ABSTRACT

Today, there is tremendous increase of the information available on electronic form. Day by day it is increasing massively. There are enough opportunities for research to retrieve knowledge from the data available in this information. Data mining and applied statistical methods are the appropriate tools to extract knowledge from such data. Although data mining is a very important and growing area, there is insufficient coverage of it in the literature, especially from a statistical viewpoint. Most of the research on data mining are either too technical and computer science oriented or too applied and marketing driven. Aim of this research is to establish a bridge between data mining methods and applications in the fields of business and industry by adopting a coherent and rigorous approach to statistical modeling.

In this book basic concepts of data mining and data warehousing are covered. Meta data, data mart, data warehousing architecture, data mining techniques are also discussed. We also used open source Java based software Weka for the analysis and comparison study of various statistical techniques and algorithms..

To implement all the models practically, we have use stock market data NSE (National Stock Exchange). Large collection of stock market data being gathered for a implementation of developed applications (tools) based on the suggested models in research study. Data Mining from such a data corpus can lead to interesting results and discoveries of patterns.

1. Fundamentals of Data Mining

1.1 Basic Concepts

The increasing availability of data in the current information society has led to the need for valid tools for its modeling and analysis. Data mining and applied statistical methods are the appropriate tools to extract knowledge from such data. Data mining can be defined as the process of selection, exploration and modeling of large databases in order to discover model and pattern that are unknown a priori. It differs from applied statistics mainly in terms of its scope; whereas applied statistics concerns the application of statistical methods to the data at hand, data mining is a whole process of data extraction and analysis aimed at the production of decision rules for specified business goals. In other words, data mining is a business intelligence process.

Although data mining is a very important and growing topic, there is insufficient coverage of it in the literature, especially from a statistical viewpoint. Most of the research on data mining are either too technical and computer science oriented or too applied and marketing driven. Our research aims to establish a bridge between data mining methods and applications in the fields of business and industry by adopting a coherent and rigorous approach to statistical modeling.

Nowadays, each individual and organization – business, family or institution can access a large quantity of data and information about itself and its environment. This data has the potential to predict the evolution of interesting variables or tends in the outside environment, but so far that potential has not been fully exploited. This is particularly true in the business field. There are two main problems, information is scattered within different archive systems that are not connected with one another, producing an inefficient organization of the data. There is a lack of awareness about statistical tools and their potential for information elaboration. This interferes with the production of efficient and relevant data synthesis.

Two developments could help to overcome these problems. First, software and hardware continually, offer more power at lower cost, allowing

organization to collect and organize data in structures that give easier access and transfer. Second, methodologically research, particularly in the field of computing and statistics, has recently led to the development of flexible and scalable procedures that can be used to analyse large data stores. These two developments have meant that data mining is rapidly spreading through many businesses as an important intelligence tool for backing up decisions.

1.2 Use of Data Mining

As recently as before few years ago, data mining was a new concept for many people and organizations. Data mining products were also new and marred by unpolished interfaces. Only the most innovative or daring early adopters was trying to apply these emerging tools. Now scenario has been changed, today data mining products have matured, and data mining is accessible to a much wider audience. We are even seeing the emergence of specialized vertical market data mining products.

What kinds of organization can use data mining technology to solve their business problem and how users understand to apply these tools effectively to get benefit that is very much important. Data mining extracts hidden information, knowledge and pattern from large volume of data that was previously not available. Data mining tools do more than query and data analysis tools.

Simple query and analysis tools can respond to questions such as, "Do sales of Product An increases in January?" or "Do sales of Product A decrease when there is a promotion on Product B?" In contrast to this, a data mining tool can be asked, "What are the factors that determine sales of Product A?

Using traditional tools, the analyst starts with a question and assumption, or perhaps just a hunch and explores the data and builds a model, step-by-step, working to prove or disprove a theory. In traditional approach the analyst propose all hypotheses, tests these proposed hypotheses. If requires, then also propose an additional or substitute hypothesis, test it also, and so on, and in this iterative way, a model can be built. Responsibility of an analyst does not disappear entirely with data mining; data mining shifts much of the

work of finding an appropriate model from the analyst to the computer. This has the following potential benefits.

- A model can be generated with requires less manual effort. (It can be more efficient.)
- Larger number of models can be evaluated; it increases the odds of finding a better model.
- The analyst needs less technical expertise because most of the step-by-step procedure is automated.

There is extensive use of statistics in data mining. So nowadays data mining is embarked with statistical techniques is also known as statistical data mining or computational data mining.

1.3 Current Scenario

In modern age all reputed and well-suited organizations are under enormous pressure to respond quickly for continuous changing trends in the competitive market. So it is obvious that in order to achieve this challenge, they need rapid access to different varieties of information before a decision can be framed. For making the right choices for organization, it is very much essential to study and analyze the past data and identify most relevant trends. Before perform any trend analysis it is very much necessary to access to entire relevant information, and naturally this information is mainly stored in very large/huge databases. To build a data warehouse is an easiest approach to gain access large volume of data for extract patterns and effective decision making for business environment. Data warehouse stores historical data and operational data. Operational data is useful to access information and generate some patterns using data mining techniques. These generated patterns can be analyzed to organize and summarize as business intelligence. Today it is business intelligence with quality information at backbone can give the power of with standing and growth.

1.4 Research in Data Mining

It is very much interesting to do analytical study and computational modeling of statistical methods useful for data mining. How various statistical methods are useful for data mining and how competitive but less expensive computational model can be built? Data mining tools available in market are very expensive and too complex. It is necessary to suggest a model for data mining that built with appropriate statistical methods. It will create a special interested in making comparison of different algorithms of data mining with effort to develop experimental paradigms that allow testing the mining algorithms.

For the different problems and applications, data mining strategies do not follow the same track and gives different pictures in different situations. The variability may cause the implementation complex and success rate not to the anticipated level. This problem is the main reason for failure data mining applications in most of organizations. To smoothen the track, efforts are made to propose the model that keeps the track of data mining and that is expected to cover many different situations.

Today the numbers of organizations are rapidly increasing which are using data mining applications for business intelligence. The model is to be proposed under this research initiative will help the developers for data mining solutions.

1.5 Data Mining – An Introduction

To understand the term ‘Data Mining’ it is useful to look at the literal translation of the word: to mine in English means to extract. The verb usually refers to mining operations that extract from the Earth her hidden, precious resources. The association of this world with data suggests an in-depth search to find additional information, which previously went unnoticed in the mass of data variables. From the viewpoint of scientific research, data mining is relatively new discipline that has developed mainly from studies carried out in other disciplines such as computing, marketing management and statistics. Many of the methodologies used in data mining come from two branches of

research, one developed in the machine learning community and the other developed in the statistical community, particularly in multivariate and computational statistics.

Machine learning is connected to computer science and artificial intelligence and is concerned with finding relations and regularities in data that can be translated into general truths. The aim of machine learning is the reproduction of the data-generating process, allowing analysts to generalize from the observed data to new, unobserved cases. Rosenblatt (1962) introduced the first machine-learning model, called the perception. Following on from this, neural networks developed in the second half of the 1980s. During the same period, some researchers perfected the theory of decision trees used mainly for dealing with problems of classification. Statistics has always been about creating models for analyzing data, and now there is the possibility of using computers to do it. Form the second half of the 1980s; given the increasing importance of computational models as the basis for statistical analysis, there was also a parallel development of statistical methods to analyze real multivariate applications. In the 1990s statisticians began showing interest in machine learning methods as well, which led to important developments in methodology.

Towards the end of the 1980s machine learning methods started to be used beyond the fields of computing and artificial intelligence. In particular, they were used in database marketing applications where the available databases were used for elaborate and specific marketing campaigns. The term knowledge discovery in databases (KDD) was coined to describe all those methods that aimed to find relations and regularity among the observed data. Gradually the term KDD was expanded to describe the whole process of extrapolating information from a database, from the identification of the initial business aims to the application of the decision rules. The term ‘data mining’ was used to describe the component of the KDD process where the learning algorithms were applied to the data.

This terminology was first formally put forward by Usama Fayyad at the First International Conference on Knowledge Discovery and Data Mining held in

Montreal in 1995 and still considered one of the main conferences on this topic. It was used to refer to a set of integrated analytical technique divided into several phases with the aim of extrapolation previously unknown knowledge from massive sets of observed data that do no appear to have any obvious regularity or important relationships. As the term ’data mining’ slowly established itself, it became a synonym for the whole process of extrapolating knowledge. The previous definition omits one important aspect – the ultimate aim of data mining. In data mining the aim is to obtain results that can be measured in terms of their relevance for the owner of the database – business advantage. Here is a more complete definition of data mining:

Data Mining is the process of selection, exploration, and modeling of large quantities of data to discover regularities or relations that are at first unknown with the aim of obtaining clear and useful results for the owner of the database.

To apply data mining methodology means following in integrated methodological process that involves translating the business needs into a problem, which has to be analyzed, retrieving the database needed to carry out the analysis, and applying a statistical technique implemented in a computer algorithm with the final aim of supportive important results useful for taking a strategic decision. The business needs, setting off what has been called ‘the virtuous circle of knowledge’ introduced by data mining (Berry and Linoff, 1997).

Data mining is not just about the use of a computer algorithm or a statistical technique: it is a process of deriving business intelligence that can be used together with what is provided by information technology to support business decision.

1.6 Data Mining – An overview

The emergence of data mining is closely connected to developments in computer technology, particularly the evolution and organization of databases, which have recently made great leaps forward. Some new terms are clarified as below.

Query and reporting tools are simple and very quick to use: they helps to explore business data at various levels. Query tools retrieve the information and reporting tools present it meaningfully. They allow the results of analysis to be transmitted across a client-server network, intranet or even on the internet. The networks allow sharing, so that the data can be analyzed by the most suitable platform. This makes it possible to exploit the analytical potential of remote servers and receive an analysis report on local PCs. A client-server network must be flexible enough to satisfy varied types of remote requests, from a simple reordering of data to ad hoc queries using Structured Query Language (SQL) for extracting and summarizing data in the database.

Data retrieval, like data mining, extracts interesting data and information from archives and databases. The difference is that, unlike data mining, the criteria for extracting information are decided beforehand so they are exogenous from the extraction itself. A classic example is a request from the marketing department of a company to retrieve all the personal details of clients who have bought product A and product B at least once in that order. This request may be based on the idea that there is some connection between having bought A and B together at least once but without any empirical evidence. The names obtained from this exploration could then be the targets of the next publicity campaign. In this way the success percentage (i.e. the customers who will actually buy the products advertised compared to the total customers contacted) will definitely be much higher than otherwise. Once again, without a preliminary statistical analysis of the data, it is difficult to predict the success percentage and it is impossible to establish whether having better information about the customers' characteristics would give improved results with a smaller campaign effort.

Data mining is different from data retrieval because it looks for relations and association between phenomena that are known beforehand. It also allows the effectiveness of a decision to be judged on the data, which allows a relational evaluation to be made, and on the objective data available. It is not to be confused with data mining methods used to create multidimensional reporting tools. Online Analytical Processing (OLAP). OLAP is usually a

graphical instrument used to highlight relations between the variables available following the logic of a two dimensional report. OLAP is an important tool for business intelligence. The query and reporting tools describe what a database contains, but OLAP is used to explain why certain relation exists.

OLAP is not a substitute for data mining: the two techniques are complementary and used together they can create useful strategies. OLAP can be used in the processing stages of data mining. This makes understanding the data easier, because it becomes possible to focus on the relevant data, identifying special cases or looking for principal interrelations. The final data mining results, expressed using specific summary variables, can be easily represented in an OLAP hypercube.
Following is the simple sequence that shows the evolution of business intelligence tools used to extrapolate knowledge from a database:

QUERY AND REPORTING ⟶ DATA RETRIEVEL ⟶ OLAP ⟶ DATA MINING

Above sequence indicates that Query and Reporting has the lowest information capacity and Data Mining has highest information capacity. This suggests a trade-off between information capacity and ease of implementation. Lack of information is one of the greatest obstacles to achieving efficient data mining.

The creation of a data warehouse can eliminate many of these problems. Efficient organization of the data in a data warehouse coupled with efficient and scalable data mining allows the data to be used correctly and efficiently to support business decision.

1.7 Data Mining and Statistics

Statistics has always been about creating methods to analyze data. The main difference between statistical methods and machine learning methods is that statistical methods are usually developed in relation to the data being analyzed but also according to a conceptual reference paradigm. Although this has made the statistical methods coherent and rigorous, it has also limited their ability to adapt quickly to the new methodologies arising from new

information technology and new machine learning applications. Statisticians have recently shown an interest in data mining and this could help its development.

For a long time statisticians saw data mining as a synonymous with ‘data fishing’, ‘data dredging’, or ‘data snooping’. In all these cases data mining had negative connotations. This idea came about because of two main criticisms. First, there is not just one theoretical reference model but also several models in competition with each other; these models are chosen depending on the data being examined. The criticism of this procedure is that it is always possible to find a model, however complex, which will adapt well to the data. Second, the great amount of data available may lead to non-existent relation being found among the data.

Although these criticisms are worth considering, we shall see that the modern methods of data mining pay great attention to the possibility of generalizing results. This means that when choosing a model, the predictive performance is considered and the more complex models are penalized. It is difficult to ignore the fact that many important findings are not known beforehand and cannot be used in developing a research hypothesis. This happens in particular when there are large databases.

This last aspect is one of the characteristics that distinguished data mining from statistical analysis. Whereas statistical analysis traditionally concerns itself with analyzing primary data that has been collected to check specific research hypotheses, data mining can also concern itself with secondary data collected for other reasons. This is the norm, for example, when analyzing company data that comes from a data warehouse. Furthermore, statistical data can be experimental data, but in data mining the data is typically observational data.

Berry and Linoff (1997) distinguish two analytical approaches to data mining. They differentiate top-down analysis (confirmative) and bottom-up analysis (explorative). Top-down analysis aims to confirm or reject hypothesis and tries to widen our knowledge of a partially understood phenomenon; it achieves this principally by using the traditional statistical methods. Bottom-up analysis

is where the user looks for useful information previously unnoticed, searching through the data and looking for ways of connecting it to create hypotheses. The bottom-up approach is typical of data mining. In reality the two approaches are complementary. In fact, the information obtained from a bottom-up analysis, which identifies important relations and tendencies, cannot explain why these discoveries are useful and to what extent they are valid. The confirmative tools of top-down analysis can be used to confirm the discoveries and evaluate the quality of decision based on those discoveries.

There are at least three other aspects that distinguish statistical data analysis from data mining. First, data mining analyses great masses of data. This implies new consideration for statistical analysis. For many applications it is impossible to analyze or even access the whole database for reasons of computer efficiency. Therefore it becomes necessary to have a sample of the data from the database being examined. This sampling must take account of the data mining aims, so it cannot be performed using traditional statistical theory, Second many databases do not lead to the classic forms of statistical organization, for example data that comes from the internet. This creates a need for appropriate analytical methods from outside the field of statistics. Third, data mining results must be of some consequence. This means that constant attention must be given to business results achieved with the data analysis models.

So it can be concluded that there are reasons for believed that the data mining is nothing new from statistical viewpoint. But there are also reasons to support the idea that, because of their nature, statistical methods should be able to study and formalize the methods used in data mining. This means that on one hand we need to look at the problems posed by data mining from a viewpoint of statistics and utility, while on the other hand we need to develop a conceptual paradigm that allows the statisticians to lead the data mining methods back to a scheme of general and coherent analysis.

1.8 Organization of the data

Data analysis requires that the data is organized into an ordered database. The data is analyzed and it depends greatly on how the data is organized within the database. In information society there is an abundance of data and a growing need for an efficient way of analyzing it. However, an efficient analysis presupposes a valid organization of the data.

It has become strategic for all medium and large organizations to have unified information system called a data warehouse; this integrates, for example, the accounting data with data arising from the production process, the contacts with the suppliers(supply chain management), and the sales trends and the contacts with the customers(customer relationship management). Another example is the increasing diffusion of electronic trade and commerce and, consequently, the abundance of data about websites visited along with any payment transactions. In this case it is essential for the services supplier, through the internet, to understand who the customers are in order to plan offers. This can be done if the transactions, which correspond to clicks on the web, are transferred to an ordered database, usually called a webhouse.

1.9 Data Warehouse

The data warehouse can be defined as ‘An integrated collection of data about a collection of subjects of subjects (units), which is not volatile in time and can support decision taken by the management’.

From this definition, the first characteristic of a data warehouse is the orientation to the subjects. This means that data in a data warehouse should be divided according to subjects rather than by business. For example, in the case of an insurance company the data put into the data warehouse should probably be divided into Customer, Policy and Insurance Premium rather than into Civil Responsibility, Life and Accident. The second characteristic is data integration, and it is certainly most important. The data warehouse must be able to integrate itself perfectly with the multitude of standard used by the different application from which data is collected. For example, various operational business applications could codify the sex of the customer in

different way and the data warehouse must be able to recognize these standards unequivocally before going on to store the information.

Third, a data warehouse can vary in time since the temporal length of a data warehouse usually oscillates between 5 to 10 years; during this period the data collected is no more than a sophisticated series of instant photos taken at specific moments in time. At the same time, the data warehouse is not volatile because data is added rather than updated. In other words, the set of photos will not change each time the data is updated but it will simply be integrated with a new photo. Finally, a data warehouse must produce information that is relevant for management decision.

This means a data warehouse is like a container of all the data needed to carry out business intelligence operations. It is the main difference between a data warehouse and other business databases. The data contained in the operational databases is used to carry out relevant statistical analysis for the business (related to various management decisions) is almost impossible. On the other hand, a data warehouse is built with this specific aim in mind.

