DIGITAL NOTES ON DATA WAREHOUSING AND DATA MINING (R20A1206)

B. TECH III Year - II Sem

(2023-24)

PREPARED BY

P.HARIKRISHNA

P.V.NARESH

T.SHILPA

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
Sponsored by CMR Educational Society
(Affiliated to JNTU, Hyderabad, Approved by AICTE- Accredited by NBA& NAAC–'A'Grade-ISO9001:2008Certified) Maisammaguda, Dhulapally(PostViaHakimpet), Secunderabad–500100,TelanganaState,India.
Contact Number: 040-23792146/64634237, E-Mail ID: mrcet2004@gmail.com, website: www.mrcet.ac.in

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

SYLLABUS

III Year B. Tech. IT–II Sem **L/T/P/C**
3/-/-/3

(R20A1206) DATA WAREHOUSING AND DATA MINING

Course Objectives:

Students will be able:

1. To study the data warehouse principles.
2. To understand the working of data mining concepts.
3. To identify the association rules in mining.
4. To define the classification algorithms.
5. To imbibe the clustering techniques.

UNIT-I

Data warehouse: Introduction to Data warehouse, Difference between operational database systems and data warehouses, Data warehouse Characteristics, Data warehouse Architecture and its Components, Extraction-Transformation-Loading, Logical(Multi-Dimensional), Data Modeling, Schema Design, Star and Snow- Flake Schema, Fact Constellation, Fact Table, OLAP Cube, OLAP Operations, OLAP Server Architecture-ROLAP, MOLAP and HOLAP.

UNIT-II

Introduction: Fundamentals of data mining, Data Mining Functionalities, Classification of Data Mining systems, Data Mining Task Primitives, Integration of a Data Mining System with a Database or Data Warehouse System, Major issues in Data Mining.

Data Preprocessing: Need for Preprocessing the Data, Data Cleaning, Data Integration & Transformation, Data Reduction, Discretization and Concept Hierarchy Generation.

UNIT-III

Association Rules: Problem Definition, Frequent Item Set Generation, The APRIORI Principle, Support and Confidence Measures, Association Rule Generation; APRIOIRI Algorithm, The Partition Algorithms, FP- Growth Algorithms, Compact Representation of Frequent Item Set- Maximal Frequent Item Set, Closed Frequent Item Set.

UNIT-IV

Classification: Problem Definition, General Approaches to solving a classification problem, Evaluation of Classifiers , Classification techniques, Decision Trees-Decision tree Construction, Algorithm for

Decision tree Induction , Naive-Bayes Classifier, Bayesian Belief Networks; K- Nearest neighbor classification-Algorithm and Characteristics.

UNIT-V

Clustering: Clustering Overview, A Categorization of Major Clustering Methods, Partitioning Methods, Hierarchical Methods, Partitioning Clustering-K-Means Algorithm, PAM Algorithm; Hierarchical Clustering- Agglomerative Methods and divisive methods, Key Issues in Hierarchical Clustering, Strengths and Weakness, Outlier Detection.

TEXT BOOKS:

1) Data Mining- Concepts and Techniques- Jiawei Han, Micheline Kamber, Morgan KaufmannPublishers, Elsevier, 2 Edition, 2006.
2) Introduction to Data Mining, Psng-Ning Tan, Vipin Kumar, Michael Steinbanch, Pearson Education.

REFERENCE BOOKS:

1) Data Mining Techniques, Arun K Pujari, 3rd Edition, Universities Press.
2) Data Warehousing Fundament's, Pualraj Ponnaiah, Wiley Student Edition.
3) The Data Warehouse Life Cycle Toolkit — Ralph Kimball, Wiley Student Edition.
4) Data Mining, Vikaram Pudi, P Radha Krishna, Oxford University Press.

Course Outcomes:

The students will be able:

- To comprehend the data warehouse in addition to database systems.
- To perform the pre-processing of data.
- To apply mining techniques on the data.
- To categorize the association rules, classification and clusters in large data sets.
- To solve real world problems in business and scientific information using data mining .

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

INDEX

UNIT - I

Introduction to Data Warehouse:

A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process.

Subject-Oriented: A data warehouse can be used to analyze a particular subject area. For example, "sales" can be a particular subject.

Integrated: A data warehouse integrates data from multiple data sources. For example, source A and source B may have different ways of identifying a product, but in a data warehouse, there will be only a single way of identifying aproduct.

Time-Variant: Historical data is kept in a data warehouse. For example, one can retrieve data from 3 months, 6 months, 12 months, or even older data from a data warehouse. This contrasts with a transactions system, where often only the most recent data is kept. For example, a transaction system may hold the most recent address of a customer, where a data warehouse can hold all addresses associated with a customer.

Non-volatile: Once data is in the data warehouse, it will not change. So, historical data in a data warehouse should never be altered.

Data Warehouse Design Process:

A data warehouse can be built using a top-down approach, a bottom-up approach, or a combination of both.

- The top-down approach starts with the overall design and planning. It is useful in cases where the technology is mature and well known, and where the business problems that must be solved are clear and well understood.
- The bottom-up approach starts with experiments and prototypes. This is useful in the early stage of business modelling and technology development. It allows an organization to move forward at considerably less expense and to evaluate the benefits of the technology before making significant commitments.
- In the combined approach, an organization can exploit the planned and strategic nature of the top-down approach while retaining the rapid implementation and opportunistic application of the bottom-up approach.

The warehouse design process consists of the following steps:

- Choose a business process to model, for example, orders, invoices, shipments, inventory, account administration, sales, or the general ledger. If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed. However, if the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.
- Choose the grain of the business process. The grain is the fundamental, atomic level of data to be represented in the fact table for this process, for example, individual transactions, individual daily snapshots, and so on.
- Choose the dimensions that will apply to each fact table record. Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
- Choose the measures that will populate each fact table record. Typical measures are numeric additive quantities like dollars sold and units sold.

Differences between operational database systems and data warehouse

Operational Database	Data Warehouse
Operational systems are designed to support high-volume transaction processing.	Data warehousing systems are typically designed to support high-volume analytical processing (i.e., OLAP).
Operational systems are usually concerned with current data.	Data warehousing systems are usually concerned with historical data.
Data within operational systems are mainly updated regularly according to need.	Non-volatile, new data may be added regularly. Once Added rarely changed.
It is designed for real-time business dealing and processes.	It is designed for analysis of business measures by subject area, categories, and attributes.
It is optimized for a simple set of transactions, generally adding or retrieving a single row at a time per table.	It is optimized for extent loads and high, complex, unpredictable queries that access many rows per table.

Characteristics of Data Warehouse

Subject-Oriented

A data warehouse target on the modeling and analysis of data for decision-makers. Therefore, data warehouses typically provide a concise and straightforward view around a particular subject, such as customer, product, or sales, instead of the global organization's ongoing operations. This is done by excluding data that are not useful concerning the subject and including all data needed by the users to understand the subject.

Integrated

A data warehouse integrates various heterogeneous data sources like RDBMS, flat files, and online transaction records. It requires performing data cleaning and integration during data warehousing to ensure consistency in naming conventions, attributes types, etc., among different data sources.

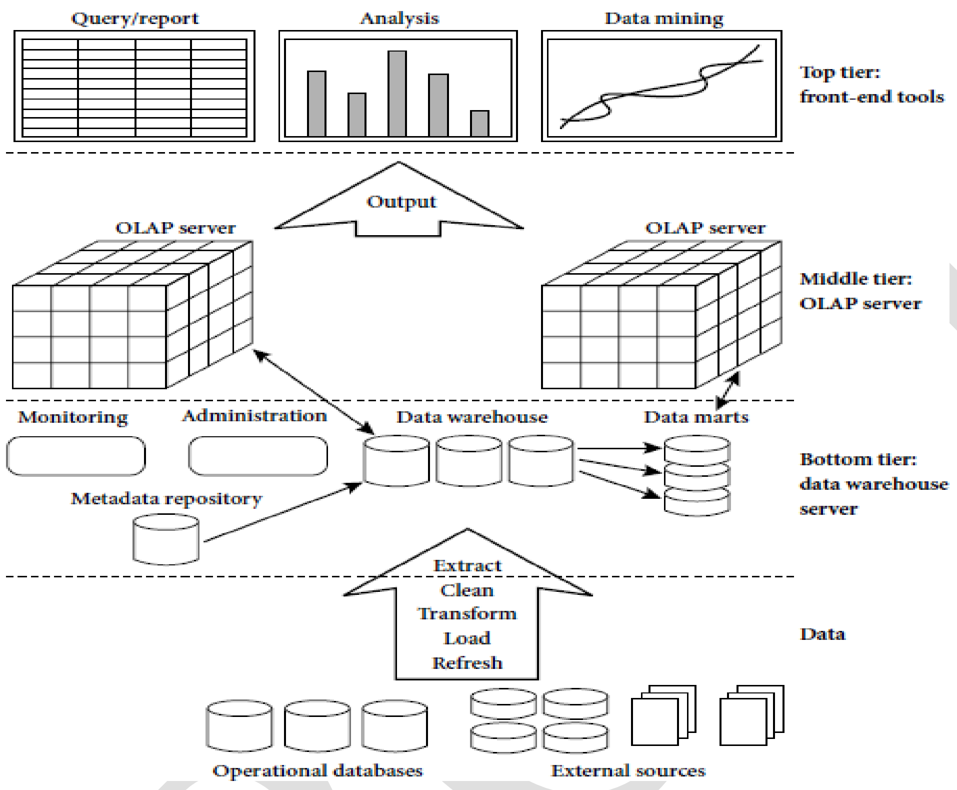
Time-Variant

Historical information is kept in a data warehouse. For example, one can retrieve files from 3 months, 6 months, 12 months, or even previous data from a data warehouse. These variations with a transactions system, where often only the most current file is kept.

Non-Volatile

The data warehouse is a physically separate data storage, which is transformed from the source operational RDBMS. The operational updates of data do not occur in the data warehouse, i.e., update, insert, and delete operations are not performed. It usually requires only two procedures in data accessing: Initial loading of data and access to data. Therefore, the DW does not require transaction processing, recovery, and concurrency capabilities, which allows for substantial speedup of data retrieval. Non-Volatile defines that once entered into the warehouse, and data should not change.

A Three Tier Data Warehouse Architecture:



Tier-1:

The bottom tier is a warehouse database server that is almost always a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (such as customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse. The data are extracted using application program interfaces known as gateways. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Open Linking and Embedding for Databases) by Microsoft and JDBC (Java Database Connection). This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

Tier-2:

The middle tier is an OLAP server that is typically implemented using either a relational OLAP (ROLAP) model or a multidimensional OLAP.

- OLAP model is an extended relational DBMS that maps operations on multidimensional data to standard relational operations.
- A multidimensional OLAP (MOLAP) model, that is, a special-purpose server that directly implements multidimensional data and operations.

Tier-3:

The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

Multidimensional Model

A multidimensional model views data in the form of a data-cube. A data cube enables data to be modelled and viewed in multiple dimensions. It is defined by dimensions and facts. The dimensions are the perspectives or entities concerning which an organization keeps records. For example, a shop may create a sales data warehouse to keep records of the store's sales for the dimension time, item, and location. These dimensions allow the save to keep track of things, for example, monthly sales of items and the locations at which the items were sold. Each dimension has a table related to it, called a dimensional table, which describes the dimension further. For example, a dimensional table for an item may contain the attributes item name, brand, and type. A multidimensional data model is organized around a central theme, for example, sales. This theme is represented by a fact table. Facts are numerical measures. The fact table contains the names of the facts or measures of the related dimensional tables.

Data Warehouse Models:

There are three data warehouse models.

1. **Enterprise warehouse:**

    - An enterprise warehouse collects all of the information about subjects spanning the entire organization.
    - It provides corporate-wide data integration, usually from one or more operational

systems or external information providers, and is cross-functional in scope.

- It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.
- An enterprise data warehouse may be implemented on traditional mainframes, computer super servers, or parallel architecture platforms. It requires extensive business modeling and may take years to design and build.

2. **Data mart:**

- A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects. For example, a marketing data mart may confine its subjects to customer, item, and sales. The data contained in data marts tend to be summarized.
- Data marts are usually implemented on low-cost departmental servers that are UNIX/LINUX- or Windows-based. The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years. However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.
- Depending on the source of data, data marts can be categorized as independent more dependent. Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area. Dependent data marts are source directly from enterprise data warehouses.

3. **Virtual warehouse:**

- A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary views may be materialized.
- A virtual warehouse is easy to build but requires excess capacity on operational database servers.

Meta Data Repository:

Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouse objects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for time stamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.

