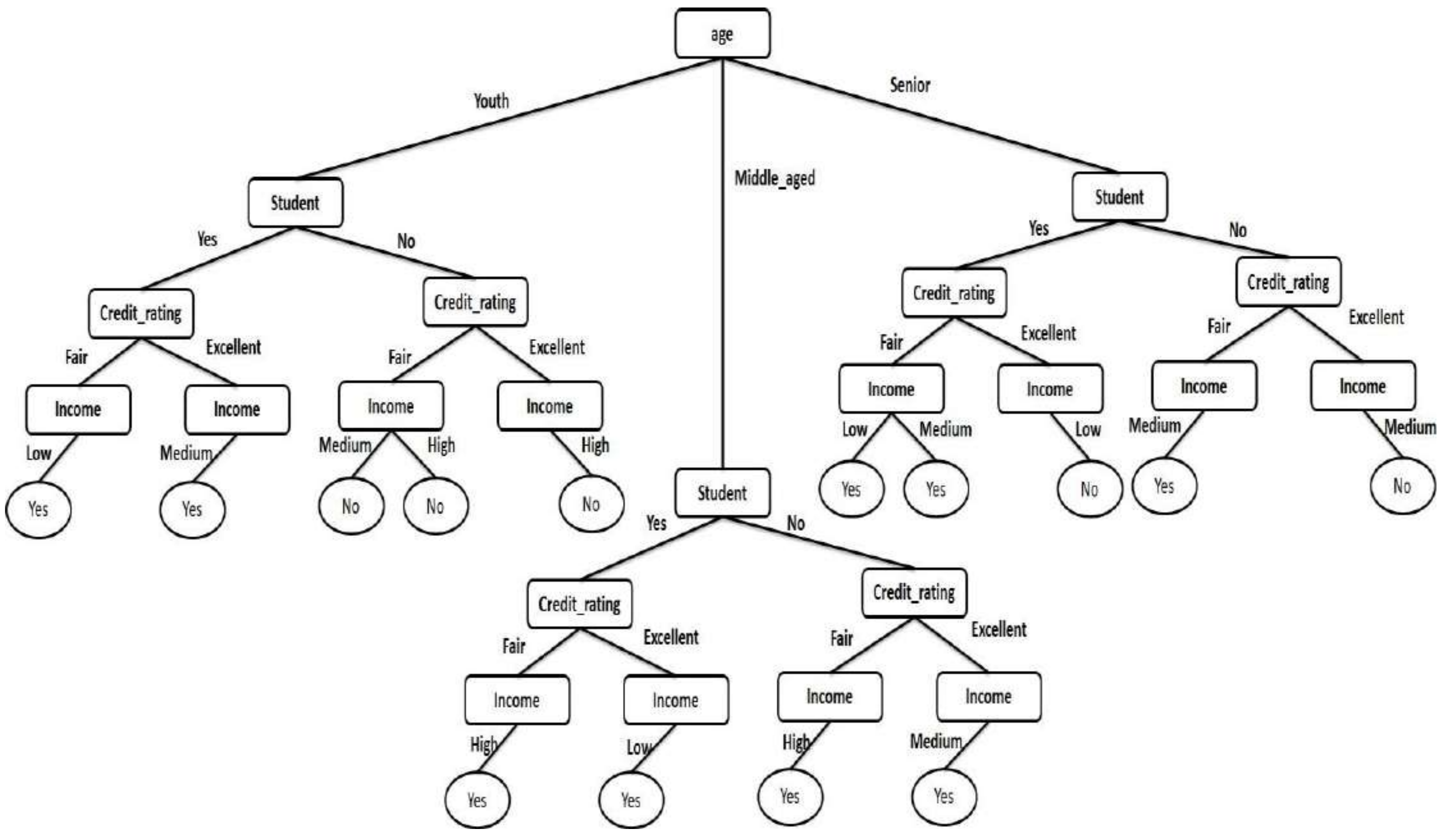


income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

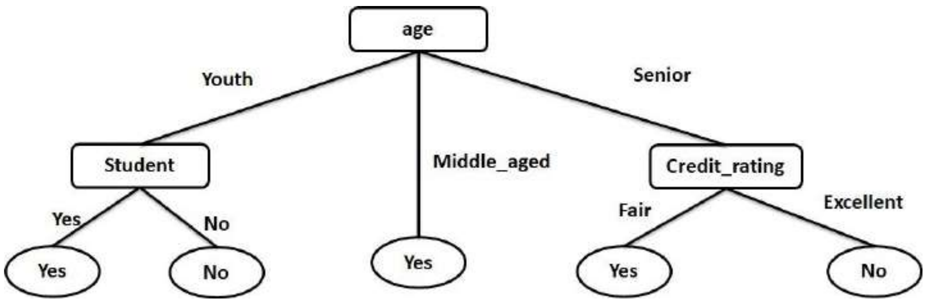
income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

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

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

Algorithm for Decision Tree Induction

A skeleton decision tree induction algorithm called Tree Growth is. The input to this algorithm consists of the training records E and the attribute set *F*. The algorithm works by recursively selecting the best attribute to split the data and expanding the leaf nodes.

Algorithm 4.1A skeleton decision tree induction algorithm.

```
Tree Growth ($E$,$F$)
1:if stopping cond($E$,$F$)=$true$ then
2:         $leaf$ =createNode().
3:         $leaf.label$=Classify($E$).
4:         return $leaf$.
5:else
6:         $root$=createNode().
7:         $root.testcond$=find_best_split($E$,$F$).
8:         let$V$ ={$v$|visapossibleoutcomeof$root.testcond$}.
9:         for each $v \in V$ do
10: $E_v$={$e$|$root.testcond$($e$)=$v$and$e \in E$}.
11: $child$= TreeGrowth($E_v$,$F$).
12: add$child$asdescendentof$root$andlabeltheedge($root \rightarrow child$)as$v$.
13:        end for
14: end if
15: return $root$.
```

EVALUATING THE PERFORMANCE OF A CLASSIFIER

The estimated **error helps** the **learning algorithm to do model** selection; i.e., to find a model of the **right complexity** that is not susceptible to overfitting. **Once the model** has been **constructed**, it can be applied to the **test set to predict** the class labels of **previously unseen records.**

Hold out Method

In the hold out method, the **original data** with labeled examples is **partitioned into two disjoint sets**, called **the training and the test sets**, respectively. A **classification model** is then induced from the **training set** and its **performance is evaluated on the test set.** The accuracy of the classifier can be estimated based on the **accuracy** of the induced model **on the test set.**
The **hold out method** has several **well-known limitations.**

First, **fewer** labeled **examples are available for training** because **some of the records are with-held for testing**. As a result, the induced **model may not be as good** as when **all the labeled examples are used for training.**
Second, the smaller the training set size, the **larger the variance of the model**. On the **other hand**, if the **training set is too large**, then the estimated **accuracy computed** from the **smaller test set is less reliable.**

Random Sub sampling

This approach is known as random sub sampling. Let acc_i be the model accuracy during the i^{th} iteration. The overall accuracy is given by $acc_{sub}=\sum_{i=1}^{k} acc_i/k$.
Random sub sampling still encounters some of the **problems associated** with the hold out method because **it does not utilize as much data as possible for training**. It also has **no control over the number of times each record is used for testing and training**. Consequently, some records might be used for training more often than others.

Cross-Validation

An **alternative to random subsampling** is **cross-validation**. In this approach, each record is used the same number of times for training and exactly once for testing. To illustrate this method, suppose **we partition the data into two equal sized sub sets**.
First, we choose **one** of the **sub sets for training and the other for testing**. We then **swap the roles** of the sub sets so that the previous training set becomes the test set and vice versa. This approach is called a **two-fold cross-validation.**
In this example, **each record is used exactly once for training and once for testing**. The *k*-fold cross-validation method generalizes this approach by segmenting the data in to *k* equal-sized partitions. During each run, one of the partitions is chosen for testing, while the rest of them are used for training. This procedure is repeated *k* times so that each partition is used for testing exactly once. The total error is found by summing up the errors for all *k* runs.

Boot strap

In the boot strap approach, the **training records are sampled with replacement**; i.e., a **record already chosen for training is put back in to the original pool** of records so that it is **equally likely to be redrawn.** If the original data has *N* records, it can be shown that, on average, a boot strap sample of size *N* contains about 63.2% of the records in the original data.
A record is chosen by a boot strap sample is $1-(1-1/N)^N$. When *N* is sufficiently large, the probability asymptotically approaches $1-e^{-1}=0.632$. Records that are not included in the boot strap sample become part of the test set. The model induced from the training set is then applied to the test set to

obtain an estimate of the accuracy of the boot strap sample, ϵ_i.

Accuracy,$acc_{boot}=1/b \sum_{i=1}^{b} (0.632\times\varrho_i+0.368\times acc_s)$.

MODEL OVERFITTING

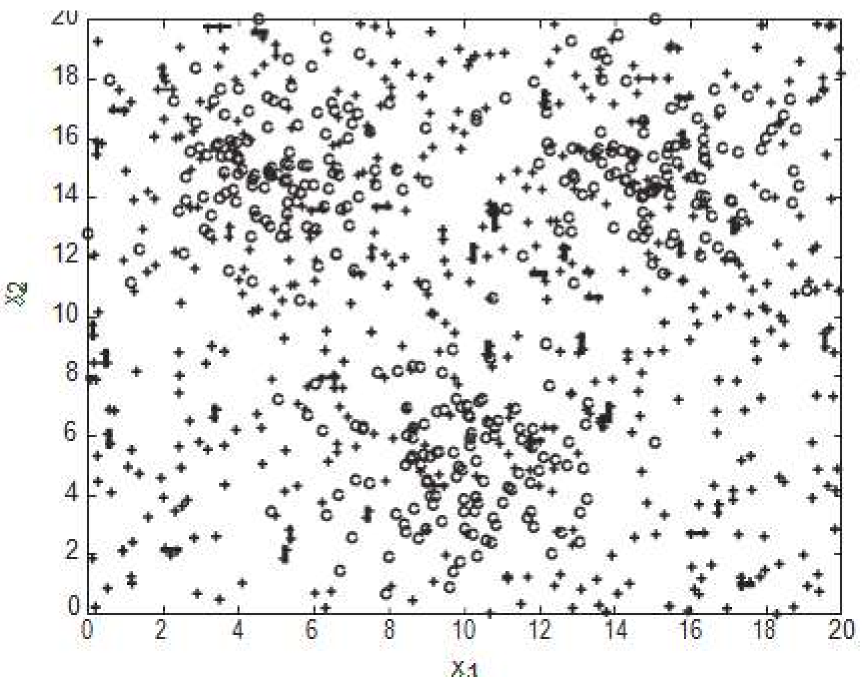
The **errors** committed by a **classification model** are generally divided **into two types: training errors and generalization errors.**

Training error, also known as resubstitution error or apparent error, **is the number of misclassification errors** committed **on training records**, where as **generalization error is the expected error of the model on previously unseen records.**

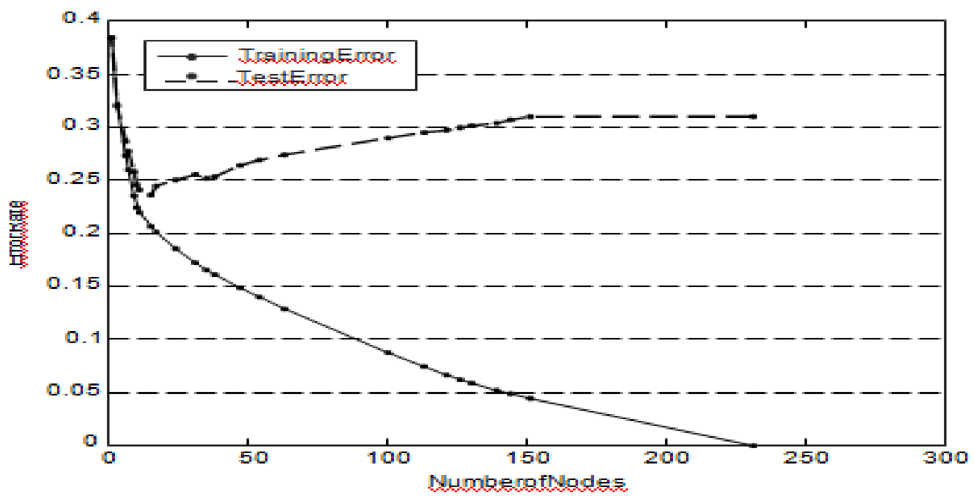
A good classification model must not only fit the training data well; it must also accurately classify records. **A good model** must have **low training error as well as low generalization error**. This is important because a model that **fits the training data too** well can have a **poorer generalization error than a model with a higher training error**. Such a **situation** is known as model **overfitting**.

Overfitting Example in Two-Dimensional Data

Consider the two-dimensional datasets shown below. The **dataset contains data points** that belong to two different classes, denoted as **class o and class +,** respectively.

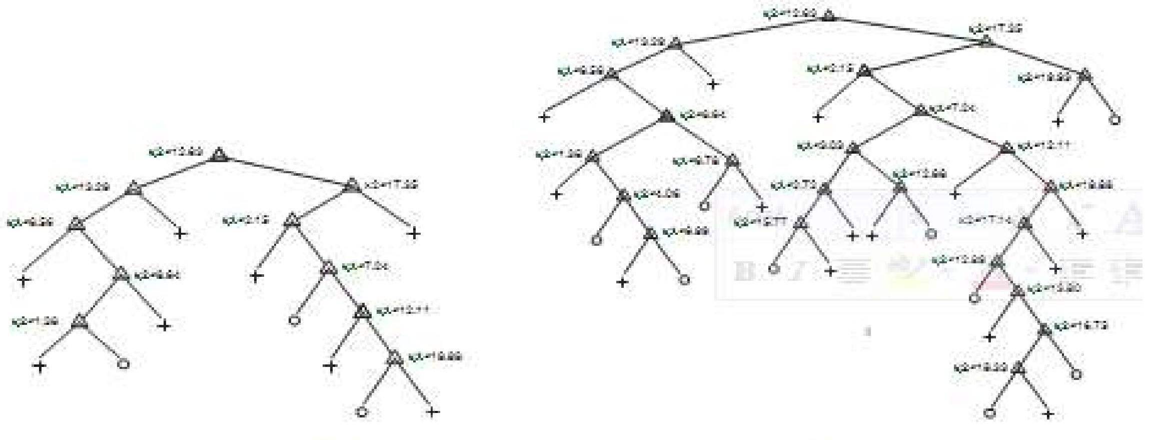


The data points for the **o class** are generated from a mixture of three Gaussian distributions, while a uniform distribution is used to generate the data points for the **+ class**. There are **altogether 1200 points belonging to the o class and 1800 points be-longing to the + class 30% of the points are chosen for training, while the remaining 70% are used for testing.** To investigate the effect of overfitting, different levels of pruning are applied to the initial, fully-grown tree. The below figure shows the **training and test error rates of the decision tree.**



The training and test error rates of the model are **large when the size of the tree is very small. This situation is known as model underfitting**. **As the number of nodes in the decision tree increases, the tree will have fewer training and test errors**. However, **once the tree becomes too large, its test error rate begins to increase even though its training error rate continues to decrease.** This phenomenon is **known** as **model overfitting.**

The below figure shows the structure of **two decision trees with different number of nodes**. The tree that contains the **smaller number of nodes has a higher training error rate**, but a **lower test error rate compared to the more complex tree.**



(a) Decisiontree with11 leafnodes. (b) Decisiontreewith24leafnodes.

Overfitting Due to Presence of Noise

Consider the training and test sets shown in below Tables for **the mammal classification problem**. **Two of the ten training records are mislabeled**: **bats and whales are classified as non-mammals instead of mammals.**

A decision tree that perfectly fits the training data is shown in below figure. Although **the training error for the tree is zero**, its **error rate on the test set is 30%.**

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warmblooded	yes	yes	no	yes
bat	warmblooded	yes	no	yes	no*
whael	warmbloodedc	yes	no	no	no*
salamander	oldblooded	no	yes	yes	no
komododragon	coldblooded	no	yes	no	no
python	coldblooded	no	no	yes	no
salmon	coldblooded	no	no	no	no
eagleg	warmbloodedc	no	no	no	no
uppy	old-blooded	yes	no	no	no

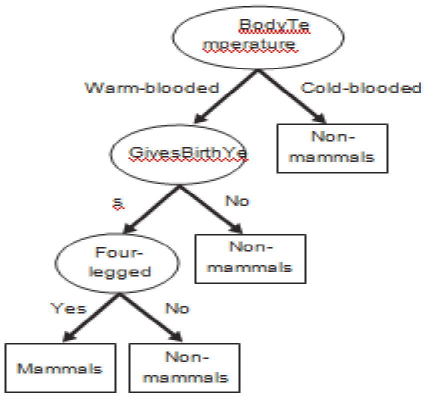
An example training set for classifying mammals. Class labels with asterisk symbols represent mislabeled records.

[illegible]

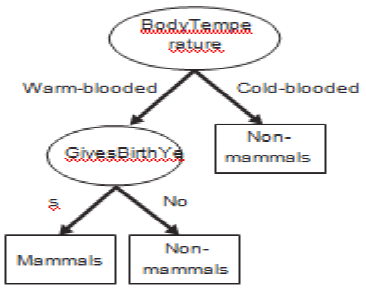
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopardshark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gilamonster	cold-blooded	no	yes	yes	no

An example test set for classifying mammals

Both humans and dolphins were misclassified as non-mammals because their **attribute values for Body Temperature,** Gives Birth, and Four-legged are identical to them is labeled records in the training set. On the other hand, represent an exceptional case in which the class label of a test record contradicts the class labels of other similar records in the training set. Errors due to exceptional cases are often unavoidable and establish the minimum error rate achievable by any classifier.



(a) ModelM1



(b)ModelM2

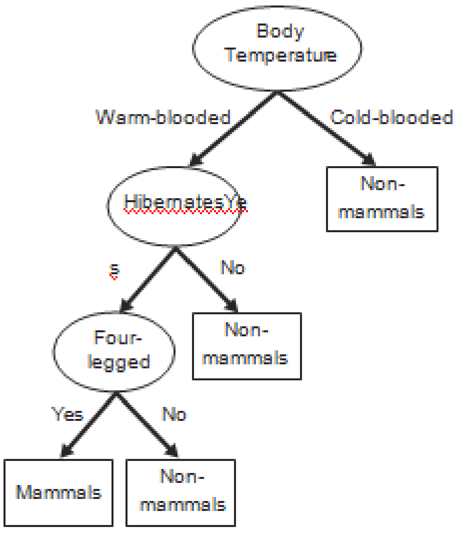
Model 1 misclassifies humans and dolphins as non mammals. **Model 2 has a lower test error rate (10%) even though its training error rate is higher (20%).**

Overfitting Due to Lack of Representative Samples

Models that make their classification decisions based on a **small number of training records** are also **likely to overfitting.** Such models can be generated because of **lack of representative samples in the training data and learning algorithms** that **continue to refine their** models even **when few training records are available.**

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
salamander	cold-blooded	no	yes	yes	no
guppy	cold-blooded	yes	no	no	no
eagle	warm-blooded	no	no	no	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes

Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded **vertebrates that do not hibernate as non-mammals. The tree arrives at this classification decision because there is only one training record, which is an eagle, with such characteristics. This example clearly demonstrates the danger of making wrong predictions when there are not enough representative examples at the leaf nodes of a decision tree.**



UNIT 4 Classification

Syllabus:

Classification: Alterative Techniques, Bayes' Theorem, Naïve Bayesian Classification, Bayesian Belief Networks

Bayesian Classifiers:

- In **many applications** the **relationship between the attribute set** and the **class variable is non-deterministic.**
- The **class label of a test record cannot be predicted even though its attribute set is identical** to **some of the training examples**
- **For example**, consider the **task of predicting whether a person is at risk for heart disease** based on the **person's diet and workout frequency.**
- Although **most people** who **eat healthily** and **exercise regularly** have **less chance of developing heart disease.**
- Still do so because of **other factors** such as **heredity**, **excessive smoking, and alcohol** abuse.
- Determining whether a **person's diet is healthy** or the **workout frequency** is **sufficient is also subject to interpretation.**
- In which in turn **may introduce uncertainties into the learning problem.**
- **An approach** for modeling **probabilistic relationships between the attribute set** and the **class variable**.
- The **Bayes theorem, a statistical principle** for **combining prior knowledge of the classes** with **new evidence gathered from data**.
- The use of the **Bayes theorem** for **solving classification problems** will be explained.
- It is followed by a description of **two implementations of Bayesian classifiers**: **naive Bayes and the Bayesian belief network**.

