DIGITAL NOTES

ON

DATA WAREHOUSING AND DATA MINING
(R22A1205)

B.TECH III Year-II Sem
(2024-25)

Prepared by:
T.Shilpa

P.Swetha

Dr.K.Suresh

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(Post Via Hakimpet), Secunderabad–500100, Telangana State, India.

Contact Number: 040-23792146/64634237, E-Mail ID: mrcet2004@gmail.com, website: www.mrcet.ac.in

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

SYLLABUS

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

(R22A1205) 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 Kaufmann Publishers, Elsevier, 2 Edition, 2006.
2) Introduction to Data Mining, Psng-Ning Tan, Vipin Kumar, Michael Steinbanch, Pearson Educatior.

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 makingprocess.

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 modeling 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-upapproach.

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.

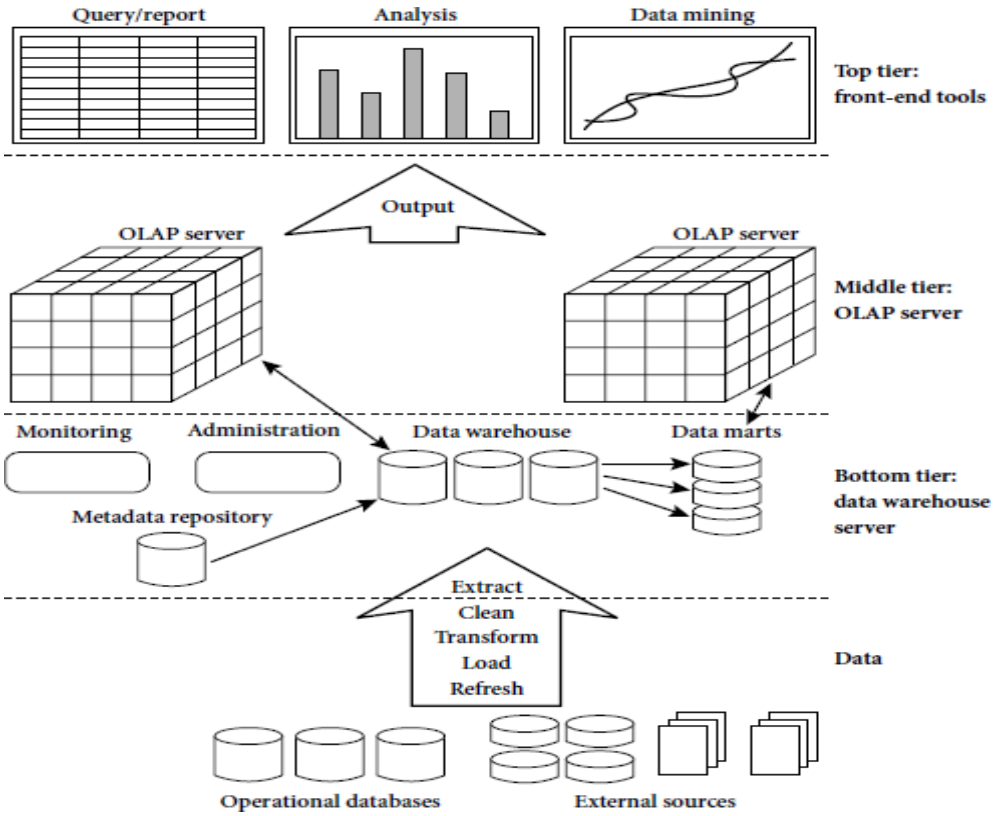
DIFFERENCES BETWEEN OLAP AND OLTP

Category	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Definition	It is well-known as an online database query management system.	It is well-known as an online database modifying system.
Data source	Consists of historical data from various Databases.	Consists of only operational current data.
Method used	It makes use of a data warehouse.	It makes use of a standard DBMS
Application	It is subject-oriented. Used for Data Mining Analytics, Decisions making, etc.	It is application-oriented. Used for business tasks.
Normalized	In an OLAP database, tables are not normalized.	In an OLTP database, tables are normalized(3NF)

Category	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Usage of data	The data is used in planning, problem-solving, and decision-making.	The data is used to perform day-to-day fundamental operations.
Task	It provides a multi-dimensional view of different business tasks.	It reveals a snapshot of present business tasks.
Purpose	It serves the purpose to extract information for analysis and decision-making.	It serves the purpose to Insert, Update, and Delete information from the database.
Volume of data	A large amount of data is stored typically in TB, PB	The size of the data is relatively small as the historical data is archived in MB, and GB.
Queries	Relatively slow as the amount of data involved is large. Queries may take hours.	Very Fast as the queries operate on 5% of the data.
Update	OLAP database is not often updated. As a result, data integrity is unaffected.	The data integrity constraint must be maintained in an OLTP database.
Backup and Recovery	It only needs backup from time to time as compared to OLTP.	The backup and recovery process is maintained rigorously
Processing time	The processing of complex queries can take a lengthy time.	It is comparatively fast in processing because of simple and straightforward queries.
Types of users	This data is generally managed by CEO, MD, and GM.	This data is managed by clerksForex and

Category	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
		managers.
Operations	Only read and rarely write operations.	Both read and write operations.
Updates	With lengthy, scheduled batch operations, data is refreshed on a regular basis.	The user initiates data updates, which are brief and quick.
Nature of audience	The process is focused on the customer.	The process is focused on the market.
Database Design	Design with a focus on the subject.	Design that is focused on the application.

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 multidimensionaldata 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).

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.

EXTRACTION,TRANFORMATION, LOADING

1. ETL stands for Extract, Transform, Load and it is a process used in data warehousing to extract data from various sources, transform it into a format suitable for loading into a data warehouse, and then load it into the warehouse. The process of ETL can be broken down into the following three stages:
2. Extract: The first stage in the ETL process is to extract data from various sources such as transactional systems, spreadsheets, and flat files. This step involves reading data from the source systems and storing it in a staging area.
3. Transform: In this stage, the extracted data is transformed into a format that is suitable for loading into the data warehouse. This may involve cleaning and validating the data, converting data types, combining data from multiple sources, and creating new data fields.

4. Load: After the data is transformed, it is loaded into the data warehouse. This step involves creating the physical data structures and loading the data into the warehouse.