There are two ways to approach the creation of the data warehouse. The first is based on the creation of a single centralized archive that collects all the business information and integrates it with information coming from outside. The second approach brings together different thematic databases, called data marts, that are initially not connected among themselves, but which can evolve to create a perfectly interconnected structure. The first approach allows the system administrators to constantly control the quality of the data introduced. But it requires careful programming to allow for future expansion to receive new data and to connect to other databases. The second approach is initially easier to implement and is therefore the most popular solution at the moment. Problems arise when the various data marts are connected among each other, as it becomes necessary to make a real effort to define, clean and transform the data to obtain a sufficiently uniform level. That is until it becomes a data warehouse in the real sense of the word.

In a system that aims to preserve and distribute data, it is also necessary to include information about the organization of the data itself. This idea is called

metadata and it can be used to increase the security level inside the data warehouse. Although it may be desirable to allow vast access to information, some specific data marts and some details might require limited access. Metadata is also essential for management, organization and the exploitation of the various activities. For an analyst it may be very useful to know how the profit variable was calculated, whether the sales areas were divided differently before a certain date, and how a multi-period event was split in time. The metadata therefore helps to increase the value of the information present in the data warehouse because it becomes more reliable.

Another important component of a data warehouse system is a collection of data marts. A data mart is a thematic database, usually represent in a very simple form that is specialized according to specific objectives.

1.10 Data Webhouse

The data webhouse developed rapidly in the beginning of 1990s, when it was vary successful and accumulated widespread use. The advent of the web with its revolutionary impact has forced the data warehouse to adapt to new requirements. In this new era the data warehouse becomes a web data warehouse or, more simply, data webhouse. The web offers an immense source data about people who use their browser to interact to websites. Despite the fact that most of the data related to the flow of users is very coarse and very simple, it gives detailed information about how internet users surf the net. This huge and undisciplined source can be transferred to the data webhouse, where it can be put together with more conventional sources of data that previously formed the data warehouse.

Another change concerns the way in which the data warehouse can be accessed. It is now possible to exploit all the interfaces of the business data warehouse that already exist through the web just by using the browser. With this it is possible to carry out various operations, from simple data entry to ad hoc queries through the web. In this way the data warehouse becomes completely distributed. Speed is a fundamental requirement in the design of a webhouse. However, in the data warehouse environment some requests need a long time before they will be satisfied. Slow time processing is intolerable in

an environment based on the web. A webhouse must be quickly reachable at any moment and any interruption, however brief, must be avoided.

1.11 Data marts

A data mart is a thematic database that was originally oriented towards the marketing field. Indeed, its name is a contraction of marketing database. In this sense it can be considered a business archive that contains all the information connected to new and/or potential customers. In other words, it refers to a database that is completely oriented to managing customer relations. As we shall see, the analysis of customer relationship management data is probably the main field where data mining can be applied. In general, it is possible to extract from a data warehouse as many data marts as there are aims we want to archive in business intelligence analysis. However, a data mart can be created, although with some difficulty, even when there is no integrated warehouse system. The creation of thematic data structures like data marts represents the first and fundamental move towards an informative environment for the data mining activity

1.12 Classification of the data

A data mart should be organized according to two principles: the statistical units, the elements in the reference population that are considered important for the aims of the analysis (e.g. the supply companies, the customers, the people who visit the site) and the statistical unit (e.g. the amounts customers buy, the payment methods they use, the socio-demographic profile of each customer).

The statistical units can refer to the whole reference population (e.g. All the customers of the company) or they can be a sample selected represent the whole population. There is a large body of work on the statistical theory of sampling and sampling strategies. If we consider an adequately representative sample rather than a whole population, there are several advantages. It might be expensive to collect complete information about the entire population and the analysis of great masses of data could waste a lot of time in analyzing and interpreting the results.

The statistical variables are the main source of information to work on in order to extract conclusions about the observed units and eventually to extend these conclusions to a wider population. It is good to have a large number of variables to achieve these aims, but there are tow main limits to having an excessively large number. First of all, for efficient and stable analyses the variables should not duplicate information. For example, the presence of the customers' annual income makes monthly income superfluous. Furthermore, for each statistical unit the data should be correct for all the variables considered. This is difficult when there are many variables, because some data can go missing: missing data causes problems for the analysis.

Once the units and the interest variables in the statistical analysis of the data have been established, each observation is related to a statistical unit, and a distinct value (level) for each variable is assigned. This process is known as classification. In general it leads to two different types of variable: qualitative and quantitative. Qualitative variables are typically expressed as an adjectival phrase, so they are classified into levels, sometimes known as categories. Some examples of qualitative variables are sex, postal code and brand preference. Qualitative data is nominal if it appears in different categories have an order that is either explicit or implicit.

The measurement at a nominal level allows us to establish a relation or equality or inequality between the different levels (=, ≠). Examples of nominal measurements are the eye colour of a person and legal status of a company. Ordinal measurements allow us to establish an order relation between the different categories but they do not allow any significant numeric assertion (or metric) on the difference between categories. More precisely, we can affirm which category is bigger or better but we cannot say by how much (=,>, <). Examples of ordinal measurements are computing skills of a person and the credit rate of a company.

Quantitative variables are linked to intrinsically numerical quantities, such as age and income. It is possible to establish connections and numerical relations among their levels. They can be divided into discrete quantitative variables when they have a finite number of levels, and continuous

quantitative variables if the levels cannot be counted. A discrete quantitative variable is the annual revenues of a company.

Very often the ordinal level of a qualitative variable is marked with a number. This does not transform the qualitative variable into a quantitative variable, so it is not possible to establish connections and relations between the levels themselves.

1.13 Overview of KDD

The term Knowledge Discovery in Databases abbreviated as KDD refers to the broad process of extracts knowledge from huge data, and emphasizes the "high-level" application of particular data mining methods. It is of interest to researchers in machine learning, pattern recognition, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, and data visualization. The unifying goal of the KDD process is to extract knowledge from data in the context of large databases.

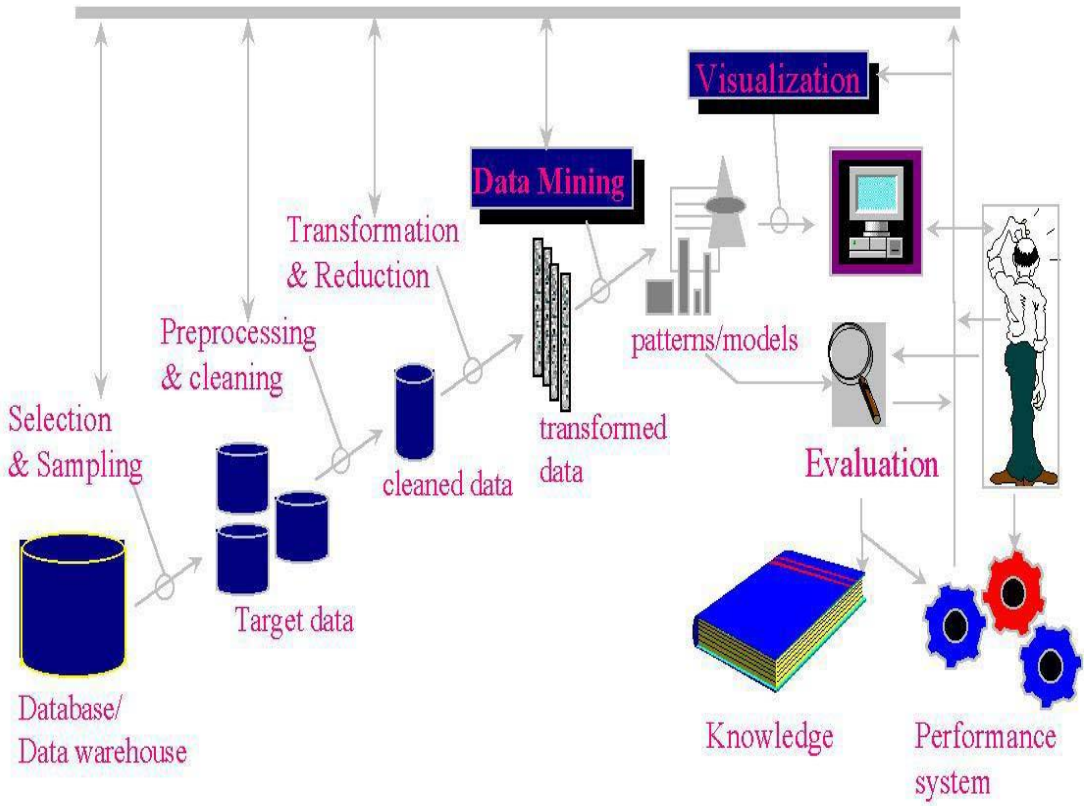


Fig 1.1:KDD

The overall process of finding and interpreting patterns from data involves the repeated application of the following steps:

1. Developing an understanding of
   - the application domain
   - the relevant prior knowledge
   - the goals of the end-user
2. Creating a target data set: selecting a data set, or focusing on subset of variables, or data samples, on which discovery is to be performed.
3. Data cleaning and preprocessing.
   - Removal of noise or outliers.
   - Collecting necessary information to model or account for noise.
   - Strategies for handling missing data fields.
   - Accounting for time sequence information and known changes.
4. Data reduction and projection.
   - Finding useful features to represent the data depending on the goal of the task.
   - Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations or the data.
5. Choosing the data mining task.
   - Deciding whether the goal of the KDD process is classification, regression, clustering etc.
6. Choosing the data mining algorithm(s).
   - Selecting method(s) to be used for searching for patterns in the data.
   - Deciding which models and parameters may be appropriate.

- Matching a particular data mining method with the overall criteria of the KDD process.

7. Data mining.
   - Searching for patterns of interest in a particular representational form or a set of such representations as classification rules or trees, regression, clustering, and so forth.

8. Interpreting mined patterns.

9. Consolidating discovered knowledge

1.15 Reasons of growing data mining research

The amount of digital data has been exploding during the past decade, while the number of scientist, engineers and analysts available to analyze the data has been static. To bridge this gap requires the solution of fundamentally new research problems, which can be grouped into the following broad challenges:

(a) Developing algorithms and systems to mine large, massive and high
(b) Dimensional data sets
(c) Developing algorithms and systems to mine new types of data
(d) Developing algorithms, protocols and other infrastructure to mine distributed data
(e) Improving the ease of use of data mining systems
(f) Developing appropriate privacy and security models for data mining

In order to respond to those challenges, there is requirement of applied, multidisciplinary and interdisciplinary research in data mining and knowledge discovery.

1.16 Data Mining: The Benefits

As a process, data mining keeps our business profitable. As a software application, it empowers to uncover previously undetected facts about customers in your databases.

According to IBM, "Data mining offers firms in many industries the ability to discover hidden patterns in their data patterns that can help them understand customer behavior and market trends. The advent of parallel processing and new software technology enable customers to capitalize on the benefits of data mining more effectively than had been possible previously."

For example, to quote from a paper on the benefits of data mining, "Retailers often wish to know what products typically sell together. Knowing that 72 percent of their customers who buy a certain brand of soda also buy a certain brand of potato chips can help the retailer determine appropriate promotional displays, optimal use of shelf space, and effective sales strategies. As a result of doing this type of affinity analysis, the retailer might decide not to discount the potato chips whenever the soda is on sale, as doing so would needlessly reduce profits. Knowing such information can help a mail-order firm increase sales by narrowing down the scope of a given mailing campaign or perhaps creating more custom mailings that each yield higher success rates."

There are a number of ways that data mining saves money and improves efficiency. It adds the benefits of automation to your existing software and hardware. It can be carried out on new technology eve if your current system is being upgraded. And today's technology can also analyze terabytes in minutes.

These are some of the ways data mining can take customer-service to the ultimate level, from attracting new prospects to re- and cross-selling existing customers to retaining the blue-chip ones. A company can minimize losing customers who are a flight risk by studying customers they have lost. Or a company can target prospects with just the right specials if they know their buying traits. If one out of a hundred consumers who had a similar profile bought product A, then marketing to the remaining 99 is imperative. Chances are many of those people will purchase product A as well.

The Two Crows Corporation queried some of the top CFOs in the country about data mining and made these enterprise-wide recommendations on how data mining can groove with your existing CRM system:

To emphasize the link between data mining and CRM, their report continued, "This is not to say that the acquisition and implementation of CRM technologies and applications cannot be pursued on a non-enterprise, departmental basis. Valuable data can be gained from almost any CRM implementation, especially through the short-term reduction of specific process costs. Overall, however, the cost of piecemeal implementations will likely be greater to the enterprise - and the potential value of the strategy, system and data will be less. As CFOs are charged with optimizing the value of the enterprise, it behooves them to make CRM - and its data - an enterprise imperative."

Here are a few snapshots how data mining can benefit certain industries.

Retail / Marketing

- Identify buying behavior patterns from customers.
- Find associations among customer demographic characteristics.
- Predict with customer will respond to mailing.

Banking

- Detect patterns of fraudulent credit card usage.
- Identify “loyal” customers.
- Predict customers that are likely to change their credit card affiliation.
- Determine credit card spending by customer groups.
- Find hidden correlation between different financial indicators.
- Identify stocks trading rules from historical market data.

Insurance and Health Care

- Claims analysis – determine which medical procedures are claimed together.
- Predict which customers will buy new policies.
- Identify behavior patterns of risky customers.
- Identify fraudulent behavior.

Transportation

- Determine the distribution schedules among outlets.
- Analyze loading patterns.

Medicine

- Characterize patient behavior to predict office visits.
- Identify successful medical therapies for different illnesses.

2. Introduction to Data Warehouse

2.1 Types of Systems

Perhaps the most important concept that has come out of the Data Warehouse movement is the recognition that there are two fundamentally different types of information systems in all organizations: operational systems and informational systems.

"Operational systems" are just what their name implies; they are the systems that help us run the enterprise operation day-to-day. These are the backbone systems of any enterprise, our "order entry', "inventory", "manufacturing", "payroll" and "accounting" systems. Because of their importance to the organization, operational systems were almost always the first parts of the enterprise to be computerized. Over the years, these operational systems have been extended and rewritten, enhanced and maintained to the point that they are completely integrated into the organization. Indeed, most large organizations around the world today couldn't operate without their operational systems and the data that these systems maintain.

On the other hand, there are other functions that go on within the enterprise that have to do with planning, forecasting and managing the organization. These functions are also critical to the survival of the organization, especially in our current fast-paced world. Functions like "marketing, planning", "engineering planning" and "financial analysis" also require information systems to support them. But these functions are different from operational ones, and the types of systems and information required are also different. The knowledge- based functions are informational systems.

"Informational systems" have to do with analyzing data and making decisions, often major decisions, about how the enterprise will operate, now and in the future. And not only do informational systems have a different focus from operational ones, they often have a different scope. Where operational data needs are normally focused upon a single area, informational

data needs often span a number of different areas and need large amounts of related operational data.

In the last few years, Data Warehousing has grown rapidly from a set of related ideas into architecture for data delivery for enterprise end- user computing.

2.2 Difference of Operational and Informational system

	OPERATIONAL	INFORMATIONAL
Data Content	Current Value	Archived, derived, Summarized
Data Structure	Optimized for transactions	Optimized for complex queries
Access Frequency	High	Medium to low
Access Type	Read, update, delete	Read
Usage	Predictable, repetitive	Ad hoc, random, heuristic
Response Time	Sub-seconds	Several seconds to minutes
Users	Large number	Relatively small number

2.3 OLTP and DSS Systems

One of the interesting differences between the operational environment and the data warehouse environment is that of the transaction that is executed in each environment. In the operational environment when a transaction executes, the execution entails very little data. As few as two or three rows of data may be required for the execution of an operational transaction. A really large operational transaction may access up to twenty-five rows of data. But the number of rows that is accessed is modest. It is necessary to keep the row size small in the operational environment if consistent, good online response time is to be maintained.

The transaction profile in the DSS data warehouse environment is very different. The transactions run in the DSS environment may access thousands and even hundreds of thousands of rows of data. Depending on what the DSS analyst is after, the data warehouse transaction may access huge amounts of data.

The response time in the DSS environment is very different from the response time found in the OLTP environment. Depending on what is being done in the DSS data warehouse environment, response time may vary from a few seconds all the way up to several hours.

There is then a marked difference in the transaction profile found in the DSS data warehouse environment and the operational transaction processing environment.

One by-product of this extreme difference in transaction profiles is that the definition of response time differs from one environment to another. In the operational environment, transaction response time is the length of time from the initiation of the transaction until the moment in time when results are FIRST returned to the end user.

In the DSS data warehouse environment, there are two response times. One response time is the length of time from the moment when the transaction is initiated until the first of the results are returned. And the second measurable response time is the length of time from the moment of the initiation of the transaction until the moment when the LAST of the results are returned. The difference between these two variables can be considerable.

Both sets of response time are needed in order to effectively measure system performance in the DSS data warehouse environment.

2.4 Introduction to Data Warehouse

The Data Warehousing is the only viable solution for providing strategic information. The information delivery for the new system environment called the data warehouse. So, followings are the functional definition of the data warehouse.

The data warehouse is an informational environment that

- Provides an integrated and total view of the enterprise.
- Makes the enterprise's current and historical information easily available for decision-making.
- Makes decision-support transaction possible without hindering operational systems
- Renders the organization's information consistent.
- Presents a flexible and interactive source of strategic information.

The data warehouse is : Subject-Oriented, Integrated, Non-Volatile, and Time variant collection of data.