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).

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

Problem:

- Consider a **football game** between two rival teams: **Team 0 and Team 1**. Suppose **Team 0 wins 65%** of the time and **Team 1 wins the remaining matches**.
- Among the games **won by Team 0**, only **30%** of them come from playing on **Team 1's football field**. On the other hand, **75% of the victories for Team 1** are obtained while **playing at home**.
- If **Team 1 is to host the next match** between the two teams, **which team** riff most likely emerge as the **winner**?

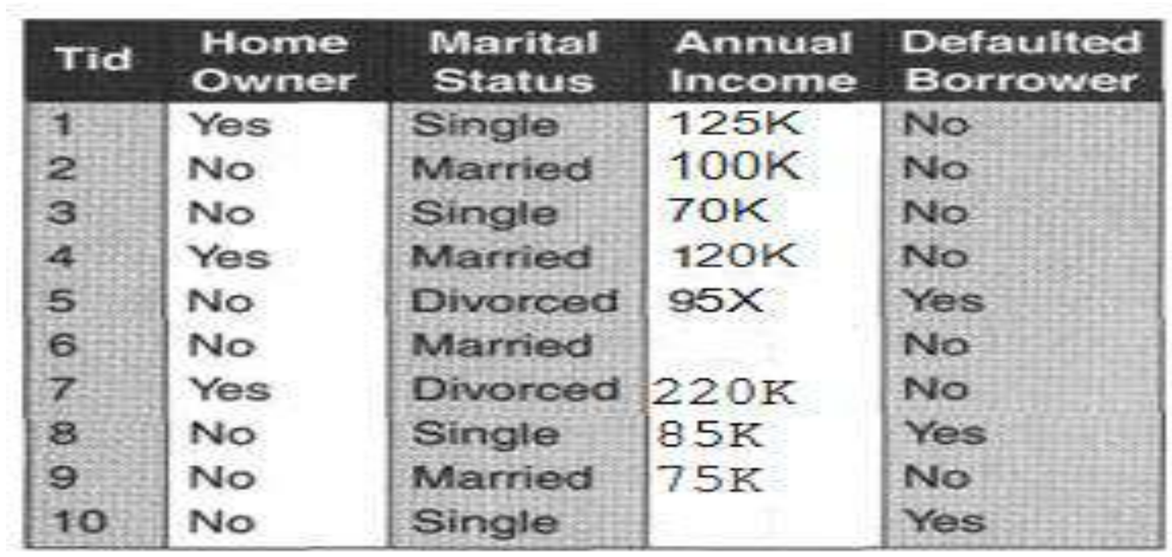
Solution:

- The **Bayes theorem can be used to solve the prediction problem**.
- For notational convenience, let **A** be the random variable that represents the **team hosting the match** and **V** be the random variable that represents the **winner of the match**. **Both A and V** can take on **values f**rom the set **(0, 1).**
- Probability Team 0 wins is $P(Y = 0) = 0.65$.
- Probability Team 1 wins is $P(V - 1) = 1 — P(V — 0) = 0.35$.
- Probability Team 1 hosted the match it won is $P(A = 1|V = 1) = 0.75$.
- Probability Team 1 hosted the match won by Team 0 is $P(A = 1|V = 0) = 0.3$.
- Our objective is to compute $P(V = 1|A = 1)$, which is the conditional probability that Team 1 wins the next match it will be hosting, and compares it against $P(V = 0|A = 1)$.
- Using the Bayes theorem, we obtain

$$
\begin{aligned}
P(Y = 1|X = 1) &= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1)} \\
&= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1, Y = 1) + P(X = 1, Y = 0)} \\
&= \frac{P(X = 1|Y = 1) \times P(Y = 1)}{P(X = 1|Y = 1)P(Y = 1) + P(X = 1|Y = 0)P(Y = 0)} \\
&= \frac{0.75 \times 0.35}{0.75 \times 0.35 + 0.3 \times 0.65} \\
&= 0.5738,
\end{aligned}
$$

Using the Bayes Theorem for Classification

- The **Bayes theorem** can be used for **classification**, Let **X denote the attribute set** and **Y denote the class variable.**
- The **relationship with the attributes**, then we can treat **X and Y as random variables** and **capture their relationship probabilistically using *P(Y/X).***
- This **conditional probability is also known** as the **posterior probability** for Y, as **opposed to** its **prior probability,** *P(Y).*
- **During the training phase**, we need to learn the **probabilities P(Y|X) for every combination** of **X and Y** based on information gathered **from the training data.**
- By knowing these probabilities, **a test record X**’ can be **classified by finding the class Y'** is $P(Y^1/X^1)$.
- To **illustrate this approach,** consider the task of **predicting** whether a **loan borrower will default on their payments.**
- The below Figure shows **a training set** with the following attributes: **Home Owner, Marital status, and Annual Income.**
- **Loan borrowers who defaulted** on their payments are **classified as Yes**, while those who **repaid their loans are classified as No.**



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

Dia: Training set for predicting the loan default problem.

- Suppose we are given a **test record** with the following attribute set: **X = (Home Owner = No, Marital Status = Married, Annual Income = $120K).**
- **To classify the record**, we need to compute the **posterior probabilities P(Yes |X) and P(No |X) based on information available in the training data**.
- If **P(Yes |X) > P(No |X),** then the record is **classified as Yes; otherwise, it is classified as No**.
- The **Bayes theorem** is useful because it allows us to express the **posterior probability** in terms of the **prior probability *P(Y),*** the class-conditional probability P(X|Y), and the evidence, P(X):

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y) \times P(Y)}{P(\mathbf{X})}.$$

Naive Bayes Classifier

- A **naive Bayes classifier** estimates the class-conditional probability by assuming that the **attributes are conditionally independent**, given the **class label p**.
- The conditional independence assumption can be formally stated as follows:

$$P(\mathbf{X}|Y=y) = \prod_{i=1}^{d} P(X_i|Y=y),$$

- Where each **attribute set** X = {x1,x2,…….xd} consists of ***d* attributes.**

Conditional Independence

- **Before delivering to naïve baye's classifier** working , we should **know** about **conditional independence**.
- Let us examine the **notion of conditional independence.**
- Let **X, Y, and Z** denote **three sets of random variables**.
- The variables in **X are said to be conditionally independent of Y, given Z,** if the **following condition holds:**

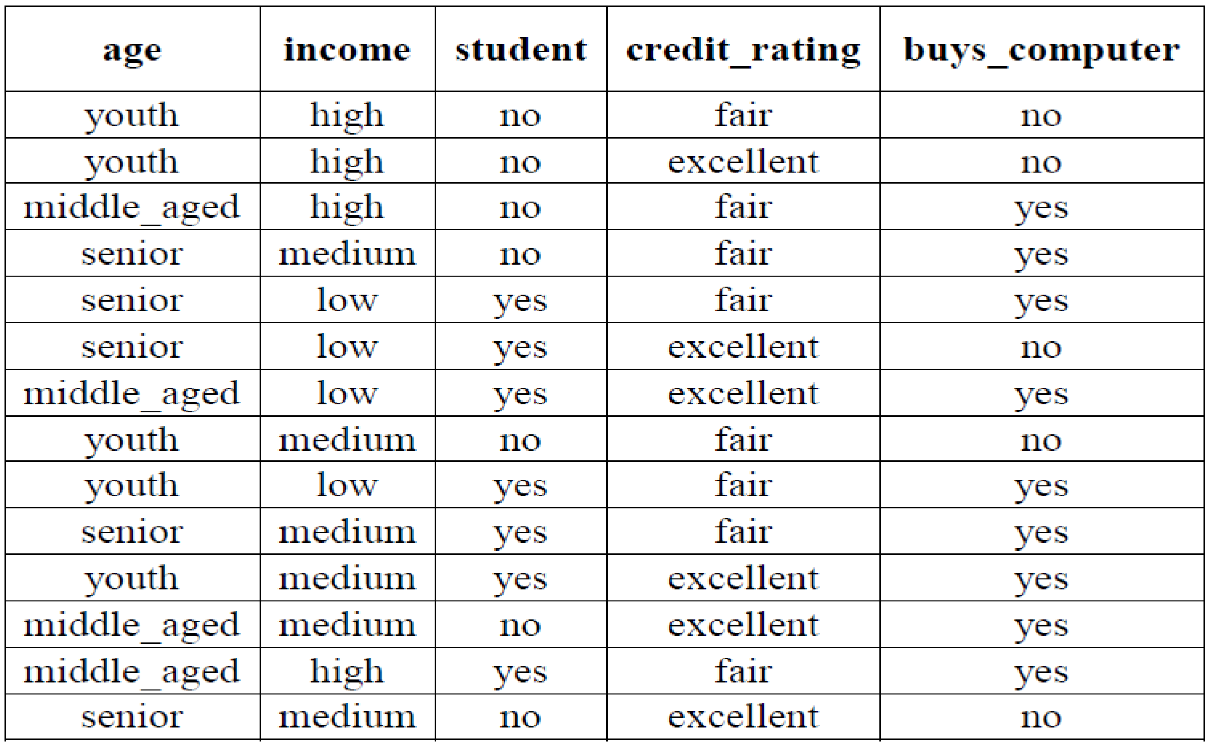
  P(X|Y, Z) = P(X|Z).
- The **conditional independence between X and Y** can also be written into a form.

$$\begin{aligned} P(\mathbf{X},\mathbf{Y}|\mathbf{Z}) &= \frac{P(\mathbf{X},\mathbf{Y},\mathbf{Z})}{P(\mathbf{Z})} \\ &= \frac{P(\mathbf{X},\mathbf{Y},\mathbf{Z})}{P(\mathbf{Y},\mathbf{Z})} \times \frac{P(\mathbf{Y},\mathbf{Z})}{P(\mathbf{Z})} \\ &= P(\mathbf{X}|\mathbf{Y},\mathbf{Z}) \times P(\mathbf{Y}|\mathbf{Z}) \\ &= P(\mathbf{X}|\mathbf{Z}) \times P(\mathbf{Y}|\mathbf{Z}), \end{aligned}$$

How a Naive Bayes Classifier Works:

- Instead of computing the class-conditional probability for every combination of X, we only have to **estimate the conditional probability of each Xi, given Y.**
- It **does not require** a **very large training set to obtain** a **good estimate of the probability**.
- To **classify a test record**, the **naive Bayes classifier computes the posterior probability** for each class Y:

$$P(Y|\mathbf{X}) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(\mathbf{X})}.$$



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 **C1 correspond** to the class **buys computer=yes** and **C2 correspond** to **buys 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) × P(income=medium | buys computer=yes) × P(student=yes | buys computer=yes) × P(credit rating=fair | buys computer=yes) = 0.222 × 0.444 × 0.667 × 0.667

= 0.044.

- Similarly, *P(**X | buys computer=no) = 0.600 × 0.400 × 0.200 × 0.400 = 0.019.** To find the class, Ci, that* ***P(X|Ci)P(Ci),***
- ***we compute***

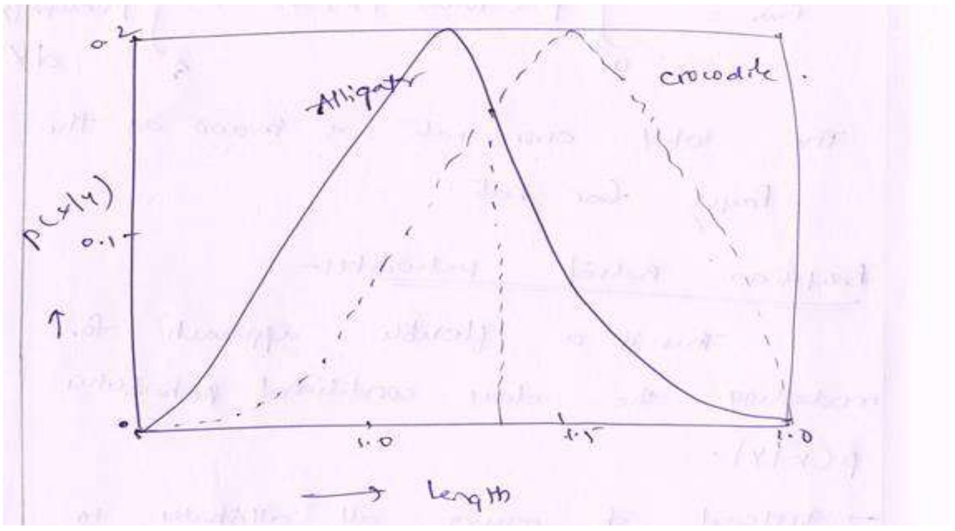
P(X | buys computer=yes) P(buys computer=yes) = 0.044 × 0.643 = 0.028

P(X | buys computer=no) P(buys computer=no) = 0.019 × 0.357 = 0.007

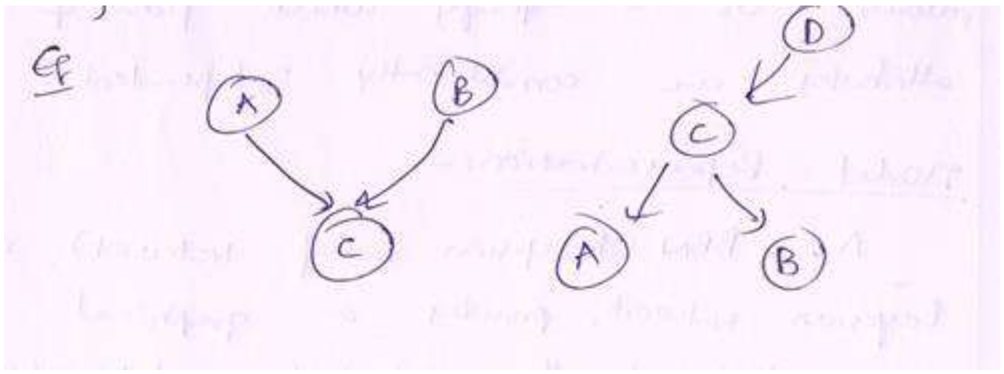
- ***Therefore, the naïve Bayesian classifier predicts buys computer = yes for tuple X.***

Baye's error rate:

- **Bayesian classification method** allow us to determine the ideal **decision boundary for classification task.**



- Probabilistic relationships are represented using DAG(Directed Acyclic graph)



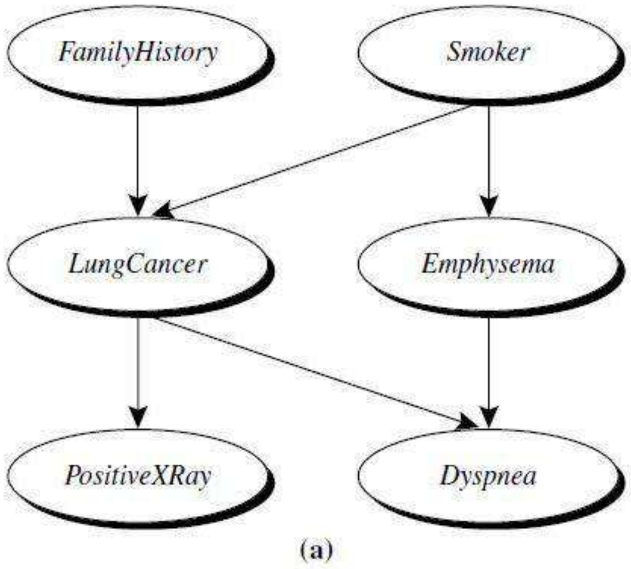
- **Error rate of classifier** is represented by the **sum of area under postirior probability curve** for **crocodiles and alligators**.

$$\text{Error} = \int_{0}^{\hat{x}} p(\text{crocodile} \mid x)\, dx + \int_{\hat{x}}^{\infty} p(\text{Alligator} \mid x)\, dx$$

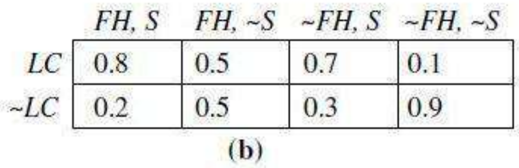
- The **total error rate** is known as the **bayes error rate** .

Bayesian Belief Networks:

- Bayesian Belief Network or Bayesian Network or Belief Network is a **Probabilistic Graphical Model** (PGM)
- It represents **conditional dependencies between random variables through a Directed Acyclic Graph (DAG).**
- **Bayesian belief networks** —**probabilistic graphical model**s, **which unlike naïve Bayesian classifiers**
- **It** allow the representation of **dependencies among subsets of attributes .**
- The **naïve Bayesian classifier** makes the **assumption of class conditional independence.**
- **It means** 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 .**
- **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.***



(a)



	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

(b)

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 cance****r 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.**

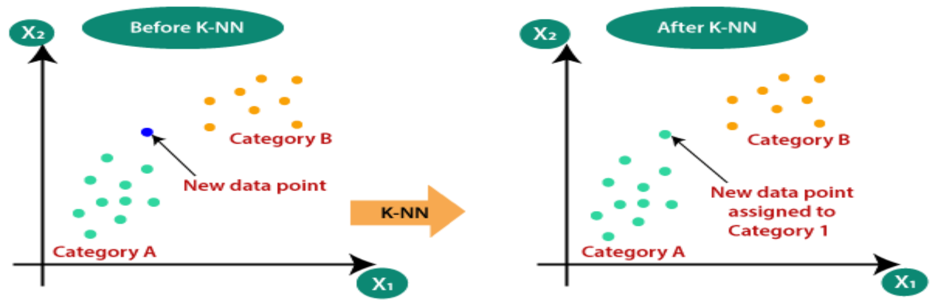
K nearest neighbour:

- **K-Nearest Neighbour** is one of the **simplest Machine Learning algorithms** based on **Supervised Learning technique**.
- K-NN algorithm assumes the **similarity between the new case/data** and available cases and **put the new case into the category** that is most similar to the available categories.

Why do we need a K-NN Algorithm?

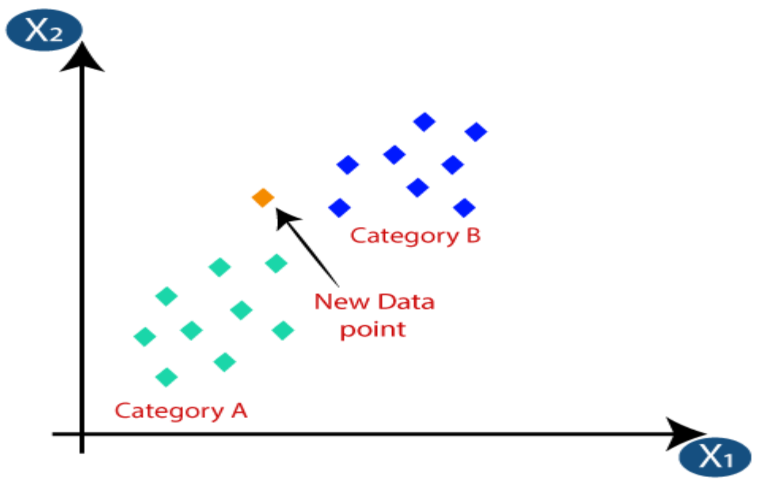
- Suppose there are **two categories**, i.e., **Category A and Category B**, and we have a **new data point x1**, so this data point will lie in which of these categories.
- To **solve this type of problem**, we **need a K-NN** algorithm.
- With the **help of K-NN, we can easily identify the category** or class of a particular dataset.