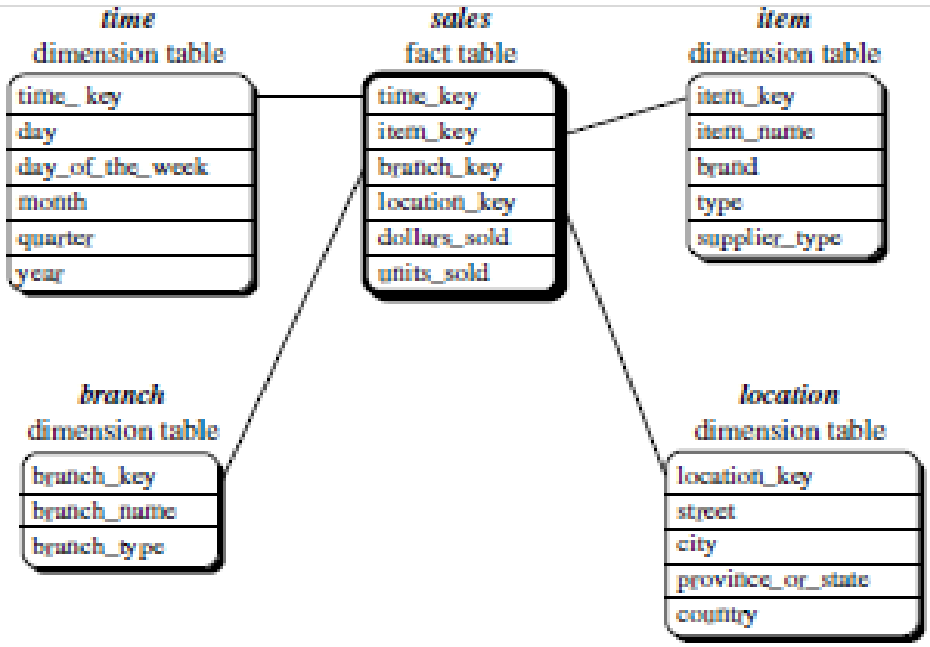
Schema Design:

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).

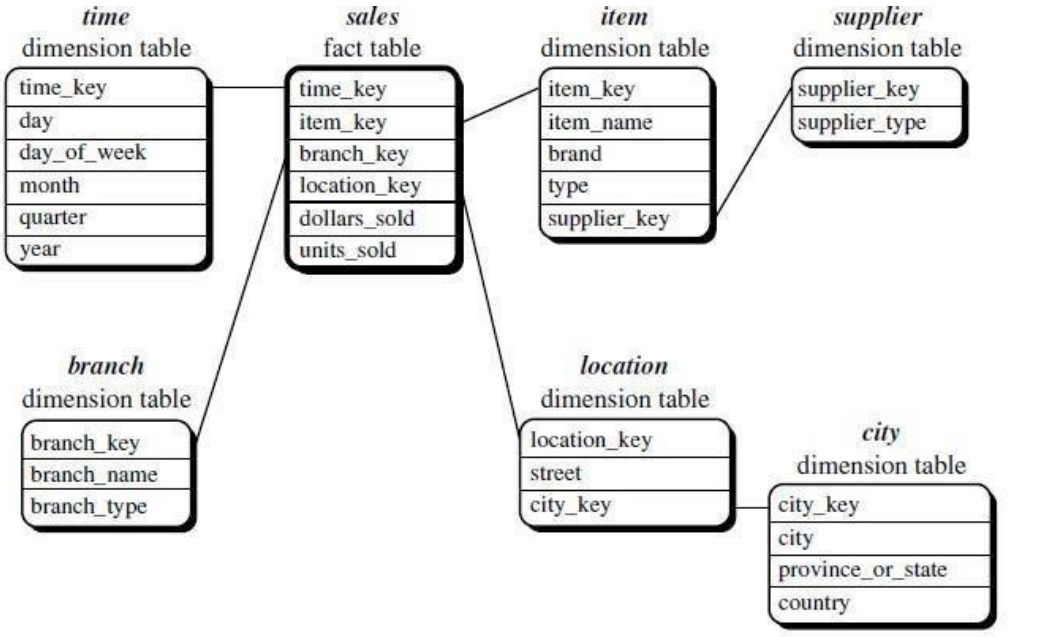


Star schema of a data warehouse for sales.

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. The city key in the new location table links to the city dimension. Notice that further normalization can be performed on province or state and country in the snowflake schema



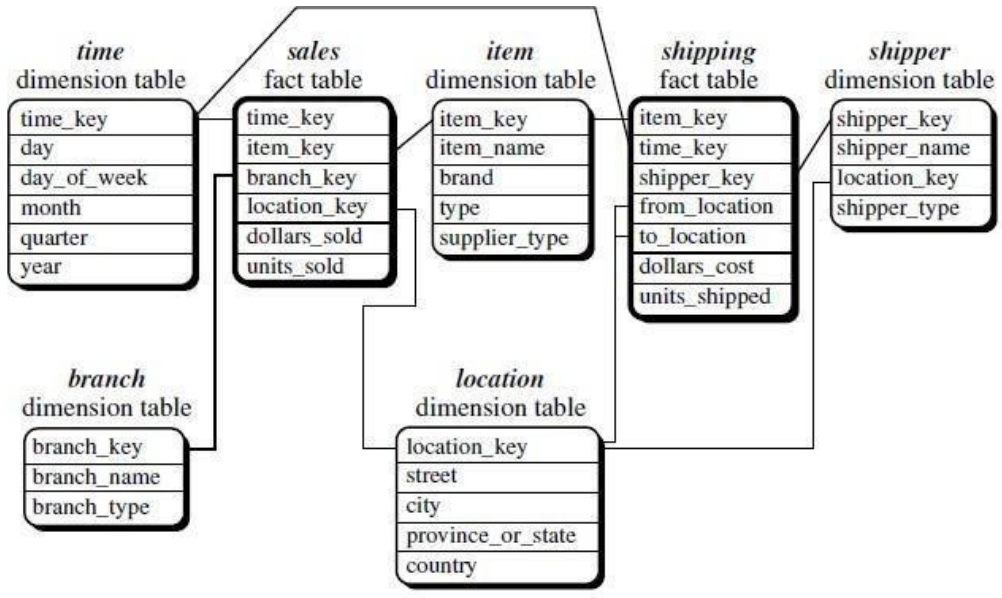
Snowflake schema of a data warehouse for sales.

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.



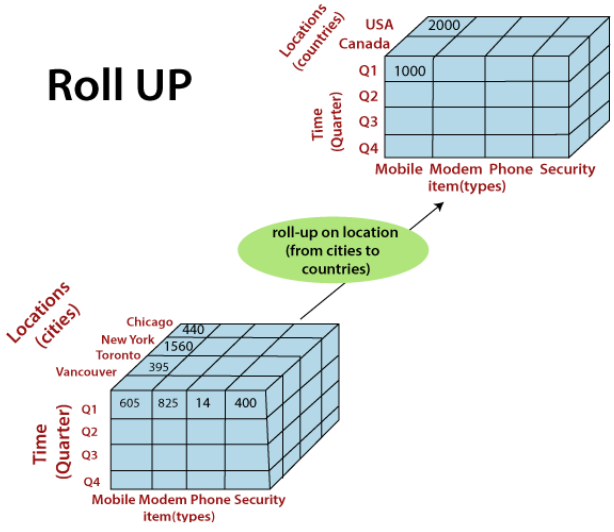
5 Fact constellation schema of a data warehouse for sales and shipping.