2.4.1 Subject-Oriented

In operational systems, data is stored by individual applications. In the data sets for an order processing applications, data is kept for that particular application. These data sets provides the data for all the functions for entering orders, checking stock, verifying customer's credit, and assigning the order for shipment. But these data sets contain only the data needed for those functions relating to this particular application. Some data sets containing data about individual orders, customers, stock status and detailed transactions, but these are structured around the processing of orders.

In enterprise, data sets are organized around individual applications to support those particular operational systems. These individual data sets a have to provide data for the specific application to perform the specific function efficiently. Therefore, the data sets for each application need to be organized around that specific application.

In the data warehouse, subjects store data, not by applications. If business subject stores the data, what are business subjects? Business subjects differ from enterprise to enterprise. These are the subjects critical for the enterprise. For a manufacturing company, sales, shipments, and inventory are critical business subjects. For a retail store, sale at the check-out counter is a critical subject.

Following figure distinguish between how data is stored in operational system and in the data warehouse. In the operational system shown, data for each application is organized separately by application: order processing, consumer loans, customer billing, accounts receivable, claims processing and saving accounts. For example *claim* is a critical business subject for an insurance company, claims under automobile insurance policies organized in that application. Similarly, claims data for workers compensation insurance is organized in the Workers Comp Insurance application. But in the data warehouse for an insurance company, claims data are organized around the subject of claims and not by individual application of Auto Insurance and Workers Comp.

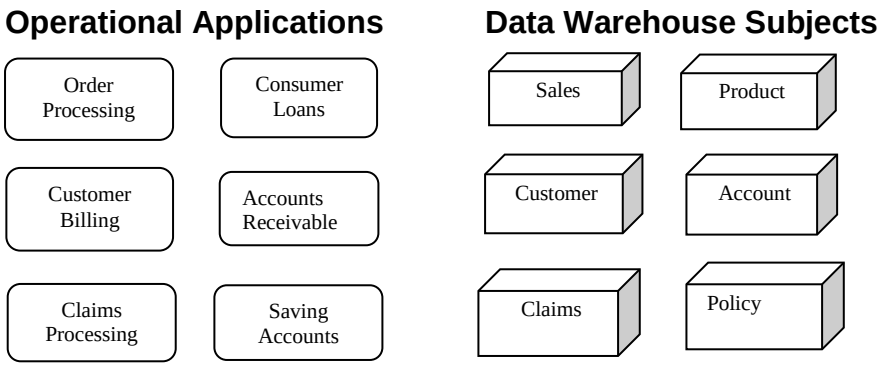


Fig 2.1: The data warehouse is subject oriented

In a data warehouse, there is no application flavor. The data in a data warehouse cut across applications.

2.4.2 Integrated

All the relevant data should pull together from various applications for proper decision making. The data in data warehouse comes from several

operational systems. Source data are in different database files, and data segments. These are disparate applications, so the operational platforms and operating systems could be different. The file layouts, character code representation, and field naming conventions all could be different. This means that there is a single key structure and a single structure of data to be found in the warehouse where there might have been many forms of the same data in the applications. In the data warehouse there is a single structure for customer. There is a single structure for product. There is a single structure for transaction, and so on.

Before the data from various disparate sources can be usefully stored in a data warehouse, the inconsistencies should be removed, various data elements should be standardized and meaning data names in each source application should clear. Before moving the data into the data warehouse, one should go through the process of transformation, consolidation, and integration of the source data.

Following figure illustrates a simple process of data integration for a banking institution. In this example the data fed into the subject area of *account* in the data warehouse comes from three different operational applications. Naming conventions could be different; attributes for data items could be different. The account number in Saving Account Application could be eight bytes long, but only six bytes in the Checking Account application.

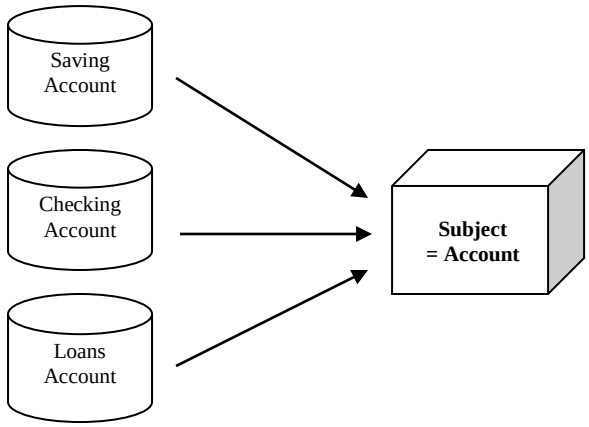


Fig 2.2: Data warehouse is integrated

2.4.3 Time-Variant Data

A data warehouse, because of the very nature of its purpose, has to contain historical data, not just current value. Data is stored as snapshots over past and current periods. Every data structure in the data warehouse contains the time element. Data warehouse contains historical snapshots of the operational data. This aspect of the data warehouse is quite significant for both the design and the implementation phase.

Time variant records are records that are created as of some moment in time. Every record in the data warehouse has some form of time valiancy attached to it. The easiest way to understand time variant records is to contrast time variant records against standard data base records. Consider a standard data base record. With the world changes, so change the values inside the database record. Data is updated, deleted, and inserted inside the standard database record. Now contrast the data warehouse record with the standard database record. Data is loaded into the data warehouse record. The moment when the data is loaded into the warehouse is usually a part of the warehouse record. And data is accessed inside the data warehouse record. But once placed inside the data warehouse, data is not changed there. Data inside the warehouse becomes an environment where the environment can be typified as load and access.

For example, in a data warehouse containing units of sale, the quantity stored in each file record for table row relates to a specific time element. Depending on the level of the details in the data warehouse, the sales quantity in a record may relate to a specific date, week, month, or quarter.

The time-variant nature of the data in a data warehouse, allows for analysis of the past, relates information to the present, enables forecasts for the future.

2.4.4 Nonvolatile Data:

Data extracted from the various operational systems and pertinent data obtained from outside sources are transformed, integrated, and stored in the data warehouse. The data in the data warehouse is not intended to run the day-to-day business. When one want to process the next order received from a

customer, it is not necessary to look into the data warehouse to find the current stock status. The operational order entry application is meant for that purpose. In the data warehouse, it is kept the extracted stock status data as snapshots over time. It is not necessary to update the data warehouse every time a single record is processed.

Data from the operational systems are moved into the data warehouse at specific intervals. Depending on the requirements of the business, these data movements take place twice a day, once a day, once a week, or once in two weeks. In fact, in a typical data warehouse, data movements to different data sets may take place at different frequencies. The changes to the attributes of the products may be moved once a week. Any revision to geographical setup may be moved once a month. The units of sales may be moved once a day. There should be planning and scheduling of data movements or data loads based on the requirements of users.

As illustrated in following figure, every business transaction does not update the data in the data warehouse. The business transactions update the operational system databases in real time. It is added, changed, or deleted data from an operational system as each transaction happens but do not usually update the data in the data warehouse. The data cannot be deleted in the data warehouse in real time. Once the data is captured in the data warehouse, an individual transaction cannot be run to change the data there. Data updates are commonplace in an operational database; not so in a data warehouse. The data in a data warehouse is not as volatile as the data in an operational database is. The data in a data warehouse is primarily for query and analysis.

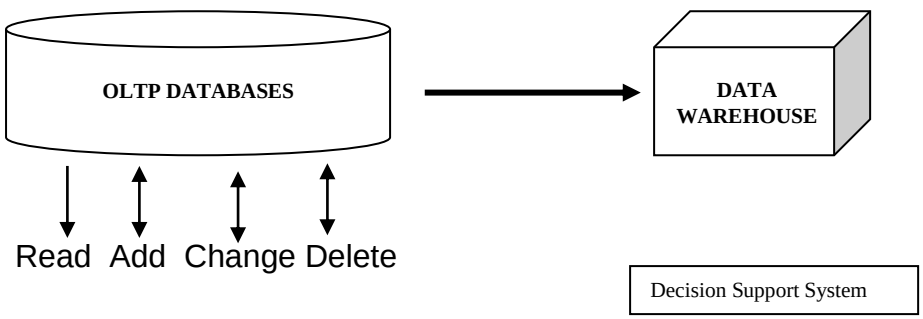


Fig 2.3: OLTP Database to Data Warehouse

2.4.5 Data Granularity

In an operational system, data is usually kept at the lowest level of detail. In a point-of-sale system for a grocery store, the units of sale are captured and stored at the level of units of a product per transaction at the check-out counter. In an order entry system, the quantity ordered is captured and stored at the level of units of a product per order received from the customer. Whenever it is needed summary data, it can be added up the individual transactions. If it is required that how many units of a product is ordered in a month, all the orders entered for the entire month for that product must be read and then add up. Operational system keep summary of data.

Data in the warehouse is granular. This means that data is carried in the data warehouse at the lowest level of granularity. So it can be found summarized data at different levels. Data granularity in a data warehouse refers to the level of detail. The lower level of detail provides the finer the data granularity. Granularity levels can be decided based on the data types and the expected system performance for queries.

2.5 Data Warehouse S/W and H/W Architecture

The architecture of a data warehouse by necessity is complex, and includes many elements. The reason for this is that a data warehouse is an amalgamation of many different systems. Integration of diverse elements is its primary concern, and to accomplish this integration, many different systems and processes are necessary.

Most software development projects require selection of the technical infrastructure, and this is true for the warehouse as well. Basic technical infrastructure includes operating system, hardware platform, database management system, and network. The DBMS selection becomes a little more complicated than a straightforward operational system because of the unusual challenges of the data warehouse, especially in its capability to support very complex queries that cannot be predicted in advance.

How does the data get into the data warehouse? The warehouse requires ongoing processes to feed it; these processes require their own

infrastructure. Many time, IS shops overlook this aspect when they plan for the data warehouse. Data layers need to be understood and planned for. Data cleansing usually involves several steps; where will the "staging area" be stored? How will ongoing data loads, cleansing, and summarizing be accomplished?

Backup and recovery are interesting challenges in the data warehouse, mainly because data warehouses are usually so large.

How will users get information out of the warehouse? The choice of query tool becomes very important, and depends upon a multiplicity of factors.

2.6 Basic steps to develop DW Architecture:

It is very much important to understand and discuss the basic steps to develop data warehouse architecture. Eachand every one of these steps needs to be performed in order to have the best opportunity of succeeding. There are six important steps to develop effective data warehouse architecture developments are as follows:

1. The first and most important step of developing effective data warehouse architecture is to enlist the full support and commitment of project sponsor/executive of the company.

2. Appointed staff in architecture team must be strongly skilled Personnel. It is not necessarily the technology you choose for your architecture, it is the personnel you have designing and developing the architecture that makes the project successful.

3. Prototype/benchmark all the technologies you are interested in using. Design and develop a prototype that can be used to test all of the different technologies that are being considered.

4. The architecture team should be given enough time to build the architecture infrastructure before development begins. For a large organization, this can be anywhere from six months to a year or more.

5. The development staff must be trained on the use of the architecture before development begins. Spend time letting the development team get full exposure to the capabilities and components of the architecture.

6. Provide freedom to the architecture team to enhance and improve the architecture as the project moves forward. No matter how much time is spent up for developing architecture, it will not be perfect the first time around.

2.7 Architectural Components of D/W (Infrastructure):

In data warehouse architecture includes a number of factors. Primarily it includes the integrated data that is the centerpiece. The architecture includes everything that is needed to prepare the data and store it. On the other hand, it also includes All Students the means for delivering information from your data warehouse. The architecture is further composed of the rules, procedures, and functions that enable your data warehouse to work and fulfill the business requirements. Finally, the architecture is made up of the technology that empowers the data warehouse.

The data warehouse consists of the following architectural components, which compose the data warehouse infrastructure:

- System infrastructure: Hardware, software, network, database management system, and personnel components of the infrastructure.
- Metadata layer: Data about data. This includes, but is not limited to, definitions and descriptions of data items and business rules.
- Data discovery: The process of understanding the current environment so it can be integrated into the warehouse.
- Data acquisition: The process of loading data from the various Sources. This is described in more detail in the ongoing maintenance section later in this chapter.

- Data distribution: The dissemination/replication of data to distributed data marts for specific segmented groups.
- User analysis: Includes the infrastructure required to support user queries and analysis. This is described in more detail in the "User Access" section, later in this chapter.

2.8 Data Warehouse System Architecture

The system architecture is the overall blueprint that is to be followed when building data warehouse platform. It is the underlying foundation that governs many of the decisions will be needed to make when building and managing data warehouse platform. Given this, it is no surprise that there are probably as many different data warehouse architectures as there are data warehouses. However, they can usually be grouped into one of two main categories: a three-tiered architecture or a two-tiered architecture.

2.8.1 Three-Tiered Architecture

In a three-tiered architecture, the first tier is comprised of operational systems that are already in place. These are the transaction processing systems that collect the data of all the events that occur within enterprise. The data collected by these systems is fed into your data warehouse. The second and third tiers of this architecture are the data warehouse and the data marts, respectively.

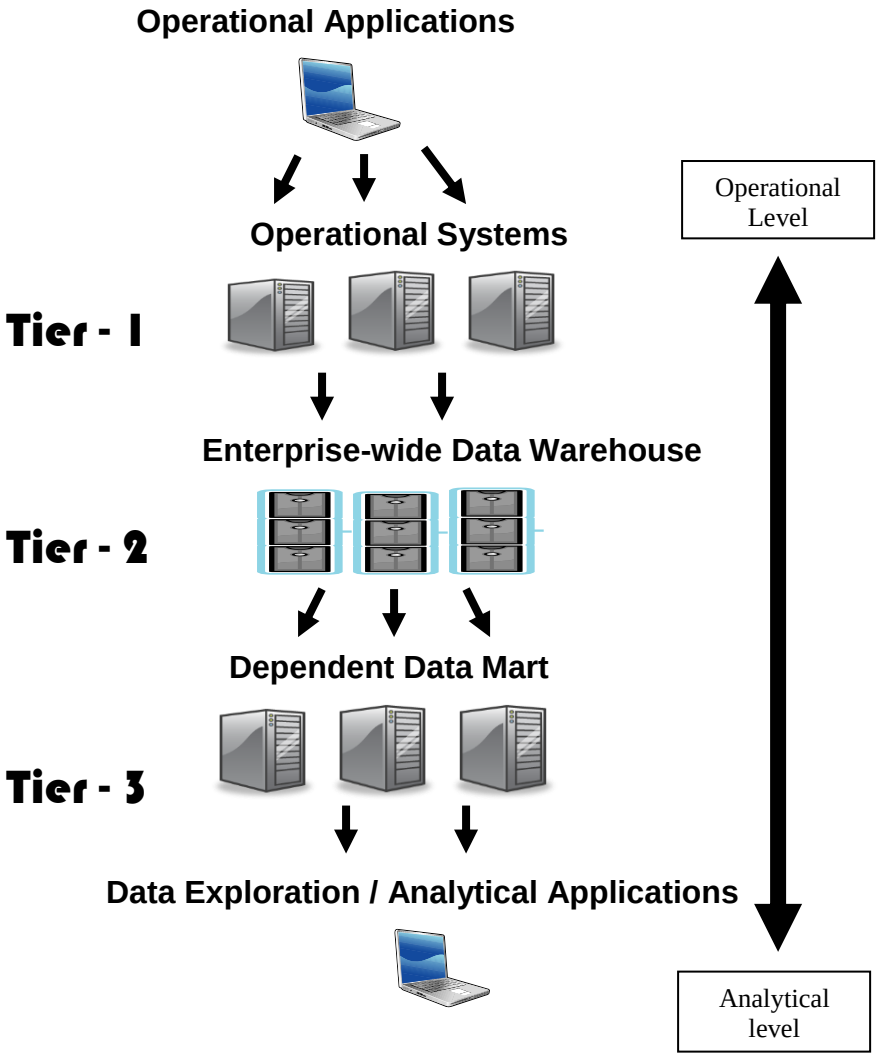


Fig 2.4: Three-Tiered Architecture

There two different types of functions that must be addressed when building a large-scale data warehouse environment that is data consolidation and data analysis.

Data consolidation refers to the process of transforming, extracting, and cleaning the data from disparate and unconsolidated operational systems into one consolidated repository. Data analysis refers the process of end-users access, manipulate, and generally analyze the data looking for useful insights. For these two different needs, it is much more scalable to split these functions into two different tiers. In addition, form of functional parallelism is used here to improve scalability, and different tasks are assigned to different computers. It is very much simple and easier to address different needs if

have one tier focused on optimally solving data consolidation issues and another tier focused on optimally solving data exploration issues.

In three-tier architecture, a tier for the data warehouse is responsible for the consolidation activities and its function is to take data from the various operational systems, consolidate the data, and then feed portions of the consolidated data into the various data marts, and the data marts are responsible for the exploration activities. Since these data marts get their data from the data warehouse, and it is referred as “dependent data marts”, they are dependent on the existence of an enterprise data warehouse.

Data warehouses and data marts can be distinguished as: the data warehouse is fed by multiple operational systems, and it performs the required extractions and transformations. On the other hand, data marts only need to extract data from a single source that is already consolidated in the data warehouse. The data marts also occasionally include additional external data, but the amount of consolidation of data performed by data mart is very much less than in comparison with the data warehouse.

The data warehouse stores its information in a form that is called **application generic**, and it is used to feed multiple data marts, each of which is focused on a different set of business problems. Data warehouse designer, always wants to keep the data stored in the data warehouse tier in its most flexible form, which is the not summarized, means in detail level form. So the design a database schema for the second tier that has much of the flavor of a traditional third-normal form schema. On other hand, data marts need to store their information in a form that is **application specific** and tailored to meet the needs of the explorer or farmer. It means, there is requirement of summarizations, subsets, and/or samples in data mart that are specific to the particular business unit that is using the data mart.