A metadata repository should contain the following:

- A description of the structure of the data warehouse, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- Operational metadata, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The algorithms used for summarization, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- The mapping from the operational environment to the data warehouse, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).
- Data related to system performance, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.
- Business metadata, which include business terms and definitions, data ownership information, and charging policies

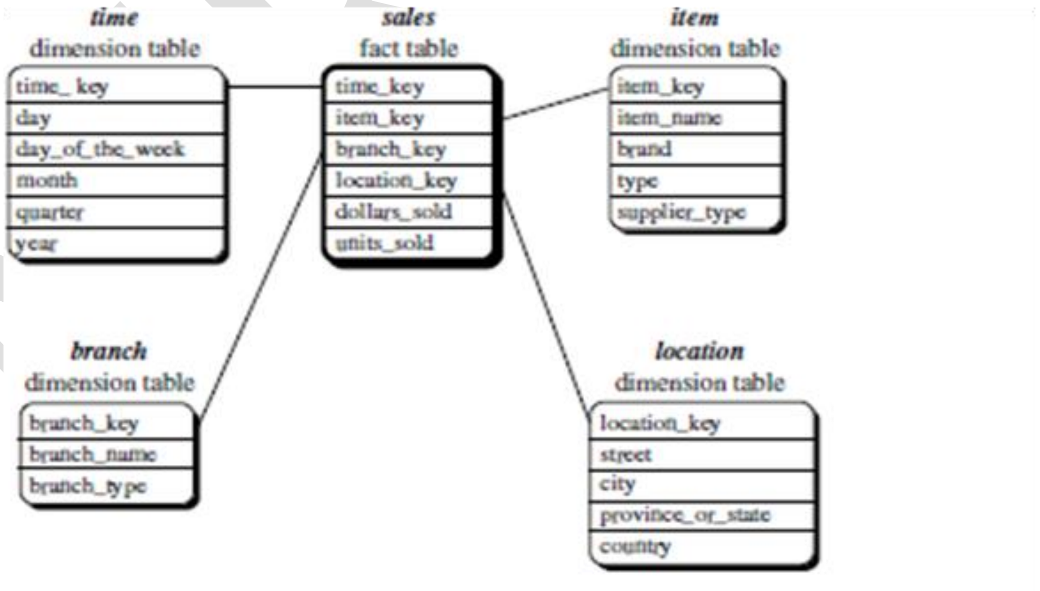
Schema Design:

Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Databases The entity- relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities and the relationships between them. Such a data model is appropriate for on- line transaction processing. A data warehouse, however, requires a concise, subject-oriented schema that facilitates on-line data analysis. The most popular data model for a data warehouse is a multidimensional model. Such a model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema. Let’s look at each of these schema types. Star schema: The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (fact table) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (dimension tables), one for each dimension.

Star schema:

A star schema for All Electronics sales is shown in Figure. Sales are considered along four dimensions, namely, time, item, branch, and location. The schema contains a central fact table for sales that contains keys to each of the four dimensions, along with two measures: dollars sold and units sold. To minimize the size of the fact table, dimension identifiers (such as time key and item key) are system-generated identifiers. Notice that in the star schema, each dimension is represented by only one table, and each table contains a set of attributes. For example, the location dimension table contains the attribute set {location key, street, city, province or state, country}. This constraint may introduce some redundancy.

For example, "Vancouver" and "Victoria" are both cities in the Canadian province of British Columbia. Entries for such cities in the location dimension table will create redundancy among the attributes province or state and country, that is, (..., Vancouver, British Columbia, Canada) and (..., Victoria, British Columbia, Canada). Moreover, the attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).

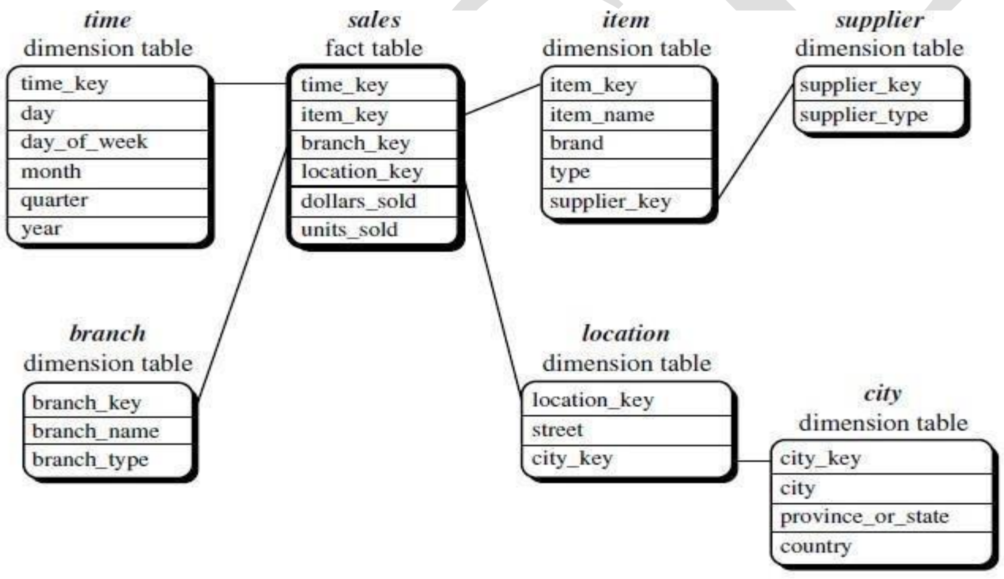


Star schema of a data warehouse for sales.

Snowflake schema.:

A snowflake schema for All Electronics sales is given in Figure Here, the sales fact table is identical to that of the star schema in Figure. The main difference between the two schemas is in the definition of dimension tables.

The single dimension table for item in the star schema is normalized in the snowflake schema, resulting in new item and supplier tables. For example, the item dimension table now contains the attributes item key, item name, brand, type, and supplier key, where supplier key is linked to the supplier dimension table, containing supplier key and supplier type information. Similarly, the single dimension table for location in the star schema can be normalized into two new tables: location and city.



Snowflake schema of a data warehouse for sales.

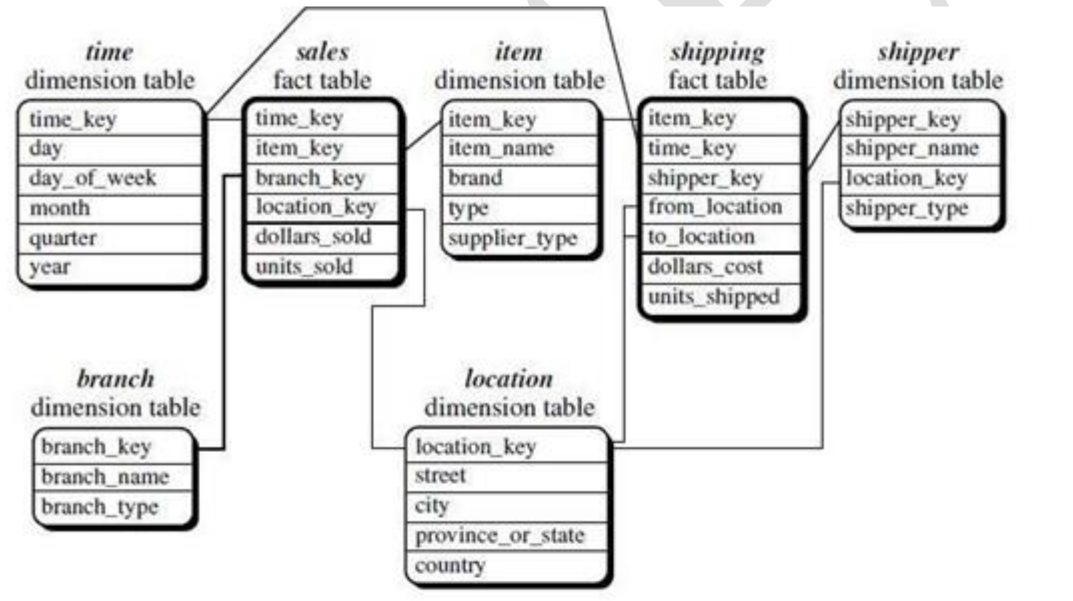
Notice that further normalization can be performed on province or state and country in the snowflake schema.

Fact constellation.

A fact constellation schema is shown in Figure. This schema specifies two fact tables, sales and shipping. The sales table definition is identical to that of the star schema . The shipping table has five dimensions, or keys: item key, time key, shipper key, from location, and to location, and two measures: dollars cost and units shipped.

A fact constellation schema allows dimension tables to be shared between fact tables. For example, the dimensions tables for time, item, and location are shared between both the sales and shipping fact tables. In data warehousing, there is a distinction between a data warehouse and a data mart.

A data warehouse collects information about subjects that span the entire organization, such as customers, items, sales, assets, and personnel, and thus its scope is enterprise-wide. For data warehouses, the fact constellation schema is commonly used, since it can model multiple, interrelated subjects. A data mart, on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is department wide. For data marts, the star or snowflake schema are commonly used, since both are geared toward modeling single subjects, although the star schema is more popular and efficient.



5 Fact constellation schema of a data warehouse for sales and shipping.

Measures: Their Categorization and Computation:

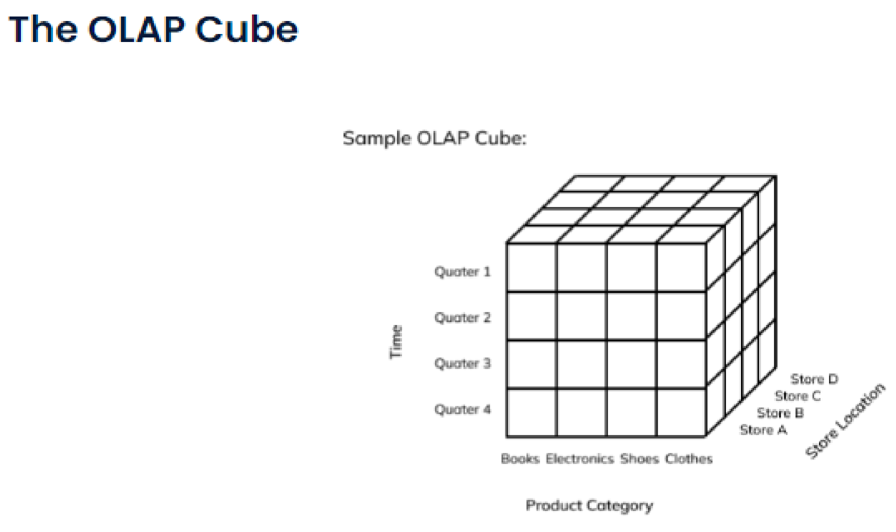
“How are measures computed?” To answer this question, we first study how measures can be categorized.1 Note that a multidimensional point in the data cube space can be defined by a set of dimension-value pairs, for example, h time = “Q1”, location = “Vancouver”, item = “computer” . A data cube measure is a numerical function that can be evaluated at each point in the data cube space. A measure value is computed for a given point by aggregating the data corresponding to the respective dimension-value pairs defining the given point.

Measures can be organized into three categories (i.e., distributive, algebraic, holistic), based on the kind of aggregate functions used.

Distributive: An aggregate function is distributive if it can be computed in a distributed manner as follows. Suppose the data are partitioned into n sets. We apply the function to each partition, resulting in n aggregate values. If the result derived by applying the function to the n aggregate values is the same as that derived by applying the function to the entire data set (without partitioning), the function can be computed in a distributed manner. For example, count() can be computed for a data cube by first partitioning the cube into a set of sub cubes, computing count() for each sub cube, and then summing up the counts obtained for each sub cube. Hence, count() is a distributive aggregate function. For the same reason, sum(), min(), and max() are distributive aggregate functions. A measure is distributive if it is obtained by applying a distributive aggregate function. Distributive measures can be computed efficiently because they can be computed in a distributive manner.

OLAP (Online analytical Processing):

- OLAP is an approach to answering multi-dimensional analytical (MDA) queries swiftly.
- OLAP is part of the broader category of business intelligence, which also encompasses relational database, report writing and data mining.
- OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives.



OLAP consists of three basic analytical operations:

- Consolidation (Roll-Up)

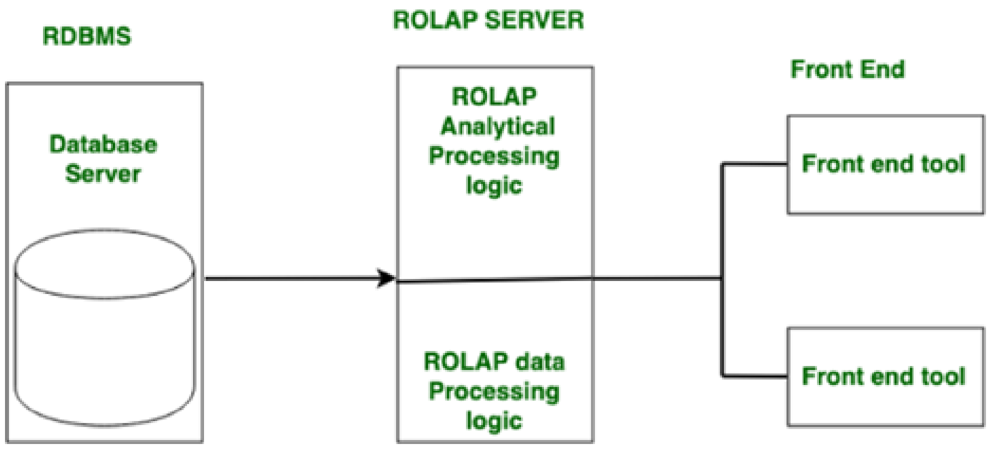
- Drill-Down
- Slicing And Dicing

- **Consolidation** involves the aggregation of data that can be accumulated and computed in one or more dimensions. For example, all sales offices are rolled up to the sales
department or sales division to anticipate sales trends.
- **The drill-down** is a technique that allows users to navigate through the details. For instance, users can view the sales by individual products that make up a region's sales.
- **Slicing and dicing** is a feature whereby users can take out (slicing) a specific set of data of the OLAP cube and view (dicing) the slices from different viewpoints.

Types of OLAP:

1. Relational OLAP (ROLAP):

- ROLAP works directly with relational databases. The base data and the dimension tables are stored as relational tables and new tables are created to hold the aggregated information. It depends on a specialized schema design.
- This methodology relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. In essence, each action of slicing and dicing is equivalent to adding a "WHERE" clause in the SQL statement. ROLAP tools do not use pre-calculated data cubes but instead pose the query to the standard relational database and its tables in order to bring back the data required to answer the question.
- ROLAP tools feature the ability to ask any question because the methodology does not limit to the contents of a cube. ROLAP also has the ability to drill down to the lowest level of detail in the database.