OLAP(Online analytical Processing) OPERATIONS

- 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.

Roll-Up:

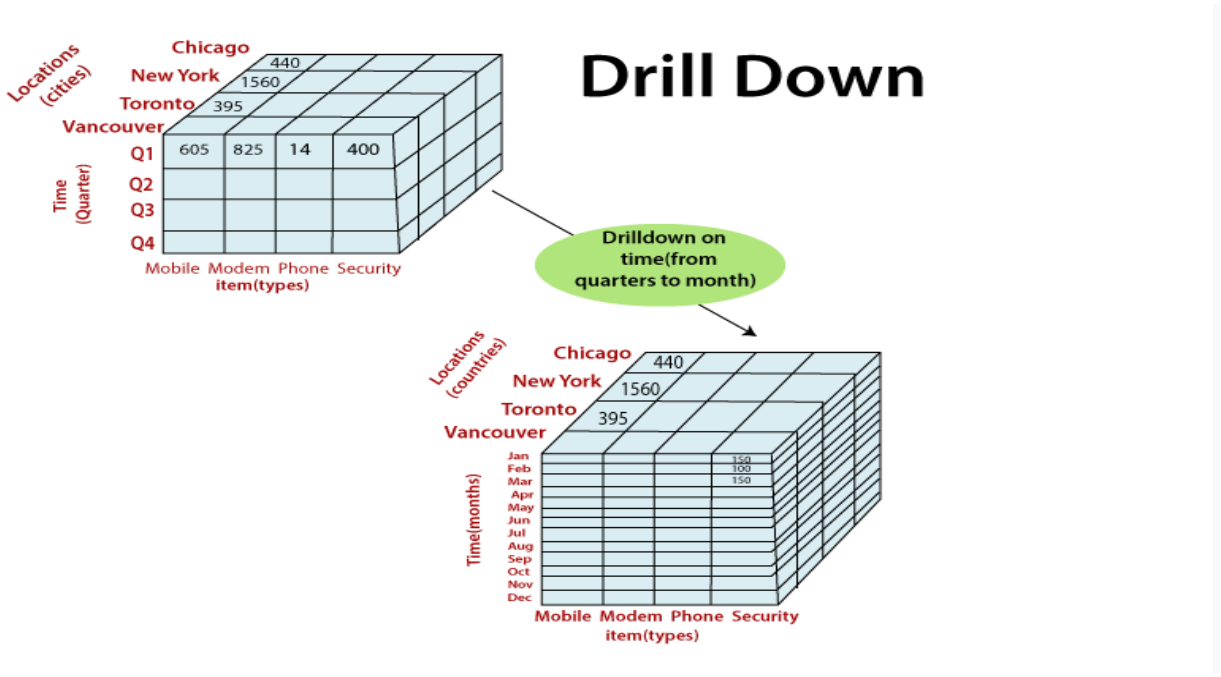
The roll-up operation (also known as drill-up or aggregation operation) performs aggregation on a data cube, by climbing down concept hierarchies, i.e., dimension reduction. Roll-up is like zooming-out on the data cubes. Figure shows the result of roll-up operations performed on the dimension location. The hierarchy for the location is defined as the Order Street, city, province, or state, country. The roll-up operation aggregates the data by ascending the location hierarchy from the level of the city to the level of the country.



Drill-Down

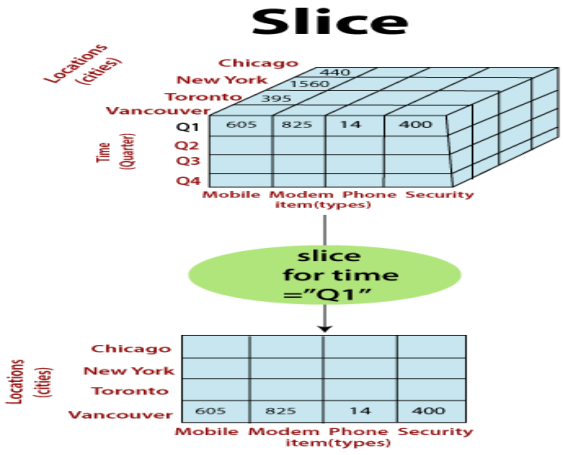
The drill-down operation (also called roll-down) is the reverse operation of roll-up. Drill-down is like zooming-in on the data cube. It navigates from less detailed record to more detailed data. Drill-down can be performed by either stepping down a concept hierarchy for a dimension or adding

additional dimensions



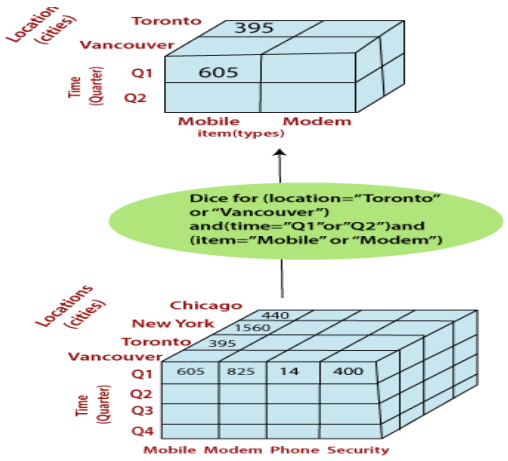
Slice

A **slice** is a subset of the cubes corresponding to a single value for one or more members of the dimension. For example, a slice operation is executed when the customer wants a selection on one dimension of a three-dimensional cube resulting in a two-dimensional site. So, the Slice operations perform a selection on one dimension of the given cube, thus resulting in a subcube.



Dice

The dice operation describes a subcube by operating a selection on two or more dimension.



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.

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.

3. Hybrid OLAP (HOLAP):

- There is no clear agreement across the industry as to what constitutes Hybrid OLAP, exceptthat 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.

UNIT-II

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 behaviors. 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, subsequences, and substructures.