It is also necessary that the data in data marts must be easily accessible by the end-user community. Traditional third-normal form schemas are excel in minimizing data redundancy, and fairly poor models for end users to try to

understand and analyze, and is, therefore, usually a poor choice for data marts. Instead of third-normal form, data marts should use dimensional models and star schema designs, which make heavy use of redundancy to make it far easier for end-users to navigate their way through large volumes of data without getting lost.

The choice of optimal hardware and DBMS may different for the data warehouse and the data marts. The data warehouse tier has to act as a repository or enormous amounts of data that span many different organizations and subject areas. In addition, more data from both existing and new subject areas is constantly being added to the data warehouse. So the data warehouse tier must be able to feed an ever-growing number of application-specific data marts. This means that the data warehouse tier is desired that it must be an industrial-strength, highly scalable, enterprise-class hardware and DBMS.

However, the data marts need only focus on a single business problem. Data mart will see growth in their particular subject area, because all the new transactions related to that subject area are continuously being collected from the operational systems, but the magnitude of this growth is far less. Therefore the data mart tier is usually a smaller but scalable, department level hardware and DBMS solution.

2.8.1.1 Benefits of Three-Tiered Architecture

The major benefits of three-tiered data warehouse architecture are high performance and scalability. The high performance comes from the fact that the inclusion of data marts allows to partition different query workloads across different data marts. It means that the workload levels of users of other data marts will not affect users of one data mart. For instance, if users of the sales data mart are executing very complex and long-running queries that are highly resource intensive, it does not effect on the performance seen by users of the completely separate finance data mart. It results the increase in end-user satisfaction levels is enormous. General experience is that, end users get frustrated by their own queries slowing down their machine, but nothing infuriates end users more than

having someone else in some other department bring a machine that everyone must share to its knees. When the workload is separated into physically separate marts, different sets of users can be prevented from adversely affecting each other.

Three-Tier architecture is also very much scalable. As explained above, extraction and consolidation functions are assigned to one-tier and end-user query and data analysis functions to another tier. This is a straightforward use of the concept of functional parallelism that is described previously as one effective method for improving scalability. Second and third tiers can be individually scaled up as well. The data warehouse tier is a large and highly scalable, and it can be scaled up by adding more resources like processors, disks and I/O controllers to it. Scaling up of the data mart tier can be done by simply adding more data marts to service new user populations, address new subject areas, or focus on providing a new type of functionality such as data mining. But data marts are usually very cheaper to build compare to data warehouses. It is also very easy to add another data mart when it is needed to explore a new business area.

2.8.1.2 Drawbacks of Three-Tiered Architecture

The main drawback is the multi-subject; enterprise-wide data warehouse is always at the center of this architecture. Designing of this data warehouse is a complex process. It is very much time consuming process to consolidate various subject areas, this can involve many long meetings and debates with the representatives from various organizations. So, there is quite a large time investment involved with building an enterprise-wide data warehouse. Defining which business problems you want to solve, finding where all the data required to solve those problems is located, writing all the required extraction, cleansing, and transformation routines, loading all the data into the database, and then tuning the resulting system is no trivial task. In fact, to build average enterprise-wide data warehouse takes about 18-24 months to build. Finally, the cost of such a system is not trivial, often reaching into the many millions of rupees.

The complexity of the project and the required time and cost investments are prohibitive for many organizations. The dynamic, organic nature of a data warehouse environment is that, it doesn't really make sense to build something that large as at very first step. Need of organization may change by the time when data warehouse is to be delivered finally. It is required to spent a large amount of effort building a wonderful system that helps to give answers of the questions, and there are no longer the most important questions that need answering.

2.8.2 Two-Tiered Architecture

Generally, to build a two-tiered architecture, two way are used. The first involves just building the enterprise-wide data warehouse without the data marts, and all the end-users can have direct access of data warehouse. In this architecture, there is no need of separate data mart hardware to store copies of the data because that already exists in the central data warehouse, main benefit is that cost and some amount of time may reduce to build such architecture compare to Three-Tier architecture. Data marts are generally not as complicated to build, so the timesavings would not be dramatic role some time. But there are some limitations with this approach. First, when building of the central data warehouse is started, at that time majority of the complexity, time, cost, and risk are main factors. Second is that, when all departments and all users will be sharing a single database, to separate workloads among different user groups is very much crucial task. With these limitations, it can be concluded that there are not advantages to build such architecture.

Another most common approach is to build Two-Tiered architecture is to build the data marts without building the centralized data warehouse. But these data marts do not depend on the existence of a consolidated data warehouse, so it can be referred as **independent data mart**.

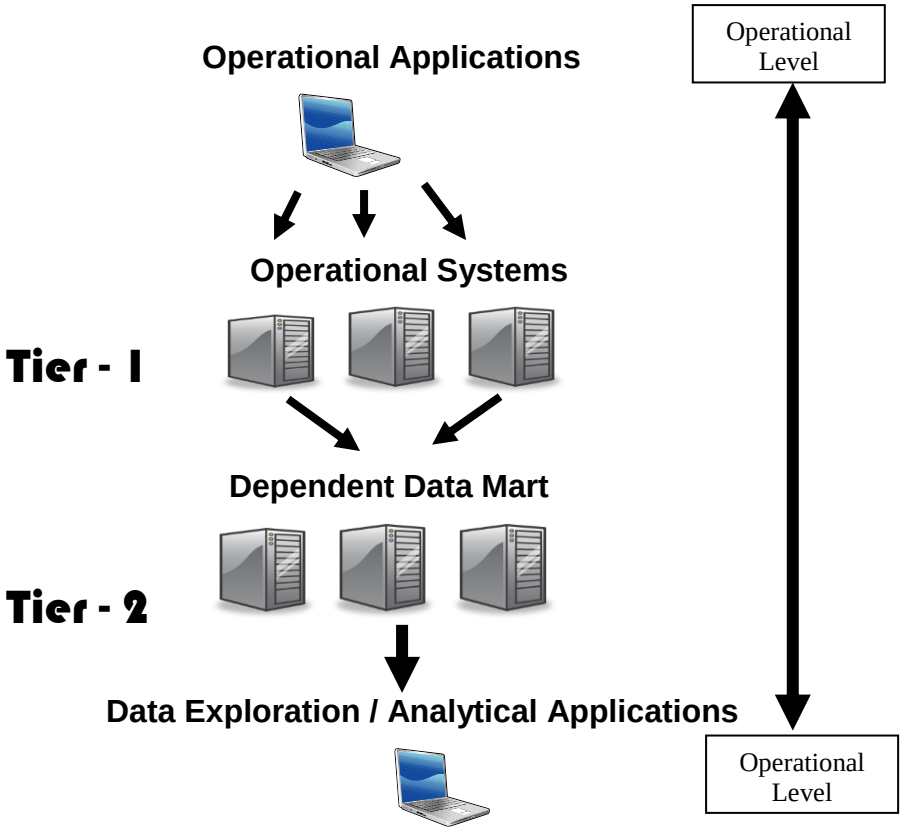


Fig 2.5: Two-Tiered Architecture

But there are some advantages of building a two-tiered environment using the independent data mart approach.

- The data mart traditionally will only have data pertaining to one or two subject areas, so there is much less complexity involved in the design and implementation of this architecture.
- This architecture is dealing with fewer data sources and less data, so amount of time is reduced to build such architecture in comparison of three-tiered architecture.
- As discussed earlier, the hardware required for the data mart, are generally much smaller departmental machines, not enterprise-class machines like in three-tiered architecture.

3. Data Marts

A data mart is a powerful and natural extension of a data warehouse to a specific functional or departmental usage. The data warehouse provides granular data and various data marts interpret and structure the granular data to suit their needs. Data mart is a container or structure in which gets the data from enterprise data warehouse. The data mart is the place, where the end user has the most interaction with the enterprise data warehouse environment. Various data marts are designed after knowing the requirements of the different departments that own the data mart. All the data mart looks different from each other, Because a different department owns each data mart.

The detailed data is found in the enterprise data warehouse, while very little detailed data is found in the data mart, because enterprise data warehouse is the source of data inside the data mart. The data stored in enterprise data warehouse is reshaped according to departmental requirement and then data is sent to the data mart. Data is often summarized and/or otherwise aggregated when it moves into the data mart.

The data stored in the data mart can be described as residing in star joins (star schemas) or snowflake structures. These star joins reflect the different ways that the departments look at their data.

The world of data mart revolves around technology and structures that are designed for end user access and analysis. The leading of these structures is the cube or the multi dimensional structure. Most of the queries are well structured before the query is submitted to the data mart. Many reports emanate from the data mart. The data mart can also spawn smaller desktop versions of the data mart that can be fed and maintained at an individual's workstation. The data mart is reconcilable back to the enterprise data warehouse. The only legitimate source of data for the data mart other than the enterprise data warehouse is external data.

3.1 Data Mart Structure

A star join or a snowflake structure is most suitable for the data structure of the data mart. There are two basic components of a star join structure. One is a fact table and another is supporting dimension tables. A fact table represents the data that is the most populous in the data mart. Stocking data are typically most populous data. In bank, transactions done by ATM are also populous data.

The fact table is a composition of many types of data that have been pre-joined together. The fact table contains:

- A primary key reflecting the entity for which the table has been built, such as an order, a transaction of stock, a transaction through ATM, etc.
- Information about primary key.
- Foreign keys relating the fact table to the dimensions.
- Non-key foreign data that is carried with the foreign key. This non-key foreign data is included if it is regularly used in the analysis of data found in the fact table.

The fact table is highly indexed. In some cases every column in the fact table is indexed. There may be 30 to 40 indexes on the fact table. The highly indexing results that data in a fact table is highly accessible. However, the amount of resources required for the loading of the indexes must be factored into the equation. By rule, fact tables are not updated any way. They have data loaded into them, but once a record is loaded properly, there is no need to go into the record and alter any of its contents.

The dimension tables surround the fact tables. The dimension tables contain the data that are non-populace. The dimension tables are related to the fact tables through a foreign key relationship. Typical dimension tables might be product list, customer list, vendor lists, etc., depending of course on the data mart being represented. Dimension data are the dimensions along which data can be analyzed. In the pet store, sales data can be analyzed by salesperson or by type of pet. Therefore, “salesperson” and “type of pet” are

dimension data. Generally, time is a dimension element in most data warehouses.

Followings are the differences between fact data and dimension data.

Fact Data	Dimension Data
• Millions or billions of rows	• Tens to a few million rows
• Multiple foreign keys	• One primary key
• Numeric	• Textual descriptions
• Does not change	• Frequently modified

The source of the data found in the data mart is the enterprise data warehouse. All data should pass through the enterprise data warehouse before finding its way into the data mart, but there is one exception. The exception is data that is specific only to the data mart that is used nowhere else in the environment. External data often fits this category. If however, the data is used anywhere else in the DSS environment, then it must pass through the enterprise data warehouse.

Generally the data mart contains two kinds of data, which is detailed data and summary data. The detailed data in the data mart is contained in the star join, as previously described. It is noteworthy that the star join may well represent a summarization as it passes out of the enterprise data warehouse. In that sense, the enterprise data warehouse contains the most elemental data while the data mart contains a higher level of granularity. However, from the point of view of the data mart user, the star join data is as detailed as data gets.

The second kind of data that the data mart contains is summary data. It is very common for the users to create summaries from the data found in the star join (detailed data). A typical summary might be monthly sales totals of sales territory. Because summaries are kept on an ongoing basis, history of the data is stored in the data mart. But the preponderance of history that is kept in the data mart is stored at the summary level. Very little history is kept at the star join level.

Whenever it is required, the data mart can be refreshed from the enterprise data warehouse. However, refreshment can be done either much more or much less frequently, depending on the needs of the department that owns the data mart.

3.2 Usage of Data Mart

The data mart is the most versatile of the data structures. The data mart allows the data to be examined from the standpoint of many perspectives - from a detailed perspective, from a summarized perspective, across much data, across few occurrences of data.

The primary user of the data called a farmer. The farmer knows what will happen, when the query is submitted. The farmer does the same activity repeatedly against different occurrences of data. The data is structured inside the data mart so that it is optimal for the access of the farmer. The farmer spends a fair amount of time with the DWA for the gathering and synthesizing requirements before the data mart is built.

The farmer looks at summarized data, at exception based data, at data created on a periodic basis, and other types of data. The farmer also looks upon the data mart as a mission critical component of the environment.

Another use of the data mart is as a spawning ground for insightful analysis by the explorer community. While the data mart is not designed to support exploration, many of the insights, which merit deeper exploration, are initiated in the data mart. The explorer has the inspiration at the data mart, does some cursory exploration there, and then moves down to the explore warehouse for the detailed analysis that is required for exploration. The data mart lacks the foundation for exploration because data is structured along the lines of a department, because data is usually summarized and the explorer needs detail, and because the data mart has a limited amount of historical data. Never the less, the data mart is a fertile breeding ground for insight.

3.3 Security in a Data Mart

When there is secretive information in the data mart it needs to be secured

well. Typically, secretive information includes financial information, medical records and human resource information etc. the data mart administrator should make necessary security arrangements such as: firewalls, log on/off security, application based security, DBMS security, encryption and decryption. The information cost of security depends upon its exclusiveness.

3.4 Data warehouse and Data Mart

Finally Data warehouse and Data Mart can be distinguished considering following factors:

Data Warehouse	Data Mart
• Corporate/Enterprise-wide • Union of all data marts • Data received from staging area • Queries on presentation resource • Structure for corporate view of data	• Departmental • A single business process • Star-join (facts & dimensions) • Technology optimal for data access and analysis • Structure to suit the departmental view of data

4. Online Analytical Processing (OLAP)

Online Analytical Processing (OLAP) systems, contrary to the regular, conventional online transaction processing (OLTP) systems, are capable of analyzing online a large number of past transactions or large number of data records (ranging from mega bytes to giga bytes and tera bytes) and summarize them on the fly. This type of data is usually multidimensional in nature. This multi-dimensionality of the key driver for OLAP technology, which happens to be central to data warehousing.

Multidimensional data may be stored in spreadsheet, cannot be processed by conventional SQL type DBMS. For a complex real-world problem, the data is usually multidimensional in nature. Even though one can manage to put such data in a conventional relational database in normalized tables, the semantics of multidimensionality will be lost and any processing of such data in the conventional SQL will not be capable of handling it effectively. As such, multidimensional query on such a database will explode into a large number of complex SQL statements each of which may involve full table scan, multiple joins, aggregation, sorting and also large temporary table space for storing temporary results.

Finally, the end-result may consume large computing resources in terms of disk space, memory, CPU time, which may not be available and even if they are available the query may take very long time. For example, conventional DBMS may not be able to handle three months' moving average or net present value calculations. These situations call for extensions to ANSI SQL, a near non-feasible requirement.

In addition to response time and other resources, OLAP is a continuously iterative process and preferably interactive one. Drilling down from summary aggregative levels to lower level details may be required to be done on an ad hoc basis by user. Such drilling down may lead the user to detect certain patterns in the data. The user may put forward yet another OLAP query based on these patterns.

This process makes it impossible to handle or tackle for a conventional

database.

4.1 OLTP and OLAP Systems

Conventional OLTP database applications are developed to meet the day-to-day database transactional requirements and operational data retrieval needs of the entire user community. On the other hand, the data warehousing based OLAP tools are developed to meet the information exploration and historical trend analysis requirements of the management or executive user communities. The conventional regular database transactions or OLTP transactions are short, high volume, provide concurrent and online update, insert, delete in addition to retrieval queries and other procedures, processing or reporting. These transactions in batch mode may be ad hoc, online or pre-planned. On the other hand, OLAP transactions are long (infrequent or occasional updates or refreshing the data warehouse), but the more efficient in processing a number of ad hoc queries. Information in a data warehouse frequently comes from different operational source systems (which are usually conventional OLTP database systems) and is interpreted, filtered, wrapped, summarized and organized in an integrated manner, making it more suitable for trend analysis and decision support data retrieval.

Following is the comparison between OLTP and OLAP systems

OLTP	OLAP
• Only current data available (old data) is replaced by current data by updating)	• Both current and historic data available (current is appended to historic data)
• Short transactions (single granularity or more)	• Long database transactions
• Online update/insert/delete transactions	• Batch update/insert/delete transactions.
• High volume of transactions in a given period	• Low volume transactions, periodic refreshing
• Concurrency control and transactions recovery	• No concurrent transactions and therefore no recovery upon failure required
• Largely online ad hoc queries, requiring low level of indexing	• Largely pre-determined queries requiring high level of indexing

A very popular and early approach for achieving analytical processing is ‘star schema’ or ‘collection model’. This approach is based on the common denomination of the user requirements. In this model, a small number of user-referenced or joint data sets are generated from the detailed data sets. This involves de-normalization of the data followed by integration it is shown in figure of three-tiered architecture of data warehouse.

4.2 Types of OLAP

In the OLAP world, there are mainly two different types: Multidimensional OLAP (MOLAP) and Relational OLAP (ROLAP). Hybrid OLAP (HOLAP) refers to technologies that combine MOLAP and ROLAP.

4.2.1 MOLAP

MOLAP is the more traditional way of OLAP analysis. MOLAP-based products organize, navigate and analyze data typically in an aggregated form. They require tight coupling with the applications and type depend upon a multidimensional database (MDDB) system. Efficient implementations store the data in a way similar to the form in which it is utilized by using improved store techniques so as to minimize storage. Many efficient techniques are used as sparse data storage management on disk so as to improve the response time. Applications requiring iterative and comprehensive time series analysis of trends are well suited for MOLAP technology.