Consider the below diagram:

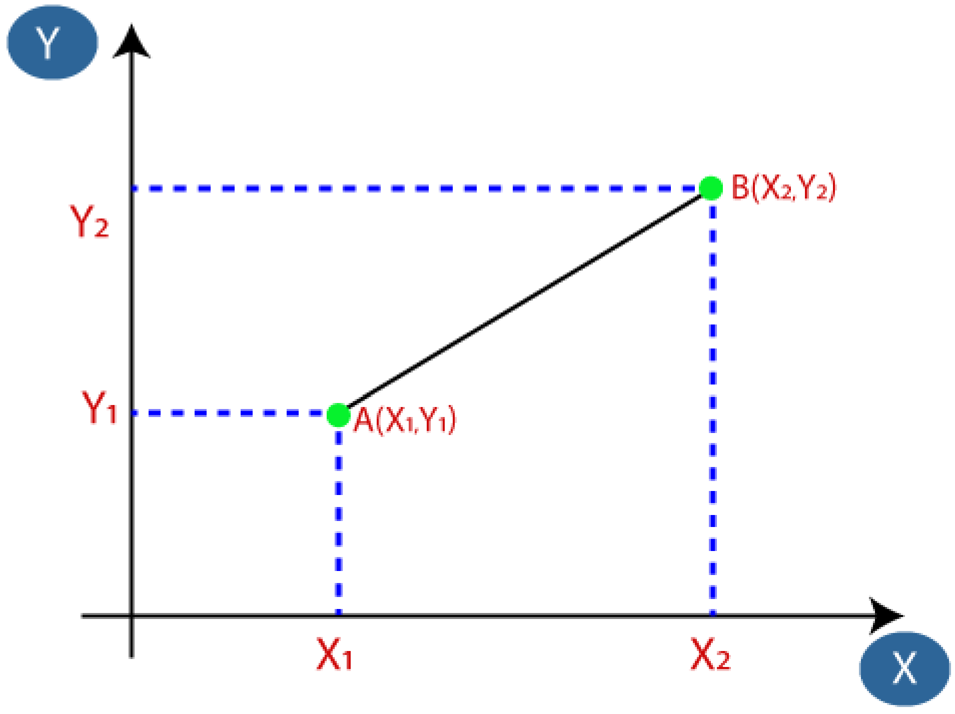


How does K-NN work?

- **Step-1: Select the number K** of the neighbors
- **Step-2: Calculate** the **Euclidean distance** of **K number of neighbors**
- **Step-3:** Take the **K nearest neighbors** as per the **calculated Euclidean distance.**
- **Step-4:** Among these **k neighbors, count the number of the data points in each category.**
- **Step-5: Assign the new data points to** that **category** for **which the number of the neighbor is maximum**.
- **Step-6:** Our model is ready.

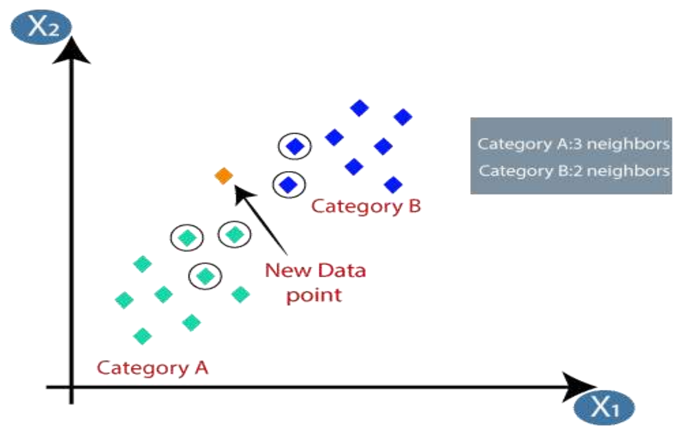


- Firstly, we will choose the **number of neighbors**, so we will **choose the k=5**.
- Next, we will **calculate** the **Euclidean distance** between the data points.
- The **Euclidean distance** is the **distance between two points.**



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

- By calculating the Euclidean distance we got the nearest neighbors, as **three nearest neighbors in category A** and **two nearest neighbors in category B.**



- As we can see the **3 nearest neighbors are from category A**, hence this **new data point** must **belong to category A.**

<u>How to select the value of K in the K-NN Algorithm?</u>

- Below are some points to remember while selecting the **value of K in the K-NN algorithm**:
- There is **no particular way to determine the best value for "K",** so we need to try some values to find the best out of them. **The most preferred value for K is 5.**
- A **very low value for K such as K=1 or K=2**, can be noisy and lead to the effects of **outliers in the model.**
- **Large values for K** are **good,** but it **may find some difficulties**

<u>Advantages of KNN Algorithm:</u>

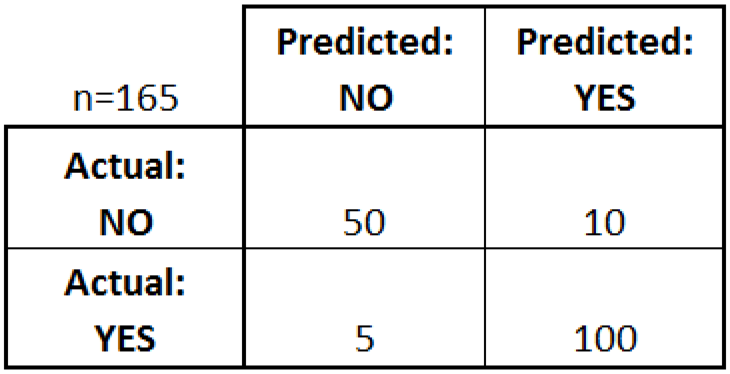
- It is **simple to implement**.
- It is **robust to the noisy training data**
- It can be **more effective** if **the training data is large.**

Disadvantages of KNN Algorithm:

- Always needs to **determine the value of K** which may be **complex some time.**
- The **computation cost is high** because of **calculating the distance** between the data points for **all the training samples.**

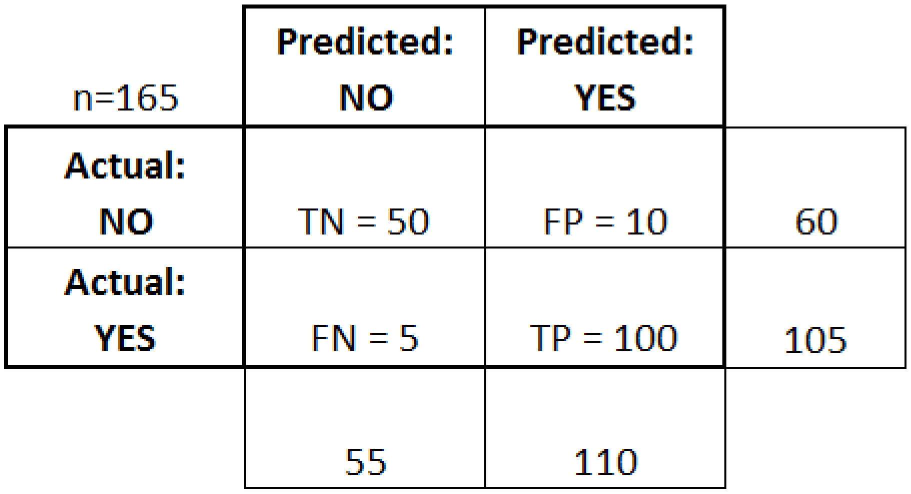
What is misclassification rate of a classifier? Describe sensitivity and specificity measures of a classifier

A) A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier")



n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")



n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- **Accuracy:** Overall, how often is the classifier correct?
    - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
    - (FP+FN)/total = (10+5)/165 = 0.09

- equivalent to 1 minus Accuracy
- also known as "Error Rate"

What Is Sensitivity

- Sensitivity is a measure of the proportion of actual positive cases that got predicted as positive (or true positive). Sensitivity is also termed as Recall. This implies that there will be another proportion of actual positive cases, which would get predicted incorrectly as negative (and, thus, could also be termed as the false negative).

Mathematically, sensitivity can be calculated as the following:
Sensitivity = (True Positive)/(True Positive + False Negative)

What Is Specificity?

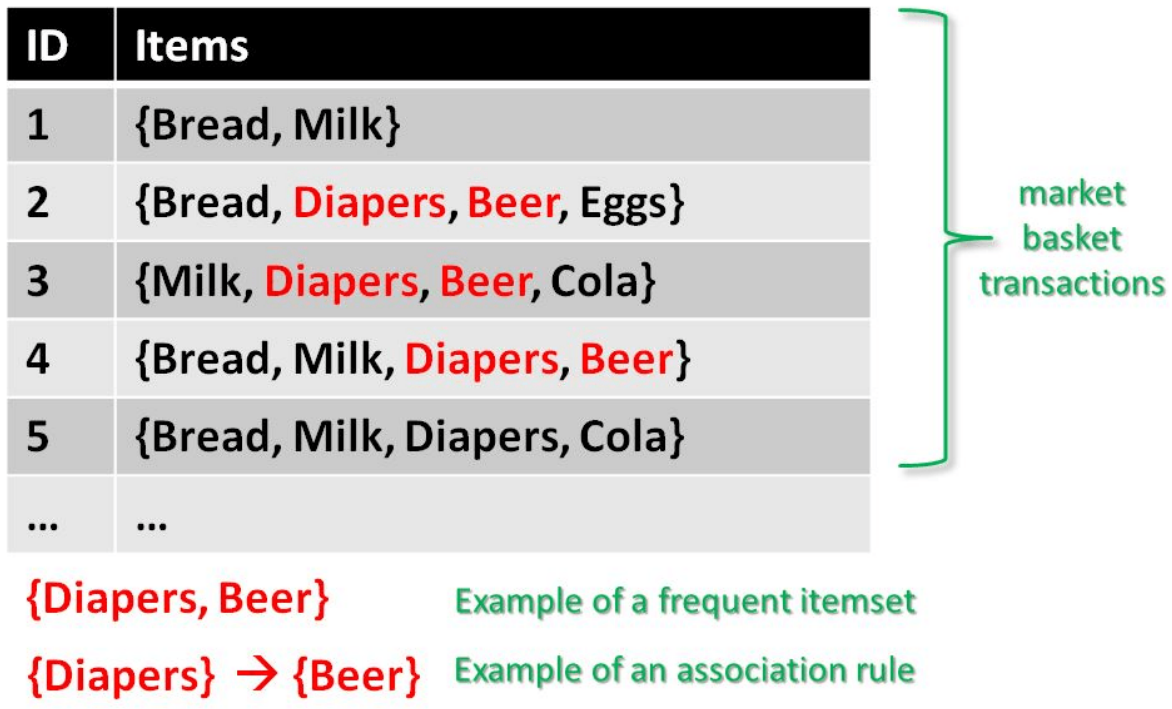
Specificity is defined as the proportion of actual negatives, which got predicted as the negative (or true negative). This implies that there will be another proportion of actual negative, which got predicted as positive and could be termed as false positives. This proportion could also be called a false positive rate. The sum of specificity and false positive rate would always be 1.
Mathematically, specificity can be calculated as the following:
Specificity = (True Negative)/(True Negative + False Positive)

Association Analysis:

Syllabus: Basic Concepts and Algorithms: Problem Defecation, Frequent Item Set generation, Rule generation, compact representation of frequent item sets, FP-Growth Algorithm.

Association Analysis:

Association analysis, which is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of association rules or sets of frequent items. For example, the following rule can be extracted from the data set shown in below Table:



ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}
…	…

market basket transactions
{Diapers, Beer} Example of a frequent itemset
{Diapers} → {Beer} Example of an association rule

The rule suggests that a strong relationship exists between the sale of diapers and beer because many customers who buy diapers also buy beer. There are two key issues that need to be addressed when applying association analysis to market basket data. Second, some of the discovered patterns are potentially spurious because they may happen simply by chance.

1. PROBLEM DEFINITION

The basic terminology used in association analysis:

Binary Representation: Market basket data can be represented in a binary format as shown in below Table, where each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Itemset and Support Count Let I = {i1,i2,….,id} be the set of all items in a market basket data and T = {t1,t2,….,tN} be the set of all transactions. In association analysis, a collection of zero or more items is termed an itemset. If an itemset contains *k* items, it is called a k-itemset. For instance, **{Beer, Diapers, Milk**} is an example of a 3-itemset. The null (or empty) set is an itemset that does not contain any items.

The transaction width is defined as the number of items present in a transaction. A transaction t_i is said to contain an itemset X if X is a subset of tj. For example, the second transaction shown in the above table contains the itemset **{Bread, Diapers}** but not **{Bread, Milk}.** Its support count, which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, σ(X), for an itemset A can be stated as follows:

$$\sigma(X) = \left|\{t_i | X \subseteq t_i,\ t_i \in T\}\right|,$$

Where the symbol |. | denote the number of elements in a set. In the data set shown in the above table, the support count for **{Beer, Diapers, Milk}** is equal to two because there are only two transactions that contain all three items.

Association Rule An association rule is an implication expression of the form X→Y, 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 definitions of these metrics are

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

Consider the rule **{Milk, Diapers} →{Beer}**. Since the support count for **{Milk, Diapers, Beer}** is 2 and the total number of transactions is 5, the rule's support is 2/5 = 0.4. The rule's confidence is obtained by dividing the support count for **{Milk, Diapers, Beer}** by the support count for **{Milk, Diape**rs}. Since there are 3 transactions that contain milk and diapers, the confidence for this rule is 2/3 = 0.67.

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 uninteresting from a business perspective because it may not be profitable to promote items that customers

Confidence, on the other hand, measures the reliability of the inference made by a rule. For a given rule X —› 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.

Formulation of Association Rule Mining Problem The association rule mining problem can be formally stated as follows:

Association Rule Discovery Given a set of transactions T, find all the rules having support ≥ minsup and confidence ≥ minconf, where minsup and minconf are the corresponding support and confidence thresholds. A brute-force approach for mining association rules is to compute the support and confidence for every possible rule.

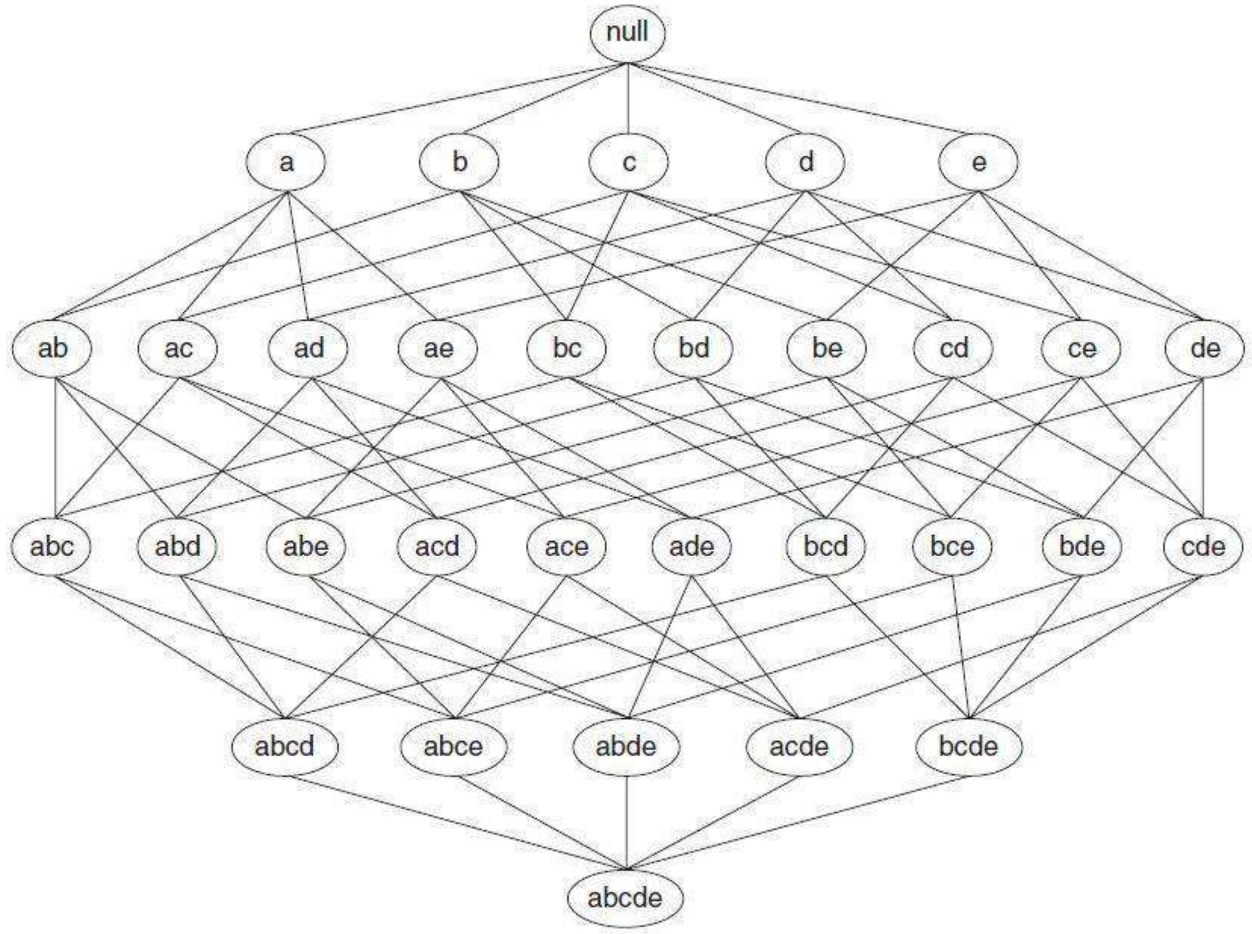
$$R = 3^d - 2^{d+1} + 1.$$

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.

1.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****.*



An itemset lattice.