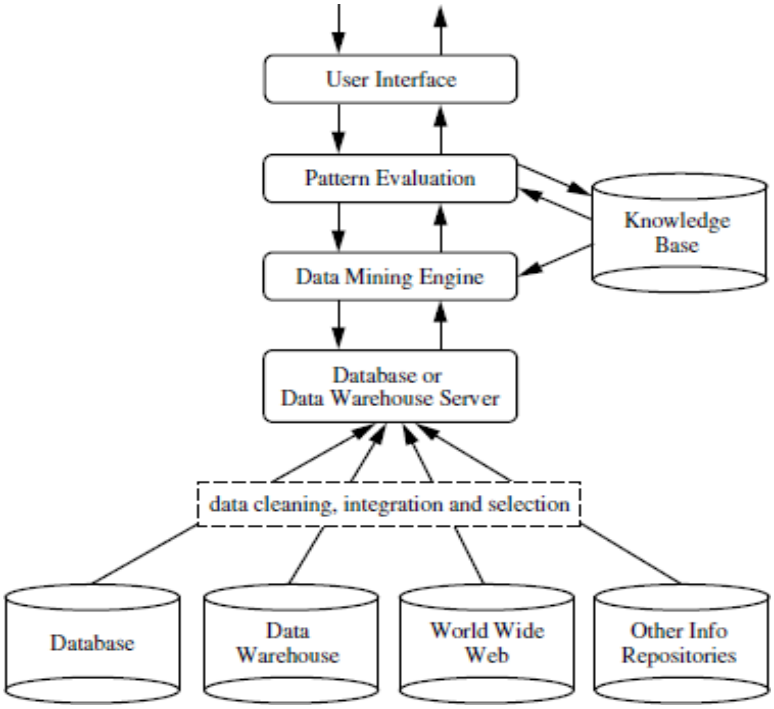
A frequent item set typically refers to a set of items that frequently appear together in a transactional dataset, such as milk and bread. A frequently occurring subsequence, such as the pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is a (frequent)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

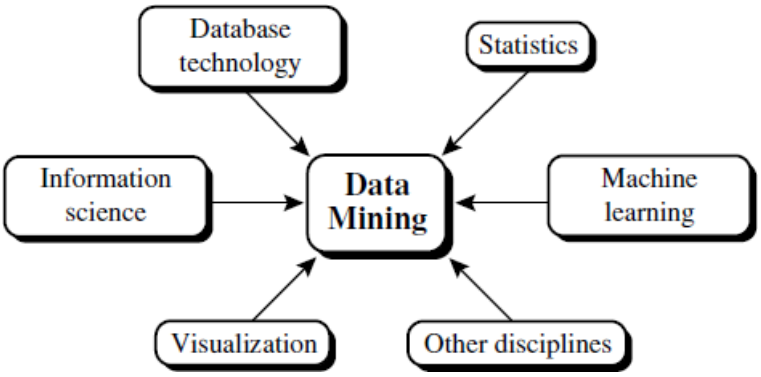
4. User interface:

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 datastructures, 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.

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.

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

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

- **Presentation and visualization of data mining results.** - Once the patterns are discovered it needs to be expressed in high level languages, visual representations. This 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 Integration:

It combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat

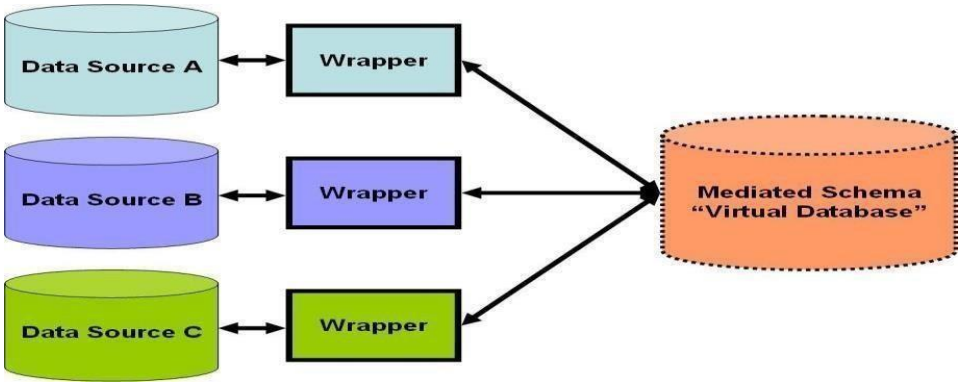
files.The data integration systems are formally defined as

triple<G,S,M>

Where G: The global schema

S:Heterogeneous source of schemas

M: Mapping between the queries of source and global schema



Issues in Data integration:

1. **Schema integration and object matching:**

   How can the data analyst or the computer be sure that customer id in one database and customer number in another reference to the same attribute.

2. **Redundancy:**

   An attribute (such as annual revenue, for instance) may be redundant if it can be derived from another attribute or set of attributes. Inconsistencies in attribute or dimension naming

can also cause redundancies in the resulting dataset.

3. **detection and resolution of data value conflicts:**

For the same real-world entity, attribute values from different sources may differ.

Data Transformation:

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

Data transformation can involve the following:

- **Smoothing**, which works to remove noise from the data. Such techniques include binning, regression, and clustering.
- **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. **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.
- **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.
- **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.

Data Preprocessing:

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

Steps of Data Preprocessing

Data preprocessing is an important step in the data mining process that involves cleaning and transforming raw data to make it suitable for analysis. Some common steps in data preprocessing include:

Data Cleaning: This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.

Data Integration: This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.

Data Transformation: This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.

Data Reduction: This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

Data Discretization: This involves dividing continuous data into discrete categories or intervals. Discretization is often used in data mining and machine learning algorithms that require categorical data. Discretization can be achieved through techniques such as equal width binning, equal frequency binning, and clustering.

Data Normalization: This involves scaling the data to a common range, such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales. Common normalization techniques include min-max normalization, z-score normalization, and decimal scaling.

Data preprocessing plays a crucial role in ensuring the quality of data and the accuracy of the analysis results. The specific steps involved in data preprocessing may vary depending on the nature of the data and the analysis goals.

By performing these steps, the data mining process becomes more efficient and the results become more accurate.

Preprocessing in Data Mining

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Preprocessing in Data MiningSteps Involved in Data Preprocessing

1. **Data Cleaning**: The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

Missing Data: This situation arises when some data is missing in the data. It can be handled in various ways.

Some of them are:

Ignore the tuples: This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

Fill the Missing values: There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

Noisy Data: Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

Binning Method: This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

Regression:Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

Clustering: This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. **Data Transformation**: This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

Normalization: It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

Attribute Selection: In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

Discretization: This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

Concept Hierarchy Generation: Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute "city" can be converted to "country".

3. **Data Reduction**: Data reduction is a crucial step in the data mining process that involves reducing the size of the dataset while preserving the important information. This is done to improve the efficiency of data analysis and to avoid overfitting of the model. Some common steps involved in data reduction are:

Feature Selection: This involves selecting a subset of relevant features from the dataset. Feature selection is often performed to remove irrelevant or redundant features from the dataset. It can be done using various techniques such as correlation analysis, mutual information, and principal component analysis (PCA).

Feature Extraction: This involves transforming the data into a lower-dimensional space while preserving the important information. Feature extraction is often used when the original features are high-dimensional and complex. It can be done using techniques such as PCA, linear discriminant analysis (LDA), and non-negative matrix factorization (NMF).

Sampling: This involves selecting a subset of data points from the dataset. Sampling is often used to reduce the size of the dataset while preserving the important information. It can be done using techniques such as random sampling, stratified sampling, and systematic sampling.

Clustering: This involves grouping similar data points together into clusters. Clustering is often used to reduce the size of the dataset by replacing similar data points with a representative centroid. It can be done using techniques such as k-means, hierarchical clustering, and density-based clustering.