Some of the problems faced by users are related to maintaining support to multiple subject areas in an RDBMS. As shown in following figure, some vendors can solve these problems by maintaining access from MOLAP tools to detailed data in RDBMS.

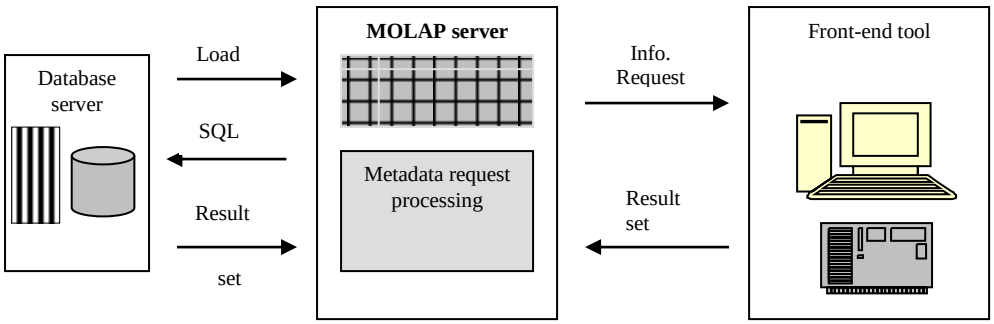


Fig 4.1: MOLAP architecture

This can be very useful for organizations with performance-sensitive multidimensional analysis requirements and that has built or is in the process of building a data warehouse architecture that contains multiple subject areas. An example would be creation of sales data measured by several dimensions (product, sales region) to be stored and maintained in a persistent structure. This structure would be provided to reduce the application overhead performing calculations and building aggregations during application initialization. These structures can be automatically refreshed at predetermined intervals established by an administrator.

4.2.1.1 Advantages

- Excellent performance: MOLAP cubes are built for fast data retrieval, and are optimal for slicing and dicing operations.
- Ability to perform complex calculations: All calculations have been pre-generated when the cube is created. Hence, complex calculations are not only doable, but they return quickly.

4.2.1.2 Disadvantages

- Limitation of handling large amount of data: Because all calculations are performed when the cube is built, it is not possible to include a large amount of data in the cube itself. This is not to say that the data in the cube cannot be derived from a large amount of data. Indeed, this is possible. But in this case, only summary-level information will be included in the cube itself.
- Requirement of additional investment: Cube technologies are often

proprietary and do not already exists in the organization. Therefore, to adopt MOLAP technology, chances are additional investments in human and capital resources are needed.

4.2.2 ROLAP

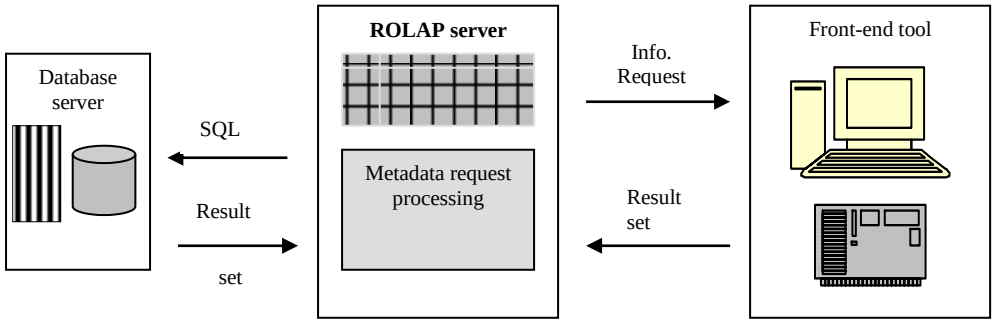


Fig 4.2: ROLAP architecture

This approach enables multiple multidimensional views of two-dimensional relational tables to be created, avoiding structuring data around the desired view. Some products in this segment have supported strong SQL engines to support the complexity of multidimensional analyst. This includes creating multiple SQL statements to handle user requests, being 'RDBMS aware' and also being capable of generating SQL statement based on the optimizer of the DBMS engine. While flexibility is the attractive feature of ROLAP, there exist products, which require the use of de-normalized database designs (as star schema). However, of late there is a noticeable change of realignment in ROLAP technology. Firstly, there is a shift towards pure middle-were technology so as to simplify the development of multidimensional applications. Secondly, the sharp delineation between ROLAP and other approaches as hybrid-OLAP is fast disappearing. Thus vendors of ROLAP tools and RDBMS products are now eager to provide multidimensional persistent structure with facilities to assist in the administrations of these structures.

4.2.2.1 Advantages

- Handling large amounts of data: The data size limitation of ROLAP technology is the limitation on data size of the underlying relational database. In other words, ROLAP itself places no limitation on amount

of data.

- Leverage of functionalities inherent in the relational database: Often, relational database already comes with a host of functionalities. ROLAP technologies, since they sit on top of the relational database, can therefore leverage these functionalities.

4.2.2.2 Disadvantages

- Slow Performance: Because each ROLAP report is essentially a SQL query (or multiple SQL queries) in the relational database, so it is obvious that the query time can be long if the underlying data size is large.

- Limited by SQL functionalities: Because ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs (for example, it is difficult to perform complex calculations using SQL), ROLAP technologies are therefore traditionally limited by what SQL can do. ROLAP vendors have mitigated this risk by building into the tool out-of-the-box complex functions as well as the ability to allow users to define their own functions.

4.2.3 HOLAP

HOLAP technologies attempt to combine the advantages of MOLAP and ROLAP, which are other possible implementation of OLAP. HOLAP allows storing part of the data in the MOLAP store and another part of the data in ROLAP store. The degree of control that cube designer has over this partitioning varies from product to product. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.

Vertical Partitioning: In this mode HOLAP stores aggregations in MOLAP for fast query performance, and detailed data in ROLAP to optimize time of cube processing.

Horizontal Partitioning: In this mode HOLAP store comes slice of data, usually the more recent one (eg. sliced by Time Dimension) in MOLAP for fast query performance, and older data in ROLAP. Moreover, some dices in MOLAP and others in ROLAP can be stored, that leverages the fact that in large cuboids, there will be dense and sparse sub regions.

Popular Tools:

- Micro Analysis Services
- Micro Strategy DSS Web
- Cognos PowerPlay
- BI Accelerator
- SAP AG
- IBI Focus Fusion
- Pilot Software
- Arbor Essbase Web
- Information Advantage Web

5. Extraction, Transformation and Loading (ETL)

ETL is process that involves extracting data from outside sources, transforming it to fit business needs, and loading into the data warehouse.

ETL is important, as it is the way data actually gets loaded into the warehouse. ETL can also be used for the integration with legacy systems.

5.1 Extraction

The first part of an ETL process is to extract the data from the source systems. Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization / format. Common data source formats are relational databases and flat files, but may include non-relational database structures such as IMS or other data structures such as VSAM or ISAM. Extraction converts the data into a format for transformation processing.

5.2 Transformation

The transformation stage applies a series of rules or functions to the extracted data to derive the data to be loaded. Some data sources will require very little manipulation of data. In other cases, one or more of the following transformations types may be required.

- Selecting only certain columns to load or selecting that null columns not to be loaded.
- Translating coded values, if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female, this is called data cleansing.
- Encoding free-form values, it is mapping of “Male” ,“1” and “Mr” into M.
- Deriving new calculated values, like sale_amt = quantity * rate.
- Joining together data from multiple sources, like lookup data and merging of data.

- Summarizing multiple rows of data, means to summarize total sale for each store for each region.
- Generating surrogate key values.
- Transporting or pivoting, it turns multiple columns into multiple rows or vice versa.
- Splitting a column into multiple columns.

Fig 5.1: Extraction and Cleansing of data

5.3 Loading

The load phase loads the data into the data warehouse. Depending on the requirements of the organization, this process ranges widely. Some data warehouses might weekly overwrite existing information with cumulative, updated data, while other data warehouse or part of them might add new data hourly. The timing and scope to replace or append are strategic design choices dependent on the time available and business needs. More complex systems can maintain a history and audit trial of all changes to the data.

ETL processes can be quite complex, and significant operational problems can occur with improperly designed ETL systems.

The range of data values or data quality in an operational system may be outside the expectations of designers at the time validation and transformation rules are specified. Data profiling of a source during data analysis is recommended to identify the data conditions that will need to be managed by

transform rules specifications.

The scalability of an ETL system across the lifetime of its usage needs to be established during analysis. This includes understanding the volumes of data that will have to be processed within Service Level Agreements (SLAs). The time available to extract from source systems may be change, which may mean the same amount of data may have to be processed in less time. Some ETL systems have to scale to process terabytes of data to update data warehouse with tens of terabytes of data. Increasing volumes of data may require designs that can scale from daily batch to intra-day micro-batch to integration with message queues for continuous transformation and update.

A recent development in ETL software is the implementation of parallel processing. This has enabled a number of methods to improve overall performance of ETL processes when dealing with large volumes of data. There are three main types of parallelisms as implemented in ETL applications

5.4 Data

By splitting a single sequential file into smaller data files to provide parallel access.

5.5 Pipeline

Allowing the simultaneous running of several components on the same data stream. An example would be looking up a value on record 1 at the same time as adding together two fields on record 2.

5.6 Component

The simultaneous running of multiple processes on different data streams in the same job. Sorting one input file while performing a duplication on another file would be an example of component parallelism.

All the three types of parallelism are usually combined in a single job. An additional difficulty is making sure the data being uploaded is relatively consistent. Since all have different update cycles, an ETL system may be required to hold back certain data until all sources are synchronized.

Likewise, where a warehouse may have to be reconciled to the contents in a source system or with the general ledger, establishing synchronization and reconciliation point is necessary.

5.6 Popular ETL Tools

- Data Junction
- Essential Data Stage
- Ab Initio
- Informatica

5.7 Meta Data

Metadata (data about data) describes the details about the data in a data warehouse or in a data mart. When structured into a hierarchical arrangement, metadata is more properly called an ontology or schema. Both terms describe, “what exists” for some purpose or to enable some action. For instance, the arrangement of subject heading in a library catalog serves not only as a guide to finding books on a particular subject in the stacks, but also as a guide to what subjects “exist” in the library’s own ontology and how more specialized topics are related to or derived from the more general subject headings.

Metadata is frequently stored in a central location and used to help organizations standardize their data. This information is typically stored in a metadata registry.

Following are the components of metadata for a data warehouse or data mart.

- Description of sources of the data.
- Description of customization that may have taken as the data passes from data warehouse into data mart.
- Descriptive information about data mart, its tables, attributes and relationships, etc.
- Definitions of all types.

The metadata of a data mart is created and updated from the load programs that move data from data warehouse to data mart. The linkages and relationships between metadata of data warehouse and metadata of data mart have to be well established or well understood by the analyst using the metadata.

This description is essential to establish drill down capability between the two environments. With this linkage, the analyst using data mart metadata can easily find the heritage of the data in data warehouse. Further, the DSS analyst also requires understanding how calculations are made and how data is selected for the data mart environment. The metadata pertaining to the individual data mart should also be available to the end-user for effective usage.

Distributed metadata resides at the data mart also. Using distributed metadata in the data mart, the end user can see:

- The metadata that applies to the local data mart
- The metadata that may resides elsewhere but which is relevant to the local data mart
- Locally protected metadata that is not open and available for any other departments
- Metadata residing at architectural entities other than data marts, such as ODS, enterprise data warehouse, etc.

The distributed metadata is very much useful for the data mart user, using this data mart user can look at both technical and business metadata.

5.7.1 Metadata Architecture

There are two basic architectures for metadata in the DSS data warehouse environment. Those architectures are a centralized architecture and a distributed architecture.

The classical metadata architecture is a centralized architecture. In centralized

architecture metadata is stored and managed centrally. The rights of creation and update are invested in a central administrator. The great appeal of the centralized approach is that data can be uniformly defined and used over the enterprise. Once defined centrally, the metadata will have no conflict in its definition.

The centralized approach to metadata architecture has the problem of not accommodating the need for local autonomy of metadata. Local autonomy of metadata refers to the need to create and manage metadata entirely within the confines of a department. There are other difficulties with the centralized approach to metadata as well. Some of these difficulties are:

- Most or all of the metadata in the enterprise must be accommodated and captured before any of the metadata is useful.
- The administrator of the centralized metadata must be taught the business of the department before the metadata becomes useful.
- The centralized metadata infrastructure cannot be built incrementally.

It is very difficult for the centralized metadata to be kept up to date once captured, and so forth.

A variant form of the centralized metadata architecture is the centralized replicated form of architecture. In this replicated form of metadata architecture, the metadata is gathered centrally, as in the case of the classical centralized metadata architecture. But once gathered there, the metadata can be copied out to any other person or environment that requests. Once copied, the person or organization that has requested the metadata can alter or otherwise manipulate the metadata. There are no controls or conditions on the metadata once copied.

Yet another alternative is that of the distributed metadata architecture. In the distributed metadata mode, metadata resides independently at the many different locales in the DSS environment, such as at the data mart environment, the ODS environment, the enterprise data warehouse environment, etc. Metadata resides and is managed locally. This means that

a data mart, for example, creates, updates, and deletes its own metadata. There is local control and ownership of metadata. However, the metadata that is owned and managed locally can be shared. Other corporate entities - other data marts, other ODS, other enterprise data warehouse, and so forth - can access the metadata as it is stored locally. In doing so, careful record is made of who the owner of the metadata is. If an organization is sharing metadata, it cannot alter the metadata that does not belong to it. The preservation of the rights of ownership of metadata is called the system of record across all participating corporate entities. The system of record is the backbone of the distributed metadata environment.

6. Introduction of Data Mining

6.1 Foundation of Data Mining

Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. Data mining takes this evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery. Data mining is ready for application in the business community because it is supported by three technologies that are now sufficiently mature:

- Massive data collection
- Powerful multiprocessor computers
- Data mining algorithms

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.

Large amount of data generated by organizations worldwide is mostly unorganized. If data is organized one can generate/extract meaningful and useful information to convert unorganized data into organized data. Normally the concept of DBMS is implemented though a database in management systems is embedded with a query language popularly known as SQL server. The use of SQL particularly in unorganized large databank is not always adequate to meat the end user requirements. Data mining is the technique of abstracting meaningful information form large and unorganized databanks. It involves the process of performing automated abstraction and generating

predictive information from large databanks. The abstraction of meaningful large databanks can also be known as knowledge discovery. The data mining process uses of varieties of analysis tools to determine the relationship between data and the databank and to use the same to make valid prediction. Data mining techniques are a result of integration of various techniques forms multiple disciplines such as statistic, machine learning, pattern recognition, neural networks, image processing, etc.

In other words it can be described as Competitive advantage requires abilities. Abilities are built through knowledge. Knowledge comes from data. The process of extracting knowledge from data is called Data Mining.

6.2 Data Mining Process

Data mining is an iterative process that typically involves the following phases. The general phases in the data mining process to abstract knowledge are outlined as under:

6.2.1 Problem Definition

This initial phase is for understanding the problem and domain environment in which the problem occurs. At this stage data mining experts, business experts, and domain experts work closely together to define the project objectives and the requirements. Problem definition specifies the limits within which problem need to be solved. The object is clear then it is translated into a data mining problem definition. In the problem definition phase, data mining tools are not yet required.

6.2.2 Data Understanding

The data-understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

6.2.3 Data Preparation

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modeling tools.

6.2.4 Creation of database for data mining

This phase is to create a database where the data to be mined are stored for knowledge acquisition. Creating a database does not require to create a specialized database management system. Even a flat file or a spreadsheet to store data. Data warehouse is also a kind of data storage where large amount of data is stored for data mining. The creation of data mining database consumes about 50% to 90% of the overall data mining process.

6.2.5 Exploring the database

This phase is to select and examine Important data sets of a data mining database in order to determine their feasibility to solve the problem. Exploring the database is a time-consuming process and requires a good user interface and computer system with good processing speed.

6.2.6 Preparation for creating a data mining model

This phase is to select variables to act as predictors. New variables are also built depending upon the existing variables along with defining the range of variables in order to support imprecise information.

6.2.7 Building a data mining model

This phase is to create multiple data mining models and to select the best of these models. Building a data mining model is an interactive process, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. The selected data mining model can be a decision tree, an artificial neural network, or an association rule model. Typically, there are several techniques for the same data mining problem

type. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase is often is selected.

6.2.8 Evaluation of data mining model

At this stage the built model or models that appear to have high quality, from a data analysis perspective. It important to evaluate the accuracy of the selected data mining model. In data mining, the evaluating parameter is data accuracy in order to test working model. This is because the information generated in the simulated environment varies from the external environment. The errors that occur during the evaluation phase needs to be recorded and the cost and time involved in rectifying the error needs to be estimated. External validation is also needs to be performed in order to check whether the selected model performs correctly when provided real world values. A key objective is to determine if there is some important issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

6.2.9 Deployment of the data mining model

This phase is to deploy the built data mining model and evaluate with external working environment. A monitoring system should monitor the working of the model and generate reports about its performance. The information in the report helps to enhance the performance of selected data mining model. The purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that the end-user can use it. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process.