Benefits:

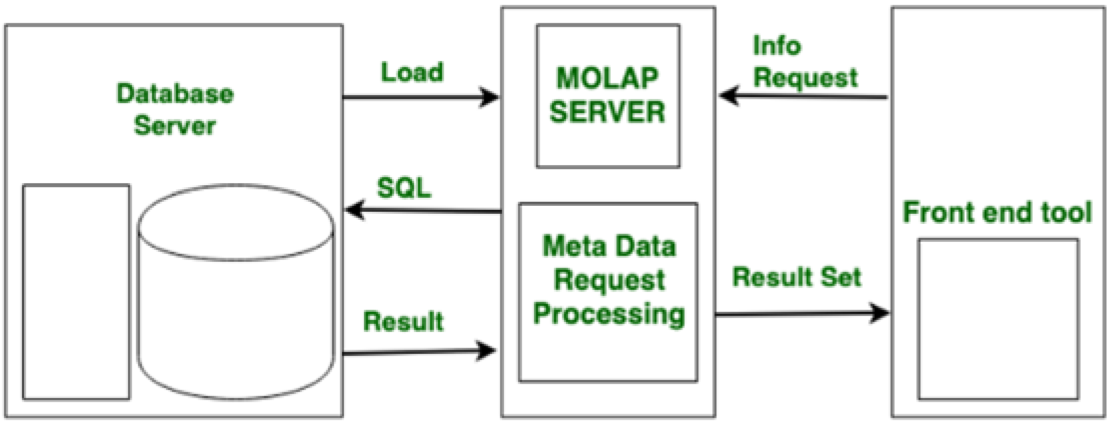
- It is compatible with data warehouses and OLTP systems.
- The data size limitation of ROLAP technology is determined by the underlying RDBMS. As a result, ROLAP does not limit the amount of data that can be stored.

Limitations:

- SQL functionality is constrained.
- It's difficult to keep aggregate tables up to date.

2. Multidimensional OLAP (MOLAP):

- MOLAP is the 'classic' form of OLAP and is sometimes referred to as just OLAP.
- MOLAP stores this data in an optimized multi-dimensional array storage, rather than in a relational database. Therefore it requires the pre-computation and storage of information in the cube - the operation known as processing.
- MOLAP tools generally utilize a pre-calculated data set referred to as a data cube. The data cube contains all the possible answers to a given range of questions.
- MOLAP tools have a very fast response time and the ability to quickly write back data into the data set.



Benefits:

• Suitable for slicing and dicing operations.

• Outperforms ROLAP when data is dense.

• Capable of performing complex calculations.

Limitations:

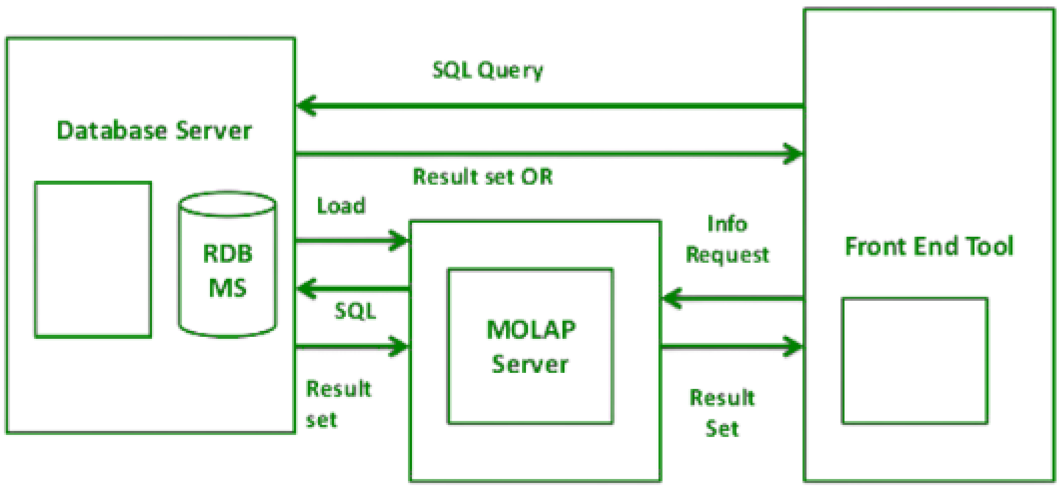
• It is difficult to change the dimensions without re-aggregating.

• Since all calculations are performed when the cube is built, a large amount of data cannot be stored in the cube itself.

3. Hybrid OLAP (HOLAP):

• There is no clear agreement across the industry as to what constitutes Hybrid OLAP, except that a database will divide data between relational and specialized storage.

- For example, for some vendors, a HOLAP database will use relational tables to hold the larger quantities of detailed data, and use specialized storage for at least some aspects of the smaller quantities of more-aggregate or less-detailed data.
- HOLAP addresses the shortcomings of MOLAP and ROLAP by combining the capabilities of both approaches.
- HOLAP tools can utilize both pre-calculated cubes and relational data sources.



Benefits:

• HOLAP combines the benefits of MOLAP and ROLAP.

• Provide quick access at all aggregation levels.

Limitations

• Because it supports both MOLAP and ROLAP servers, HOLAP architecture is extremely complex.

• There is a greater likelihood of overlap, particularly in their functionalities.

UNIT-2

Fundamentals of Data Mining:

Data mining refers to extracting or mining knowledge from large amounts of data. The term is actually a misnomer. Thus, data mining should have been more appropriately named as knowledge mining which emphasis on mining from large amounts of data.

It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems.

The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

The key properties of data mining are

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Focus on large datasets and databases

The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides.

Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

Automated prediction of trends and behaviours

Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands- on analysis can now be answered directly from the data — quickly.

A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

Automated discovery of previously unknown patterns.
Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

Data Mining Functionalities:
We have observed various types of databases and information repositories on which datamining can be performed. Let us now examine the kinds of data patterns that can be mined. Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories: descriptive and predictive. Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions.

In some cases, users may have no idea regarding what kinds of patterns in their data may be interesting, and hence may like to search for several different kinds of patterns in parallel. Thus, it is important to have a data mining system that can mine multiple kinds of patterns to accommodate different user expectations or applications. Furthermore, data mining systems should be able to discover patterns at various granularity (i.e., different levels of abstraction). Data mining systems should also allow users to specify hints to guide or focus the search for interesting patterns. Because some patterns may not hold for all of the data in the database, a measure of certainty or "trustworthiness" is usually associated with each discovered pattern.

Data mining functionalities, and the kinds of patterns they can discover, are described Mining Frequent Patterns, Associations, and Correlations Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including item sets, subsequence's, and substructures.

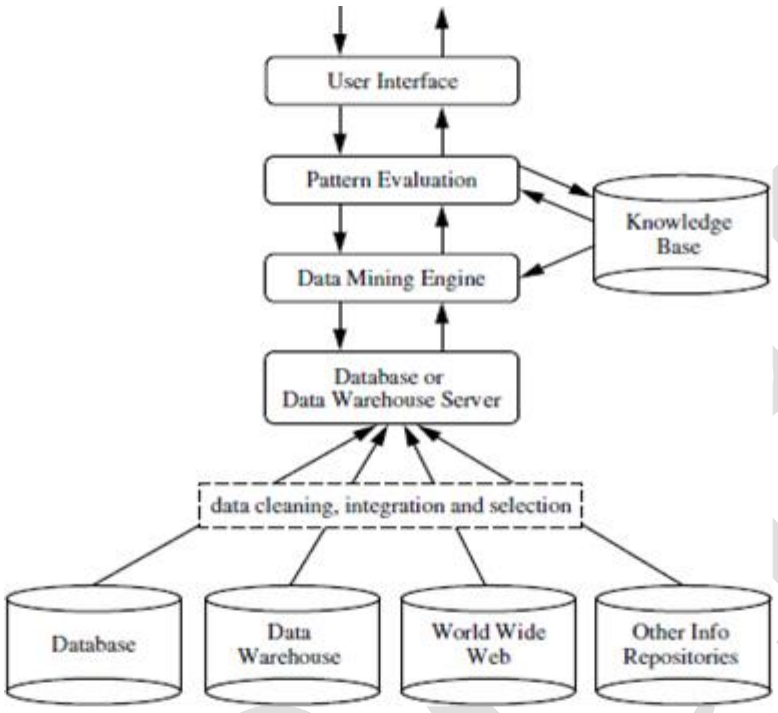
A frequent item set typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread. A frequently occurring subsequence, such as the pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with item sets or subsequence's. If a substructure occurs frequently, it is called a (frequent) structured pattern. Mining frequent patterns leads to the discovery of interesting associations and correlations within data below.

Data mining involves six common classes of tasks:

- **Anomaly detection (Outlier/change/deviation detection)** – The identification of unusual data records, that might be interesting or data errors that require further investigation.
- **Association rule learning (Dependency modelling)** – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.
- **Clustering** – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.
- **Classification** – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".
- **Regression** – attempts to find a function which models the data with the least error.
- **Summarization** – providing a more compact representation of the data set, including Visualization and report generation.

Architecture of Data Mining

A typical data mining system may have the following major components.



1. Knowledge Base:

This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern's interestingness based on its unexpectedness, may also be included. Other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

2. Data Mining Engine:

This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

3. Pattern Evaluation Module:

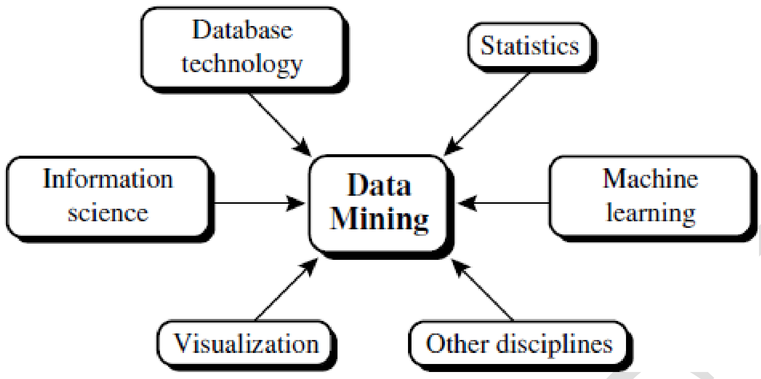
This component typically employs interestingness measures interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process as to confine the search to only the interesting patterns.

4. User interface:

This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

Classification of Data Mining Systems

Data mining is an interdisciplinary field, the confluence of a set of disciplines, including database systems, statistics, machine learning, visualization, and information science. Moreover, depending on the data mining approach used, techniques from other disciplines may be applied, such as neural networks, fuzzy and/or rough set theory, knowledge representation, inductive logic programming, or high-performance computing. Depending on the kinds of data to be mined or on the given data mining application, the data mining system may also integrate techniques from spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, computer graphics, Web technology, economics, business, bioinformatics, or psychology. Because of the diversity of disciplines contributing to data mining, data mining research is expected to generate a large variety of data mining systems. Therefore, it is necessary to provide a clear classification of data mining systems, which may help potential users distinguish between such systems and identify those that best match their needs.



Data mining systems can be categorized according to various criteria, as follows:

Classification according to the kinds of databases mined: A data mining system can be classified according to the kinds of databases mined. Database systems can be classified according to different criteria (such as data models, or the types of data or applications involved), each of which may require its own data mining technique. Data mining systems can therefore be classified accordingly.

For instance, if classifying according to data models, we may have a relational, transactional, object- relational, or data warehouse mining system. If classifying according to the special types of data handled, we may have a spatial, time-series, text, stream data, multimedia data mining system, or a World Wide Web mining system.

Classification according to the kinds of knowledge mined: Data mining systems can be categorized according to the kinds of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis. A comprehensive data mining system usually provides multiple and/or integrated data mining functionalities.

Moreover, data mining systems can be distinguished based on the granularity or levels of abstraction of the knowledge mined, including generalized knowledge (at a high level of abstraction), primitive-level knowledge (at a raw data level), or knowledge at multiple levels (considering several levels of abstraction). An advanced data mining system should facilitate the discovery of knowledge at multiple levels of abstraction

Data mining systems can also be categorized as those that mine data regularities (commonly occurring patterns) versus those that mine data irregularities (such as exceptions, or outliers). In general, concept description, association and correlation analysis, classification, prediction, and clustering mine data regularities, rejecting outliers as noise. These methods may also help detect outliers.

Classification according to the kinds of techniques utilized: Data mining systems can be categorized according to the underlying data mining techniques employed. These techniques can be described according to the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems, query-driven systems) or the methods of data analysis employed (e.g., database-oriented or data warehouse–oriented techniques, machine learning, statistics, visualization, pattern recognition, neural networks, and so on). A sophisticated data mining system will often adopt multiple data mining techniques or work out an effective, integrated technique that combines the merits of a few individual approaches.

Classification according to the applications adapted: Data mining systems can also be categorized according to the applications they adapt. For example, data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on. Different applications often require the integration of application-specific methods. Therefore, a generic, all-purpose data mining system may not fit domain-specific mining tasks.

Data Mining Process:

Data Mining is a process of discovering various models, summaries, and derived values from a given collection of data.
The general experimental procedure adapted to data-mining problems involves the following steps:
1. State the problem and formulate the hypothesis
Most data-based modeling studies are performed in a particular application domain. Hence, domain-specific knowledge and experience are usually necessary in order to come up with a meaningful problem statement. Unfortunately, many application studies tend to focus on the data- mining technique at the expense of a clear problem statement. In this step, a modeler usually specifies a set of variables for the unknown dependency and, if possible, a general

form of this dependency as an initial hypothesis. There may be several hypotheses formulated for a single problem at this stage.

The first step requires the combined expertise of an application domain and a data-mining model. In practice, it usually means a close interaction between the data-mining expert and the application expert. In successful data-mining applications, this cooperation does not stop in the initial phase; it continues during the entire data-mining process.

2. **Collect the data**

This step is concerned with how the data are generated and collected. In general, there are two distinct possibilities. The first is when the data-generation process is under the control of an expert (modeler): this approach is known as a designed experiment.

The second possibility is when the expert cannot influence the data- generation process: this is known as the observational approach. An observational setting, namely, random data generation, is assumed in most data-mining applications.