Compression: This involves compressing the dataset while preserving the important information. Compression is often used to reduce the size of the dataset for storage and transmission purposes. It can be done using techniques such as wavelet compression, JPEG compression, and gif compression.

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.

Problem Definition:

The problem of association rule mining is defined as:

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*.

Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*.

Each transaction in D has a unique transaction ID and contains a subset of the items in I.

A *rule* is defined as an implication of the form $X \Rightarrow Y$

where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

The sets of items (for short *item sets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule respectively.

Example:

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is $I = \{\text{milk}, \text{bread}, \text{butter}, \text{beer}\}$ and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table.

An example rule for the supermarket could be $\{\text{butter}, \text{bread}\} \Rightarrow \{\text{milk}\}$ meaning that if butter and bread are bought, customers also buy milk.

Example database with 4 items and 5 transactions

Transaction ID	milk	bread	butter	beer
1	1	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Important concepts of Association Rule Mining:

- The **support** $\text{supp}(X)$ of an item set X is defined as the proportion of transactions in the data set which contain the item set. In the example database, the item set

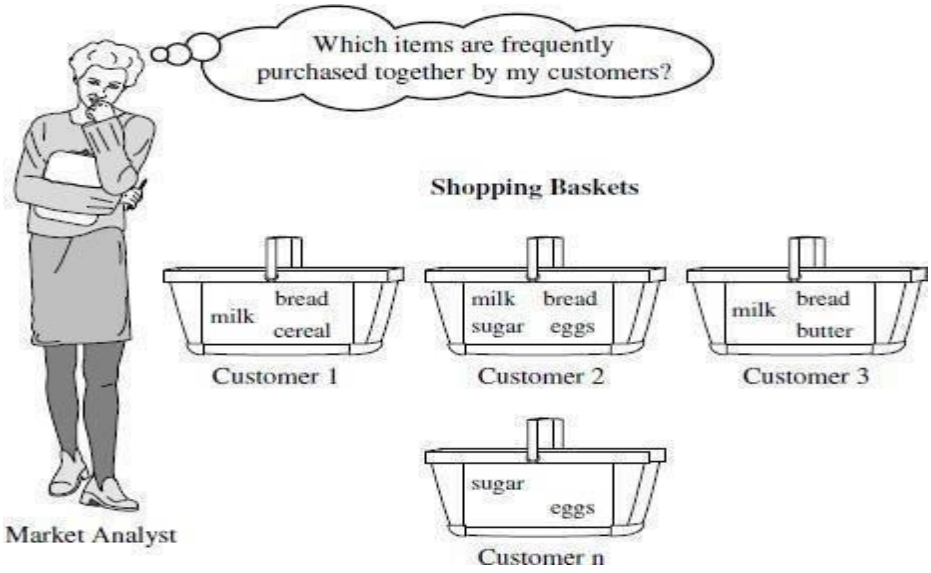
  $\{\text{milk}, \text{bread}, \text{butter}\}$ has a support of $1/5 = 0.2$ since it occurs in 20% of all transactions (1 out of 5 transactions).
- The **confidence** of a rule is defined
  $\text{conf}(X \Rightarrow Y) = \text{supp}(X \cup Y)/\text{supp}(X)$.
  For example, the rule $\{\text{butter}, \text{bread}\} \Rightarrow \{\text{milk}\}$ has a confidence of $0.2/0.2 = 1.0$ in the database, which means that for 100% of the transactions containing butter and bread the rule is correct (100% of the times a customer buys butter and bread, milk is bought as well). Confidence can be interpreted as an estimate of the probability $P(Y|X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.
- The ***lift*** of a rule is defined as
  $$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)}$$
  or the ratio of the observed support to that expected if X and Y were independent. The rule $\{\text{milk}, \text{bread}\} \Rightarrow \{\text{butter}\}$ has a lift of $\frac{0.2}{0.4 \times 0.4} = 1.25$.
- The **conviction** of a rule is defined as
  $$\text{conv}(X \Rightarrow Y) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \Rightarrow Y)}.$$

The rule $\{\text{milk, bread}\} \Rightarrow \{\text{butter}\}$ has a conviction of $\frac{1-0.4}{1-.5} = 1.2$,

and can be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions.