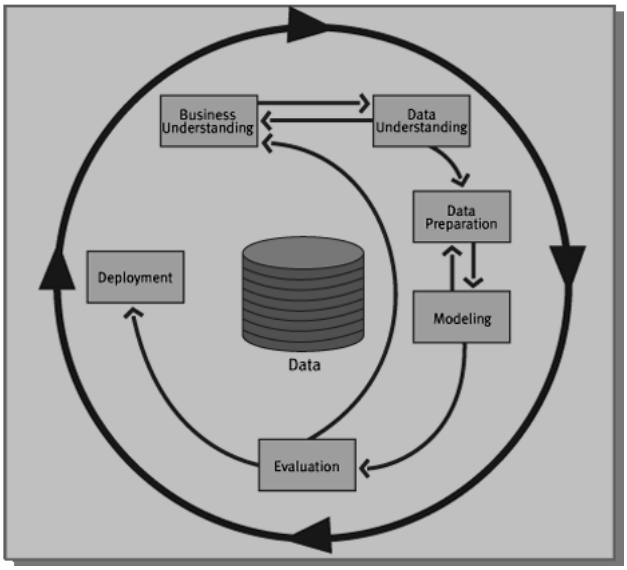


Fig 6.1: Phases of data mining process

6.3 Architecture for Data Mining

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.

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 techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing

systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions.

To best apply these advanced techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. Following figure illustrates architecture for advanced analysis in a large data warehouse.

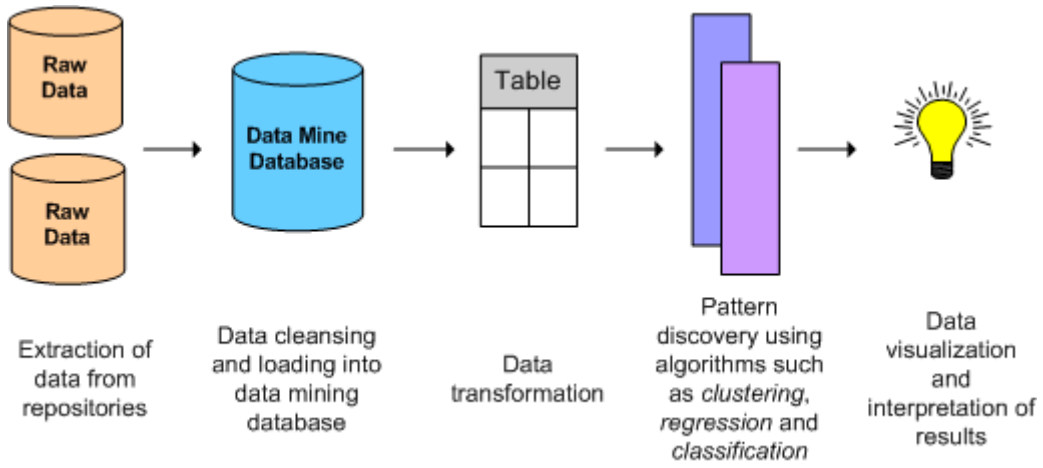


Fig 6.2: Steps taken in Data Mining

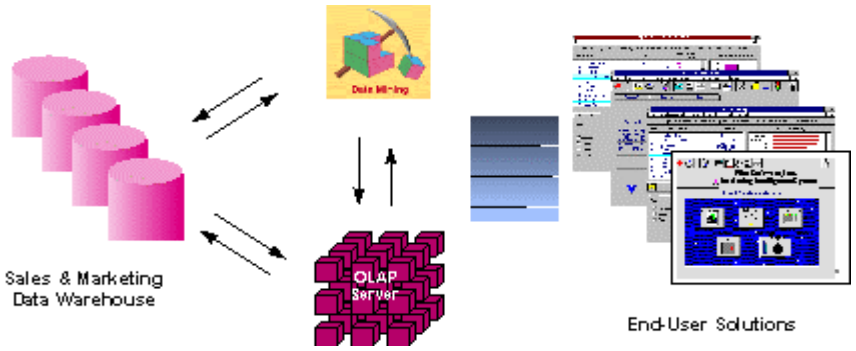


Fig 6.3: Integrated Data Mining Architecture

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access.

An OLAP (On-Line Analytical Processing) server enables a more sophisticated end-user business model to be applied when navigating the data warehouse. The multidimensional structures allow the user to analyze the data as they want to view their business summarizing by product line, region, and other key perspectives of their business. The Data Mining Server must be integrated with the data warehouse and the OLAP server to embed ROI-focused business analysis directly into this infrastructure. An advanced, process-centric metadata template defines the data mining objectives for specific business issues like campaign management, prospecting, and promotion optimization. Integration with the data warehouse enables operational decisions to be directly implemented and tracked. As the warehouse grows with new decisions and results, the organization can continually mine the best practices and apply them to future decisions.

This design represents a fundamental shift from conventional decision support systems. Rather than simply delivering data to the end user

through query and reporting software, the Advanced Analysis Server applies users' business models directly to the warehouse and returns a proactive analysis of the most relevant information. These results enhance the metadata in the OLAP Server by providing a dynamic metadata layer that represents a distilled view of the data. Reporting, visualization, and other analysis tools can then be applied to plan future actions and confirm the impact of those plans.

7. Data Mining Techniques

There are many different techniques used to perform data mining tasks. These techniques not only require specific types of data structure, but also imply certain types of algorithmic approaches. Data mining techniques provide a way to use data mining tasks in order to predict solution sets for a problem and a level of confidence about the predicted solution interns of consistency of prediction and interns of frequency of correct predictions. In this chapter, we briefly introduced some of the common data mining techniques. Data mining techniques include:

1. Statistics
2. Machine learning
3. Decision Trees
4. Neural Networks
5. Genetic Algorithms
6. Association Rules
7. Clustering

7.1 Statistics

There have been many statistical concepts that are the basis for data mining techniques.

7.1.1 Point Estimation

Point estimation refers to the process of estimating a population parameter, θ, by an estimate of the parameter, θ. This can be done to estimate mean, variance, standard deviation, or any other statistical parameter. Often the estimate of the parameter for a general population may be made by actually calculating the parameter value for a population sample. An estimator technique may also be sued to estimate or predict the value of missing data. The *bias* of an estimator is the difference between the expected value of the estimator and the actual value:

$$\text{Bias} = E(\theta) - \theta$$

An *unbiased* estimator is one whose bias is 0. Wile point estimator for small

data sets may actually be unbiased, for larger database applications we could expect that most estimators are biased.

One measure of the effectiveness of an estimate is the *mean square error* (MSE), which is defined as the expected value of the squared difference between the estimate and the actual value:

$$MSE(\theta) = E(\theta - \theta)^2$$

The *square error* is often examined for a specific prediction to measure accuracy rather than to look at the average difference.

7.1.2 Model Based on Summarization

There are many basic concepts that provide an abstraction and summarization of the data as a whole. The basic well-known statistical concepts such as *mean, standard deviation, median* and *mode* are simple models of the underlying population. Fitting a population to a specific *frequency distribution* provides an even better model of the data. Of course, doing this with large databases that have multiple attributes, have complex and/or multimedia attributes, and are constantly changing is not practical.

There are also many well-known techniques to display the structure of the data graphically: For example, *histogram* shows the distribution of the data. A *Box plot* is more sophisticated technique that illustrates several different features of the population at once.

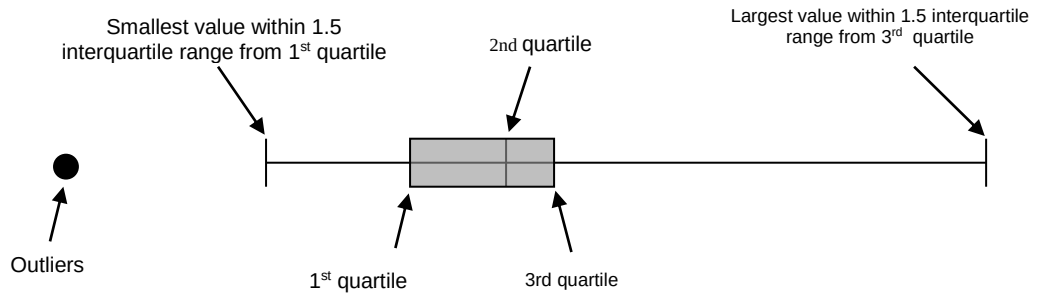


Fig 7.1: Example of Box Plot Technique

Above figure shows a sample box plot. The *total range* of the data values is divided into four equal parts called *quartiles.* The box in the center of the figure shows the range between the first, second, and third quartiles. The line in the box shows the median. The lines extending from either end of the box are the values that are a distance of 1.5 of the interquartile range from the first and third quartiles, respectively. Outliers are shown as points beyond these values.

Another visual technique to display data is called a *scatter diagram*. This is a graph on a two-dimensional axis of points representing the relationships between *x* and *y* values. By plotting the actually observable (*x, y)* points as seen in a sample, a visual image of some derivable may be seen.

7.1.3 Bayes Theorem

With statistical inference, information about a data distribution are inferred by examining data that follow that distribution. Given set of data X = {x_1, x_2, x_3, …, x_n}, a data mining problem is to uncover properties of the distribution from which the set comes. Bayes rule is a technique to estimate the likelihood of a property given the set of data as evidence or input. Suppose that either hypothesis h_1 or hypothesis h_2 must occur, but not both. Also suppose that x_i is an observable event.

In Bayes Rule or Bayes Theorem $P(h_1 / x_i)$ is called the posterior probability, while $P(h_1)$ is the prior probability associated with hypothesis h1. $P(x_i)$ is the

probability of the occurrence of data value x_i and $P(x_1 / h_1)$ is the conditional probability that, given a hypothesis, the tuple satisfies it.

$$P(h_1 / x_i) = \frac{P(x_i / h_1)\, P(h_1)}{P(x_i)}$$

Bayes rule allows us to assign probabilities of hypothesis of given a data value, $P(h_1 / x_i)$.

7.1.4 Hypothesis Testing

Hypothesis testing attempts to find a model that explains the observed data by first creating a hypothesis and then testing the hypothesis against the data. This is in contrast to most data mining approaches, which create the model from the actual data without guessing what it is first. The actual data itself drive the model creation. The hypothesis usually is verified by examining a data sample. If the hypothesis holds for the sample, it is assumed to hold for the population in general. Given a population, the initial or assumed hypothesis to be tested, $\mathbf{H_0}$ is called the *null hypothesis*. Rejection of the null hypothesis causes another hypothesis, $\mathbf{H_1}$ called the *alternative hypothesis,* to be made.

One technique to perform hypothesis testing is based on the use of the chi-squared statistic. Actually, there is a set of procedures referred to as chi-squared. These procedures can be used to test the association between two observed variable values and to determine if a set of observed variable values is statistically significant. A hypothesis is first made, and then the observed values are compared based on this hypothesis. Assuming that ***O*** represents the observed data and ***E*** is the expected values based on the hypothesis, the *chi-squared statistic* is defined as:

$$\textit{Chi-squared statistic} = \frac{\sum (O - E)^2}{E}$$

When comparing a se of observed variable values to determine statistical

significance, the values are compared to those of the expected case. This may be the uniform distribution. We could look at the ration of the difference of each observed score from the expected value over the expected value. However, since the sum of these score will always be 0, this approach cannot be used to compare different samples to determine how they differ from the expected values. The solution to this is the same as we saw with mean square error – 0. Statistical tables allow the actual value to be evaluated to determine its significance.

7.1.5 Regression and Correlation

Both *bivariate regression and correlation* can be used to evaluate the strength of a relationship between two variables. Regression is generally used to predict future values based on past values by fitting a set of points to a curve. Correlation, however, is used to examine the degree to which the values for two variables behave similarly.

Linear regression assumes that a linear relationship exists between the input data and the output data. The common formula for a linear relationship is used in the model:

$$y = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + \ldots + c_nx_n$$

Here there are n input variables, which are called *predictor or regressions;* one output variable (the variable being predicted), which is called the *response;* and n+1 constants, which are chosen during the modeling process to match the input samples. This is sometimes called *multiple linear regression* because there is more than one predictor.

Two different data variables X and Y, may behave very similarly, *Correlation* is the problem of determining how much alike the two variables actually are. One standard formula to measure linear correlation is the *correlation coefficient* ‘r’. Given two variables X and Y, the correlation coefficient is a real value r Є [-1, 1]. A positive number indicates a positive correlation, whereas a negative number indicates a negative correlation. Here negative correlation indicates that one variable increases while the other decreases in value. The closer the value of r to 0, the smaller the correlation. A prefect relationship

exists with a value 1 or -1, whereas no correlation exists with a value of 0. When looking at a scatter plot of the two variables, the closer the values are to a straight line, the closer the r-value is to 1 or -1. The value for r is defined as

$$r = \frac{\sum (x_i - \overline{X})\,(y_i - \overline{Y})}{\sqrt{\sum (x_i - \overline{X})^2\,(y_i - \overline{Y})^2}}$$

Where X and Y are the means of X and Y, respectively. When two data variables have a strong correlation, they are similar. Thus, the correlation can be used to define similarity for clustering or classification.

7.2 Machine Learning

Machine Learning is a process capable of independently acquiring data and integrating that data to generate useful knowledge. The concept of machine learning is implemented by way of computing software system that act as human being who learns from experience, analyses the observations made and self-improves providing increased efficiency and effectiveness.

The processes by which most modern predictive models are developed are *adaptive,* using numerical methods that adjust model parameters based upon analysis of data and model performance. The machine learning process generates a predictive model through mechanical analysis of data and introspection; model parameters are determined without human involvement.

Machine learning is the area of Artificial Intelligence (AI) that examines how to write programs that can learn. In data mining, machine learning is often used for prediction or classification. With machine learning, the computer makes a prediction and then, based on feedback as to whether it is correct, “learns” from this feedback. It learns through examples, domain knowledge, and feedback. When a similar situation arises in the future, this feedback is used to make the same prediction or to make a completely different prediction. Statistics are very important in machine learning programs

because the results of the predictions must be statistically significant and must perform better than a naive prediction. Applications that typically use machine learning techniques include speech recognition, training moving robots, classification of astronomical structures, and game playing.

When machine learning is applied to data mining tasks, a model is used to represent the data (such as a graphical structure like a neural network or a decision tree). During the learning process, a sample of the database is used to train the system to properly perform the desired task. Then the system is applied to the general database to actually perform the task. This predictive modeling approach is divided into two phases. During the training phase, historical or sampled data are used to create a model that represents those data. it is assumed that this model is representative not only for this sample data, but also for the database as a whole and for future data as well. The testing phase then applies this model to the remaining and future data.

7.3 Decision Trees

Predicting future outcomes and identifying factors that can produce a desired effect are often the main goals of data analysis and data mining. Decision trees are one of the most popular methods of predictive modeling for data mining purposes because they provide interpretable rules and logic statements that enable more intelligent decision-making. Decision tree is a tree-shaped structure, which represents a predictive model used in classification, clustering, and prediction tasks. Decision trees use a “divide and conquer” technique to split the problem search space into subsets. In data mining and machine learning, a decision tree is a predictive model that is a mapping from observations about an item to conclusions about its target value. More descriptive names for such tree models are classification tree. In a decision tree, each branch of the tree represents a classification question while the leaves of the tree represents the partition of the classified information. Decision trees are generally suitable for tasks related to clustering and classification. It helps generate rules that can be used to explain the decision being taken. Decision Trees are excellent tools for helping to choose between several courses of action. They provide a highly

effective structure within which it can be laid out options and investigate the possible outcomes of choosing those options. They also help you to form a balanced picture of the risks and rewards associated with each possible course of action.

7.3.1 Decision Tree Representation

A decision tree is an arrangement of tests that prescribes an appropriate test at every step in an analysis. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

More specifically, decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

An instance is classified by starting at the root node of the decision tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is then repeated at the node on this branch and so on until a leaf node is reached.

Diagram

- Each non-leaf node is connected to a test that splits its set of possible answers into subsets corresponding to different test results.
- Each branch carries a particular test result's subset to another node.
- Each node is connected to a set of possible answers.

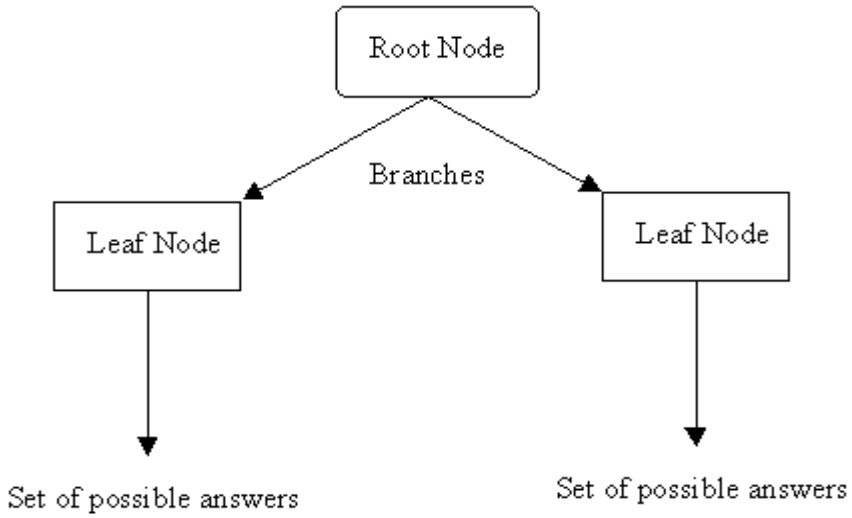
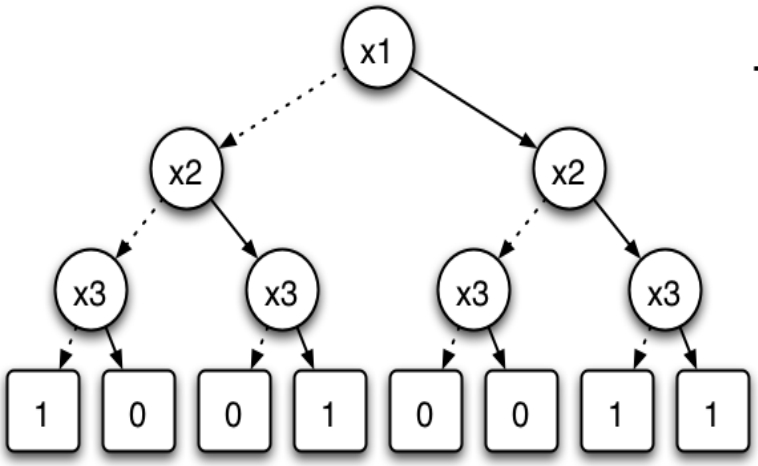


Fig 7.2: Decision Tree Representation

7.3.2 Binary Decision Tree

The following figure shows a binary decision tree and a truth table, each representing the function (x1, x2, x3, x3, x4). In the tree on the left, the value of the function can be determined for a given variable assignment by following a path down the graph to a terminal. In the figure below, a dotted (solid) line represents an edge to a low (high) child. Therefore, to find (x1=0, x2=1, x3=1), begin at x1, traverse down the dotted line to x2 (since x1 has an assignment to 0), then down two solid lines (since x2 and x3 each have an assignment to one). This leads to the terminal,1 which is the value of f(x1=0, x2=1, x3=1).