Typically, the sampling distribution is completely unknown after data are collected, or it is partially and implicitly given in the data-collection procedure. It is very important, however, to understand how data collection affects its theoretical distribution, since such a priori knowledge can be very useful for modeling and, later, for the final interpretation of results. Also, it is important to make sure that the data used for estimating a model and the data used later for testing and applying a model come from the same, unknown, sampling distribution. If this is not the case, the estimated model cannot be successfully used in a final application of the results.

Data Mining Task Primitives

A data mining task can be specified in the form of a data mining query, which is input to the data mining system. A data mining query is defined in terms of data mining task primitives. These primitives allow the user to interactively communicate with the data mining system during discovery to direct the mining process or examine the findings from different angles or depths. The data mining primitives specify the following,

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

Integration of a data mining system with a database system

The data mining system is integrated with a database or data warehouse system so that it can do its tasks in an effective presence. A data mining system operates in an environment that needed it to communicate with other data systems like a database system. There are the possible integration schemes that can integrate these systems which are as follows –

No coupling − No coupling defines that a data mining system will not use any function of a database or data warehouse system. It can retrieve data from a specific source (including a file system), process data using some data mining algorithms, and therefore save the mining results in a different file.

Loose Coupling − In this data mining system uses some services of a database or data warehouse system. The data is fetched from a data repository handled by these systems. Data mining approaches are used to process the data and then the processed data is saved either in a file or in a designated area in a database or data warehouse. Loose coupling is better than no coupling as it can fetch some area of data stored in databases by using query processing or various system facilities.

Semitight Coupling − In this adequate execution of a few essential data mining primitives can be supported in the database/Datawarehouse system. These primitives can contain sorting, indexing, aggregation, histogram analysis, multi-way join, and pre-computation of some important statistical measures, including sum, count, max, min, standard deviation, etc.

Tight coupling − Tight coupling defines that a data mining system is smoothly integrated into the database/data warehouse system. The data mining subsystem is considered as one functional element of an information system.

Major Issues in Data Mining:

Mining different kinds of knowledge in databases. - The need of different users is not the same. And Different user may be in interested in different kind of knowledge. Therefore, it is necessary for data mining to cover broad range of knowledge discovery task.

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 returned results.

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, visual representations. These representations should be easily understandable by the users.

Handling noisy or incomplete data. - The data cleaning methods are required that can handle the noise, incomplete objects while mining the data regularities. If data cleaning methods are not there then the accuracy of the discovered patterns will be poor.

Pattern evaluation. - It refers to interestingness of the problem. The patterns discovered should be interesting because either they represent common knowledge or lack novelty.

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 parallel. Then the results from the partitions are merged. The incremental algorithms, updates the databases without having to mine the data again from the scratch.

Data preprocessing: Data preprocessing is converting raw data into legible and defined sets that allow businesses to conduct data mining, analyze the data, and process it for business activities. It's important for businesses to preprocess their data correctly, as they use various forms of input to collect raw data, which can affect its quality. Preprocessing data is an important step, as raw data can be inconsistent or incomplete in its formatting. Effectively preprocessing raw data can increase its accuracy, which can increase the quality of projects and improve its reliability.

Importance of data preprocessing

Preprocessing data is an important step for data analysis. The following are some benefits of preprocessing data:

1. It improves accuracy and reliability. Preprocessing data removes missing or inconsistent data values resulting from human or computer error, which can improve the accuracy and quality of a dataset, making it more reliable.
2. It makes data consistent. When collecting data, it's possible to have data duplicates, and discarding them during preprocessing can ensure the data values for analysis are consistent, which helps produce accurate results.

3. It increases the data's algorithm readability. Preprocessing enhances the data's quality and makes it easier for machine learning algorithms to read, use, and interpret it.

Data Cleaning

Data cleaning is an essential step in the data mining process. It is crucial to the construction of a model. The step that is required, but frequently overlooked by everyone, is data cleaning. The major problem with quality information management is data quality. Problems with data quality can happen at any place in an information system. Data cleansing offers a solution to these issues.

Data cleaning is the process of correcting or deleting inaccurate, damaged, improperly formatted, duplicated, or insufficient data from a dataset. Even if results and algorithms appear to be correct, they are unreliable if the data is inaccurate. There are numerous ways for data to be duplicated or incorrectly labeled when merging multiple data sources.

In general, data cleaning lowers errors and raises the caliber of the data. Although it might be a time-consuming and laborious operation, fixing data mistakes and removing incorrect information must be done. A crucial method for cleaning up data is data mining. A method for finding useful information in data is data mining. Data quality mining is a novel methodology that uses data mining methods to find and fix data quality issues in sizable databases. Data mining mechanically pulls intrinsic and hidden information from large data sets. Data cleansing can be accomplished using a variety of data mining approaches.

To arrive at a precise final analysis, it is crucial to comprehend and improve the quality of your data. To identify key patterns, the data must be prepared. Exploratory data mining is understood. Before doing business analysis and gaining insights, data cleaning in data mining enables the user to identify erroneous or missing data.

Data Integration

Data integration is the process of combining data from multiple sources into a cohesive and consistent view. This process involves identifying and accessing the different data sources, mapping the data to a common format, and reconciling any inconsistencies or discrepancies between the sources. The goal of data integration is to make it easier to access and analyze data that is spread across multiple systems or platforms, in order to gain a more complete and accurate understanding of the data.

Data Transformation:

In data transformation, the data are transformed or consolidated into forms appropriate for mining.

Data transformation can involve the following:

1. **Smoothing,** which works to remove noise from the data. Such techniques include binning, regression, and clustering.
2. **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 analysis of the data at multiple granularities.
3. **Generalization of the data,** where low-level or—primitive‖ (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher- level concepts, like city or country.
4. **Normalization,** where the attribute data are scaled so as to fall within a small specified range, such as 1:0 to 1:0, or 0:0 to 1:0.
5. **Attribute construction** (or feature construction),where new attributes are constructed and added from the given set of attributes to help the mining process.

Data Reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Strategies for data reduction include the following:

Data cube aggregation: where aggregation operations are applied to the data in the construction of a data cube.

Attribute subset selection: where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

Dimensionality reduction: where encoding mechanisms are used to reduce the dataset size.

Numerosity reduction: where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data) or nonparametric methods such as clustering, sampling, and the use of histograms.

Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Data discretization is a form of numerosity reduction that is very useful for the automatic generation of concept hierarchies. Discretization and concept hierarchy generation are powerful tools for data mining, in that they allow the mining of data at multiple levels of abstraction.

UNIT-III

Association Rule Mining:

Association rule mining is a popular and well researched method for discovering interesting relations between variables in large databases.

It is intended to identify strong rules discovered in databases using different measures of interestingness.

Based on the concept of strong rules, Rakesh Agrawal et al. introduced association rules.

Market basket analysis:

This process analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket. Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

Example:

If customers who purchase computers also tend to buy anti virus software at the same time, then placing the hardware display close to the software display may help increase the sales of both items.

In an alternative strategy, placing hardware and software at opposite ends of the store may entice customers who purchase such items to pick up other items along the way. For instance, after deciding on an expensive computer ,a customer may observe security systems for sale while heading toward the software display to purchase antivirus software and may decide to purchase a

home security system as well. Market basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers as well as computers.

Frequent Pattern Mining:

Frequent pattern mining can be classified in various ways, based on the following criteria:

1. Based on the completeness of patterns to be mined:

   We can mine the complete set of frequent item sets, the closed frequent item sets, and the maximal frequent item sets, given a minimum support threshold.

   We can also mine constrained frequent item sets, approximate frequent item sets, near- match frequent item sets, top-k frequent item sets and soon.

2. Based on the levels of abstraction involved in the rules et:

   Some methods for association rule mining can find rules at differing levels of abstraction.

   For example, suppose that a set of association rules mined includes the following rules where X is a variable representing a customer:

   buys(X, ―computer‖))=>buys(X, ―HP printer‖) (1)

   buys(X, ―laptop computer‖)) =>buys(X, ―HP printer‖) (2)

In rule (1) and (2), the items bought are referenced at different levels of abstraction (e.g.,

―computer‖ is a higher-level abstraction of ―laptop computer‖).

3. Based on the number of data dimensions involved in the rule:

If the items or attributes in an association rule reference only one dimension, then it is a single-dimensional association rule.

buys(X, ―computer‖))=>buys(X, ―antivirus software‖).

If a rule references two or more dimensions, such as the dimensions age, income, and buys, then it is a multidimensional association rule. The following rule is an example of a multidimensional rule:

age(X, ―30,31…39‖) ^ income(X, ―42K,…48K‖))=>buys(X, ―high resolution TV‖)

4. Based on the types of values handled in the rule:

If a rule involves associations between the presence or absence of items, it is a Boolean association rule.

If a rule describes associations between quantitative items or attributes, then it is a quantitative association rule.

5. Based on the kinds of rules to be mined:

Frequent pattern analysis can generate various kinds of rules and other interesting relationships.

Association rule mining can generate a large number of rules, many of which are redundant or do not indicate a correlation relationship among item sets.

The discovered associations can be further analyzed to uncover statistical correlations, leading to correlation rules.

6. Based on the kinds of patterns to be mined:

Many kinds of frequent patterns can be mined from different kinds of data sets. Sequential pattern mining searches for frequent subsequences in a sequence data set, where a sequence records an ordering of events.

For example, with sequential pattern mining, we can study the order in which items are frequently purchased. For instance, customers may tend to first buy a PC, followed by a digital camera, and then a memory card.

Structured pattern mining searches for frequent sub structures in a structured data set. Single items are the simplest form of structure.

Each element of an item set may contain a subsequence, a sub tree, and so on. Therefore, structured pattern mining can be considered as the most general form of frequent pattern mining.

Apriori Algorithm:

Finding Frequent Item sets Using Candidate Generation: The Apriori Algorithm

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent item sets for Boolean association rules.

The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent item set properties.

Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-item sets.

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

The finding of each Lkrequires one full scan of the database.

A two-step process is followed in Apriori consisting of join and prune action.

Example:

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

Steps:

1. In the first iteration of the algorithm, each item is a member of the set of candidate1- item sets, C1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

2. Suppose that the minimum support count required is 2, that is, min sup = 2. The set of frequent 1-itemsets, L1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. In our example, all of the candidates in C1 satisfy minimum support.

3. To discover the set of frequent 2-itemsets, L2, the algorithm uses the join L1 on L1 to generate a candidate set of 2-itemsets, C2.No candidates are removed fromC2 during the prune step because each subset of the candidates is also frequent.

4. Next, the transactions in D are scanned and the support count of each candidate itemsetInC2 is accumulated.

5. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate2- item sets in C2 having minimum support.

6. The generation of the set of candidate 3-itemsets,C3, From the join step, we first getC3

=L2x L2 = ({I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4},{I2, I3, I5}, {I2, I4, I5}. Based on the Apriori property that all subsets of a frequent item set must also be frequent, we can determine that the four latter candidates cannot possibly be frequent.

7. The transactions in D are scanned in order to determine L3, consisting of those candidate 3-itemsets in C3 having minimum support.

8. The algorithm uses L3x L3 to generate a candidate set of 4-itemsets, C4.

FP-growth (finding frequent item sets without candidate generation).

We re-examine the mining of transaction database, D using the frequent pattern growth approach.

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or list is denoted L.

An FP-tree is then constructed as follows. First, create the root of the tree, labeled with “null.” Scan database D a second time. The items in each transaction are processed in L order (i.e., sorted according to descending support count), and a branch is created for each transaction. For example, the scan of the first transaction, “T100: I1, I2, I5,” which contains three items (I2, I1, I5 in L order), leads to the construction of the first branch of the tree with three nodes, hI2: 1i, hI1:1i, and hI5: 1i, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T200, contains the items I2 and I4 in L order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix, I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, hI4: 1i,which is linked as a child of hI2: 2i. In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in tree via a chain of node-links. The tree obtained after scanning all of the transactions with the associated node-links. In this way, the problem of mining frequent patterns in databases is transformed to that of mining the FP-tree.

The FP-tree is mined as follows.

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

We first consider I5, which is the last item in L, rather than the first. The reason for starting at the end of the list will become apparent as we explain the FP-tree mining process. I5 occurs in two branches of the FP-tree (The occurrences of I5 can easily be found by following its chain of node-links.) The paths formed by these branches are hI2, I1, I5: 1i and hI2, I1, I3, I5: 1i.

Therefore, considering I5 as a suffix, its corresponding two prefix paths are hI2, I1: 1i and hI2, I1, I3: 1i, which form its conditional pattern base. Its conditional FP-tree contains only a single path, hI2: 2, I1: 2i; I3 is not included because its support count of 1 is less than the minimum support count.

The single path generates all the combinations of frequent patterns: fI2, I5: 2g, fI1, I5: 2g, fI2, I1, I5: 2g. Generating Association Rules from Frequent Item sets:

Once the frequent item sets from transactions in a database D have been found, it is straightforward to generate strong association rules from them.

Compact Representation of Frequent Item Set:

For many applications, it is difficult to find strong associations among data items at low or primitive levels of abstraction due to the sparsity of data at those levels.

Strong associations discovered at high levels of abstraction may represent commonsense knowledge.

Therefore, data mining systems should provide capabilities for mining association rules at multiple levels of abstraction, with sufficient flexibility for easy traversal among different abstraction spaces.

Association rules generated from mining data at multiple levels of abstraction are called multiple-level or multilevel association rules.

Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework.

In general, a top-down strategy is employed, where counts are accumulated for the calculation of frequent item sets at each concept level, starting at the concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent item sets can be found.

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher level, more general concepts. Data can be generalized by replacing low-level concepts within the data by their higher-level concepts, or ancestors, from a concept hierarchy.

The concept hierarchy has five levels, respectively referred to as levels 0to 4, starting with level 0 at the root node for all.

Here, Level 1 includes computer, software, printer&camera, and computer accessory. Level 2 includes laptop computer, desktop computer, office software, antivirus software Level 3 includes IBM desktop computer, . . . , Microsoft office software, and so on.