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 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 headingtoward 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 themaximal 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 $L_k$requires one full scan of the database.
- A two-step process is followed in Apriori consisting of join and prune action.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)      Ck = apriori_gen(Lk−1);
(4)      for each transaction t ∈ D { // scan D for counts
(5)         Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)         for each candidate c ∈ Ct
(7)            c.count++;
(8)      }
(9)      Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)      for each itemset l2 ∈ Lk−1
(3)         if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)            c = l1 ⋈ l2; // join step: generate candidates
(5)            if has_infrequent_subset(c, Lk−1) then
(6)               delete c; // prune step: remove unfruitful candidate
(7)            else add c to Ck;
(8)         }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)      if s ∉ Lk−1 then
(3)         return TRUE;
(4)   return FALSE;
```

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

There are nine transactions in this database, that is, |D| = 9.

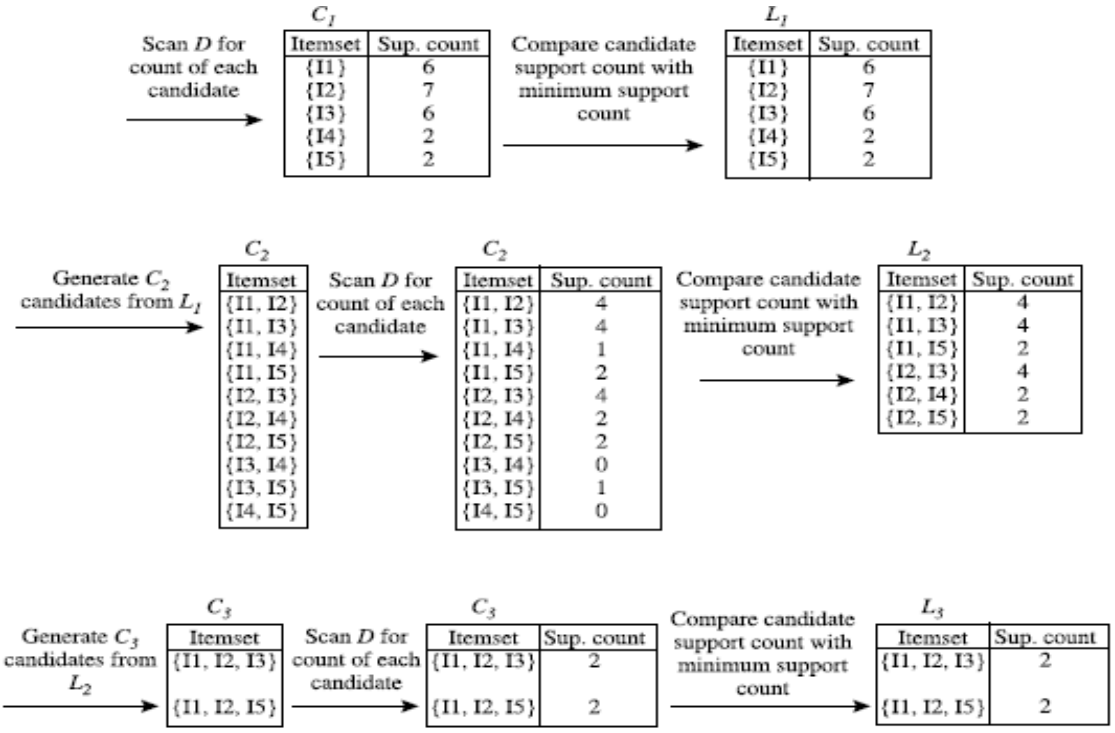
Steps:

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 occurrencesof 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 minimumsupport. 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 setsin 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.



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

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} \bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\} = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

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

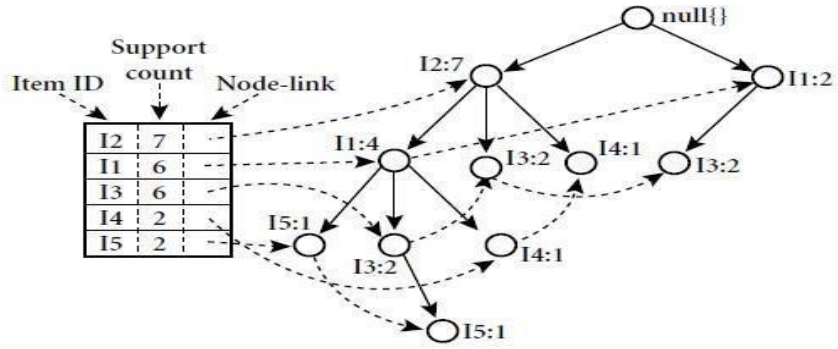
We re-examine the mining of transaction database, *D*, of Table 5.1 in Example 5.3 using the frequent patterngrowth approach.

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 the tree via a chain of node-links. The tree obtained after scanning all of the transactions is shown in Figure 5.7 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.



An FP-tree registers compressed, frequent pattern information.

Mining the FP-tree by creating conditional (sub-)pattern bases.

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

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 generatedfrom a conditional FP-tree.

Mining of the FP-tree is summarized in Table 5.2 and detailed as follows. 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 of Figure 5.7. (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 togenerate strong association rules from them.

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

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

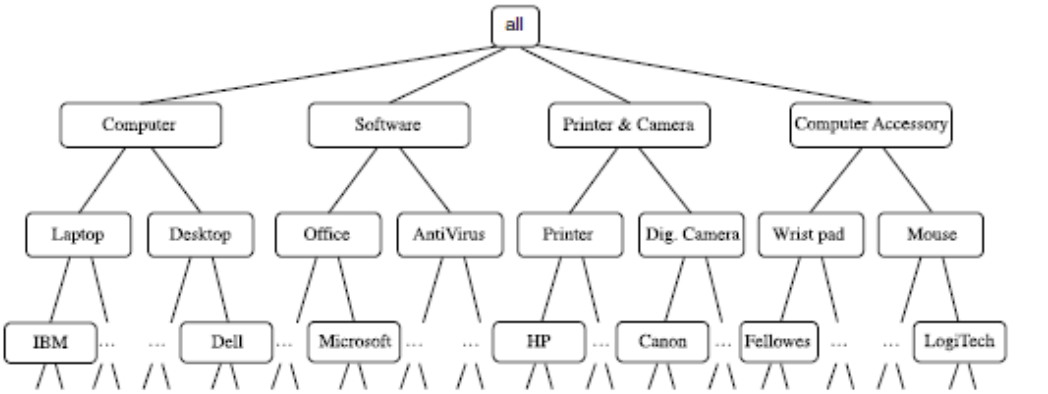
- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ *min_conf*, where *min_conf* is the minimum confidence threshold.

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 calledmultiple-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.

TID	Items Purchased
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...



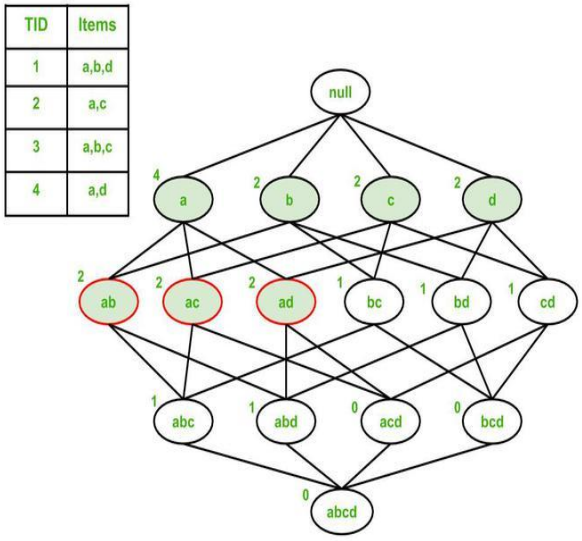
A concept hierarchy for *AllElectronics* computer items.

The concept hierarchy has five levels, respectively referred to as levels 0to 4, starting with level0 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 softwareLevel 3 includes IBM desktop computer, . . . , Microsoft office software, and so on.
- Level 4 is the most specific abstraction level of this hierarchy.

A **maximal frequent itemse**t 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).



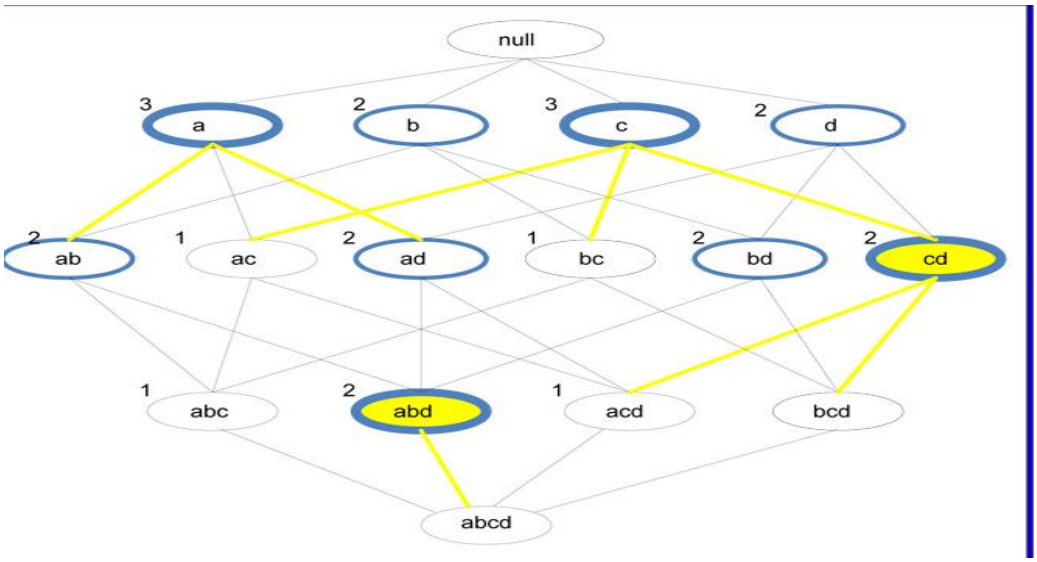
Out of these 7 frequent itemsets, 3 are identified as maximal frequent (having red outline):

- **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.

Closed Frequent Itemset:

It is a frequent itemset that is both closed and its support is greater than or equal to minsup. An itemset is closed in a data set if there exists no superset that has the same support count as this original itemset.



the lattice diagram above shows the maximal, closed and frequent itemsets. The itemsets that are circled with blue are the frequent itemsets. The itemsets that are circled with the thick blue are the closed frequent itemsets. The itemsets that are circled with the thick blue and have the yellow fill are the maximal frequent itemsets. In order to determine which of the frequent itemsets are closed, all you have to do is check to see if they have the same support as their supersets, if they do they are not closed. For example **ad** is a frequent itemset but has the same support as **abd** so it is NOT a closed frequent itemset; c on the other hand is a closed frequent itemset because all of its supersets, ac, bc, and cd have support that are less than3.

As you can see there are a total of 9 frequent itemsets, 4 of them are closed frequent itemsets and out of these 4, 2 of them are maximal frequent itemsets. This brings us to the relationship between the three representations of frequent itemsets.

UNIT-IV

Classification :