To find **frequent item** sets we have two algorithms,

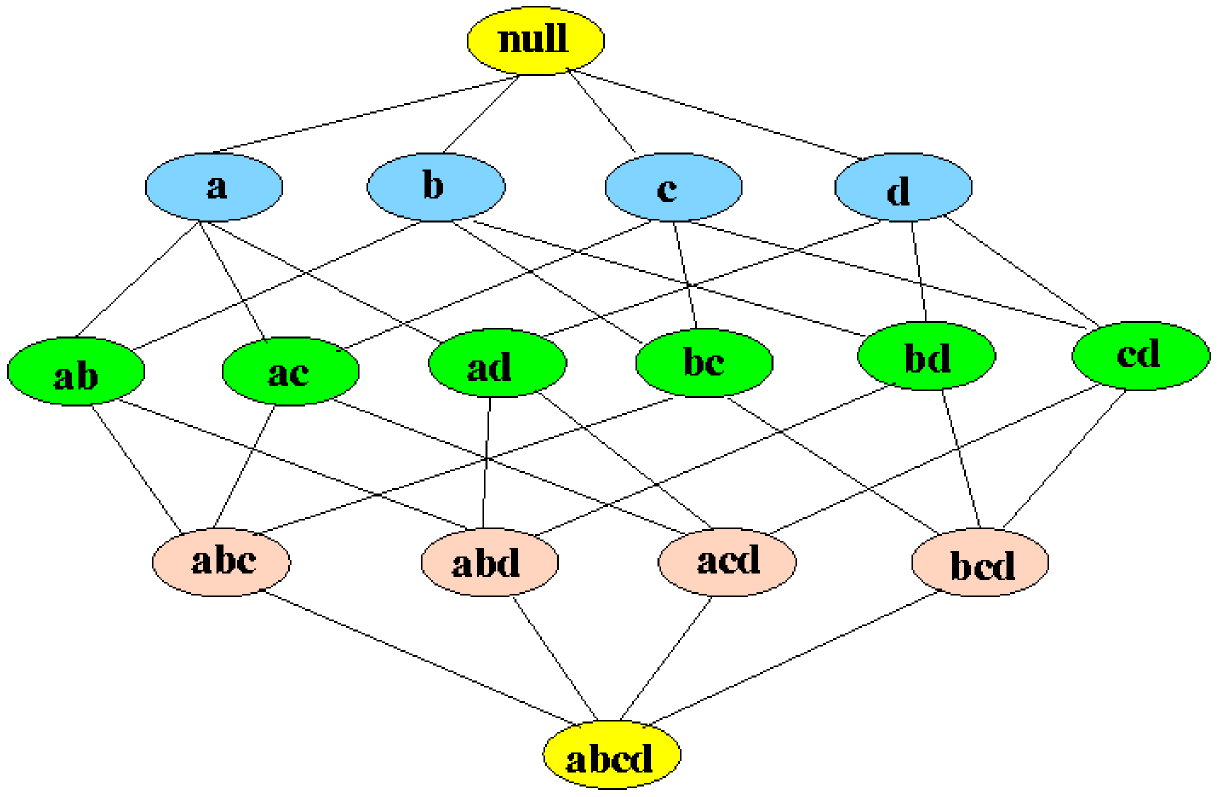
- a) **Apriori Algorithm**
- b) **FP-Growth**

Apriori Principle

- If **an item set is frequent** , then all of **its subsets must also be frequent.**
- If an **item set is infrequent** then all of its **supersets must be infrequent.**

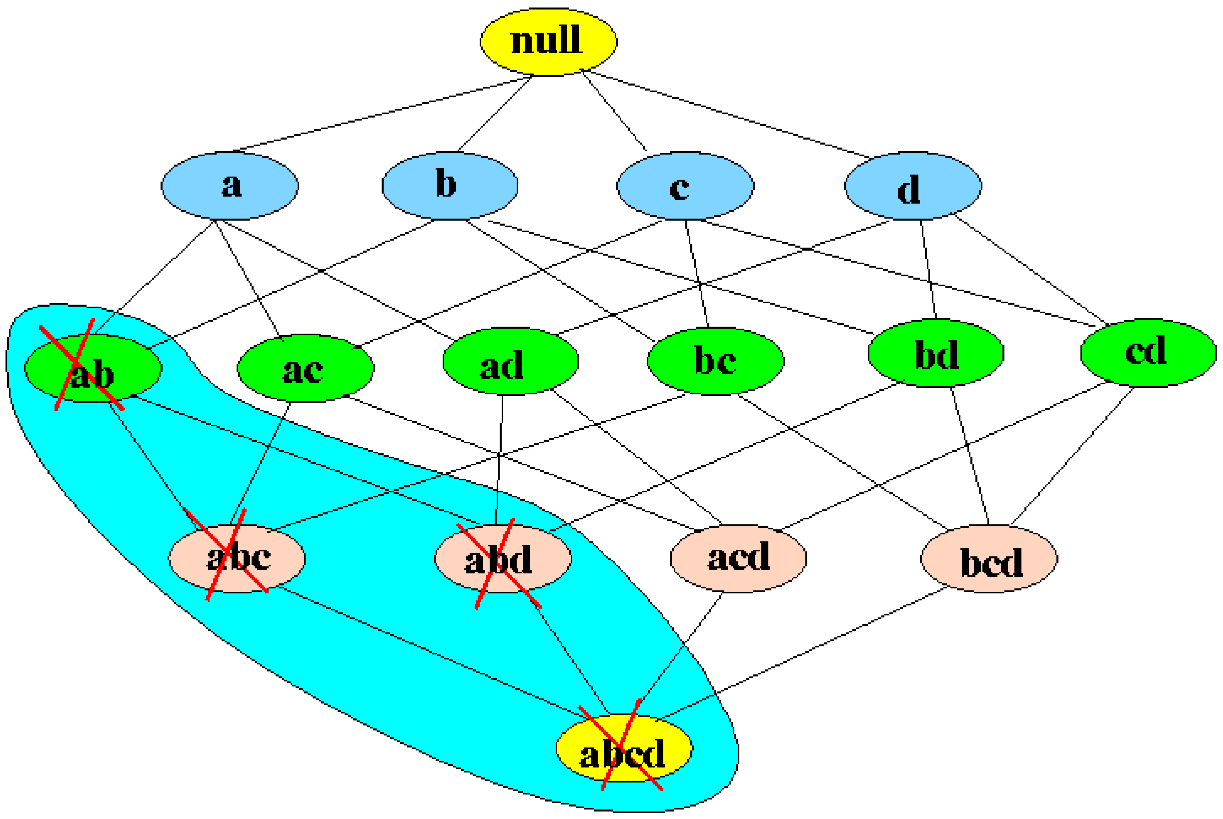
The **bottom is all the items in the transaction** data and then **you start moving upwards creating**

subsets till the null set.



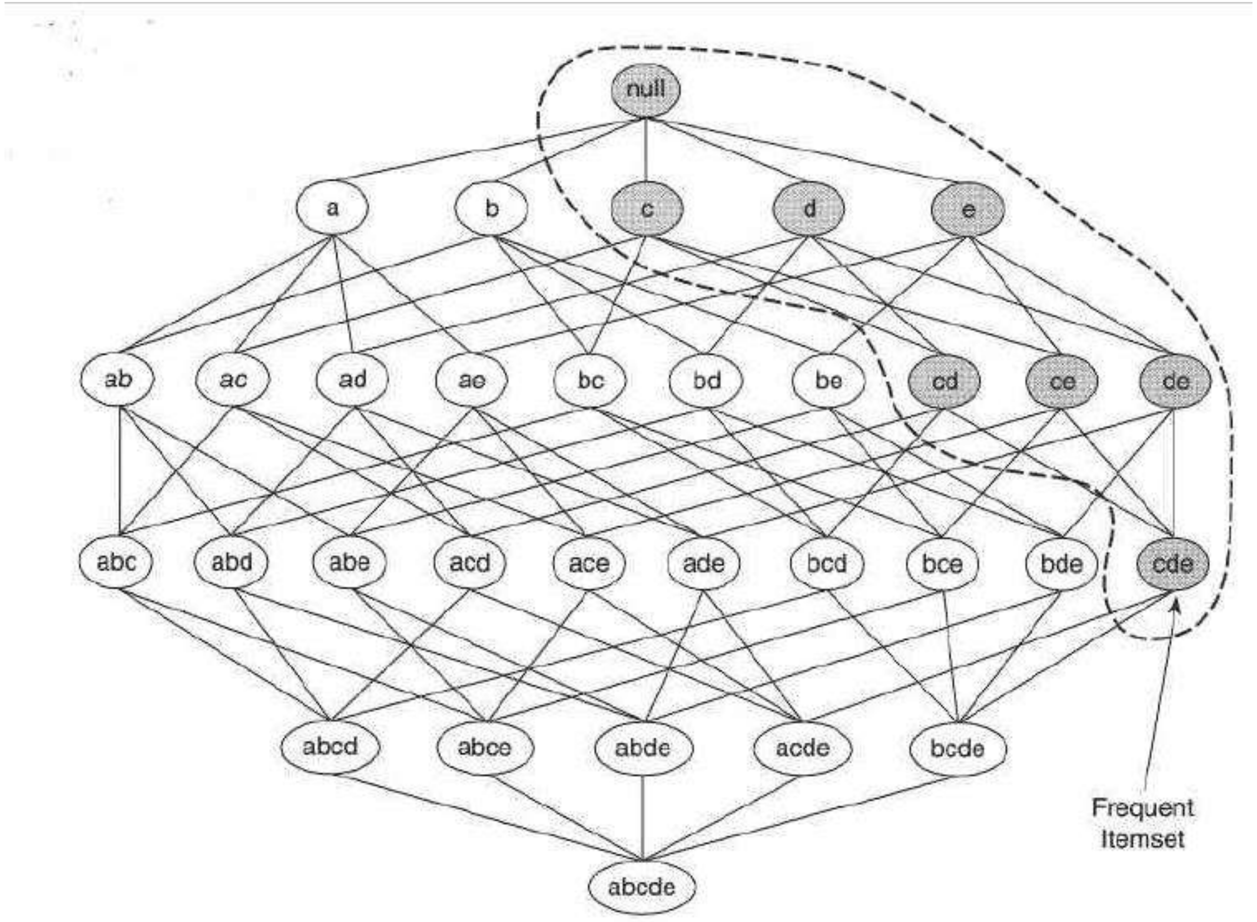
- For ***d* number of items size** of the **lattice will become 2^d**.
- This shows how **difficult** it will be to **generate Frequent Item-set by finding support** for **each combination**.
- The following figure shows how much **APRIORI helps to reduce the number of sets** to be generated:

If **item-set *{a,b}* is infrequent** then we **do not** need to take into account **all its super-sets**. **If {a, b} is infrequent**, **then all supersets of {a, b} are infrequent.**



- If an itemset such as **{a,b} is infrequent**, then **all of its supersets must be infrequent too.**
- As illustrated in the below Figure, the entire sub graph containing the **supersets of {a,b} can be pruned** immediately **once {a, b} is found to be infrequent.**
- This strategy of **trimming is known as support-based pruning.**
- Such **a pruning strategy** is **made possible by a key property of the support measure**.
- This **property** is also **known** as the **anti-monotone property of the support measure**.

If {c, d, e} is frequent, then all subsets of this itemset are frequent.

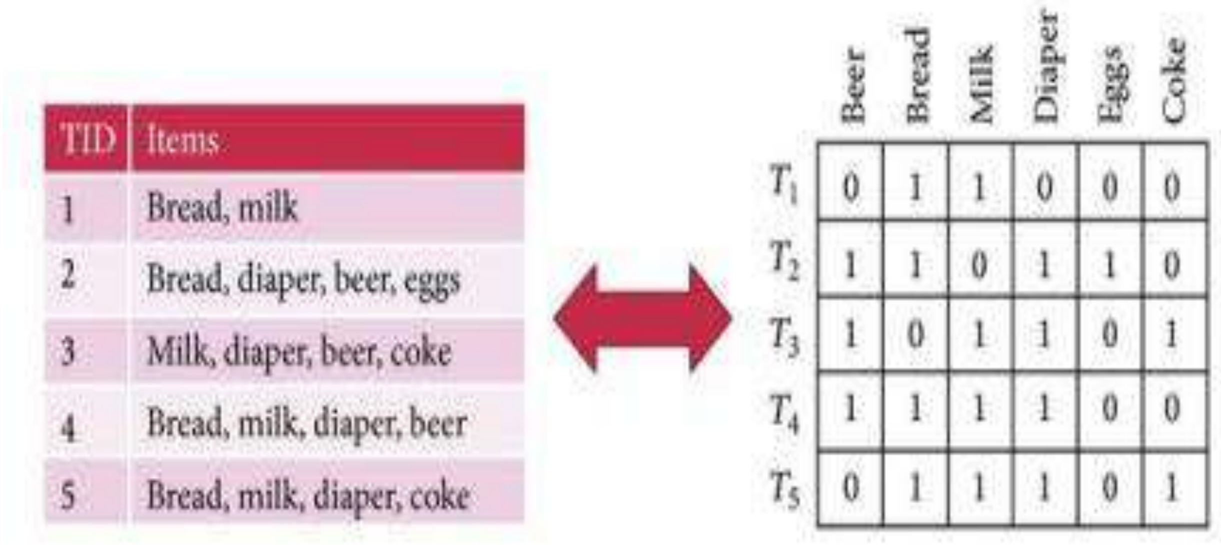


- Suppose **{c, d, e} is a frequent itemset**.
- Clearly, **any transaction that contains {c, d, e} must also contain its subsets**, **{c, d}, {c, e}, {d, e}, {c}, {d}, and {e}.**
- As a result, **if {c, d, e} is frequent**, then **all subsets of {c, d, e} must also be frequent.**

Frequent Item set Generation in the Aprior Algorithm:

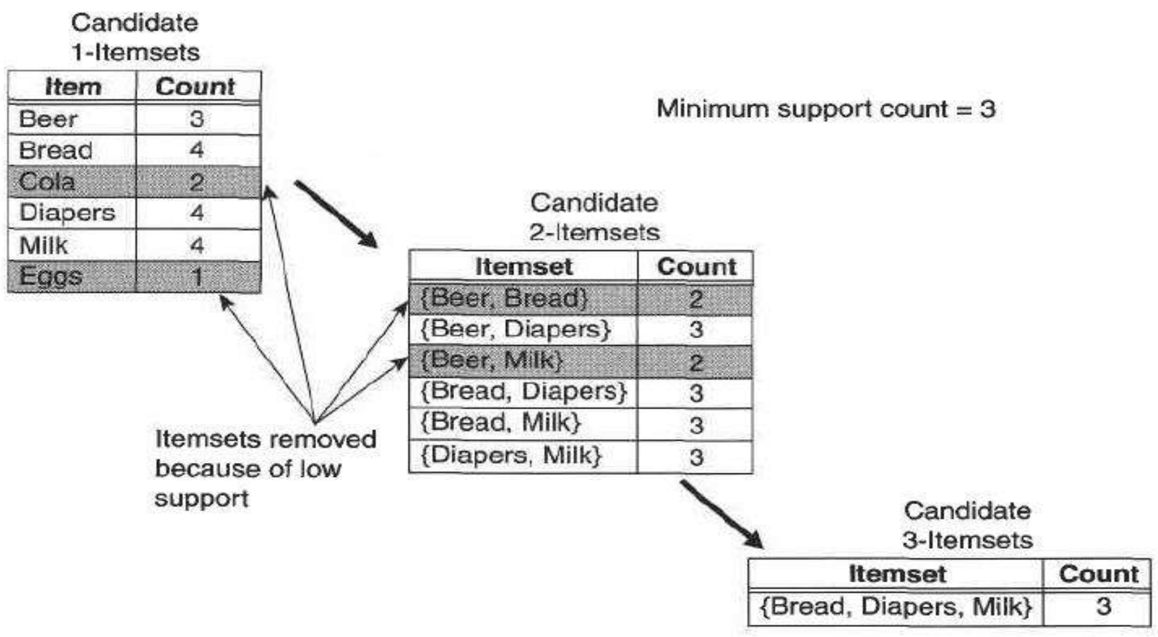
- Apriori is the first association rule mining algorithm, the **use of support-based pruning** to systematically **control** the exponential **growth of candidate itemsets**.
- The below Figure provides a high-level illustration of the **frequent itemset generation part of the Apriori algorithm for the transactions.**

Find frequent item set where min support is 3 or 50%



TID	Items
1	Bread, milk
2	Bread, diaper, beer, eggs
3	Milk, diaper, beer, coke
4	Bread, milk, diaper, beer
5	Bread, milk, diaper, coke

	Beer	Bread	Milk	Diaper	Eggs	Coke
T_1	0	1	1	0	0	0
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	1	1	0	0
T_5	0	1	1	1	0	1


Minimum support count = 3
Itemsets removed because of low support

Candidate 1-Itemsets

Item	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

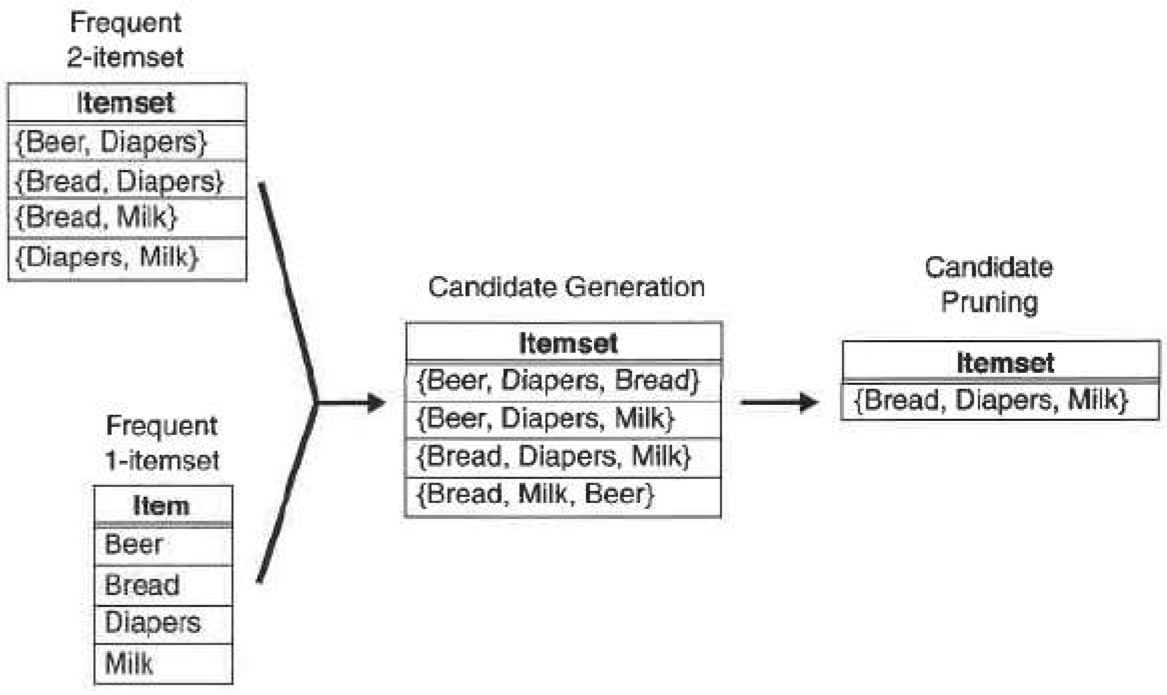
Candidate 2-Itemsets

Itemset	Count
{Beer, Bread}	2
{Beer, Diapers}	3
{Beer, Milk}	2
{Bread, Diapers}	3
{Bread, Milk}	3
{Diapers, Milk}	3

Candidate 3-Itemsets

Itemset	Count
{Bread, Diapers, Milk}	3

Dia: Illustration of frequent item set generation using the *Apriori* algorithm.



- **Initially, every item** is considered as a **candidate l-itemset.**
- After counting their supports, the candidate itemsets **{Cola} and {Eggs} are discarded** because they appear in **fewer than three transactions**.
- In the next iteration, **candidate 2-itemsets are generated using only the frequent 1-itemsets** because of **the *Apriori* principle.**
- **Two of these six candidates, {Beer, Bread} and {Beer, Milk}, are** subsequently found **to be infrequent** after computing their support values.
- The **remaining four candidates are frequent**, and thus will **be used to generate candidate 3-itemsets**.
- With the ***Apriori* principle**, we only need to keep **candidate 3-itemsets whose subsets are frequent**.
- The **only candidate** that has this property is **{Bread, Diapers, Milk}.**
- The **effectiveness** of the **Apriori *pruning* strategy** can be shown by **counting the number of candidate itemsets generated.**
- A brute-force strategy of enumerating all itemsets (up to size 3) as candidates will produce 41 candidates.