Level 4 is the most specific abstraction level of this hierarchy.

2.5.1 Approaches For Mining Multilevel Association Rules:

1. Uniform Minimum Support:

The same minimum support threshold is used when mining at each level of abstraction. When a uniform minimum support threshold is used, the search procedure is simplified.

The method is also simple in that users are required to specify only one minimum support threshold.

The uniform support approach, however, has some difficulties. It is unlikely that items at lower levels of abstraction will occur as frequently as those at higher levels of abstraction.

If the minimum support threshold is set too high, it could miss some meaningful associations occurring at low abstraction levels. If the threshold is set too low, it may generate many uninteresting associations occurring at high abstraction levels.

2. Reduced Minimum Support:

Each level of abstraction has its own minimum support threshold.

The deeper the level of abstraction, the smaller the corresponding threshold is.

For example, the minimum support thresholds for levels 1 and 2 are 5% and %,respectively. In this way, ―computer,‖ ―laptop computer,‖ and ―desktop computer‖ are all considered frequent.

3. Group-Based Minimum Support:

Because users or experts often have insight as to which groups are more important than others, it is sometimes more desirable to set up user-specific, item, or group based minimal support thresholds when mining multilevel rules.

For example, a user could set up the minimum support thresholds based on product price, or on items of interest, such as by setting particularly low support thresholds for laptop computers and these categories. Mining Multidimensional Association Rules from Relational **Databases and Data Warehouses:**

Single dimensional or intra dimensional association rule contains a single distinct predicate (e.g., buys)with multiple occurrences i.e., the predicate occurs more than once within the rule. buys(X, ―digital camera‖)=>buys(X, ―HP printer‖)

Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules.

age(X, “20…29”)^occupation(X, “student”)=>buys(X, “laptop”)

Above Rule contains three predicates (age, occupation, and buys), each of which occurs only once in the rule. Hence, we say that it has no repeated predicates.

Multidimensional association rules with no repeated predicates are called inter dimensional association rules.

We can also mine multidimensional association rules with repeated predicates, which contain multiple occurrences of some predicates. These rules are called hybrid- dimensional association rules. An example of such a rule is the following, where the predicate buys is repeated:

age(X, ―20…29‖)^buys(X, ―laptop‖)=>buys(X, ―HP printer‖)

Mining Quantitative Association Rules:

Quantitative association rules are multidimensional association rules in which the numeric attributes are dynamically discretized during the mining process so as to satisfy some mining criteria, such as maximizing the confidence or compactness of the rules mined.

In this section, we focus specifically on how to mine quantitative association rules having two quantitative attributes on the left-hand side of the rule and one categorical attribute on the right- hand side of the rule. That is

Aquan1 ^Aquan2 =>Acat

Where Aquan1 and Aquan2 are tests on quantitative attribute interval

A cat tests a categorical attribute from the task-relevant data.

Such rules have been referred to as two-dimensional quantitative association rules, because they contain two quantitative dimensions.For instance, suppose you are curious about the association relationship between pairs of quantitative attributes, like customer age and income, and the type of television (such as high-definition TV, i.e., HDTV) that customers like to buy.

An example of such a 2-D quantitative association rule is age(X, ―30…39‖)^income(X, ―42K…48K‖)=>buys(X, ―HDTV‖)

From Association Mining to Correlation Analysis:

A correlation measure can be used to augment the support-confidence framework for association rules. This leads to correlation rules of the form

A=>B [support, confidence, correlation]

That is, a correlation rule is measured not only by its support and confidence but also by the correlation between item sets A and B. There are many different correlation measures from which to choose. In this section, we study various correlation measures to determine which would be good for mining large data sets.

Lift is a simple correlation measure that is given as follows. The occurrence of item set A is independent of the occurrence of item set B if = P(A)P(B); otherwise, item sets A and B are dependent and correlated as events. This definition can easily be extended to more than two item sets.

The lift between the occurrence of A and B can be measured by computing If the lift(A,B) is less than 1, then the occurrence of A is negatively correlated with the occurrence of B.

If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then A and B are independent and there is no correlation between them. Maximal Frequent Itemsets

The number of frequent itemsets generated by the Apriori algorithm can often be very large, so it is beneficial to identify a small representative set from which every frequent itemset can be derived. One such approach is using maximal frequent itemsets.

A maximal frequent itemset is a frequent itemset for which none of its immediate supersets are frequent. To illustrate this concept, consider the example given below:

The support counts are shown on the top left of each node. Assume support count threshold = 50%, that is, each item must occur in 2 or more transactions. Based on that threshold, the frequent itemsets are a, b, c, d, ab, ac, and ad (shaded nodes).

Out of these 7 frequent itemsets, 3 are identified as maximal frequent (having red outline):

• ab: Immediate supersets abc and abd are infrequent.

• ac: Immediate supersets abc and acd are infrequent.

• ad: Immediate supersets abd and bcd are infrequent.

The remaining 4 frequent nodes (a, b, c, and d) cannot be maximal frequent because they all have at least 1 immediate superset that is frequent.

Advantage: Maximal frequent itemsets provide a compact representation of all the frequent itemsets for a particular dataset.

Disadvantage: The support count of maximal frequent itemsets does not provide any information about the support count of their subsets.

Closed Frequent Itemsets A frequent itemset is closed, when no (immediate) superset has the same support. If an itemset is not closed, we can look at the next larger closed itemset instead – support does not change. we can recover all frequent itemsets and their support from the closed frequent itemsets.

If customers who purchase computers also tend to buy antivirus software at the same time, then placing the hardware display close to the software display may help increase the sales of both items.

In an alternative strategy, placing hardware and software at opposite ends of the store may entice customers who purchase such items to pick up other items along the way. For instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading toward the software display to purchase antivirus software and may decide to purchase a home security system as well. Market basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers as well as computers.

Frequent Pattern Mining:

Frequent pattern mining can be classified in various ways, based on the following criteria:

1. Based on the completeness of patterns to be mined:

- We can mine the complete set of frequent item sets, the closed frequent item sets, and the maximal frequent item sets, given a minimum support threshold.
- We can also mine constrained frequent item sets, approximate frequent item sets, near-match frequent item sets, top-k frequent item sets and soon.

UNIT-IV

Classification and Prediction:

- Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Classification predicts categorical (discrete, unordered) labels, prediction models continuous valued functions.
- For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures of potential customers on computer equipment given their income and occupation.
- A predictor is constructed that predicts a continuous-valued function, or ordered value, as opposed to a categorical label.
- Regression analysis is a statistical methodology that is most often used for numeric prediction. Many classification and prediction methods have been proposed by researchers in machine learning, pattern recognition, and statistics.
- Most algorithms are memory resident, typically assuming a small data size. Recent data mining research has built on such work, developing scalable classification and prediction techniques capable of handling large disk-resident data.

Classification General Approaches:

1. Preparing the Data for Classification and Prediction:

The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

(i) Data cleaning:

This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques) and the treatment of missing values (e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics).

Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.

(ii) Relevance analysis:

- Many of the attributes in the data may be redundant.
- Correlation analysis can be used to identify whether any two given attributes are statistically related.
- For example, a strong correlation between attributes A1 and A2 would suggest that one of the two could be removed from further analysis.
- A database may also contain irrelevant attributes. Attribute subset selection can be used in these cases to find a reduced set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.
- Hence, relevance analysis, in the form of correlation analysis and attribute subset selection, can be used to detect attributes that do not contribute to the classification or prediction task. Such analysis can help improve classification efficiency and scalability.

(iii) Data Transformation and Reduction

- The data may be transformed by normalization, particularly when neural networks or methods involving distance measurements are used in the learning step.
- Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as -1 to +1 or 0 to 1.
- The data can also be transformed by generalizing it to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous valued attributes.
- For example, numeric values for the attribute income can be generalized to discrete ranges, such as low, medium, and high. Similarly, categorical attributes, like street, can be generalized to higher-level concepts, like city.
- Data can also be reduced by applying many other methods, ranging from wavelet transformation and principle components analysis to discritization techniques, such as binning, histogram analysis, and clustering.

Comparing Classification and Prediction Methods:

➢ Accuracy:

The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data (i.e., tuples without class label information).

The accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.

➢ **Speed:**

This refers to the computational costs involved in generating and using the given classifier or predictor.

➢ **Robustness:**

This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.

➢ **Scalability:**

This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

➢ **Interpretability:**

This refers to the level of understanding and insight that is provided by the classifier or predictor.
Interpretability is subjective and therefore more difficult to assess.

Decision Tree Algorithm:

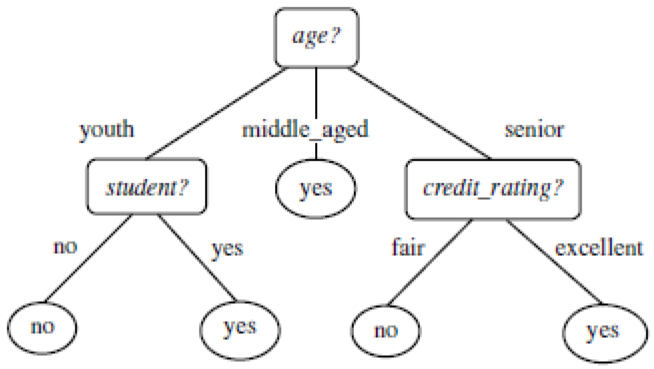
Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, were

➢ Each internal node denotes a test on an attribute.

➢ Each branch represents an outcome of the test.

➢ Each leaf node holds a class label.

➢ The topmost node in a tree is the root node.



- The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore I appropriate for exploratory knowledge discovery.
- Decision trees can handle high dimensional 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. In general, decision tree classifiers have good accuracy.
- 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.