The binary decision tree of the left figure can be transformed into a binary decision diagram by maximally reducing it according to the two reduction rules. The resulting binary decision diagram is shown in the right figure.



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Binary decision tree and truth table for the function f(x1, x2, x3) = -x1 * -x2 * -x3 + x1 * x2 + x2 * x3

Fig 7.3: Binary Decision Tree with Truth Table

7.3.3 Solution of problem using Decision Tree

Following example of golf club is illustrating the decision tree. The manager of golf club having some trouble with his customer attendance. There are days when everyone wants to play golf and the staff is overworked. On other days, for no apparent reason, no one plays golf and staff has too much slack time. Manager's objective is to optimize staff availability by trying to predict when people will play golf. To accomplish that he needs to understand the reason people decide to play and if there is any explanation for that. He assumes that weather must be an important underlying factor, so he decides to use the weather forecast for the upcoming week. So during two weeks he has been recording following factors.

- The outlook, whether it was sunny, overcast or raining.
- The temperature (in Fahrenheit) .
- The relative humidity in percent.
- Whether it was windy or not
- Whether people attended the golf club on that day.

The dataset is to be compiled into a table containing 14 rows and 5 columns as shown below.

Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

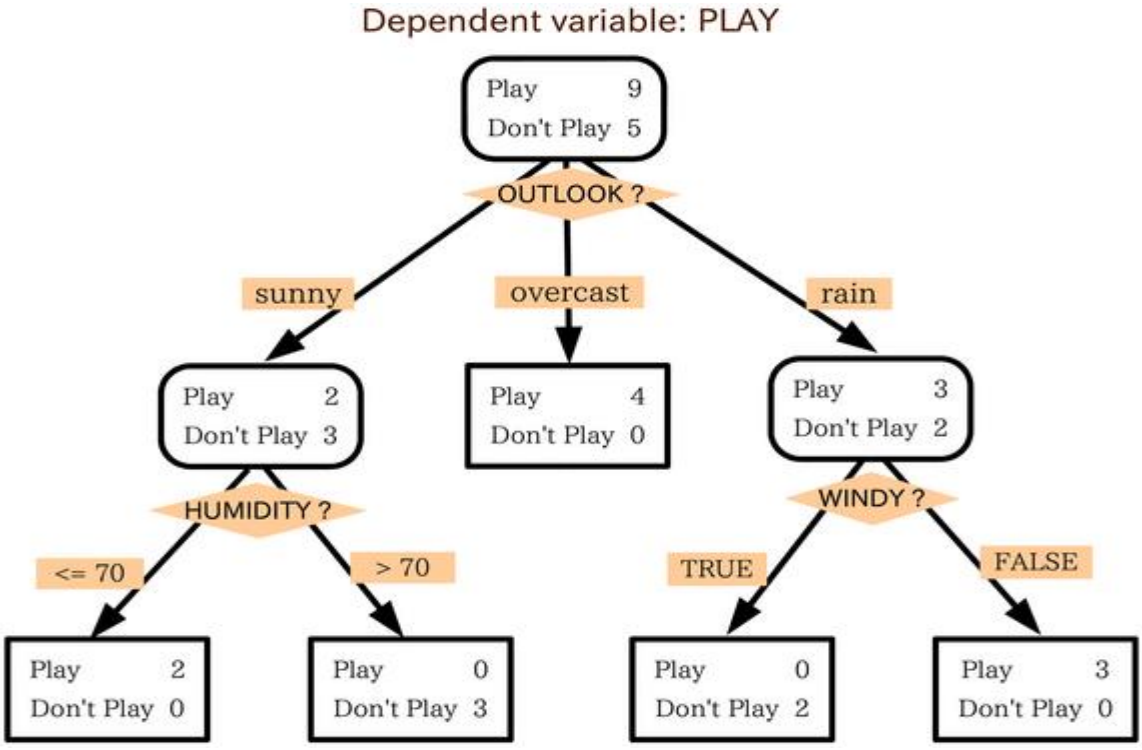


Fig 7.4: Example of Decision Tree

A decision tree is a model of the data that encode the distribution of the class label in terms of the predictor attributes. It is a directed acyclic graph in form

of a tree. The top node represents all the data. the classification tree algorithm concludes that the best way to explain the dependent variable, play, is by using the variable “outlook”. Using the categories of the variable outlook, three different groups were found.

- One that plays golf when the whether is sunny.
- One that plays when the weather is clouded.
- One that plays when it’s raining.

So it can be concluded that, when the outlook is overcast, people always play golf, and there are some fanatical who plays golf even in the rain. Then sunny group should be divided into to groups. It is also seen from the decision tree that the people do not like to play golf when humidity is higher than seventy percent.

Rain category can also be divided into two and it is founded that people will also not play golf, if it is windy.

Lastly, short solution of this problem given by the classification tree is that, most of staff should be dismissed on days that are sunny and humid or on rainy days that ware windy, because almost no one is going to play golf on those days. On days when lot of people will play golf, more staff can be hired.

Following are the decision tree algorithms.

- CART
- ID3
- C4.5
- CHAID
- Rainforest
- Approximate Methods
- CLOUDS

7.4 Neural Networks

Neural networks is non-predictive mode, often referred to as *artificial neural networks* to distinguish them from biological neural networks, are modeled after the working of the human brain. The *neural networks* are actually an

information processing system that consists of a graph representing the processing system as well as various algorithms that access that graph. As with the human brain, the *neural networks* consists of many connected processing elements. The *neural networks*, then, is structured as a directed graph with many nodes (processing elements) and arcs (interconnections) between them. The nodes in the graph are like individual neurons, while the arcs are their interconnections. Each of these processing elements functions independently from the other and uses only local data (input and output to the node) to direct it's processing. This feature facilitates the use of *neural networks* in a distributed and/or parallel environment.

The *neural networks* approach, like decision trees, requires that a graphical structure to built to represent the model and then that the structure be applied to the data. The *neural networks* can be viewed as a directed graph with source (*input*), sink (*output*), and internal (*hidden*) nodes. The input nodes exist in an *input layer*, while the output nodes exist in an *output layer*. The hidden nodes exist over one or more *hidden* layer.

A typical feedforward network has neurons arranged in a distinct layered topology. The input layer is not really neural at all: these units simply serve to introduce the values of the input variables. The hidden and output layer neurons are each connected to all of the units in the preceding layer. Again, it is possible to define networks that are partially-connected to only some units in the preceding layer; however, for most applications fully-connected networks are better.

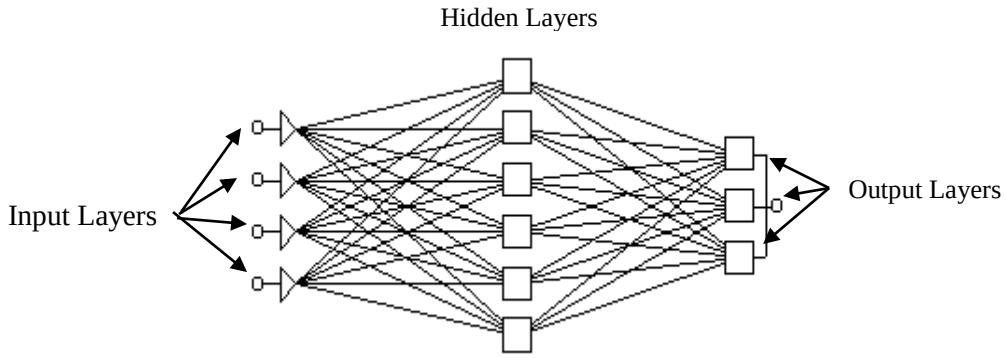


Fig 7.5: Representation of Neural Network

When the network is executed (used), the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer, and subtracting the threshold. The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network.

To perform the mining task, a tuple is input through the input nodes and the output node determines what the prediction is. Unlike decision trees, which have only input node (the root of the tree), the *neural networks* has one input node for each attribute value to be examined to solve the data mining function. Unlike decision trees, after a tuple is processed, the *neural networks* may be changed to improve future performance. Although the structure of the graph does not change, the labeling of the edges may change.

In addition to solving complex problems, *neural networks* can “learn” from prior applications. That is, if a poor solution to the problem is made, the network is modified to produce a better solution to this problem the next time. The major drawback to the use of *neural networks* is the fact that they are difficult to explain to the end users (unlike decision trees, which are easy to understand). Also, unlike decision trees, *neural networks* usually work only with numeric data.

A Neural network (NN) model is a computational model consisting of three parts:

1. Neural network graph that defines the data structure of the neural network.
2. Learning algorithm that indicates how learning takes place.
3. Recall techniques that determine how information is obtained from the network.

Neural network have been used in pattern recognition, speech recognition and synthesis, medical applications (diagnosis, drug design), fault detection,

problem diagnosis, robot control and computer vision. Although *Neural networks* can solve problems that seem more elusive to other AI techniques, they have a long training time (time during which the learning takes place) and thus are not appropriate for real-time applications. *Neural networks* may contain many processing elements and thus can be used in massively parallel systems.

Artificial neural networks can be classified based on the type of connectivity and learning. The basic type of connectivity is *feedforward*, where connections are only to layers later in the structure. Alternatively, a *neural network* may be *feedback* where some links are back to earlier layers. Learning can be either supervised or unsupervised.

7.4.1 Single Layer Linear Network

Following figure shows a sample node, *i,* in a neural network. Here there are *k* input arcs coming from nodes 1,2,3, …, *k*. with weights of $w_1, w_2, w_3, \ldots, w_{ki}$ and input values of $x_{1i}, x_{2i}, x_{ki}, \ldots, x_{ki}$. The values that flow on these arcs are shown on dashed arcs because they do not really exist as part of the graph itself. There is one output value y_i produced. During propagation this value is output on all output arcs of the node. The activation function, f_i, is applied to the inputs, which are scaled by applying the corresponding weights. The weights in the *Neural network* may be determined in two ways. In simple cases where much is known about the problem, the weights may be predetermined by a domain expert. The more common approach is to have them determined via a learning process.

The structure of the *Neural network* may also be viewed from the perspective of matrices. Input and weight on the arcs into node *i* are

$[\, x_{1i},\, x_{2i},\, x_{ki},\, \ldots,\, x_{ki}\,]^T$, $[w_{1i}, w_{2i}, w_{3i}, \ldots, w_{ki}\,]$

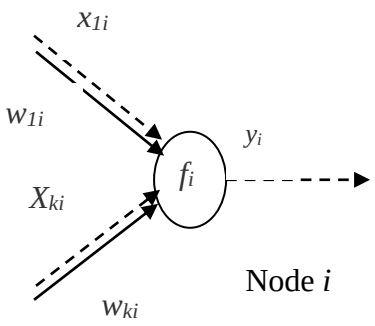


Fig 7.6: Node *i*

There is one output value from node *i*, y_i, which is propagated to all output arcs during the propagation process. Using summation to combine the inputs, then, the output of a node is

$$y_i = fi\left(\sum_{J=1}^{k} w_{ji}\, x_{ji}\right) = f_i\left([w_{1i}, w_{2i}, w_{3i}, \ldots, w_{ki}\,]\begin{pmatrix} X_{i1} \\ \cdots \\ x_{ik} \end{pmatrix}\right)$$

Here f_i, is the activation function. Because *neural networks* are complicated, domain experts and data mining experts are often advised to assist in their use. This in turn complicates the process.

Overfitting occurs when the *neural network* is trained to fit one set of data almost exactly. The error that occurs with the given training data is quite small. However, when new data are examined, the error is very large. In effect, the *neural network* has “memorized” the training set, and cannot generalize to more data. Larger and more complicated *neural network* can be trained to represent more complex functions. To avoid overfitting, smaller *neural networks* are advisable. However, this is difficult to determine beforehand. Another approach that can be used to avoid overfitting is to stop the learning process early.

7.5 Genetic Algorithm

Genetic algorithms are example of *evolutionary computing* methods and are optimization-type algorithms. Given a population of potential problem solution,

evolutionary computing expands this population with new and potentially better solutions. The basis for evolutionary computing algorithms is biological evolutions, where over time evolution produces the best or “fittest” individuals. Chromosomes, which are DNA strings, provide the abstract model for a living organism. Subsections of the chromosomes, which are called *genes*, are used to define different traits of the individual. During reproduction, genes from the parents are combined to produce the genes for the child.

When using genetic algorithms to solve a problem, the first thing, and perhaps the most difficult task, that must be determined is how to model the problem as a set of individuals. In real world, individuals may be identified by a complete encoding of the DNA structure. An individual typically is viewed as an array or tuple of values. Based on the recombination (crossover) algorithms, the values are usually numeric and may be binary strings. These individuals are like a DNA encoding structure for each individual represents an encoding of the major feature needed to model the problem. Each individual in the population is represented as a string of characters from the given alphabet.

A genetic algorithm (GA) is a computational model consisting of following parts:

1. Starting set of individuals, *P*.
2. Crossover technique
3. Mutation algorithm
4. Fitness function

7.5.1 Starting set of individuals, *P*

Given an alphabet A, an **individual** or **chromosome** is a string $I = I_1, I_2, I_3, \ldots I_n$ where $I_j \in A$. Each character in the string, I_j, is called a **gene**. The values that each character can have are called **alleses**. A **population**, is a set of individuals.

Although individuals are often represented as bit strings, any encoding is possible. An array with non-binary characters could be used, as could more

complicated data structures including trees and arrays. The only real restriction is that the genetic operators (mutation, corssover) must be defined.

7.5.2 Crossover technique

In genetic algorithms, reproduction is defined by precise algorithms that indicate how to combine the given set of individuals to produce new ones. These are called *crossover* algorithms. Given two individuals (*parents*) from the population, the crossover technique generates new individuals (*offspring* or *children)* by switching subsequences of the strings. Following figure illustrates the process of crossover. The locations indicating the crossover points are shown in the figure with the vertical lines.

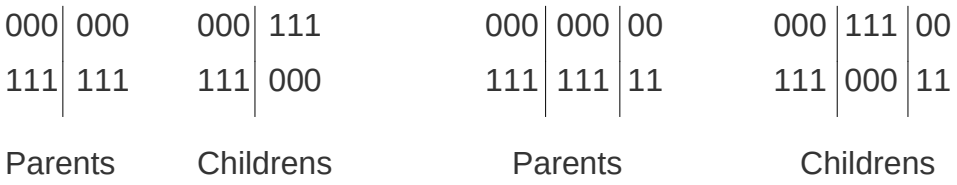


Fig 7.7: (a) Single crossover (b) Multiple crossover

In figure 7.7(a) crossover is achieved by interchanging the last three bits of the two strings. In figure 7.7(b) the center three bits are interchanged. The figure shows single and multiple crossover points. There are many variations of the crossover approach, including determining crossover points randomly. A crossover probability is used to determine how many new offspring are created via crossover. In addition, the actual crossover point may vary within one algorithm.

7.5.3 Mutation function

As in nature, however, mutations sometimes appear, and these also may be present in genetic algorithms. The mutation operation randomly changes characters in the offspring. A very small probability of mutation is set to determine whether a character should change.

Since genetic algorithms attempt to model nature, only the strong survive. When new individuals are created, a choice must be made about which individuals will survive. This may be the new individuals, the old ones, or more likely a combination of the two. The third major component of genetic

algorithms, then, is the part that determines the best (or fittest) individuals to survive.

7.5.4 Fitness function

One of the most important components of a genetic algorithm is determining how to select individuals. A fitness function, *f*, is used to determine the best individuals in a population. This is then used in the selection process to choose parents. Given an objective by which the population can be measured, the fitness function indicates how well the goodness objective is being met by an individual.

Given a population, *P*, a **fitness function**, *f* is a mapping $f : P \rightarrow R$. The simplest selection process is to select individuals based on their fitness.

$$pI_i = \frac{f(I_i)}{\sum f(I_j)}$$

Here pI_i is the probability of selecting individual I_i. This type of selection is called *roulette wheel selection*. One problem with this approach is that it is still possible to select individuals with a very low fitness value. In addition, when the distribution is quite skewed with a small number of extremely fit individuals, these individuals may be chosen repeatedly. In addition, as the search continues, the population becomes less diverse so that the selection process has little effect.