- 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 valuewith 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 *A*1 and *A*2 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:**

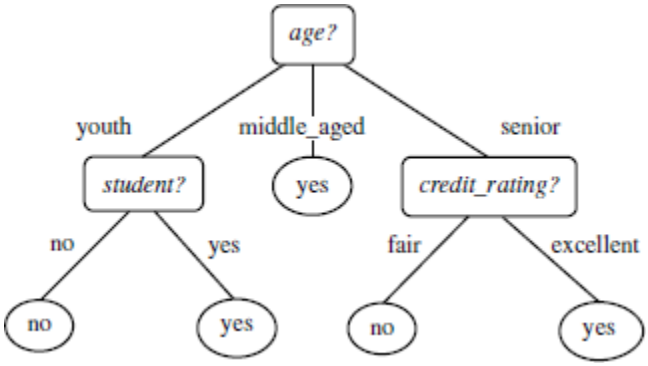
  This refers to the computational costs involved in generating and using the given classifier or predictor.
- **Robustness:**

  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, where
  - 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 molecularbiology.

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

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,

{a_1, a_2, … ,a_v}, 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:

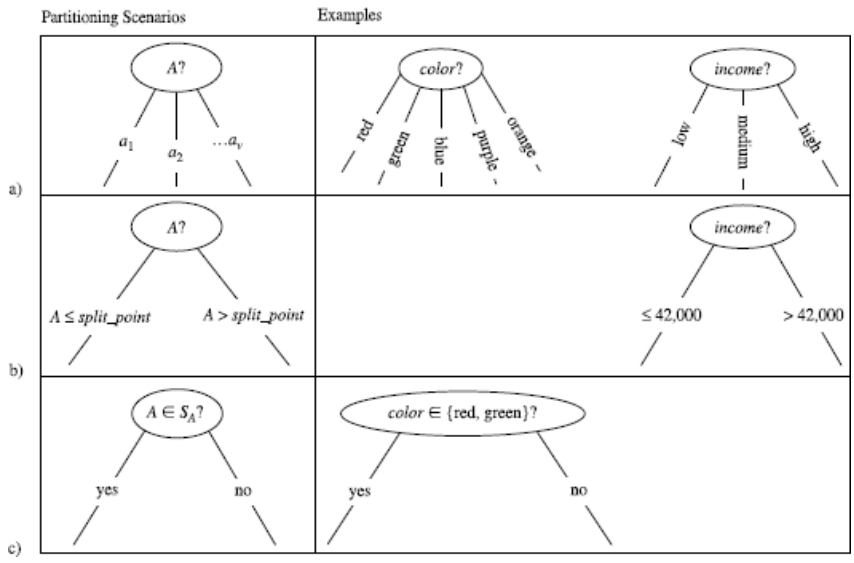
In this case, the test at node N has two possible outcomes, corresponding to the conditions

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) IfA 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 hypothesis H holds given the ―evidence or observed data tuple X.
- P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.
- Bayes' theorem is useful in that it provides a way of calculating the posterior probability, $P(H|\boldsymbol{X})$, from $P(H)$, $P(\boldsymbol{X}|H)$, and $P(\boldsymbol{X})$.

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

Naïve Bayes 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

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

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

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

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

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

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

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 $P(\boldsymbol{X}|Ci)$, we consider the following:

- If $A_k$is categorical, then $P(x_k|Ci)$ is the number of tuples of class *Ci* in *D* having the value $x_k$for A_k, divided by $|C_{i,D}|$ the number of tuples of class $C_i$in *D*.
- If $A_k$is 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*(*X*j*Ci*)*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

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

k-Nearest-Neighbor Classifiers

The *k*-nearest-neighbor method was first described in the early 1950s. The method is labor intensive when given large training sets, and did not gain popularity until the 1960s when increased computing power became available. It has since been widely used in the area of pattern recognition.

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuplewith 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, $\boldsymbol{X}1 = (x11, x12, : : : , x1n)$ and $\boldsymbol{X}2 = (x21, x22, : : : , x2n)$, is

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

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1 and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count. Typically, we normalize the values of each attribute before using Equation This helps prevent attributes with initially large ranges (such as income) fromoutweighing attributes with initially smaller ranges (such as binary attributes). Min-max normalization,

for example, 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 minA and maxA are the minimum and maximum values of attribute A. For k-nearest-neighbor classification, the unknown tuple is assigned the most common 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."But how can distance be computed for attributes that not numeric, but categorical, such as color?" The above discussion assumes that the attributes used to describe the tuples are all numeric.

For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple X1 with that in tuple X2. If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0.

If the two are different (e.g., tuple X1 is blue but tuple X2 is red), then the difference is considered to be 1. Other methods may incorporate more sophisticated schemes for differential grading (e.g., where a larger difference score is assigned, say, for blue and white than for blue and black). "What about missing values?" In general, if the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference. Suppose that each of the attributes have been mapped to the range [0, 1]. For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing. If A is numeric and missing from both tuples X1 and X2, then the difference is also taken to be 1. If only one value is missing and the other (which we'll call v0) is present and normalized, then we can take the difference to be either j1v0j or j0v0j (i.e., 1v0 or v0), whichever is greater. "How can I determine a good value for k, the number of neighbors?" This can be determined experimentally. Starting with k = 1, we use a test set to estimate the error rate of the classifier.

This process can be repeated each time by incrementing k to allow for one more neighbor. The k value that gives the minimum error rate may be selected. In general, the larger the number of training tuples is, the larger the value of k will be (so that classification and prediction decisions can be based on a larger portion of the stored tuples). As the number of training tuples approaches infinity and k =1, the error rate can be no worse then twice the Bayes error rate (the latter being the theoretical minimum). If k also approaches infinity, the error rate approaches the Bayes error rate.