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 D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)     if D_j is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(D_j, attribute_list) to node N;
     endfor
(15) return N;
```

Algorithm For Decision Tree Induction:

The algorithm is called with three parameters:

➢Data partition

➢Attribute list

➢Attribute selection method

- The parameter attribute list is a list of attributes describing the tuples.
- Attribute selection method specifies a heuristic procedure for selecting the attribute that best discriminates the given tuples according to class.
- The tree starts as a single node, N, representing the training tuples in D.
- If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class.
- All of the terminating conditions are explained at the end of the algorithm. Otherwise, the algorithm calls Attribute selection method to determine the splitting criterion.
- The splitting criterion tells us which attribute to test at node N by determining the―best‖ way to separate or partition the tuples in D into individual classes.
- There are three possible scenarios. Let A be the splitting attribute. A has v distinct values, {a1, a2, …, av}, based on the training data.

1 A is discrete-valued:

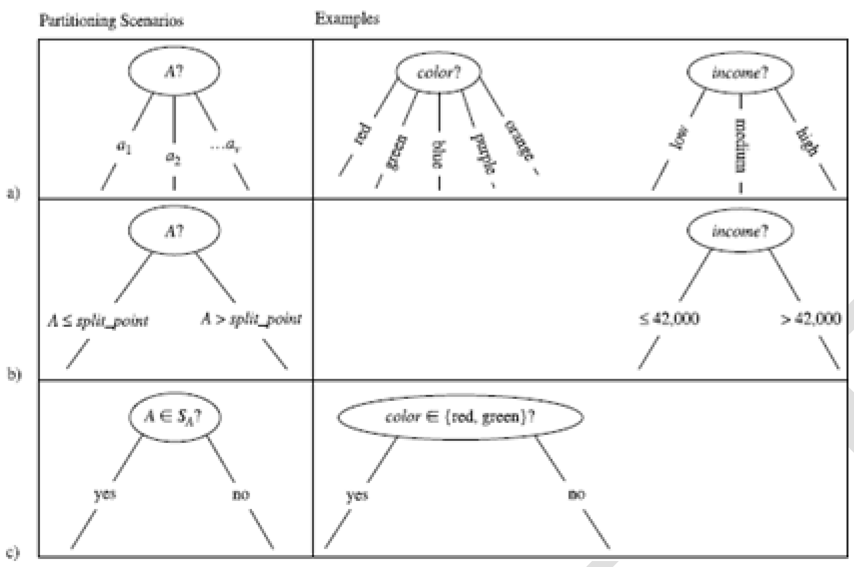
In this case, the outcomes of the test at node N correspond directly to the known values of A. A branch is created for each known value, aj, of A and labeled with that value. A need not be considered in any future partitioning of the tuples.

2 A is continuous-valued:

In this case, the test at node N has two possible outcomes, corresponding to the conditions A <=split point and A >split point, respectively where split point is the split-point returned by Attribute selection method as part of the splitting criterion.

3 A is discrete-valued and a binary tree must be produced:

The test at node N is of the form―A€SA? ‖. SA is the splitting subset for A, returned by Attribute selection method as part of the splitting criterion. It is a subset of the known values of A.



(a) If A is Discrete valued (b)If A is continuous valued (c) If A is discrete-valued and a binary tree must be produced:

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 hypothesisH holds giventhe ―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,$P(H|\boldsymbol{X})$, from $P(H)$, $P(\boldsymbol{X}|H)$, and $P(\boldsymbol{X})$.

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

Naïve Bayesian Classifier:

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

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

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

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

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

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

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

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

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

We can easily estimate the probabilities P(x1|Ci), P(x2|Ci), : : : , P(xn|Ci) from the training tuples. For each attribute, we look at whether the attribute is categorical or continuous-valued.

For instance, to compute P(X|Ci), we consider the following:

➢If Akis categorical, then P(xk|Ci) is the number of tuples of class Ci in D having the value for Ak, divided by |Ci,D| the number of tuples of class Ciin D.

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

A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation , defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

5. In order to predict the class label of X, P(XjCi)P(Ci) is evaluated for each class Ci. The classifier predicts that the class label of tuple X is the class Ci if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \le j \le m, j \ne i.$$

Bayesian Belief Network:

Bayesian Belief Networks specify joint conditional probability distributions. They are also known as Belief Networks, Bayesian Networks, or Probabilistic Networks.

•A Belief Network allows class conditional independencies to be defined between subsets of variables.

• It provides a graphical model of causal relationship on which learning can be performed.

• We can use a trained Bayesian Network for classification.

There are two components that define a Bayesian Belief Network −

• Directed acyclic graph

• A set of conditional probability tables

Directed Acyclic Graph

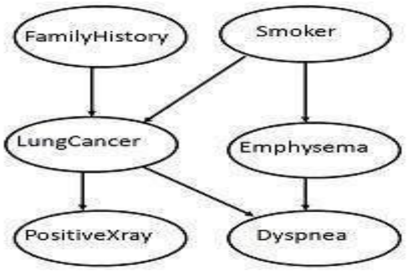
• Each node in a directed acyclic graph represents a random variable.

• These variables may be discrete or continuous valued.

• These variables may correspond to the actual attribute given in the data.

Directed Acyclic Graph Representation

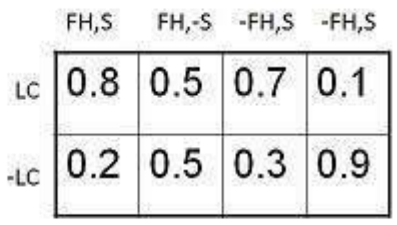
The following diagram shows a directed acyclic graph for six Boolean variables.



The arc in the diagram allows representation of causal knowledge. For example, lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker. It is worth noting that the variable Positive Xray is independent of whether the patient has a family history of lung cancer or that the patient is a smoker, given that we know the patient has lung cancer.

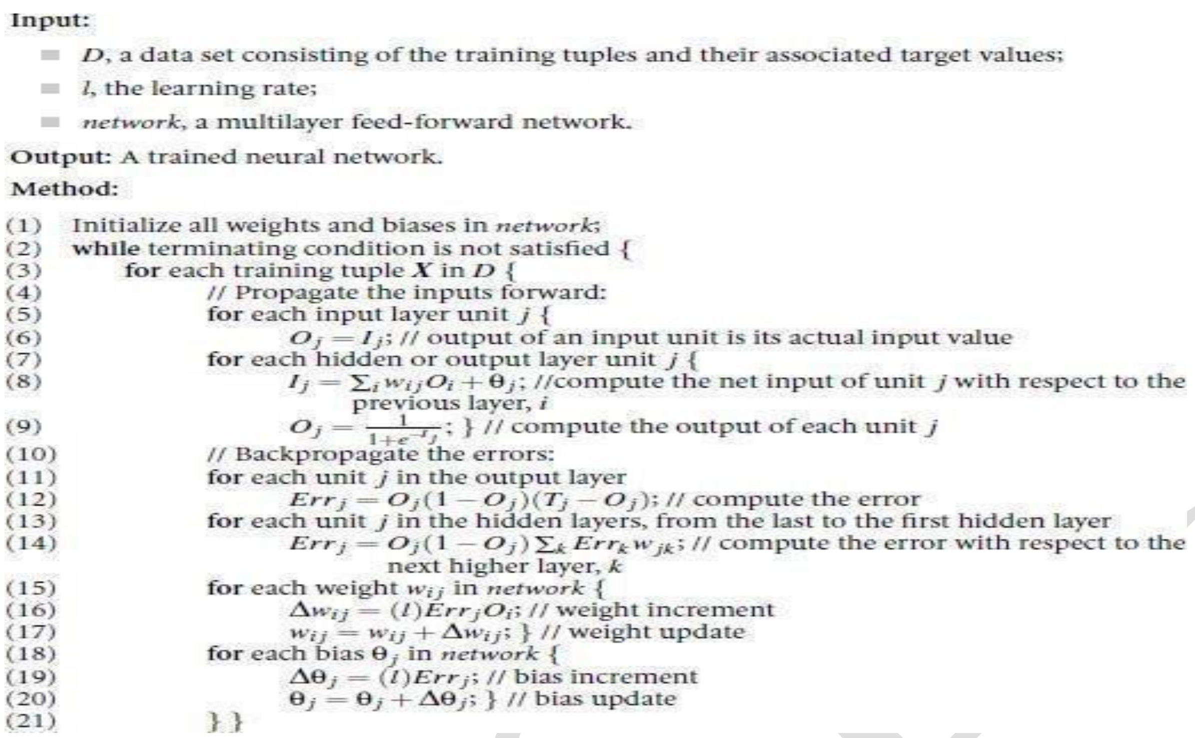
Conditional Probability Table

The conditional probability table for the values of the variable Lung Cancer (LC) showing each possible combination of the values of its parent nodes, Family History (FH), and Smoker (S) is as follows –



	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

K Nearest Neighbor Classification-Algorithm and Characteristics.



Input:

- D, a data set consisting of the training tuples and their associated target values;
- l, the learning rate;
- *network*, a multilayer feed-forward network.

Output: A trained neural network.

Method:

```
(1)  Initialize all weights and biases in network;
(2)  while terminating condition is not satisfied {
(3)      for each training tuple X in D {
(4)          // Propagate the inputs forward:
(5)          for each input layer unit j {
(6)              $O_j = I_j$; // output of an input unit is its actual input value
(7)          for each hidden or output layer unit j {
(8)              $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to the
                     previous layer, i
(9)              $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit j
(10)         // Backpropagate the errors:
(11)         for each unit j in the output layer
(12)             $Err_j = O_j(1-O_j)(T_j - O_j)$; // compute the error
(13)         for each unit j in the hidden layers, from the last to the first hidden layer
(14)             $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to the
                     next higher layer, k
(15)         for each weight $w_{ij}$ in network {
(16)             $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(17)             $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(18)         for each bias $\theta_j$ in network {
(19)             $\Delta \theta_j = (l) Err_j$; // bias increment
(20)             $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
(21)         } }
```

- Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.
- The training tuples are described by n attributes. Each tuple represents a point in an n-dimensional space. In this way, all of the training tuples are stored in an n-dimensional pattern space. When given an unknown tuple, a k-nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k nearest neighbors of the unknown tuple.
- Closeness is defined in terms of a distance metric, such as Euclidean distance.
- The Euclidean distance between two points or tuples, say, X1 = (x11, x12, … , x1n) and X2 = (x21, x22, … ,x2n), is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count.

Min-Max normalization can be used to transform a value v of a numeric attribute A to v0 in the range [0, 1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

Where minAand maxAare the minimum and maximum values of attribute A

- For k-nearest-neighbor classification, the unknown tuple is assigned the mostcommon class among its k nearest neighbors.
- When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space.
- Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple.
- In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple.

Characteristics of kNN

• Between-sample geometric distance.

• Classification decision rule and confusion matrix.

• Feature transformation.

• Performance assessment with cross-validation.

UNIT-V

Cluster Analysis:

- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.
- A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression.
- Cluster analysis tools based on k-means, k-medoids, and several methods have also been built into many statistical analysis software packages or systems, such as S-Plus, SPSS, and SAS.

Applications:

- Cluster analysis has been widely used in numerous applications, including market research, pattern recognition, data analysis, and image processing.
- In business, clustering can help marketers discover distinct groups in their customer bases and characterize customer groups based on purchasing patterns.
- In biology, it can be used to derive plant and animal taxonomies, categorize genes with similar functionality, and gain insight into structures inherent in populations.
- Clustering may also help in the identification of areas of similar land use in an earth observation database and in the identification of groups of houses in a city according to house type, value, and geographic location, as well as the identification of groups of automobile insurance policy holders with a high average claim cost.
- Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity.
- Clustering can also be used for outlier detection, Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce.

Typical Requirements of Clustering In Data Mining:

➢ **Scalability**:

Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions of objects. Clustering on a sample of a given large data set may lead to biased results. Highly scalable clustering algorithms are needed.

➢**Ability to deal with different types of attributes:**

Many algorithms are designed to cluster interval-based (numerical) data. However, applications may require clustering other types of data, such as binary, categorical (nominal), and ordinal data, or mixtures of these data types.

➢ **Discovery of clusters with arbitrary shape:**

Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. Algorithms based on such distance measures tend to find spherical clusters with similar size and density.

However, a cluster could be of any shape. It is important to develop algorithms that can detect clusters of arbitrary shape.

➢ **Minimal requirements for domain knowledge to determine input parameters:** Many clustering algorithms require users to input certain parameters in cluster analysis (such as the number of desired clusters). The clustering results can be quite sensitive to input parameters. Parameters are often difficult to determine, especially for data sets
containing high-dimensional objects. This not only burdens users, but it also makes the quality of clustering difficult to control.

➢ **Ability to deal with noisy data:**

Most real-world databases contain outliers or missing, unknown, or erroneous data. Some clustering algorithms are sensitive to such data and may lead to clusters of poor quality.

➢ **Incremental clustering and insensitivity to the order of input records:**

Some clustering algorithms cannot incorporate newly inserted data (i.e., database updates) into existing clustering structures and, instead, must determine a new clustering from scratch. Some clustering algorithms are sensitive to the order of input data. That is, given a set of data objects, such an algorithm may return dramatically different clustering depending on the order of presentation of the input objects. It is important to develop incremental clustering algorithms and algorithms that are insensitive to the order of input.

➢ **High dimensionality:**

A database or a data warehouse can contain several dimensions or attributes. Many clustering algorithms are good at handling low-dimensional data, involving only two to three dimensions. Human eyes are good at judging the quality of clustering for up to three dimensions. Finding clusters of data objects in high dimensional space is challenging, especially considering that such data can be sparse and highly skewed.

➢ **Constraint-based clustering:**

Real-world applications may need to perform clustering under various kinds of constraints. Suppose that your job is to choose the locations for a given number of new automatic banking machines (ATMs) in a city. To decide upon this, you may cluster households while considering constraints such as the city's rivers and highway networks, and the type and number of customers per cluster. A challenging task is to find groups of data with good clustering behavior that satisfy specified constraints.

➢**Interpretability and usability:**

Users expect clustering results to be interpretable, comprehensible, and usable. That is, clustering may need to be tied to specific semantic interpretations and applications. It is important to study how an application goal may influence the selection of clustering features and methods.

A Categorization of Major Clustering Methods:

➢ Partitioning Methods

➢ Hierarchical Methods

➢ Density-Based Methods

➢ Grid-Based Methods

➢ Model-Based Methods

Partitioning Methods:

A partitioning method constructs k partitions of the data, where each partition represents a cluster and k <= n. That is, it classifies the data into k groups, which together satisfy the following requirements:

Each group must contain at least one object, and Each object must belong to exactly one group.

A partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are close or related to each other, whereas objects of different clusters are far apart or very different.

Hierarchical Methods:

A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed.

❖ The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one or until a termination condition holds.

❖ The divisive approach, also called the top-down approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone. This rigidity is useful in that it leads to smaller computation costs by not having to worry about a combinatorial number of different choices.

There are two approaches to improving the quality of hierarchical clustering:

❖ Perform careful analysis of object ―linkages‖ at each hierarchical partitioning, such as in Chameleon, or Integrate hierarchical agglomeration and other approaches by first using a hierarchical agglomerative algorithm to group objects into micro clusters, and then performing macro clustering on the micro clusters using another clustering method such as iterative relocation.

Density-based methods:

❖ Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.

❖ Other clustering methods have been developed based on the notion of density. Their general idea is to continue growing the given cluster as long as the density in the neighborhood exceeds some threshold; that is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers)and discover clusters of arbitrary shape.

❖ DBSCAN and its extension, OPTICS, are typical density-based methods that grow clusters according to a density-based connectivity analysis. DENCLUE is a method that clusters objects based on the analysis of the value distributions of density functions.

Grid-Based Methods:

❖ Grid-based methods quantize the object space into a finite number of cells that form a grid structure.

❖ All of the clustering operations are performed on the grid structure i.e., on the quantized space. The main advantage of this approach is its fast-processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

❖ STING is a typical example of a grid-based method. Wave Cluster applies wavelet transformation for clustering analysis and is both grid-based and density-based.

Model-Based Methods:

❖ Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model.

❖ A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points.

❖ It also leads to a way of automatically determining the number of clusters based on standard statistics, taking ―noise‖ or outliers into account and thus yielding robust clustering methods.

Tasks in Data Mining:

➢ Clustering High-Dimensional Data

➢ Constraint-Based Clustering

Clustering High-Dimensional Data:

- It is a particularly important task in cluster analysis because many applications require the analysis of objects containing a large number of features or dimensions. For example, text documents may contain thousands of terms or keywords as features, and

DNA micro array data may provide information on the expression levels of thousands of genes under hundreds of conditions.

- Clustering high-dimensional data is challenging due to the curse of dimensionality. Many dimensions may not be relevant. As the number of dimensions increases, the data become increasingly sparse so that the distance measurement between pairs of points become meaningless and the average density of points anywhere in the data is likely to be low. Therefore, a different clustering methodology needs to be
- developed for high-dimensional data.
- CLIQUE and PROCLUS are two influential subspace clustering methods, which search for clusters in subspaces of the data, rather than over the entire data space.
- Frequent pattern–based clustering, another clustering methodology, extracts distinct frequent patterns among subsets of dimensions that occur frequently. It uses such patterns to group objects and generate meaningful clusters.

Constraint-Based Clustering:

- It is a clustering approach that performs clustering by incorporation of user-specified or application-oriented constraints.
- A constraint expresses a user's expectation or describes properties of the desired clustering results, and provides an effective means for communicating with the clustering process.
- Various kinds of constraints can be specified, either by a user or as per application requirements.
- Spatial clustering employs with the existence of obstacles and clustering under user-specified constraints. In addition, semi-supervised clustering employs for pair wise constraints in order to improve the quality of the resulting clustering.

Classical Partitioning Methods:

The most well-known and commonly used partitioning methods are

❖ The k-Means Method

❖ k-Medoids Method

Partitioning Clustering: The K-Means Method:

The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra cluster similarity is high but the inter cluster similarity is low. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

The k-means algorithm proceeds as follows.

First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean.

It then computes the new mean for each cluster. This process iterates until the criterion function converges.

Typically, the square-error criterion is used, defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2,$$

Where E is the sum of the square error for all objects in the data set p is the point in space representing a given object

Mi is the mean of cluster Ci

The k-means partitioning algorithm:

Input:

- k: the number of clusters,
- D: a data set containing n objects.

Output: A set of k clusters.

Method:

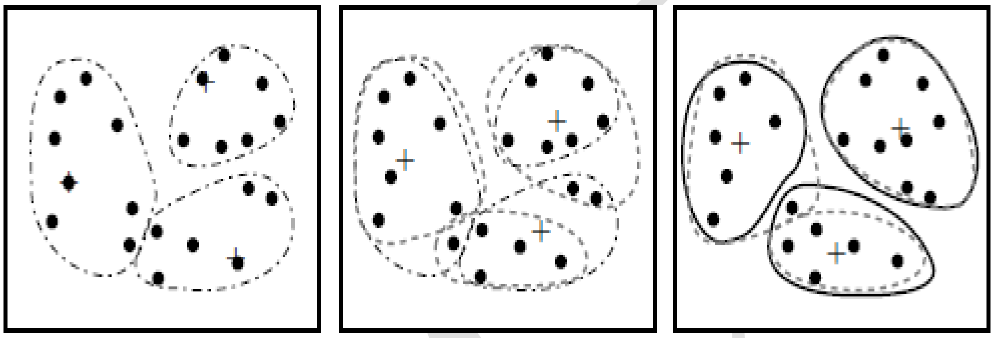
```
(1) arbitrarily choose $k$ objects from $D$ as the initial cluster centers;
(2) repeat
(3)     (re)assign each object to the cluster to which the object is the most similar,
            based on the mean value of the objects in the cluster;
(4)     update the cluster means, i.e., calculate the mean value of the objects for
            each cluster;
(5) until no change;
```

Clustering of a set of objects based on the *k*-means method

K-Medoids (also called Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as a point in the cluster, whose dissimilarities with all the other points in the cluster are minimum. The dissimilarity of the medoid(Ci) and object(Pi) is calculated by using E = |Pi – Ci|.

1. Initialize: select k random points out of the n data points as the medoids.

2. Associate each data point to the closest medoid by using any common distance metric methods.



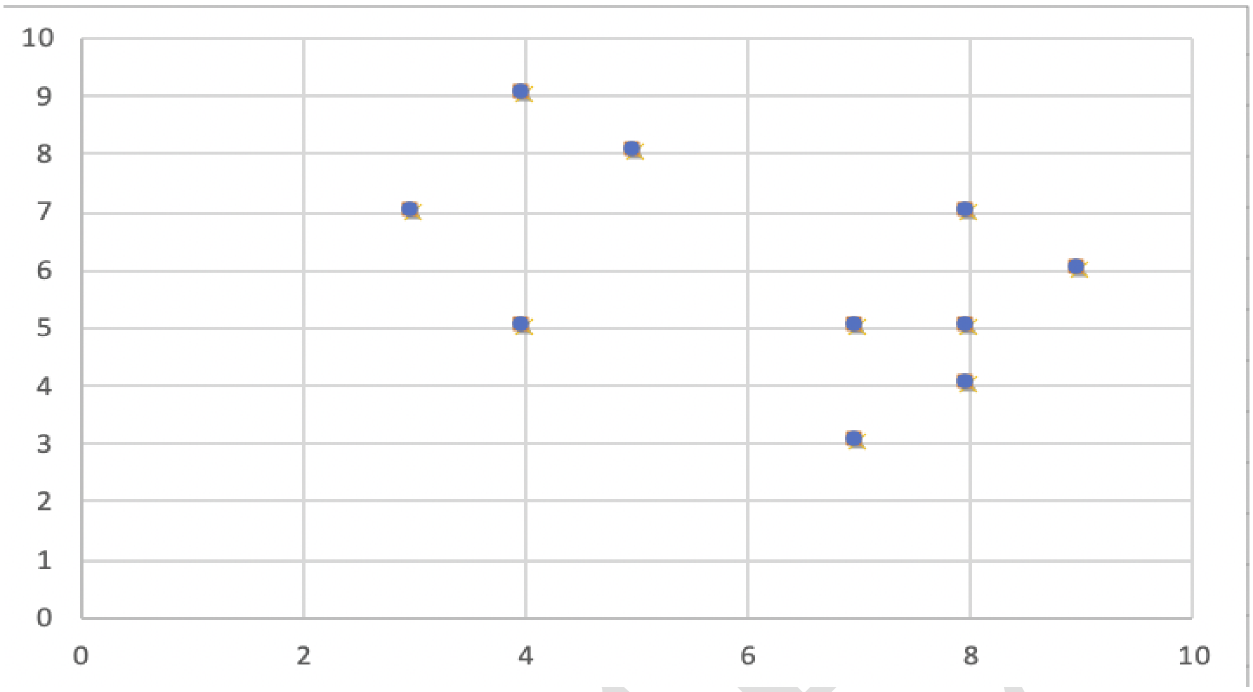
3. While the cost decreases: For each medoid m, for each data o point which is not a medoid:

• Swap m and o, associate each data point to the closest medoid, and recompute the cost.

•If the total cost is more than that in the previous step, undo the swap.

	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5

Let’s consider the following example: If a graph is drawn using the above data points, we obtain the following:



Step 1: Let the randomly selected 2 medoids, so select k = 2, and let C1 -(4, 5) and C2 -(8, 5) are the two medoids.

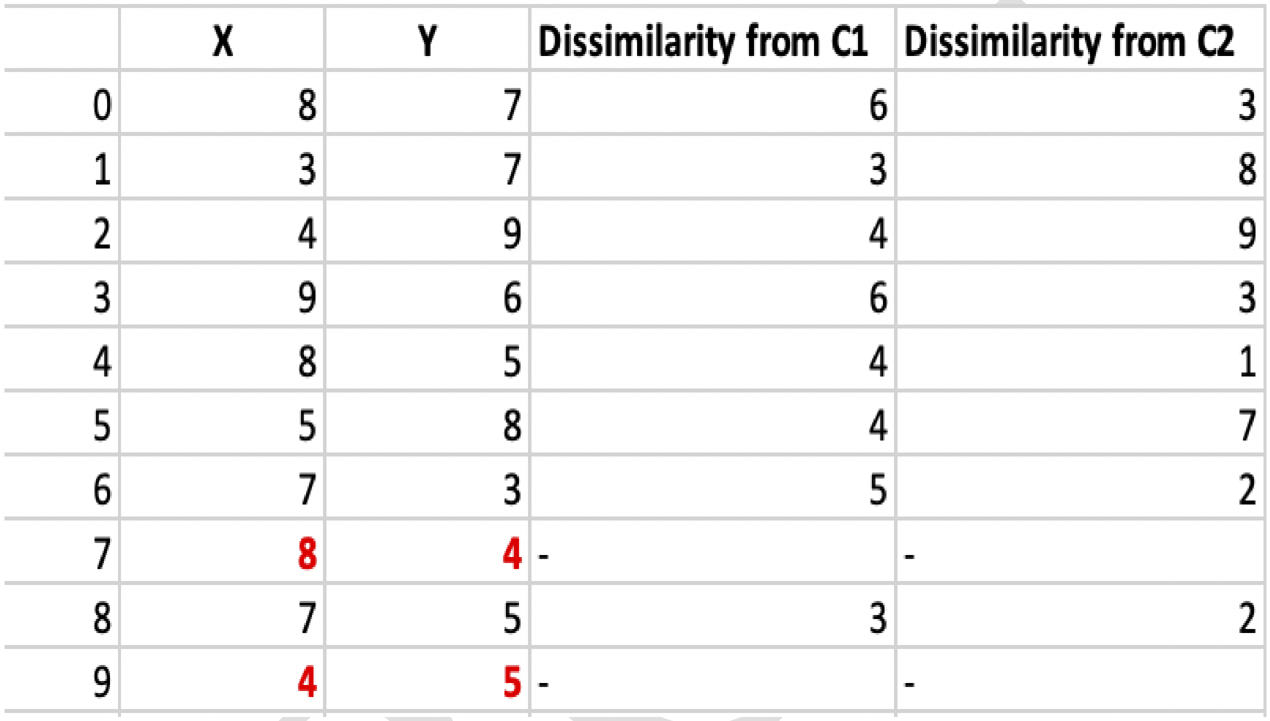
Step 2: Calculating cost. The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:

	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	2
1	3	7	3	7
2	4	9	4	8
3	9	6	6	2
4	8	5	-	-
5	5	8	4	6
6	7	3	5	3
7	8	4	5	1
8	7	5	3	1
9	4	5	-	-

Here we have used Manhattan distance formula to calculate the distance matrices between medoid and non-medoid points. That formula tell that Distance = |X1-X2| + |Y1-Y2|.

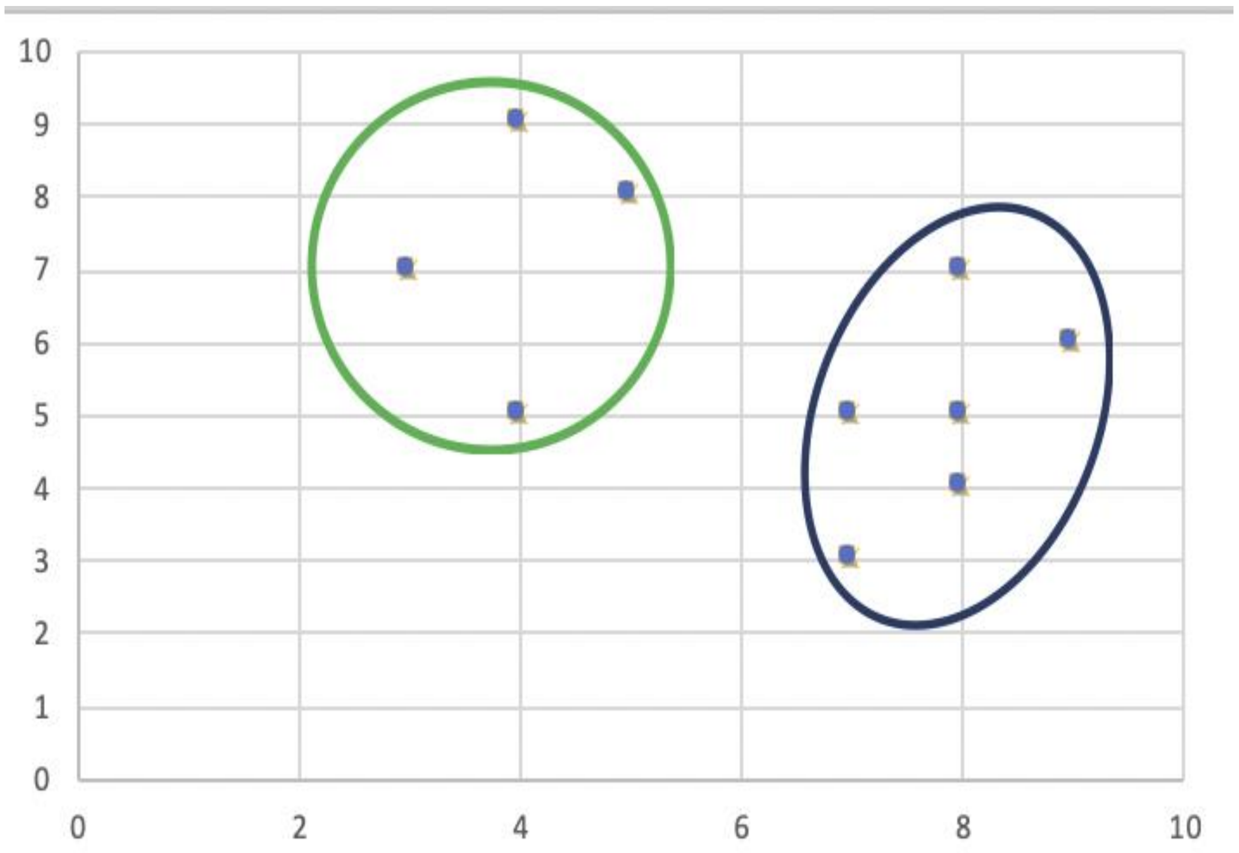
Each point is assigned to the cluster of that medoid whose dissimilarity is less. Points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The Cost = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20

Step 3: randomly select one non-medoid point and recalculate the cost. Let the randomly selected point be (8, 4). The dissimilarity of each non-medoid point with the medoids – C1 (4, 5) and C2 (8, 4) is calculated and tabulated.



	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
7	8	4	-	-
8	7	5	3	2
9	4	5	-	-

Each point is assigned to that cluster whose dissimilarity is less. So, points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The New cost = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 22 Swap Cost = New Cost – Previous Cost = 22 – 20 and **2 >0** As the swap cost is not less than zero, we undo the swap. Hence (4, 5) and (8, 5) are the final medoids. The clustering would be in the following way The **time complexity** is



Advantages:

1. It is simple to understand and easy to implement.

2. K-Medoid Algorithm is fast and converges in a fixed number of steps.

3. PAM is less sensitive to outliers than other partitioning algorithms.

Disadvantages:

1. The main disadvantage of K-Medoid algorithms is that it is not suitable for clustering non-spherical (arbitrarily shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster center) – briefly, it uses compactness as clustering criteria instead of connectivity.

2. It may obtain different results for different runs on the same dataset because the first k medoids are chosen randomly.

Hierarchical Clustering Methods:

- A hierarchical clustering method works by grouping data objects into a tree of clusters.
- The quality of a pure hierarchical clustering method suffers from its inability to

perform adjustment once a merge or split decision has been executed. That is, if a particular merge or split decision later turns out to have been a poor choice, the method cannot backtrack and correct it.

- Hierarchical clustering methods can be further classified as either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up or top-down fashion.

Agglomerative hierarchical clustering:

- This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.
- Most hierarchical clustering methods belong to this category. They differ only in their definition of inter cluster similarity.