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

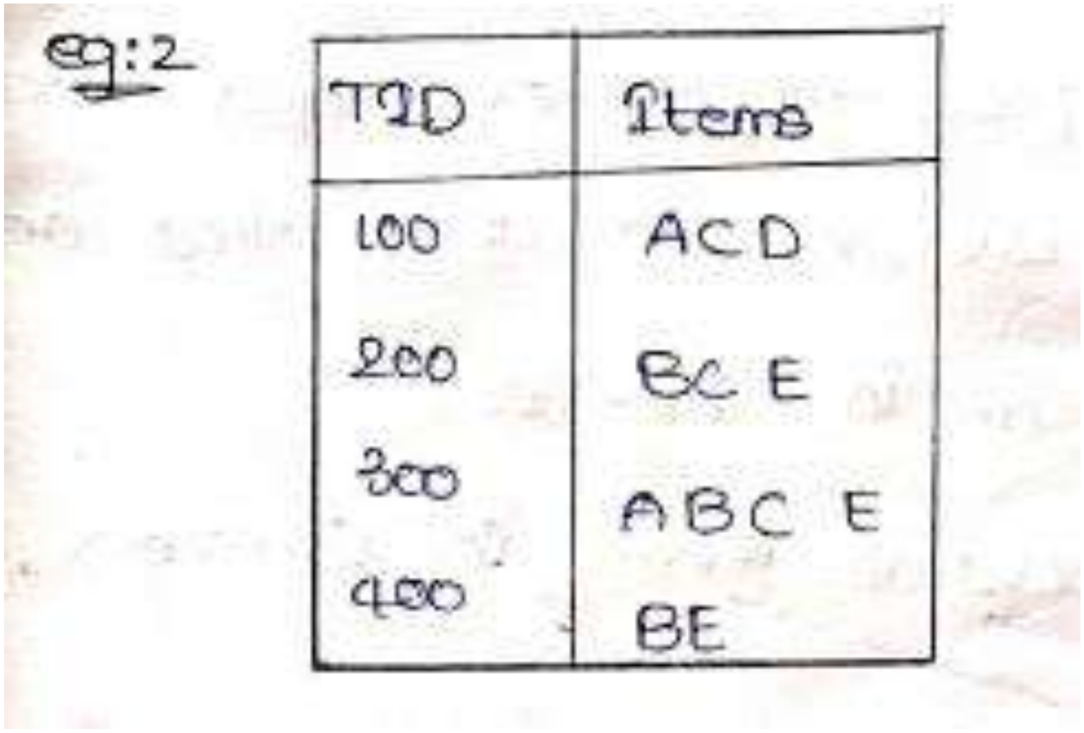
- With the Apriori principle, this number decreases to 13 candidates .

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

which represents a 68% reduction in the number of candidate item sets even in this simple example.

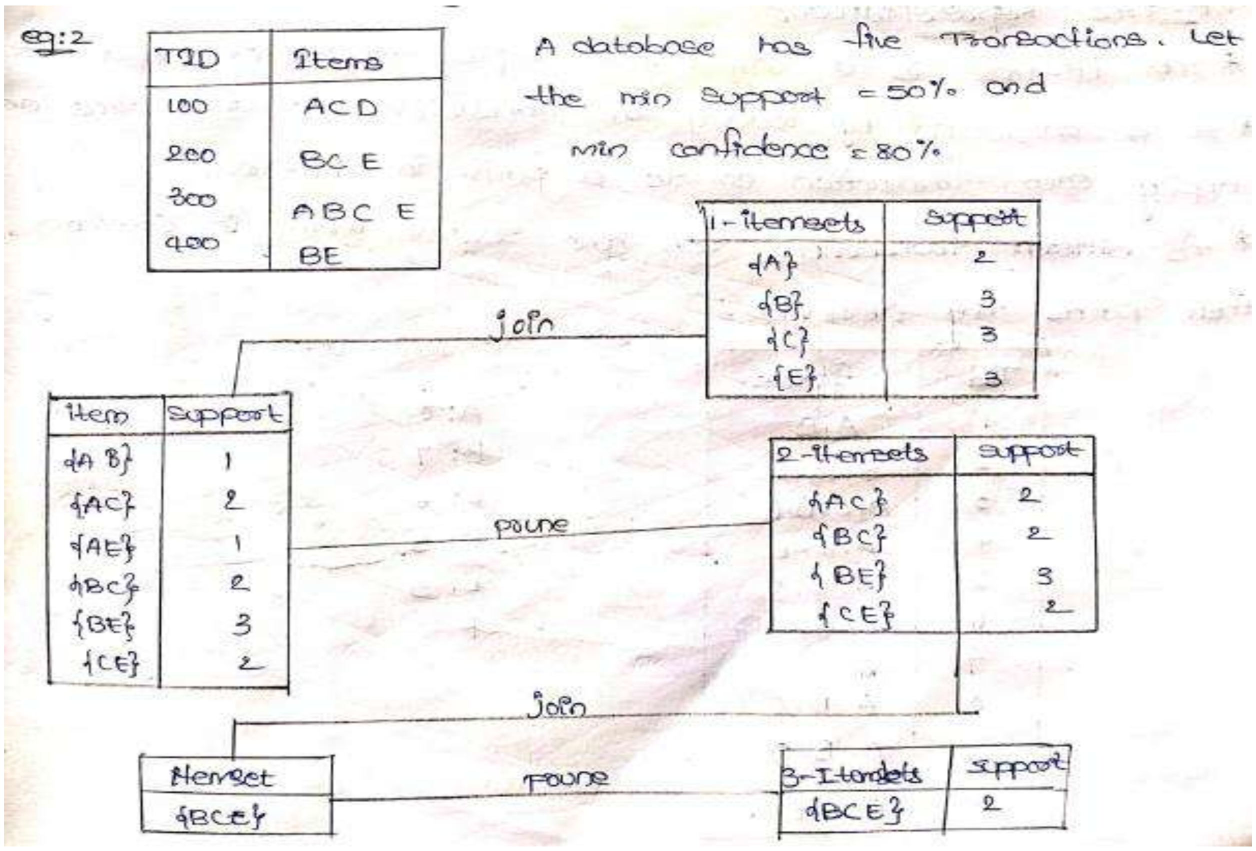
Example 2:

Min Support 50% min confidence 80%



eg:2

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E


eg:2

TID	Items
100	ACD
200	BCE
300	ABCE
400	BE

A database has five Transactions. Let the min support = 50% and min confidence = 80%.

1-itemsets	support
{A}	2
{B}	3
{C}	3
{E}	3

join

item	Support
{A B}	1
{AC}	2
{AE}	1
{BC}	2
{BE}	3
{CE}	2

prune

2-itemsets	support
{AC}	2
{BC}	2
{BE}	3
{CE}	2

join

Itemset
{BCE}

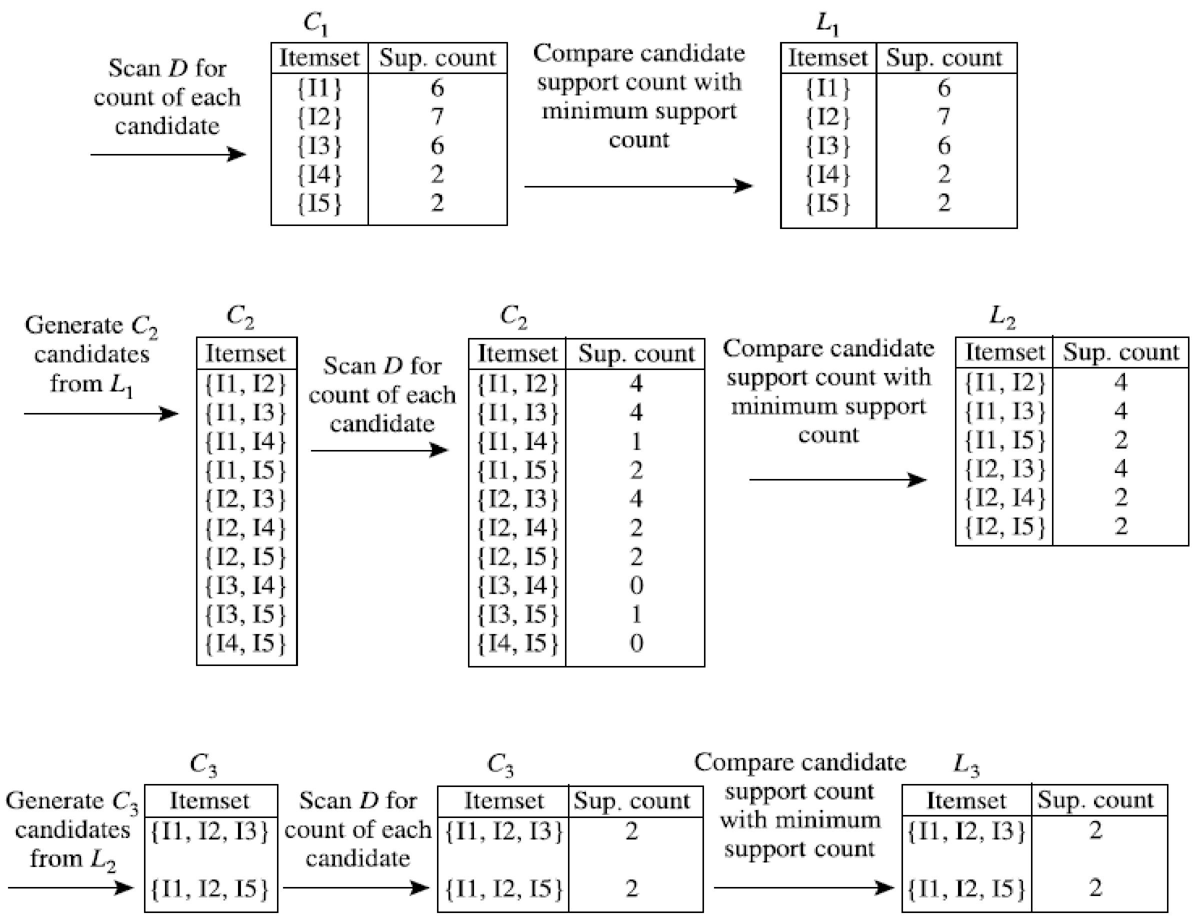
prune

3-Itemsets	support
{BCE}	2

Example 3:

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


Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C2 candidates from L1
Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C3 candidates from L2
Scan D for count of each candidate
Compare candidate support count with minimum support count

C_1

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

L_1

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

C_2

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

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

L_2

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

C_3

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

C_3

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

L_3

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

A **two-step** process is followed, consisting of **join** and **prune** actions.

- 1. The join step: To **find Lk**, a set of candidate k-itemsets is generated by **joining Lk-1** with **itself.** This set of candidates is **denoted Ck**.
- 2. The prune step: **Ck** is a **superset** of **Lk**, 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 **Ck would result** in the determination of **Lk**.

Apriori algorithm:

- The **pseudo code** for the frequent itemset generation part of the ***Apriori* algorithm** is shown in the below Algorithm.
- Let **C$_k$ denote the set of candidate k-itemsets** and ***F$_k$*** denote the set of **frequent k-itemsets:**

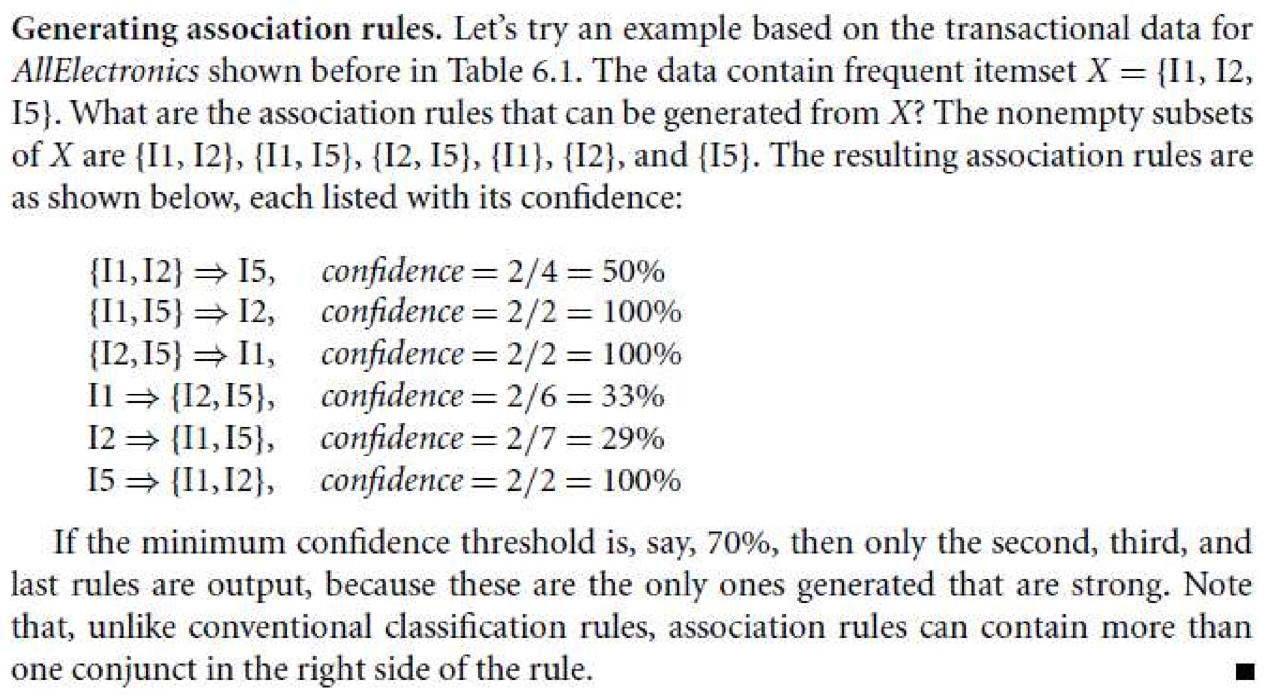
Algorithm 1 Frequent itemset generation of the *Apriori* algorithm.

```
1:  $k = 1$.
2:  $F_k = \{\ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$.    {Find all frequent 1-itemsets}
3:  **repeat**
4:    $k = k + 1$.
5:    $C_k$ = apriori-gen($F_{k-1}$).    {Generate candidate itemsets}
6:    **for** each transaction $t \in T$ **do**
7:      $C_t$ = subset($C_k$, $t$).    {Identify all candidates that belong to $t$}
8:      **for** each candidate itemset $c \in C_t$ **do**
9:        $\sigma(c) = \sigma(c) + 1$.    {Increment support count}
10:     **end for**
11:   **end for**
12:   $F_k = \{\ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$.    {Extract the frequent $k$-itemsets}
13: **until** $F_k = \emptyset$
14: Result = $\bigcup F_k$.
```

Generating Association **Rules** from Frequent Item sets:

- 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.*

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



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:

$$\begin{array}{ll}
\{I1,I2\} \Rightarrow I5, & \textit{confidence} = 2/4 = 50\% \\
\{I1,I5\} \Rightarrow I2, & \textit{confidence} = 2/2 = 100\% \\
\{I2,I5\} \Rightarrow I1, & \textit{confidence} = 2/2 = 100\% \\
I1 \Rightarrow \{I2,I5\}, & \textit{confidence} = 2/6 = 33\% \\
I2 \Rightarrow \{I1,I5\}, & \textit{confidence} = 2/7 = 29\% \\
I5 \Rightarrow \{I1,I2\}, & \textit{confidence} = 2/2 = 100\%
\end{array}$$

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. ■

Second method for frequent item set generation: 2. **Frequent Pattern Growth Algorithm.**

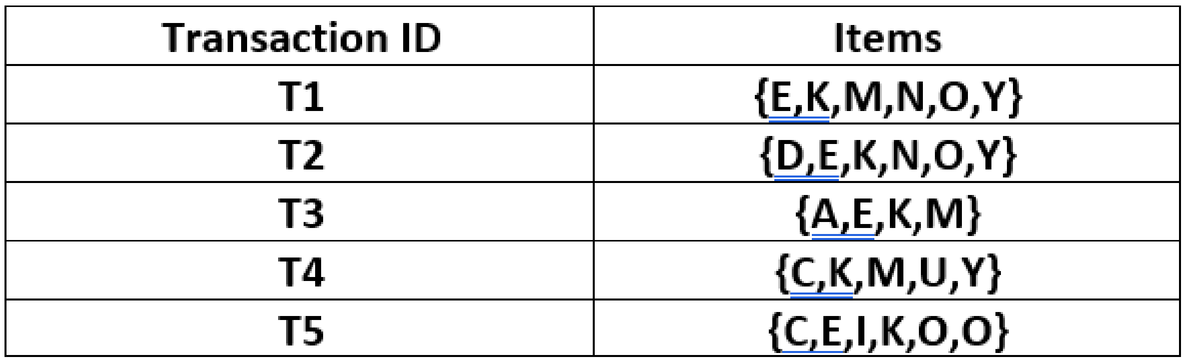
Frequent Pattern Growth Algorithm

The two primary drawbacks of the Apriori Algorithm are:-

1. At each step, **candidate sets have to be built**.
2. **To build the candidate sets**, the algorithm has to **repeatedly scan the database**.

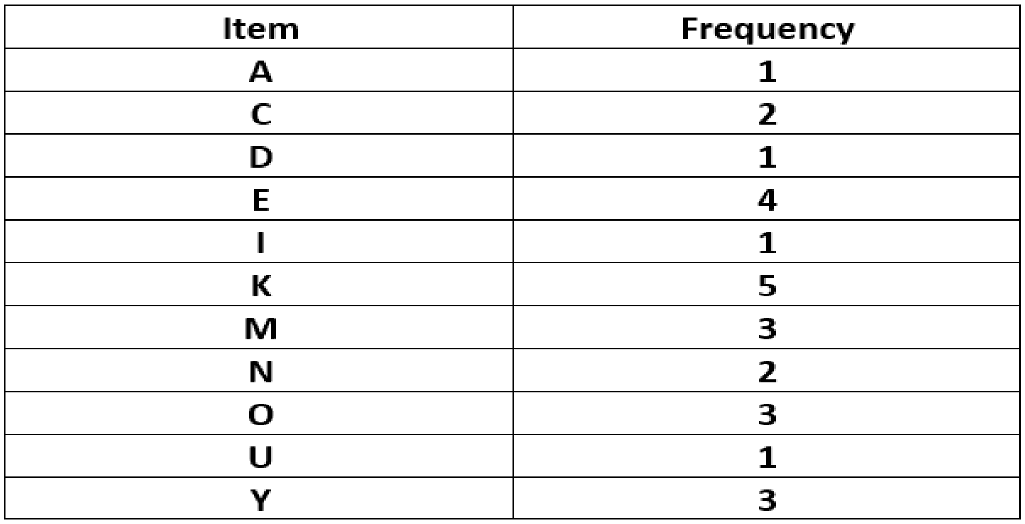
These **two properties** inevitably make the **algorithm slower**. To **overcome these redundant steps**, a new **association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm.** It **overcomes** the **disadvantages of the Apriori algorithm** by **storing all the transactions in a Trie Data Structure.**

Consider the following data:-



Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

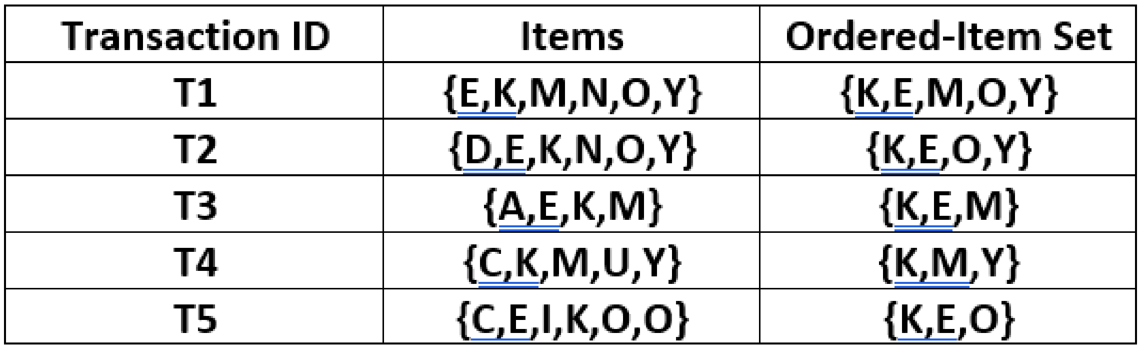
The above-given data is a hypothetical dataset of transactions with each letter representing an item. The **frequency of each individual item** is computed:-



Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

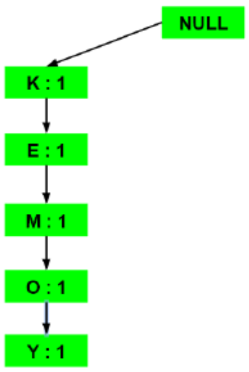
Let the minimum support be 3.