Nearest-neighbor classifiers use distance-based comparisons that intrinsically assign equal weight to each

attribute. They therefore can suffer from poor accuracy when given noisy or irrelevant attributes. The method, however, has been modified to incorporate attribute weighting and the pruning of noisy data tuples. The choice of a distance metric can be critical. The Manhattan (city block) distance ,or other distance measurements, may also be used.

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 thatcan 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 o fthe 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.

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

M_i is the mean of cluster C_i

The k-means partitioning algorithm:

The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

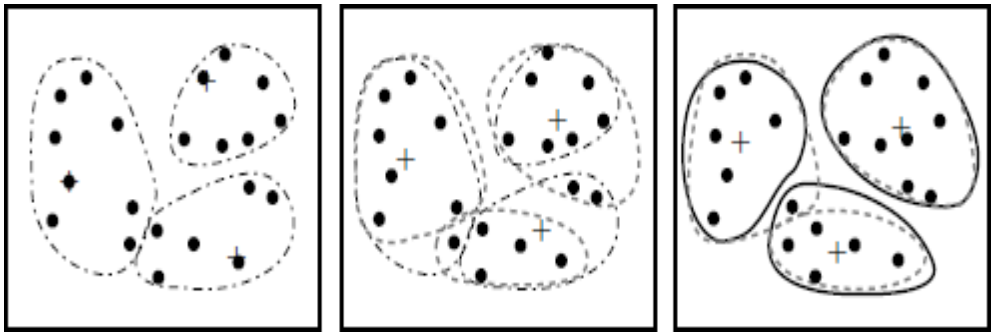
Input:

- *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

The *k*-Medoids Method:

- The k-means algorithm is sensitive to outliers because an object with an extremely large value may substantially distort the distribution of data. This effect is particularly exacerbated due to the use of the square-error function.
- Instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative object per cluster. Each remaining object is clustered with the representative object to which it is the most similar.
- The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used, defined as

$$E = \sum_{j=1}^{k} \sum_{\boldsymbol{p} \in C_j} |\boldsymbol{p} - \boldsymbol{o_j}|,$$

Where E is the sum of the absolute error for all objects in the data set

$\boldsymbol{p}$is the point in space representing a given object in clusterC_j

$\mathbf{o_j}$is the representative object of C_j

- The initial representative objects are chosen arbitrarily. The iterative process of replacing representative objects by non representative objects continues as long as the quality of the resulting clustering is improved.
- This quality is estimated using a cost function that measures the average dissimilarity between an object and the representative object of its cluster.
- To determine whether a non representative object, oj random, is a good replacement for a current representative object, oj, the following four cases are examined for each of the non representative objects.

Case 1:

'p' currently belongs to representative object, o_j. If $o_j$is replaced by o_{random}asa representative object and p is closest to one of the other representative objects, o_i,$i \neq j$, then p is reassigned to o_i.

Case 2:

'p' currently belongs to representative object, oj. If ojis replaced by o_{random}asa representative

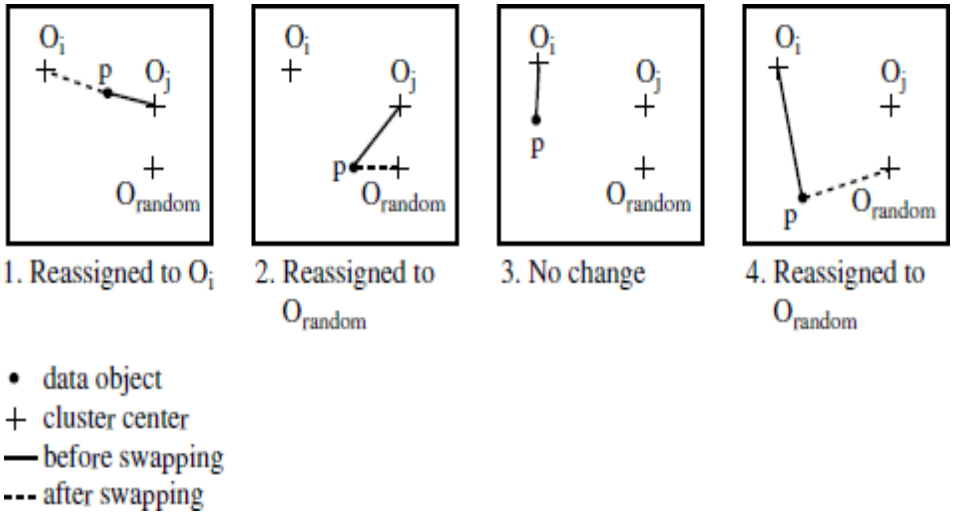
object and p is closest to o_{random}, then p is reassigned to o_{random}.

Case 3:

'p' currently belongs to representative object, o_i, $i \neq j$. If ojis replaced by o_{random}as a representative object and p is still closest to o_i, then the assignment does not change.

Case 4:

'p' currently belongs to representative object, oi, $i \neq j$. If $o_j$is replaced byo_{random}as a representative object and p is closest to o_{random}, then p is reassigned too$_{random}$.



Four cases of the cost function for *k*-medoids clustering

The *k*-Medoids Algorithm:

The k-medoids algorithm for partitioning based on medoid or central objects.

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 in D as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, S, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of k representative objects;
(7) **until** no change;

The *k*-medoids method is more robust than *k*-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the *k*-means method.

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.

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,

$$H : o_i \in F, \quad \text{where } i = 1, 2, \ldots, n.$$

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 o_i comes from another distribution model, G, is adopted. The result is very much dependent on which model F is chosen because $o_i$may 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 thesame 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 objectthat 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, once*M*+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 $O(n^2)$ computational complexity, a cell-based algorithm was developed for memory-resident data sets. Its complexity is $O(c^k+n)$, 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 $\frac{dmin}{2\sqrt{k}}$. 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, o_0 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 $N_{kdistance(p)(}p)$, or $N_k(p)$for short. By setting k to MinPts, we get $N_{MinPts}(p)$. 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 defined as

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

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, $\{x_1, \ldots, x_n\}$, a possible dissimilarity function is the variance of the numbers in the set, that is,

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

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 D_{j-1} does not cover any string in D_j that is not in D_{j-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.