Divisive hierarchical clustering:

- This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster.
- It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold.

Advantages of Hierarchical Clustering

Hierarchical clustering is a widely used technique in data analysis, which involves the grouping of objects into clusters based on their similarity. This method of clustering is advantageous in a variety of ways and can be used to solve various types of problems. Here are 10 advantages of hierarchical clustering:

1. Robustness: Hierarchical clustering is more robust than other methods since it does not require a predetermined number of clusters to be specified. Instead, it creates hierarchical clusters based on the similarity between the objects, which makes it more reliable and accurate.

2. Easy to interpret: Hierarchical clustering produces a tree-like structure that is easy to interpret and understand. This makes it ideal for data analysis as it can provide insights into the data without requiring complex algorithms or deep learning models.

3. Flexible: Hierarchical clustering is a flexible method that can be used on any type

of data. It can also be used with different types of similarity functions and distance measures, allowing for customization based on the application at hand.

Disadvantages of Hierarchical Clustering

While hierarchical clustering is a powerful tool for discovering patterns and relationships in data sets, it also has its drawbacks. Here are 10 disadvantages of hierarchical clustering:

1. It is sensitive to outliers. Outliers have a significant influence on the clusters that are formed, and can even cause incorrect results if the data set contains these types of data points.
2. Hierarchical clustering is computationally expensive. The time required to run the algorithm increases exponentially as the number of data points increases, making it difficult to use for large datasets.
3. The results of Agglomerative or divisive clustering can sometimes be difficult to interpret the results due to its complexity. The dendrogram representation of the clusters can be hard to understand and visualize, making it difficult to draw meaningful conclusions from the results.
4. It does not guarantee optimal results or the best possible clusterings. Since it is an unsupervised learning algorithm, it relies on the researcher’s judgment and experience to assess the quality of the results.
5. Hierarchical clustering methods require a predetermined number of clusters before they can begin clustering, which may not be known beforehand. This makes it difficult to use in certain applications where this information is not available.

Outlier Analysis:

- There exist data objects that do not comply with the general behavior or model of the data. Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called outliers.
- Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information because one person’s noise could be another person’s signal. In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity. Thus, outlier detection and analysis is

an interesting data mining task, referred to as outlier mining.

- It can be used in fraud detection, for example, by detecting unusual usage of credit cards or telecommunication services. In addition, it is useful in customized marketing for identifying the spending behavior of customers with extremely low or extremely high incomes, or in medical analysis for finding unusual responses to various medical treatments.
- Outlier mining can be described as follows: Given a set of *n* data points or objects and *k*, the expected number of outliers, find the top k objects that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data. The outlier mining problem can be viewed as two sub problems

- Define what data can be considered as inconsistent in a given data set, and Find an efficient method to mine the outliers so defined.

Types of outlier detection:

- Statistical Distribution-Based Outlier Detection
- Distance-Based Outlier Detection
- Density-Based Local Outlier Detection
- Deviation-Based Outlier Detection

Statistical Distribution-Based Outlier Detection:

The statistical distribution-based approach to outlier detection assumes a distribution or probability model for the given data set (e.g., a normal or Poisson distribution) and then identifies outliers with respect to the model using a discordancy test. Application of the test requires knowledge of the data set parameters knowledge of distribution parameters such as the mean and variance and the expected number of outliers.

A statistical discordancy test examines two hypotheses: A working hypothesis

An alternative hypothesis

A working hypothesis, H, is a statement that the entire data set of n objects comes from an initial distribution model, F, that is,

The hypothesis is retained if there is no statistically significant evidence supporting its rejection. A discordancy test verifies whether an object, oi, is significantly large (or small) in relation to the distribution F. Different test statistics have been proposed for use as a discordancy test, depending on the available knowledge of the data. Assuming that some statistic, T, has been chosen for discordancy testing, and the value of the statistic for object oi is vi, then the distribution of T is constructed. Significance probability, SP(vi)=Prob(T > vi), is evaluated. If SP(vi) is sufficiently small, then oi is discordant and the working hypothesis is rejected.

An alternative hypothesis, H, which states that oi comes from another distribution model, G, is adopted. The result is very much dependent on which model F is chosen because oimay be an outlier under one model and a perfectly valid value under another. The alternative distribution is very important in determining the power of the test, that is, the probability that the working hypothesis is rejected when oi is really an outlier. There are different kinds of alternative distributions.

Inherent alternative distribution:
In this case, the working hypothesis that all of the objects come from distribution F is rejected in favor of the alternative hypothesis that all of the objects arise from another distribution, G:
H :oi € G, where i = 1, 2,…, n
F and G may be different distributions or differ only in parameters of the same distribution.
There are constraints on the form of the G distribution in that it must have potential to produce outliers. For example, it may have a different mean or dispersion, or a longer tail.

Mixture alternative distribution:
The mixture alternative states that discordant values are not outliers in the F population, but contaminants from some other population,
G. In this case, the alternative hypothesis is

$$\overline{H} : o_i \in (1-\lambda)F + \lambda G, \quad \text{where } i = 1, 2, \ldots, n.$$

Slippage alternative distribution:
This alternative states that all of the objects (apart from some prescribed small number) arise independently from the initial model, F, with its given parameters, whereas the remaining objects are independent observations from a modified version of F in which the parameters have been shifted.
There are two basic types of procedures for detecting outliers:

Block procedures:

In this case, either all of the suspect objects are treated as outliers or all of them are accepted as consistent.

Consecutive procedures

An example of such a procedure is the inside out procedure. Its main idea is that the object that is least likely to be an outlier is tested first. If it is found to be an outlier, then all of the more extreme values are also considered outliers; otherwise, the next most extreme object is tested, and so on. This procedure tends to be more effective than block procedures.

Distance-Based Outlier Detection:

The notion of distance-based outliers was introduced to counter the main limitations imposed by statistical methods. An object, o, in a data set, D, is a distance-based (DB)outlier with parameters pct and dmin, that is, a DB(pct;dmin)-outlier, if at least a fraction, pct, of the objects in D lie at a distance greater than dmin from o.

In other words, rather that relying on statistical tests, we can think of distance-based outliers as those objects that do not have enough neighbors, where neighbors are defined based on distance from the given object.

In comparison with statistical-based methods, distance based outlier detection generalizes the ideas behind discordancy testing for various standard distributions. Distance-based outlier detection avoids the excessive computation that can be associated with fitting the observed distribution into some standard distribution and in selecting discordancy tests.

For many discordancy tests, it can be shown that if an object, o, is an outlier according to the given test, then o is also a DB(pct, dmin)-outlier for some suitably defined pct and dmin.

For example, if objects that lie three or more standard deviations from the mean are considered to be outliers, assuming a normal distribution, then this definition can be generalized by a DB(0.9988, 0.13s) outlier.

Several efficient algorithms for mining distance-based outliers have been developed.

Index-based algorithm:

Given a data set, the index-based algorithm uses multidimensional indexing structures, such as R-trees or k-d trees, to search for neighbors of each object o within radius dmin around that object.

Let M be the maximum number of objects within the dmin-neighborhood of an outlier. Therefore, onceM+1 neighbors of object o are found, it is clear that o is not an outlier.

This algorithm has a worst-case complexity of O(n2k), where n is the number of objects in the

data set and k is the dimensionality. The index-based algorithm scales well as k increases. However, this complexity evaluation takes only the search time into account, even though the task of building an index in itself can be computationally intensive.

Nested-loop algorithm:

The nested-loop algorithm has the same computational complexity as the index-based algorithm but avoids index structure construction and tries to minimize the number of I/Os. It divides the memory buffer space into two halves and the data set into several logical blocks. By carefully choosing the order in which blocks are loaded into each half, I/O efficiency can be achieved.

Cell-based algorithm:

To avoid O(n2) computational complexity, a cell-based algorithm was developed for memory- resident data sets. Its complexity is O(ck+n), where c is a constant depending on the number of cells and k is the dimensionality.

In this method, the data space is partitioned into cells with a side length equal to   Each cell has two layers surrounding it. The first layer is one cell thick, while the second is cells thick, rounded up to the closest integer.

The algorithm counts outliers on a cell-by-cell rather than an object-by-object basis. For a given cell, it accumulates three counts—the number of objects in the cell, in the cell and the first layer together, and in the cell and both layers together. Let's refer to these counts as cell count, cell + 1 layer count, and cell + 2 layers count, respectively.

Let M be the maximum number of outliers that can exist in the dmin-neighborhood of an outlier.

An object, o, in the current cell is considered an outlier only if cell + 1 layer count is less than or equal to M. If this condition does not hold, then all of the objects in the cell can be removed from further investigation as they cannot be outliers.

If cell_+ 2_layers_count is less than or equal to M, then all of the objects in the cell are considered outliers. Otherwise, if this number is more than M, then it is possible that some of the objects in the cell may be outliers. To detect these outliers, object-by-object processing is used where, for each object, o, in the cell, objects in the second layer of o are examined. For objects in the cell, only those objects having no more than M points in their dmin-neighborhoods are outliers. The dmin-neighborhood of an object consists of the object's cell, all of its first layer, and some of its second layer.

A variation to the algorithm is linear with respect to n and guarantees that no more than three passes over the data set are required. It can be used for large disk-resident data sets, yet does not scale well for high dimensions.

Density-Based Local Outlier Detection:

Statistical and distance-based outlier detection both depend on the overall or global distribution of the given set of data points, D. However, data are usually not uniformly distributed. These methods encounter difficulties when analyzing data with rather different density distributions.

To define the local outlier factor of an object, we need to introduce the concepts of k-distance, k-distance neighborhood, reachability distance,13 and local reachability density. These are defined as follows:

The k-distance of an object p is the maximal distance that p gets from its k- nearest neighbors. This distance is denoted as k-distance(p). It is defined as the distance, d(p, o), between p and an object o 2 D, such that for at least k objects, o0 2 D, it holds that d(p, o')_d(p, o). That is, there are at least k objects in D that are as close as or closer to p than o, and for at most k-1 objects, o00 2 D, it holds that d(p;o'') <d(p, o).

That is, there are at most k-1 objects that are closer to p than o. You may be wondering at this point how k is determined. The LOF method links to density-based clustering in that it sets k to the parameter rMinPts, which specifies the minimum number of points for use in identifying clusters based on density.

Here, MinPts (as k) is used to define the local neighborhood of an object, p.

The k-distance neighborhood of an object p is denoted Nkdistance(p)(p), or Nk(p)for short. By setting k to MinPts, we get NMinPts(p). It contains the MinPts- nearestneighbors of p. That is, it contains every object whose distance is not greater than the MinPts-distance of p.

The reachability distance of an object p with respect to object o (where o is within the MinPts-nearest neighbors of p), is defined as reach

distMinPts(p, o) = max{MinPts distance(o), d(p, o)}.

Intuitively, if an object p is far away , then the reachability distance between the two is simply their actual distance. However, if they are sufficiently close (i.e., where p is within the

MinPts-

distance neighborhood of o), then the actual distance is replaced by the MinPts- distance of o. This helps to significantly reduce the statistical fluctuations of d(p, o) for all of the p close to o.

The higher the value of MinPts is, the more similar is the reach ability distance for objects with in the same neighborhood.

Intuitively, the local reachability density of p is the inverse of the average reach ability density based on the Min Pts-nearest neighbors of p. It is defined as

$$lrd_{MinPts}(p) = \frac{|N_{MinPts}(p)|}{\sum_{o \in N_{MinPts}(p)} reach\_dist_{MinPts}(p, o)}.$$

The local outlier factor (LOF) of p captures the degree to which we call p an outlier. It is

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}.$$

defined as

It is the average of the ratio of the local reachability density of p and those of p's MinPts-nearest neighbors. It is easy to see that the lower p's local reachability density is, and the higher the local reachability density of p's MinPts-nearest neighbors are, the higher LOF(p) is.

Deviation-Based Outlier Detection:

Deviation-based outlier detection does not use statistical tests or distance-basedmeasures to identify exceptional objects. Instead, it identifies outliers by examining themain characteristics of objects in a group.Objects that ―deviate‖ fromthisdescription areconsidered outliers. Hence, in this approach the term deviations is typically used to referto outliers. In this section, we study two techniques for deviation-based outlier detection.The first sequentially compares objects in a set, while the second employs an OLAPdata cube approach.

Sequential Exception Technique:

The sequential exception technique simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects. It uses implicit redundancy of the data. Given a data set, D, of n objects, it builds a sequence of subsets,{D1, D2, …,Dm}, of these objects with 2<=m <= n such that

$$D_{j-1} \subset D_j, \quad \text{where } D_j \subseteq D.$$

Dissimilarities are assessed between subsets in the sequence. The technique introduces the following key terms.

Exception set:

This is the set of deviations or outliers. It is defined as the smallest subset of objects whose removal results in the greatest reduction of dissimilarity in the residual set.

Dissimilarity function:

This function does not require a metric distance between the objects. It is any function that, if given a set of objects, returns a low value if the objects are similar to one another. The greater the dissimilarity among the objects, the higher the value returned by the function. The dissimilarity of a subset is incrementally computed based on the subset prior to it in the sequence. Given a subset of n numbers, {x1, …,xn}, a possible dissimilarity function is the

$$\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2,$$

variance of the numbers in the set, that is,

where x is the mean of the n numbers in the set. For character strings, the dissimilarity function may be in the form of a pattern string (e.g., containing wild card characters that is used to cover all of the patterns seen so far. The dissimilarity increases when the pattern covering all of the strings in Dj-1 does not cover any string in Dj that is not in Dj-1.

Cardinality function:

This is typically the count of the number of objects in a given set.

Smoothing factor:

This function is computed for each subset in the sequence. It assesses how much the dissimilarity can be reduced by removing the subset from the original set of objects.