A Frequent Pattern set is built which will contain all the elements **whose frequency is greater than or equal to the minimum support.** These elements are **stored in descending order** of their respective frequencies. After insertion of the relevant items, the set L looks like this:-
L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

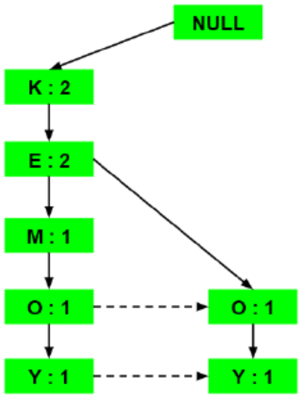


Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

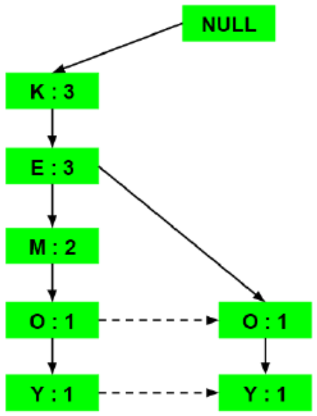
Now, all the **Ordered-Item sets are inserted into a Trie Data Structure.**
a) **Inserting the set {K, E, M, O, Y}:**
Here, all the items are simply **linked one after the other in the order** of occurrence in the set and initialize the **support count for each item as 1.**



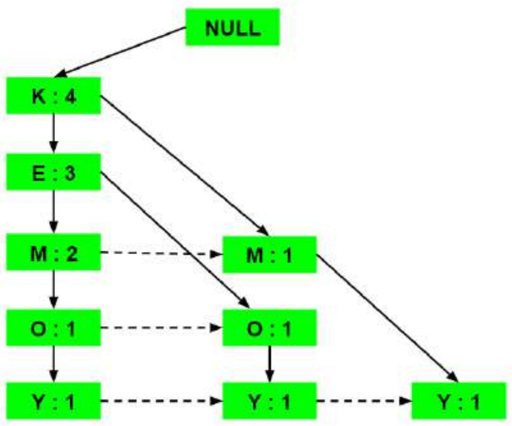
b**) Inserting the set {K, E, O, Y}:**
Till the insertion of the elements **K and E**, simply the **support count is increased by 1**. On inserting O we can see that there is no direct link between **E and O**, therefore a **new node for the item O** is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.



c) **Inserting the set {K, E, M}:**
Here simply the **support count of each element is increased by 1.**

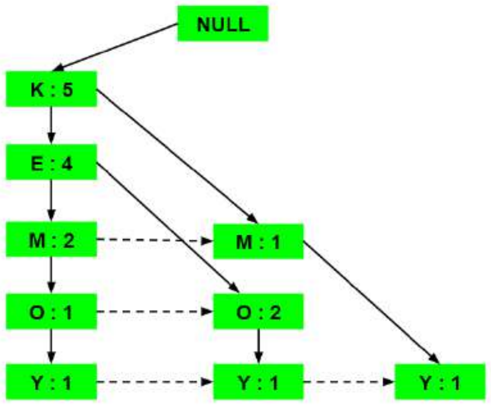


d) **Inserting the set {K, M, Y}:**
Similar to step b), first the support count of K is increased, then new nodes for M and Y are initialized and linked accordingly.



e) **Inserting the set {K, E, O}:**

Here simply the support counts of the respective elements are increased. Note that the support count of the new node of item O is increased.



Now, for each item, the **Conditional Pattern Base** is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies.

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

Now for each item the **Conditional Frequent Pattern Tree is built.** It is done by taking the set of elements which is common in all the paths in the Conditional Pattern Base of that item and calculating it's **support count by summing the support counts of all the paths in the Conditional Pattern Base.**

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

From the Conditional Frequent Pattern tree, the **Frequent Pattern rules** are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item as given in the below table.

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 3>}
K	

For each row, **two types of association rules can be inferred** for example for the **first row which contains the element, the rules K -> Y and Y -> K can be inferred**. To **determine the valid rule, the confidence of both the rules is calculated** and the one with **confidence greater than or equal to the minimum confidence value is retained**.

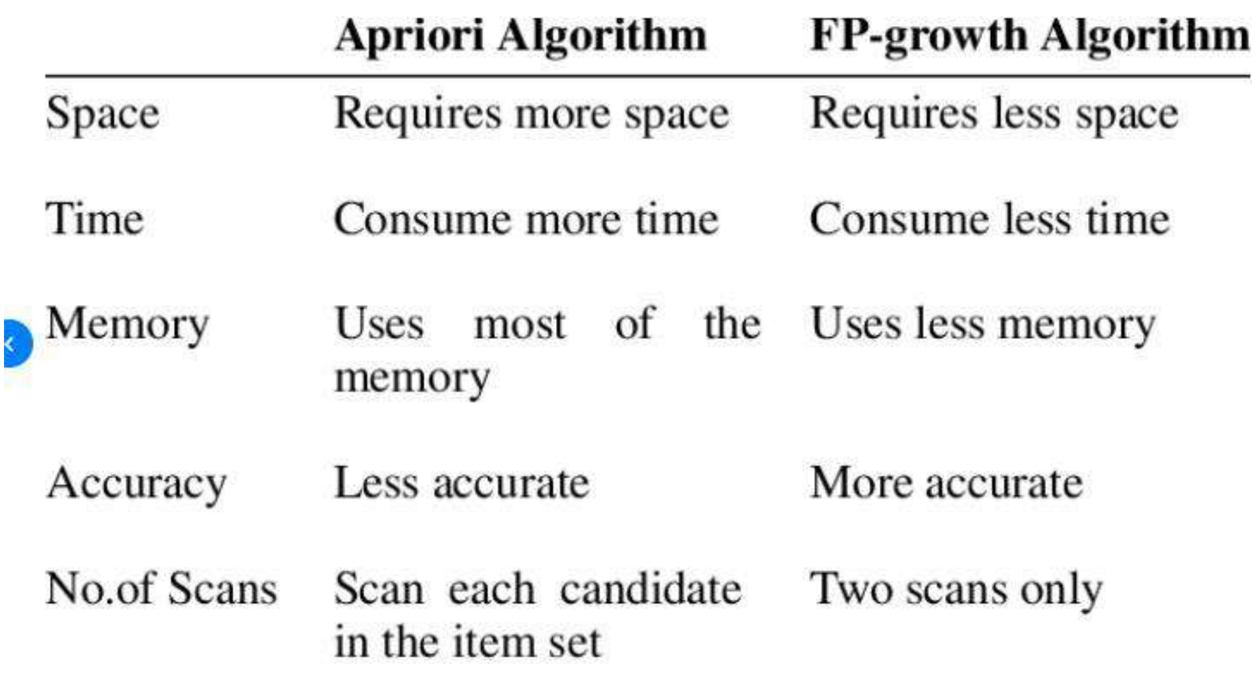
Advantages Of FP Growth Algorithm

- This algorithm needs to scan the database **only twice when compared to Apriori** which **scans the transactions for each iteration.**
- The **pairing of items is not done in this algorithm** and this **makes it faster.**
- The **database is stored in a compact version** in memory.
- It is **efficient and scalable for mining both long and short frequent patterns.**

Disadvantages Of FP-Growth Algorithm

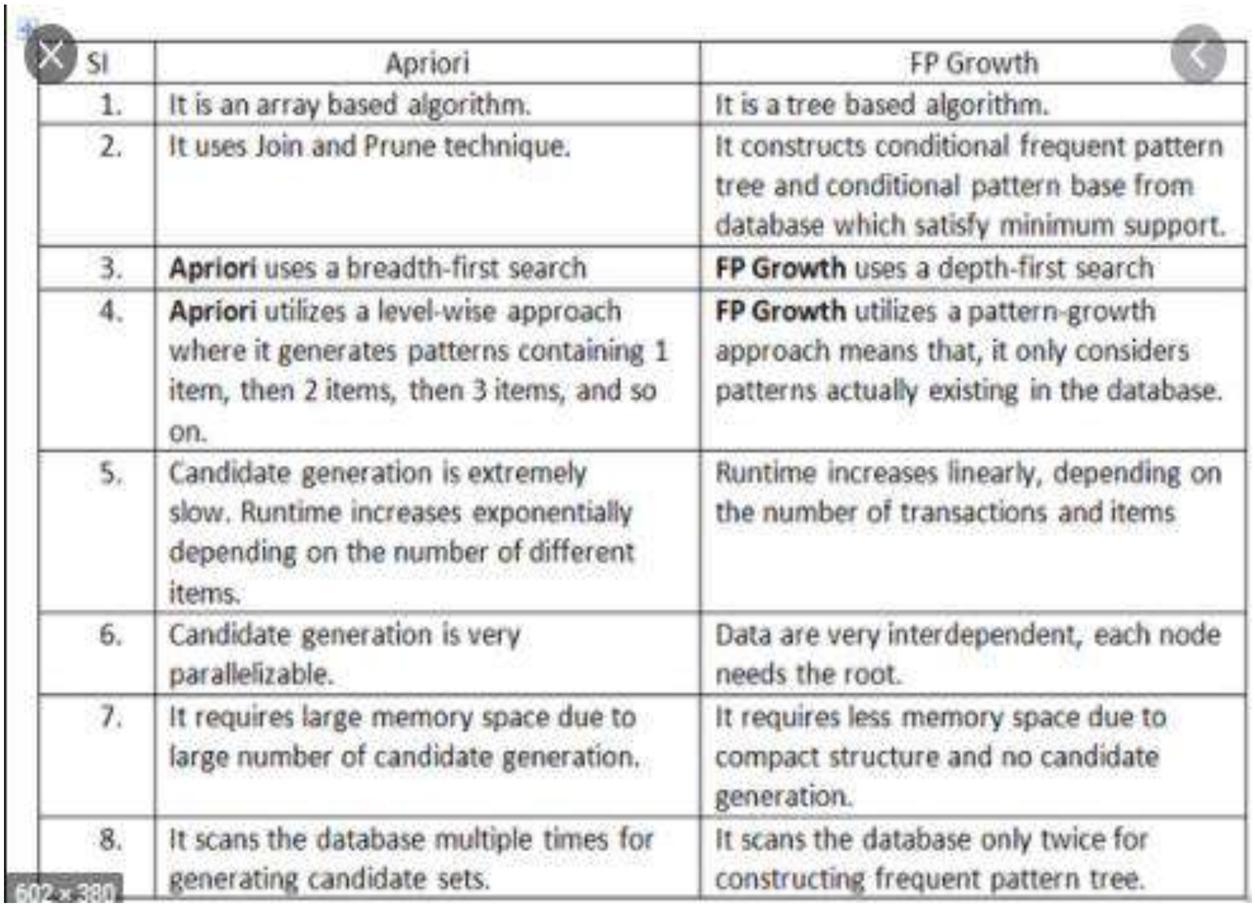
- FP Tree is more **cumbersome** and **difficult to build than Apriori.**
- It may be **expensive**.
- **When the database is large**, the **algorithm may not fit in the shared memory.**

Apriori Algorithm vs FP –Growth Algorithm.



	Apriori Algorithm	FP-growth Algorithm
Space	Requires more space	Requires less space
Time	Consume more time	Consume less time
Memory	Uses most of the memory	Uses less memory
Accuracy	Less accurate	More accurate
No.of Scans	Scan each candidate in the item set	Two scans only

Comparison between Apriori and FP-growth Algorithms.

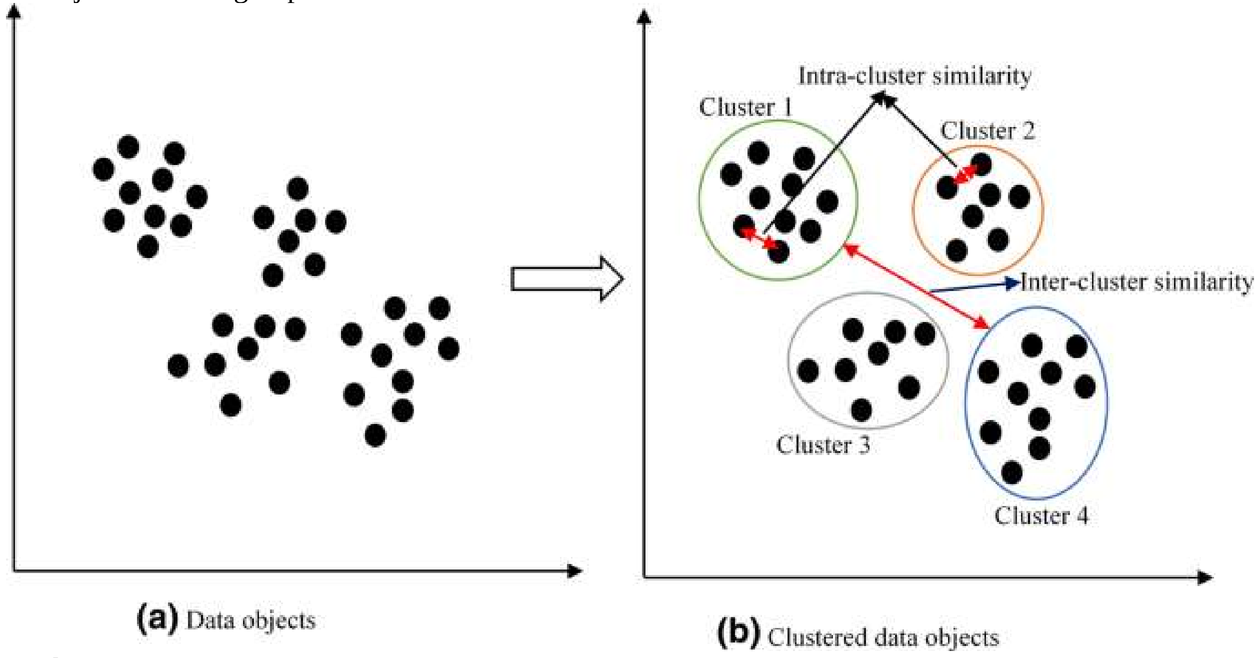


Sl	Apriori	FP Growth
1.	It is an array based algorithm.	It is a tree based algorithm.
2.	It uses Join and Prune technique.	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy minimum support.
3.	**Apriori** uses a breadth-first search	**FP Growth** uses a depth-first search
4.	**Apriori** utilizes a level-wise approach where it generates patterns containing 1 item, then 2 items, then 3 items, and so on.	**FP Growth** utilizes a pattern-growth approach means that, it only considers patterns actually existing in the database.
5.	Candidate generation is extremely slow. Runtime increases exponentially depending on the number of different items.	Runtime increases linearly, depending on the number of transactions and items
6.	Candidate generation is very parallelizable.	Data are very interdependent, each node needs the root.
7.	It requires large memory space due to large number of candidate generation.	It requires less memory space due to compact structure and no candidate generation.
8.	It scans the database multiple times for generating candidate sets.	It scans the database only twice for constructing frequent pattern tree.

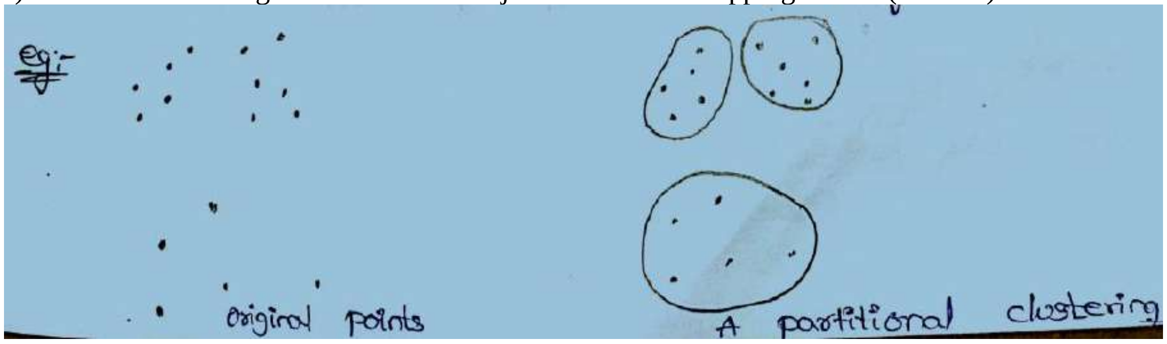
UNIT 6 Cluster Analysis

Syllabus: Basic Concepts and Algorithms: Overview: What Is Cluster Analysis? Different Types of Clustering, Different Types of Clusters; K-means: The Basic K-means Algorithm, K-means Additional Issues, Bisecting K-means, Strengths and Weaknesses; Agglomerative Hierarchical Clustering: Basic Agglomerative Hierarchical Clustering Algorithm DBSCAN: Traditional Density Center-Based Approach, DBSCAN Algorithm, Strengths and Weaknesses.

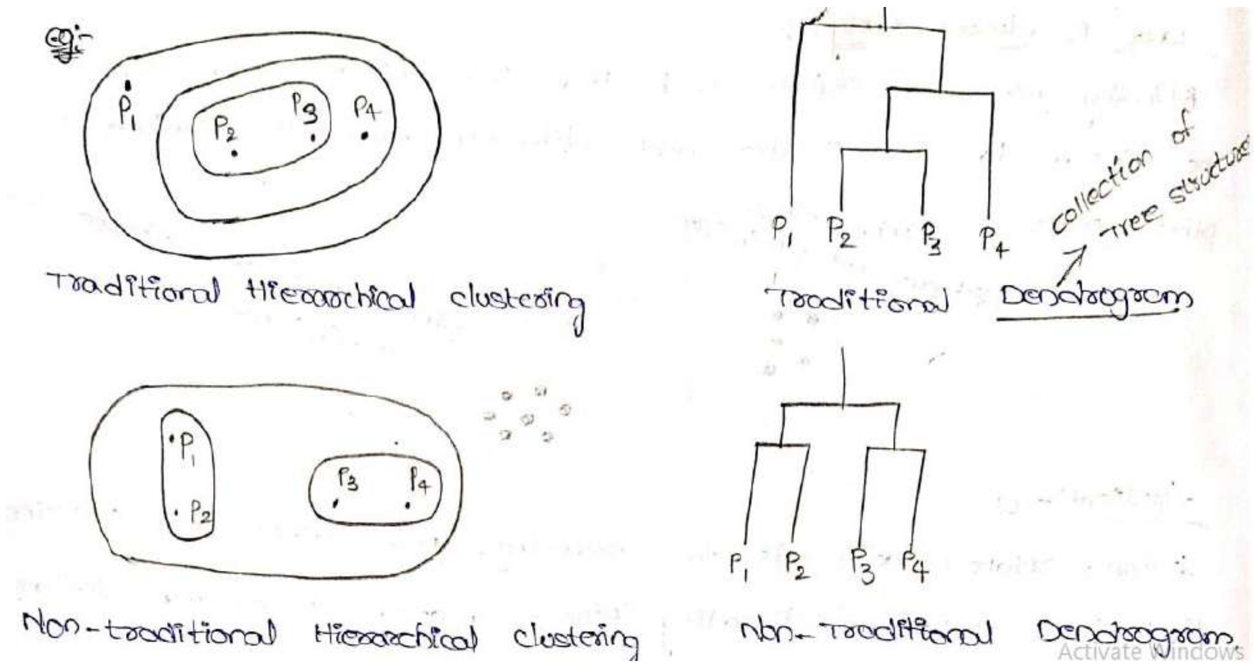
What is cluster analysis:---
Finding group of objects such that the objects in a group will be similar to one another and different from the objects in other groups.



Applications:
Group related documents for browsing , group genes and protienes that have similar functionality.
What is not cluster analysis?
Supervised classification: have class label information.
Simple segmentation: Dividing students into different registration groups alphabetically, by last name.
Results of query: grouping are a result of external specification.
Graph partitioning.
Types of clustering:
A clustering is a set of clusters.
1)Partitional clustering: A division data object into non overlapping subsets(clusters)



2)hierarchical clustering:
A set of nested clusters organized as a hierarchical tree.


eg:-
P1
P2
P3
P4
Traditional Hierarchical clustering
Traditional Dendrogram
collection of tree structure
Non-traditional Hierarchical clustering
Non-Traditional Dendrogram
Activate Windows

Difference between sets of clustering:

Exclusive versus non exclusive :

In non exclusive clustering, point may belong to multiple clusters.

Exclusive Clustering: Assignment is to one cluster

¤ Non-Exclusive Clustering: Data objects may belong to multiple clusters

Complete vs. Partial :

Complete Clustering: Every object is assigned to a cluster

Partial Clustering: Not every object needs to be assigned

Fuzzy versus non fuzzy:

In fuzzy clustering a point belongs to every cluster with some weight between 0 to 1.

Heterogeneous versus homogeneous :

Cluster of widely different sizes, shapes and densities.

Partitional vs. Hierarchical :

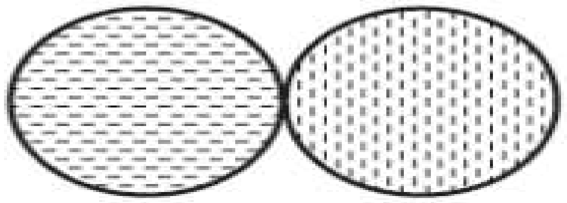
Partitional Clustering: A division of data into nonoverlapping clusters, such that each data object is in exactly one subset

Hierarchical Clustering: A set of nested clusters organized as a hierarchical tree n Each node (cluster) is union of its children (subclusters) n Root of tree: cluster containing all data objects n Leaves of tree: singleton clusters

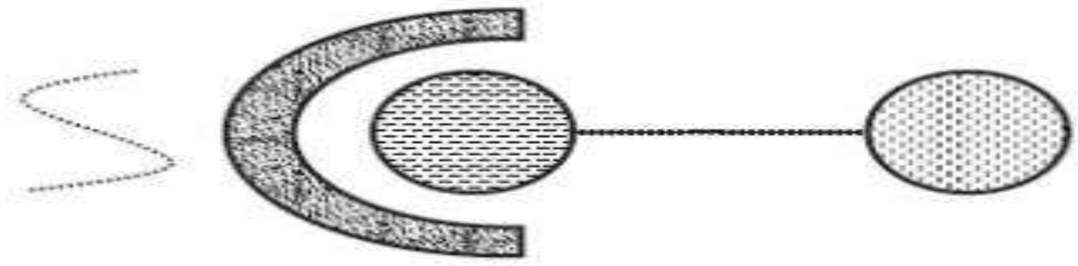
Types of clusters:
Well separated clusters: a cluster is a set of points such that any point in a cluster is closer to every other point in the cluster than to any point not in the cluster .

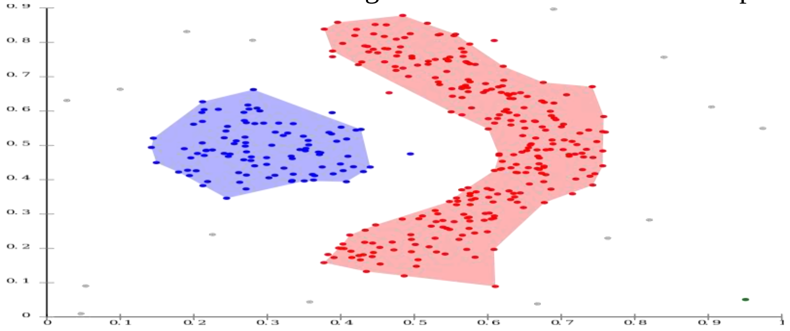
(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

Center based clusters: A cluster is a set of objects such that an object in a cluster is closer to center of cluster of a cluster, than to the center of any other cluster.
The center of a cluster is often a centroid, the average of all the points in cluster or a medoid the most representative point of cluster.

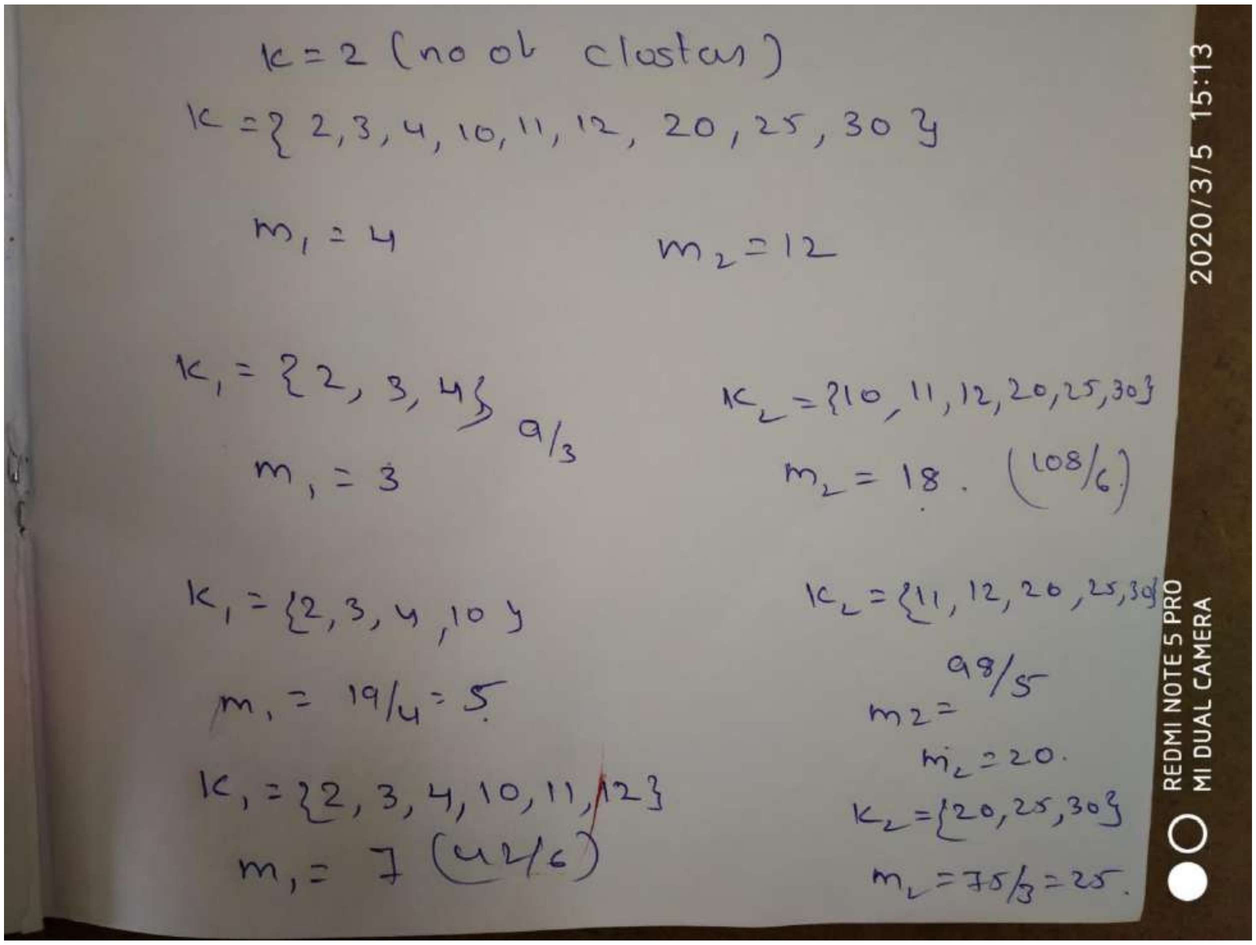

(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

Contiguous Clusters : a point in a cluster is closer to one or more other points in the cluster than to any point not in the cluster


(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

Density based clustering:
Density clustering groups data points by **how densely populated they** are . a cluster is a density region of points, which is separated by low density regions, from other regions of high density.
Used when the clusters are irregular and noise and outliers are present .


Clustering algorithms:
k-means algorithm:--
It follows partitional clustering approach. each cluster is associated with a cetroid (center point). Each point is assigned to cluster with closest centroid . The basic algorithm is simple.
k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.
Algorithm:
- Step 1: Take mean value(Random)
- Step 2:Find nearest number on mean put it in cluster
- Step 3: Repeat the one and two steps to get same mean .



$k = 2$ (no of clustas)

$K = \{2, 3, 4, 10, 11, 12, 20, 25, 30\}$

$m_1 = 4$ $\qquad$ $m_2 = 12$

$K_1 = \{2, 3, 4\}$ $9/3$ $\qquad$ $K_2 = \{10, 11, 12, 20, 25, 30\}$

$m_1 = 3$ $\qquad$ $m_2 = 18.$ $(108/6)$

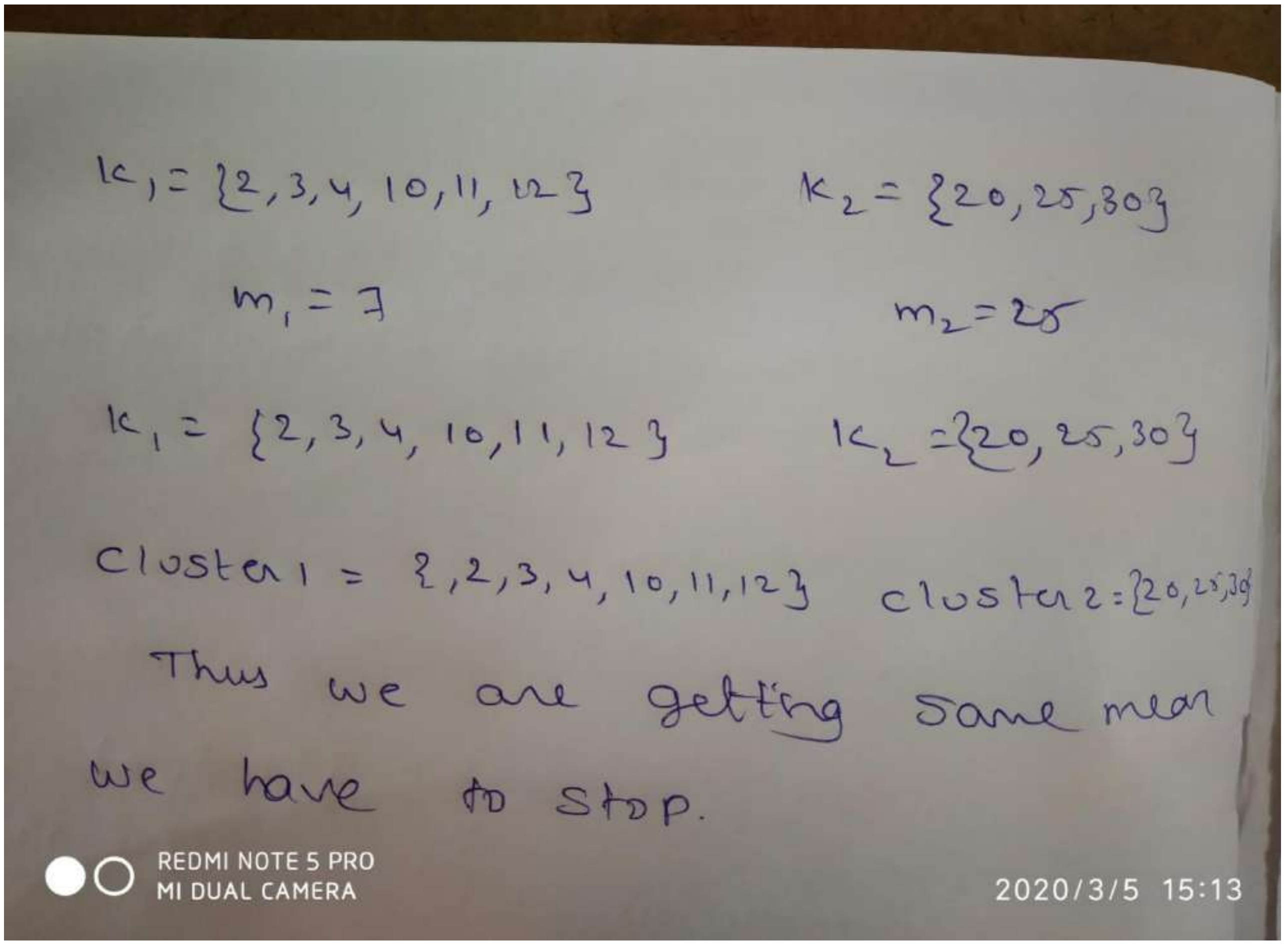
$K_1 = \{2, 3, 4, 10\}$ $\qquad$ $K_2 = \{11, 12, 20, 25, 30\}$

$m_1 = 19/4 = 5$ $\qquad$ $m_2 = 98/5$

$m_2 = 20.$

$K_1 = \{2, 3, 4, 10, 11, 12\}$ $\qquad$ $K_2 = \{20, 25, 30\}$

$m_1 = 7$ $(42/6)$ $\qquad$ $m_2 = 75/3 = 25.$



$k_1 = \{2, 3, 4, 10, 11, 12\}$ $k_2 = \{20, 25, 30\}$

$m_1 = 7$ $m_2 = 25$

$k_1 = \{2, 3, 4, 10, 11, 12\}$ $k_2 = \{20, 25, 30\}$

cluster1 = $\{, 2, 3, 4, 10, 11, 12\}$ cluster2 = $\{20, 25, 30\}$

Thus we are getting same mean we have to stop.

REDMI NOTE 5 PRO
MI DUAL CAMERA
2020/3/5 15:13

Time complixity =0(I*K*M*N)
I- No of iterations
It is offen small and safe bounded
K is linear in m – no of points
k- no of clusters and significantly less than m.
K-Means Algorithm has a few limitations which are as follows:

- It only **identifies spherical shaped clusters** i.e it cannot identify, if the **clusters are non-spherical** or of **various size and density.**
- It suffers from local minima and **has a problem when the data contains outliers.**

Bisecting K-means :

- **Bisecting K-Means Algorithm** is a **modification of the K-Means algorithm.**
- It can **produce partitional/hierarchical clustering.**
- It can **recognize clusters of any shape and size.**
- The Bisecting K-Means algorithm is a **Variant of K-means that can produce a partitional or a hierarchical clustering.**
- Bisecting k-Means is like a **combination of k-Means and hierarchical clustering**.
- **Instead** of **partitioning the data into 'k' clusters in each iteration**, **Bisecting k-means splits one cluster into two sub clusters at each bisecting step**(by using k-means) **until k clusters are obtained**

Basic Bisecting K-means Algorithm for finding K clusters

1. **Pick a cluster to split**.
2. **Find 2 sub-clusters using the basic K-means algorithm**. (Bisecting step)
3. **Repeat step 2, the bisecting step**, for ITER times and take the **split** that **produces the clustering with the highest overall similarity**.
4. **Repeat steps 1, 2 and 3 until the desired number of clusters is reached**.

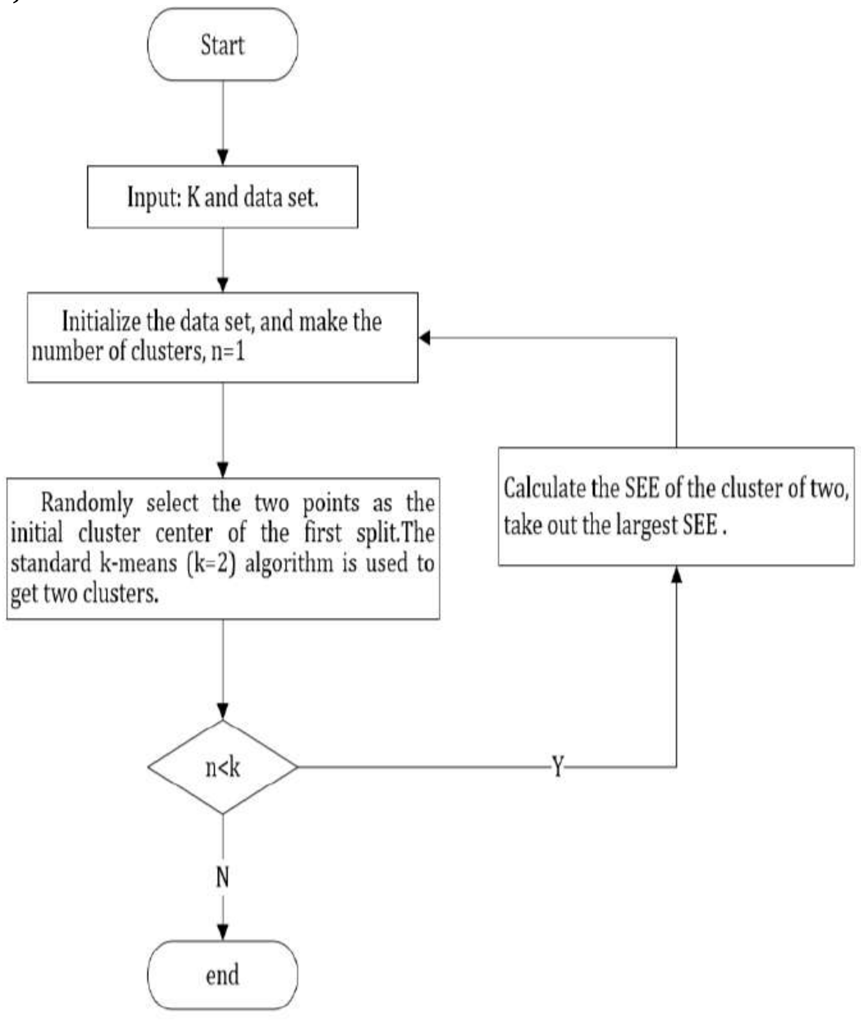

Start
Input: K and data set.
Initialize the data set, and make the number of clusters, n=1
Randomly select the two points as the initial cluster center of the first split.The standard k-means (k=2) algorithm is used to get two clusters.
Calculate the SEE of the cluster of two, take out the largest SEE .
n<k
Y
N
end

Fig. 1. Bisecting k-means algorithm flow chart.

Strengths of bisecting k means(BSK):

- As **Bisecting k-means** is **based on k-means, it keeps the merits of k-means** and **also has some advantages over k-means.**
- First, **Bisecting k-means is more efficient when 'k' is large.**
- For the **k-means algorithm**, the **computation involves every data point of the data set** and **k centroids.**
- On the other hand, in **each Bisecting step of Bisecting k-means,** only the **data points of one cluster and two centroids are involved** in the **computation**.
- Thus, the **computation time is reduced.**
- **Secondly, Bisecting k-means produce clusters of similar sizes,** while **k-means is known to produce clusters of widely different sizes** .

Weaknesses of bisecting k means:

- It is here worth noting that the **algorithm above recalled** is the **very classical and basic version of K-means, also known** as **Forgy's algorithm** (with a **slight modification of the initialization step**).
- **Many variations of this basic version of the algorithm have been proposed**, **aiming** to **reduce the computational demand,** at the **little sub-optimality**.

Agglomerative clustering

Hierarchical clustering techniques are a second important category of clustering methods. As with K-means, these approaches are relatively old compared to many clustering algorithms, but they still enjoy widespread use.

*The **agglomerative clustering** is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as* AGNES *(Agglomerative Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named dendrogram.*

Algorithm

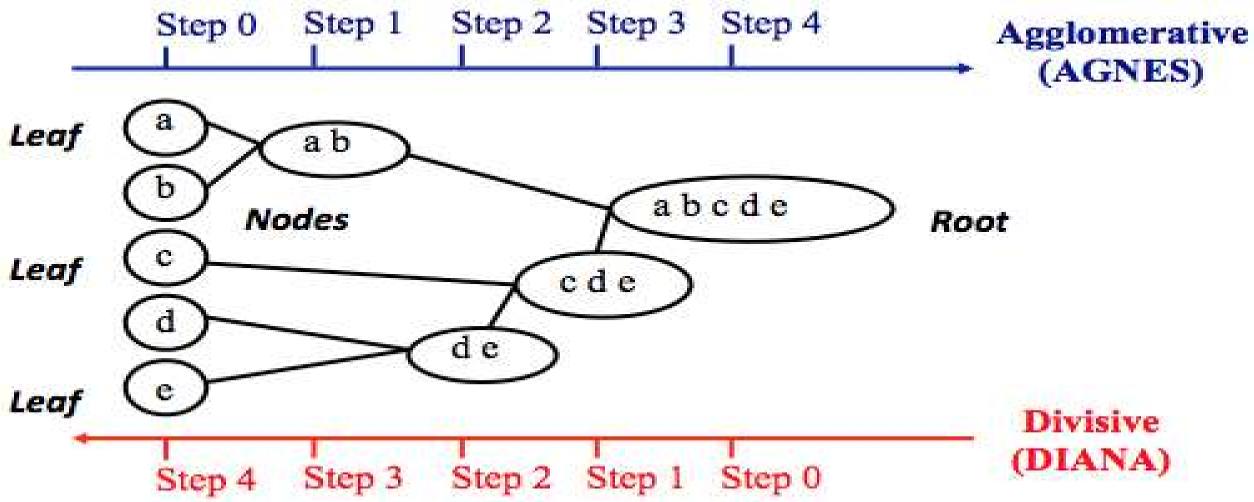
Hierarchical clustering can be divided into two main types: agglomerative and divisive.

Agglomerative **clustering**: It's also known as AGNES (Agglomerative Nesting). It works in a bottom-up manner.

Divisive **hierarchical clustering**: It's also known as DIANA (Divise Analysis) and it works in a top-down manner.

Agglomerative clustering works in a "bottom-up" manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below).

The inverse of agglomerative clustering is *divisive clustering*, which is also known as DIANA (*Divise Analysis*) and it works in a "top-down" manner. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.



```
Algorithm    Basic agglomerative hierarchical clustering algorithm.
1: Compute the proximity matrix, if necessary.
2: repeat
3:     Merge the closest two clusters.
4:     Update the proximity matrix to reflect the proximity between the new
       cluster and the original clusters.
5: until Only one cluster remains.
```

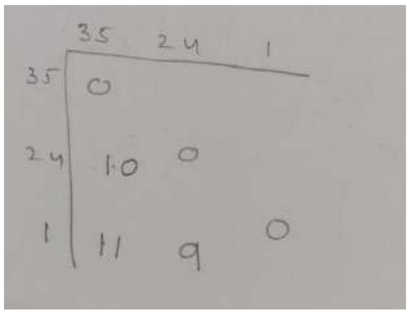
Example: Agglomerative Hierarchical Clustering
Clustering starts by computing a distance between every pair of units that you want to cluster. A distance matrix will be symmetric (because the distance between x and y is the same as the distance between y and x) and will have zeroes on the diagonal (because every item is distance zero from itself). The table below is an example of a distance matrix. Only the lower triangle is shown, because the upper triangle can be filled in by reflection.

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	$\boxed{2}$	8	0

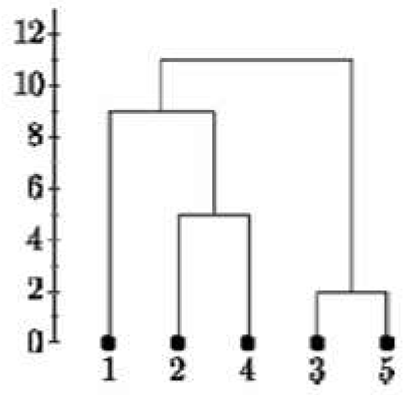
Now lets start clustering. The smallest distance is between three and five and they get linked up or merged first into a the cluster '35'.
To obtain the new distance matrix, we need to remove the 3 and 5 entries, and replace it by an entry "35" . Since we are using complete linkage clustering, the distance between "35" and every other item is the maximum of the distance between this item and 3 and this item and 5. For example, d(1,3)= 3 and d(1,5)=11. So, D(1,"35")=11. This gives us the new distance matrix. The items with the smallest distance get clustered next. This will be 2 and 4.

	35	1	2	4
35	0			
1	11	0		
2	10	9	0	
4	9	6	5	0

Continuing in this way, after 6 steps, everything is clustered. This is summarized below. On this plot, the y-axis shows the distance between the objects at the time they were clustered. This is called the cluster height. Different visualizations use different measures of cluster height.



	35	24	1
35	0		
24	10	0	
1	11	9	0



Time and Space Complexity:
The basic agglomerative hierarchical clustering algorithm just presented uses a proximity matrix. This requires the storage of l/2m^2 proximities where rn is the number of data points. The total space complexity is O(m^2).
overall time required for a hierarchical clustering based on Algorithm is O(m^{2}logm).

Some pros and cons of Hierarchical Clustering:
Pros
- No assumption of a particular number of clusters (i.e. k-means)
- May correspond to meaningful taxonomies

Cons
- Once a decision is made to combine two clusters, it can't be undone
- Too slow for large data sets, O(n2 log(n))

DBSCAN:
- **Density-based spatial clustering of applications with noise** (**DBSCAN**) is a data **clustering algorithm**
- Density-based clustering **locates regions** of **high density that are separated from one another by regions of low density**.
- DBSCAN is a **simple and effective density-based clustering algorithm.**

Traditional Density: Center-Based Approach:
- There are **not as many approaches for defining density** as there are **for defining similarity**, there are several distinct methods.
- In the **center-based approach**, **density is estimated for a particular point in the data set** by counting the **number of points within a specified radius, Eps,** of that point. This **includes the point itself**.
- The **number of points within a radius of Eps of point A is 7**, **including A itself**.

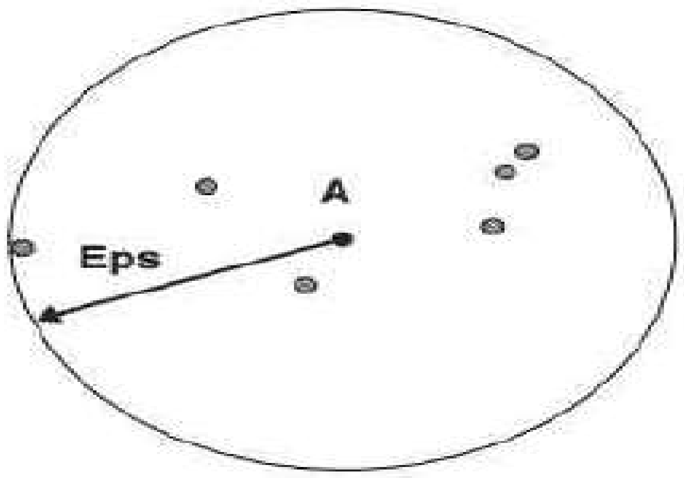


Figure Center-based density.

• This method is simple to inclement, but the density of any point will depend on the specified radius.

Classification of Points According to Center-Based Density:

The center-based approach to density allows us to classify a point as being

(1) in the interior of a dense region (a core point),

(2) on the edge of a dense region (a border point), or

(3) in a sparsely occupied region (a noise or background point).

The below Figure graphically illustrates the concepts of core, border, and noise points using a collection of two-dimensional points.

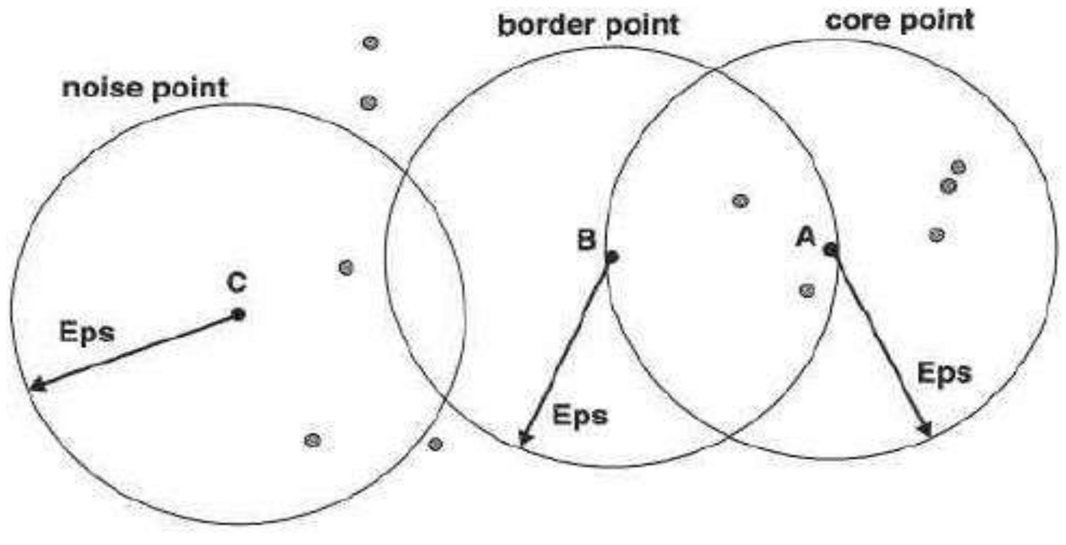


Figure Core, border, and noise points.

• **Core points: These points are in the interior of a density-based cluster**.

• **Border points: A border point is not a core point**, but falls within the **neighborhood of a core point.** In the below Figure, point **B is a border point.**

- **A border point** can fall within the **neighborhoods of several core points.**
- **Noise points:** A noise point is any point that is **neither a core point nor a border point**.
- In the below Figure, **point C is a noise point**.

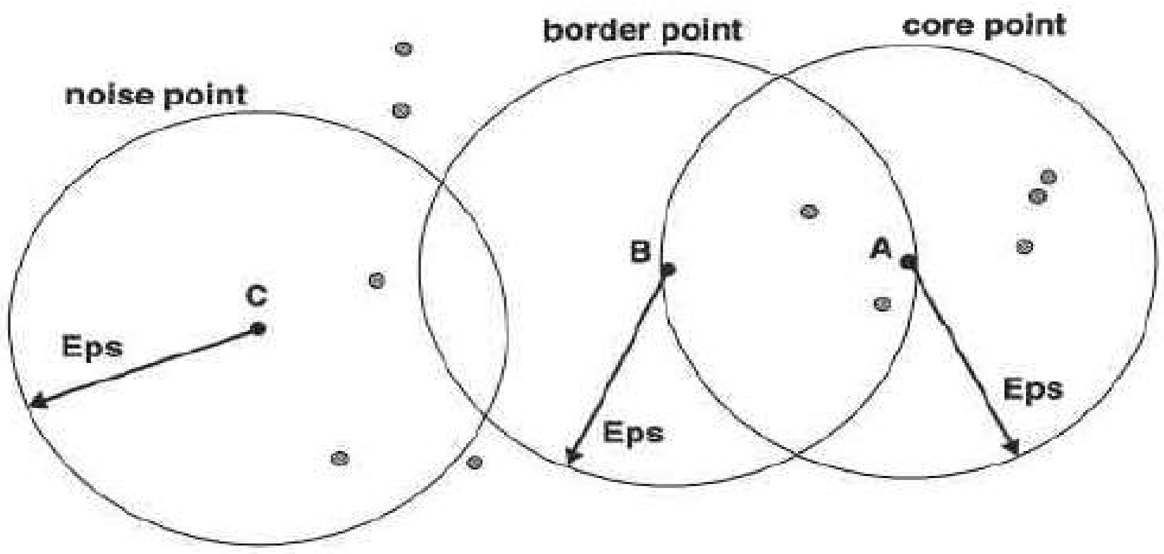
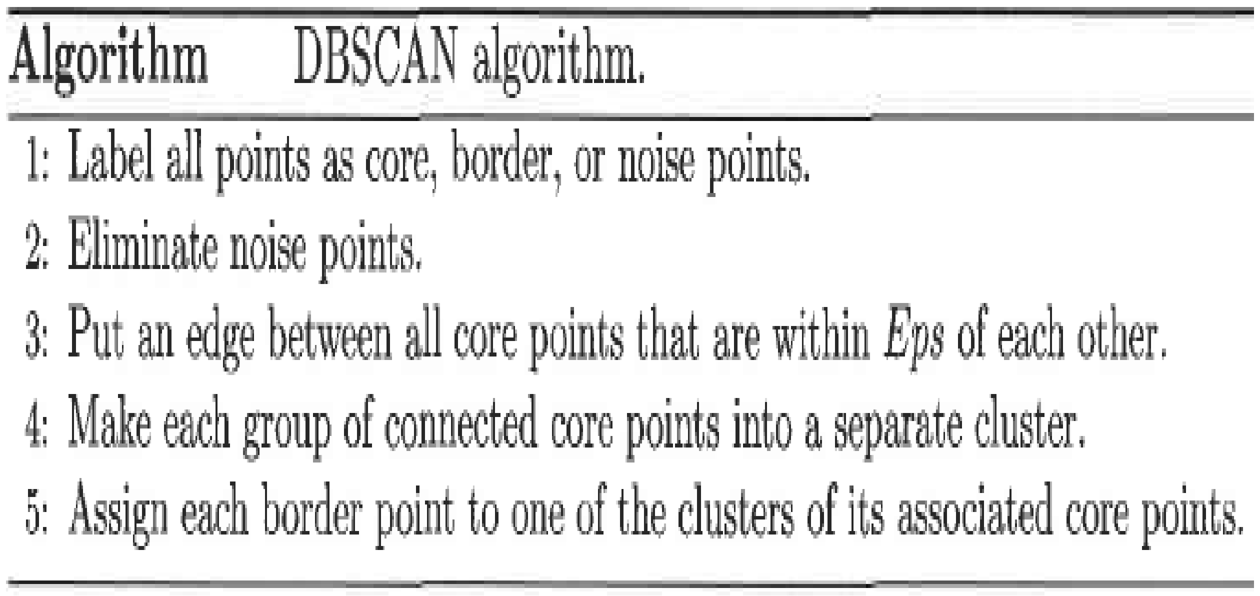


Figure Core, border, and noise points.

The DBSCAN Algorithm:

The **DBSCAN algorithm** can be informally described as follows.

- **Any two core points** that are **close enough within a distance Eps** of **one another are put in the same cluster.**
- **Any border point that is close enough** to a **core point is put in the same cluster** as the core point.
- **Noise points are discarded**.



Algorithm DBSCAN algorithm.

```
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within $Eps$ of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.
```

Time and Space Complexity

- In the worst case, this complexity is **$O(m^2)$**
  - where **m is the number of points.**

Strengths and Weaknesses:

• DBSCAN uses **a density-based definition** of a cluster, it is relatively resistant to noise and can **handle clusters of arbitrary shapes and sizes.**

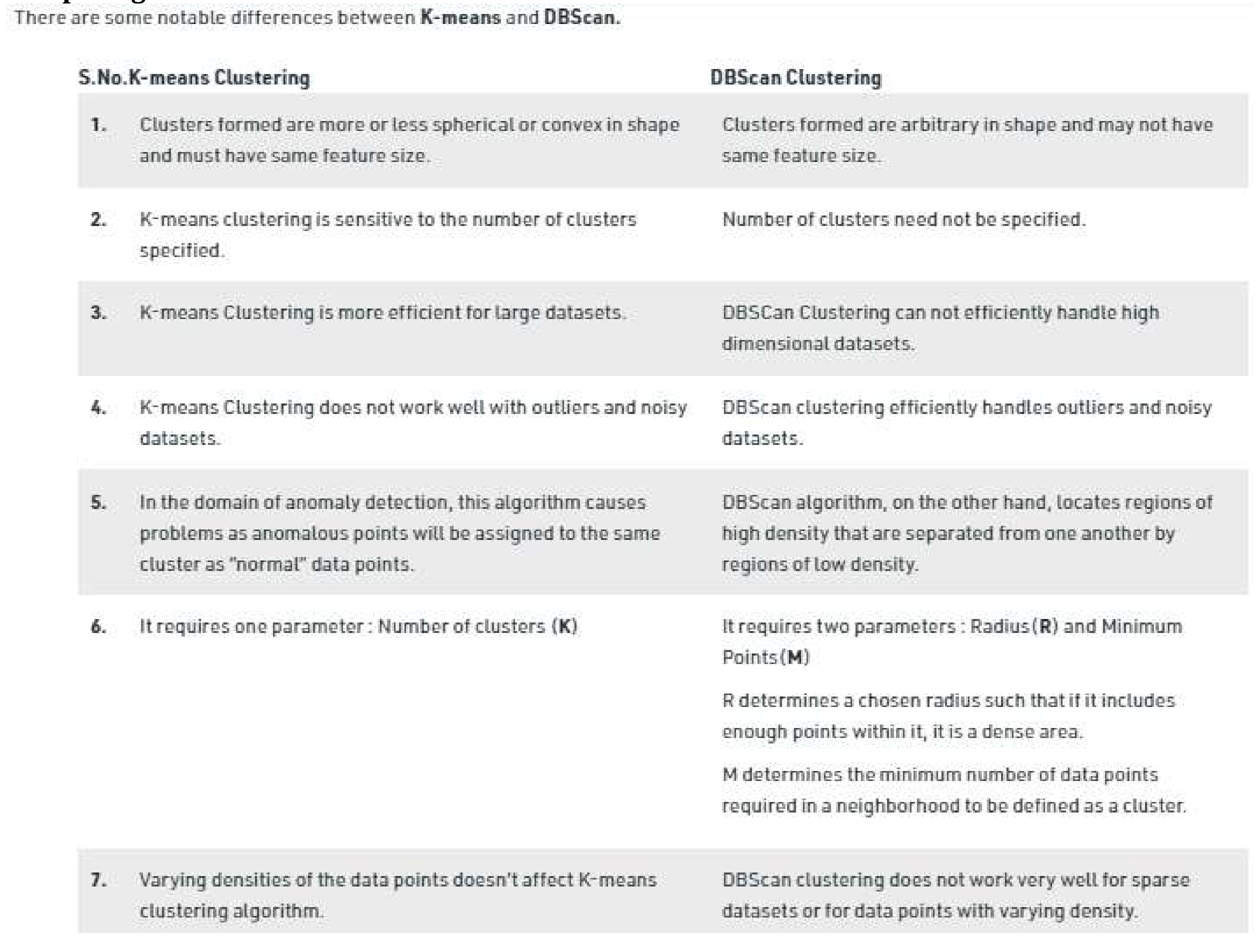
• Thus, **DBSCAN can find many clusters that could not be found using K-means**.

• **DBSCAN has trouble** when **the clusters have widely varying densities.**

• It also has **trouble with high-dimensional data** because density is **more difficult to define for such data.**

• **DBSCAN can be expensive** when the **computation** of **nearest neighbors requires computing all pairwise proximities,** as is usually the **case for high-dimensional data**.

Comparing K means and DBSCAN



There are some notable differences between **K-means** and **DBScan.**

S.No.	K-means Clustering	DBScan Clustering
1.	Clusters formed are more or less spherical or convex in shape and must have same feature size.	Clusters formed are arbitrary in shape and may not have same feature size.
2.	K-means clustering is sensitive to the number of clusters specified.	Number of clusters need not be specified.
3.	K-means Clustering is more efficient for large datasets.	DBSCan Clustering can not efficiently handle high dimensional datasets.
4.	K-means Clustering does not work well with outliers and noisy datasets.	DBScan clustering efficiently handles outliers and noisy datasets.
5.	In the domain of anomaly detection, this algorithm causes problems as anomalous points will be assigned to the same cluster as "normal" data points.	DBScan algorithm, on the other hand, locates regions of high density that are separated from one another by regions of low density.
6.	It requires one parameter : Number of clusters (**K**)	It requires two parameters : Radius(**R**) and Minimum Points(**M**) R determines a chosen radius such that if it includes enough points within it, it is a dense area. M determines the minimum number of data points required in a neighborhood to be defined as a cluster.
7.	Varying densities of the data points doesn't affect K-means clustering algorithm.	DBScan clustering does not work very well for sparse datasets or for data points with varying density.

MODEL BASED CLUSTERING:

In order **to understand our data**, we will assume that there is a **generative process** (a model) that **creates/describes the data**, and **we will try to find the model that best fits the data**.

• **Models of different complexity can be defined**, but we will assume that **our model is a distribution from which data points are sampled**.

• Example: **the data is the height of all people in US**

In most cases, **a single distribution is not** good **enough to describe all data** points: different parts of the data follow a different distribution.

• Example: **the data is the height of all people in US and China**

• **We need a mixture model**

• **Different distributions correspond to different clusters in the data**.