7.6 Association Rules

Association rule discovery techniques are generally applied to databases of transactions where each transaction consists of a set of items. In such a framework the problem is to discover all associations and correlations among data items where the presence of one set of items in a transaction implies (with a certain degree of confidence) the presence of other items. Association rules provide information of this type in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

In addition to the antecedent (the "if" part) and the consequent (the "then" part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets) that are disjoint (do not have any items in common).

The first number is called the **support** for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

The other number is known as the confidence of the rule. **Confidence** is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

For example, if a supermarket database has 100,000 point-of-sale transactions, out of which 2,000 include both items A and B and 800 of these include item C, the association rule "If A and B are purchased then C is purchased on the same trip" has a support of 800 transactions (alternatively 0.8% = 800/100,000) and a confidence of 40% (=800/2,000). One way to think of support is that it is the probability that a randomly selected transaction from the database will contain all items in the antecedent and the consequent, whereas the confidence is the conditional probability that a randomly selected transaction will include all the items in the consequent given that the transaction includes all the items in the antecedent.

Lift is one more parameter of interest in the association analysis. Lift is nothing but the ratio of Confidence to Expected Confidence. Expected Confidence in this case means, using the above example, "confidence, if buying A and B does not enhance the probability of buying C." It is the number of transactions that include the consequent divided by the total number of transactions. Suppose the number of total number of transactions for C is 5,000. Thus Expected Confidence is 5,000/ 1,00,000 = 5%. For our supermarket example the Lift = Confidence/Expected Confidence = 40%/5% =

8. Hence Lift is a value that gives us information about the increase in probability of the "then" (consequent) given the "if" (antecedent) part.

7.6.1 Basic Algorithms for Association Rules

7.6.1.1 Apriori Algorithm:

The Apriori algorithm is the most well known association rule algorithm and is used in most commercial products. It uses the following property, which we call the *large itemset property*: Any subset of a large itemset must be large.
The large itemsets are also said to be *downward closed* because if an itemset satisfies the minimum support requirements, so do All Students of its subsets. Looking at the contra positive of this, if we know that an itemset is small, we need not generate any superset of it as candidates because they also must be small. We use the lattice shown in figure (a) to illustrate the concept of this property. In this case there are four items {A, B, C, D}. The lines in the lattice represent the subset relationship, so the large itemset property says that any set in a path above an itemset must be large if the original itemset is large. In figure (b) the nonempty subsets of ACD are seen as {AC, AD, CD, A, C, D}.

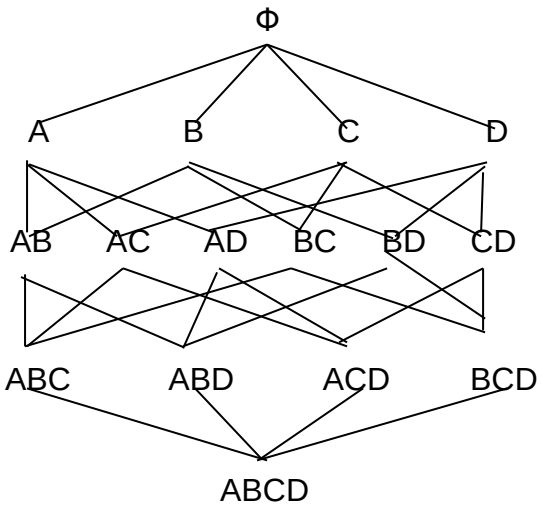


Fig 7.8(a): Lattice of itemsets {A, B, C, D}

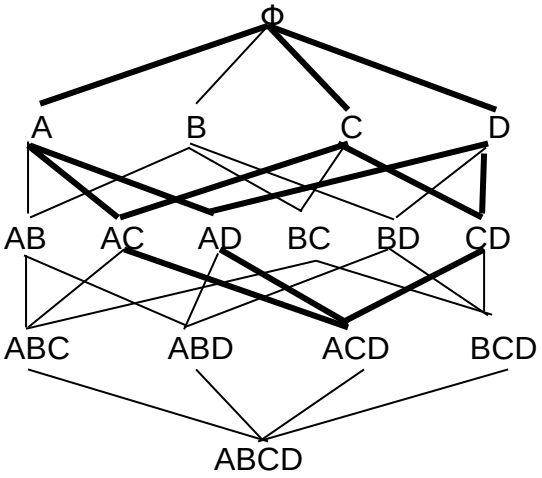


Fig 7.9 (b): Subsets of ACD

The basic idea of the Apriori algorithm is to generate candidate itemsets of a particular size and then scan the database to count these to see if they are large. During scan *i*, candidates of size *i*, C_i are counted. Only those candidates that are large are used to generate candidates for the next pass. That is L_i are used to generate C_{i+1}. An itemset is considered as a candidate only if All Students its subsets also are large. To generate candidates of size *i+1*, joins are made of large itemsets found in the previous pass.

An algorithm called Apriori-Gen is used to generate the candidate itemsets for each pass after the first. All Students singleton itemsets are used as candidates in the first pass. Here the set of large itemsets of the previous pass, L_{i-1}, is joined with itself to determine the candidates. Individual itemsets must have All Students but one item in common in order to be combined.

7.6.1.2 Sampling Algorithm

To facilitate efficient counting of itemsets with large databases, sampling of the database may be used. The original sampling algorithm reduces the number of database scans to one in the best case and two in the worst case. The database sample is drawn such that it can be memory-resident. Then any algorithm, such as Apriori, is used to find the large itemsets for the sample. These are viewed as *potentially large* (PL) itemsets and used as candidates to be counted using the entire database. Additional candidates are

determined by applying the *negative border* function, BD^-, against the large itemsets from the sample. The entire set of candidates is then $C = BD^-(PL) \cup PL$. the negative border function is a generation of the Apriori-Gene algorithm. It is defined as the minimal set of itemsets that are not in PL, but whose subsets are All Students in PL.

7.6.1.3 Partitioning

Various approaches to generating large itemsets have been proposed based on a partitioning of the set of transactions. In this case, *D* is divided into *p* partitions D1, D2, D3, … DP. Partitioning may improve the performance of finding large itemsets in several ways:

- By taking advantage of the large itemset property, we know that a large itemset must be larger in at least one of the partitions. This idea can help to design algorithms more efficiently than those based on looking at the entire database.
- Partitioning algorithms may be able to adapt better to limited main memory. Each partition can be created such that it fits into main memory. In addition, it would be expected that the number of itemsets to be counted per partition would be smaller than those needed for the entire database.
- By using partitioning, parallel and/or distributed algorithms can be easily created, where a separate machine could handle each partition.
- Incremental generation of association rules may be easier to perform by treating the current state of the database as one partition and treating the new entries as a second partition

The basic partition algorithm reduces the number of database scans to two and divides the database into partitions such that each can be placed into main memory. When it scans the database, it brings that partition of the database into main memory and counts the items in that partition alone. During the first database scan, the algorithm finds all large itemsets in each

partition. Although any algorithm could be used for this purpose, the original proposal assumes that some level-wise approach, such as Apriori.

7.6.1.4 Pincer-Search Algorithm

Discovering frequent itemsets is a key problem in important data mining applications, such as the discovery of association rules, strong rules, episodes, and minimal keys. Typical algorithms for solving this problem operate in a bottom-up, breadth-first search direction. The computation starts from frequent 1-itemsets (the minimum length frequent itemsets) and continues until all maximal (length) frequent itemsets are found. During the execution, every frequent itemset is explicitly considered. Such algorithms perform well when all maximal frequent itemsets are short. However, performance drastically deteriorates when some of the maximal frequent itemsets are long. We present a new algorithm, which combines both the bottom-up and the top-down searches. The primary search direction is still bottom-up, but a restricted search is also conducted in the top-down direction. This search is used only for maintaining and updating a new data structure, the maximum frequent candidate set. It is used to prune early candidates that would be normally encountered in the bottom-up search. A very important characteristic of the algorithm is that it does not require explicit examination of every frequent itemset. Therefore, the algorithm performs well even when some maximal frequent itemsets are long. As its output, the algorithm produces the maximum frequent set, i.e., the set containing all maximal frequent itemsets, thus specifying immediately all frequent itemsets. We evaluate the performance of the algorithm using well-known synthetic benchmark databases, real-life census, and stock market databases. The improvement in performance can be up to several orders of magnitude, compared to the best previous algorithms.

7.6.1.5 FP-Tree Growth Algorithm

FP-growth algorithm is an efficient algorithm for mining frequent patterns. It scans database only twice and does not need to generate and test the candidate sets that is quite time consuming. The efficiency of the FP-growth algorithm outperforms previously developed algorithms. But, it must

recursively generate huge number of conditional FP-trees that requires much more memory and costs more time. In this paper, we present an algorithm, CFPmine, that is inspired by several previous works. CFPmine algorithm combines several advantages of existing techniques. One is using constrained sub-trees of a compact FP-tree to mine frequent pattern, so that it is doesn't need to construct conditional FP-trees in the mining process. Second is using an array-based technique to reduce the traverse time to the CFP-tree. And an unified memory management is also implemented in the algorithm. The experimental evaluation shows that CFPmine algorithm is a high performance algorithm. It outperforms Apriori, Eclat and FP-growth and requires less memory than FP-growth.

7.6.2 Advance Association Rule Techniques

7.6.2.1 Generalized Association Rules

Using a concept of hierarchy that shows the set relationship between different items, generalized association rules allow rules at different levels. Association rules can be generalized for any and all levels in the hierarchy. A *generalized association rule,* X $\Longrightarrow$ *Y,* is defined like a regular association rule with the restriction that no item in *Y* may be above any item *X*. When generating generalized association rules, all possible rules are generated using one or more given hierarchies. Several algorithms have been proposed to generate generalized rules. The simples would be to expand each transaction by adding all items above it in any hierarchy.

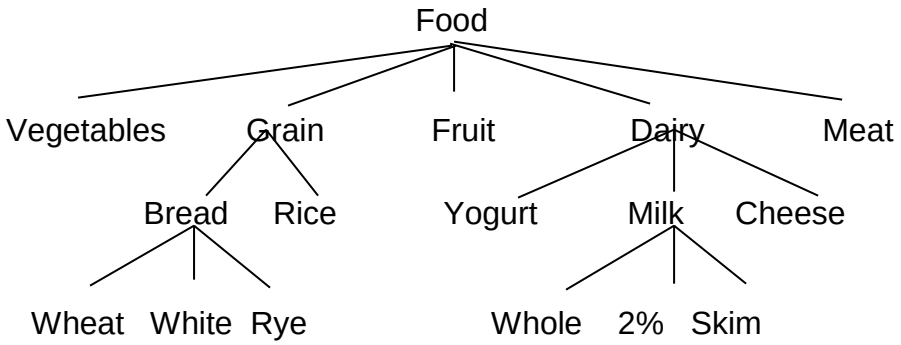


Fig 7.10: Hierarchy of Association Rule

Above figure shows a partial concept hierarchy for food. This hierarchy shows that Wheat Bread is a type of Bread, which is a type of grain. An association rule of the form Bread $\Longrightarrow$ Peanut Butter has a lower support and threshold than one of the form Grain $\Longrightarrow$ Peanut Butter. There obviously are more transactions containing any type of grain than transactions containing Bread. Likewise, Wheat Bread $\Longrightarrow$ Peanut butter a has lower threshold and support than Bread $\Longrightarrow$ Peanut Butter.

7.6.2.2 Multiple-Level Association Rules

A variation of generalized rules is *multiple-level association rules*. With multiple-level rules, itemsets may occur from any level in the hierarchy. Using a variation of the Apriori algorithm, the concept hierarchy is traversed

in a top-down manner and large itemsets are generated. When large itemsets are found at level *I*, large itemsets are generated for level *I* + 1. large *k*-itemsets at one level in the concept hierarchy are used as candidates to generate large *k*-itemsets for children at the next level.

Modification to the basic association rule ideas may be changed. We expect that there is more support for itemsets occurring at higher levels in the concept hierarchy. Thus, the minimum support required for association frequency of itemsets at higher levels is much greater than the frequency of itemsets at lower level. Thus, for the reduced minimum support concept, the following rules apply:

- The minimum support for all nodes in the hierarchy at the same level is identical.
- If α_i is the minimum support for level *I* in the hierarchy and α_{i-1} is the minimum support for level *i-1*, then $\alpha_{i-1} > \alpha_i$.

7.7 Clustering

Clustering is an example of data mining task that fits in the descriptive model of data mining. The use of clustering enables you to create new groups and classes based on the study of patterns and relationship between values of data in a data bank. It is similar to classification but does not require you to predefine the groups or classes. Clustering technique is otherwise known as unsupervised learning or segmentation. All those data items that resembles more closely with each other are clubbed together in a single group, also known as clusters.

Clustering is similar to classification in that data are grouped. However, unlike classification, the groups are not predefined. Instead, the grouping is accomplished by finding similarities between data according to characteristics found in the actual data. the groups are called *clusters,* some researchers view clustering as a special type of classification. Here we follow a more conventional view in that the two are different. Many definitions for cluster have been proposed:

- Set of like elements, elements from different clusters are not alike.
- The distance between points in a cluster is less than the distance between a point in the cluster and any point outside it.

The term similar to clustering is *database segmentation*, where like tuples in a database are grouped together. This is done to partition or segment the database into components that then give the user a more general view of the data.

As illustrated in the following figure, a given set of data may be clustered on different attributes. Here a group of homes in a geographic area is shown. The first type of clustering is based on the location of the home. Homes that are geographically close to each other are clustered together. In the second clustering, homes are grouped based on the size of the house.

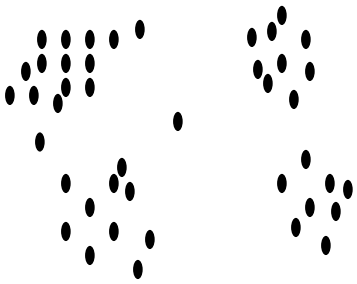


Fig 7.11 (a): Group of homes

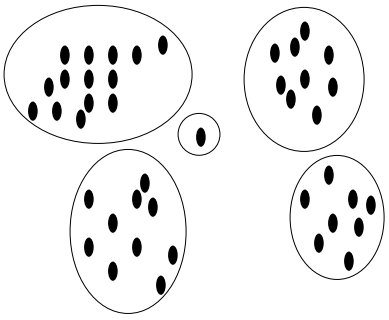


Fig 7.12 (b): Geographic distance based

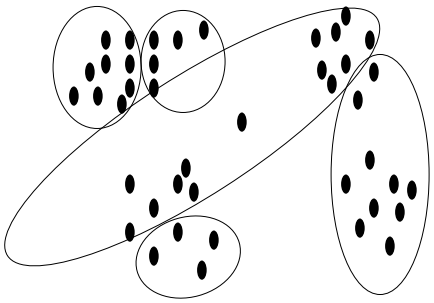


Fig 7.13 (c): Size – based

Clustering has been used in many domains, including biology, medicine, anthropology, marketing, and economics. Clustering applications include plant and animal classification, dieses classification, image processing, pattern recognition, and document retrieval. One of the first domains in which clustering was used was biological taxonomy. Recent uses include examining Web log data to detect usage patterns.

When clustering is applied to a real-world database, many interesting problems occur:

- Outlier handling is difficult. Here the elements do not naturally fall into any cluster. They can be viewed as solitary clusters. However, if a clustering algorithm attempts to find larger clusters, these outliers will be forced to be placed in some cluster. This process may result in the creation of poor clusters by combining two existing clusters and leaving the outlier in its own cluster.

- Dynamic data in the database implies that cluster membership may change over time.

- Interpreting the semantic meaning of each cluster may be difficult. However, with clustering, this may not be the case. Thus, when the clustering process finishes creating a set of clusters, the exact meaning of each cluster may not be obvious. Here is where a domain expert is needed to assign a label or interpretation for each cluster.

- There is no one correct answer to a clustering problem. In fact, many answers may be found. The exact number of clusters required is not easy to determine. Again, a domain expert may be required. For example, suppose we have a set of data about plants that have been collected during a field trip, without any prior knowledge of plant classification, if we attempt to divide this set of data into similar groupings, it would not be clear how many groups should be created.

- Another related issue is what data should be used for clustering. Unlike learning during a classification process, where there is some a priori knowledge concerning what the attributes of each classification should be, in clustering we have no supervised learning to aid the process. Indeed, clustering can be viewed as similar to unsupervised learning.

It can be summarized some basic features of clustering against the classification:

- The number of clusters is not known,

- There may not be any a priori knowledge concerning the clusters.

- Cluster results are dynamic.

The clustering problem is stated as shown in following definition. Here we assume that the number of clusters to be created is an input value, k. the actual content (and interpretation) of each cluster, K_j, $1 \leq j \leq k$, is determined as a result of the function definition. Without loss of generality, we will view that the result of solving a cluster problem is that a s to clusters is created: $K = \{K_1, K_2, K_3, \ldots, K_k\}$.

Given a database $D = \{t_1, t_2, t_3, \ldots, t_n\}$ of tuples and an integer value k, the **clustering problem** is to define a mapping $f : D \rightarrow \{1, 2, 3, \ldots, k\}$ where each ti is assigned to one cluster K_j, $1 \leq j \leq k$. A **cluster**, K_j, contains precisely those tuples mapped to it; that is, $K_j = \{t_i \mid f(t_i) = K_j, 1 \leq j \leq k,$ and $t_i \in D\}$.

A classification of the different types of clustering algorithms is shown in following figure. Clustering algorithms themselves may be viewed as hierarchical or partitional. With *hierarchical* clustering, a nested set of clusters is created.

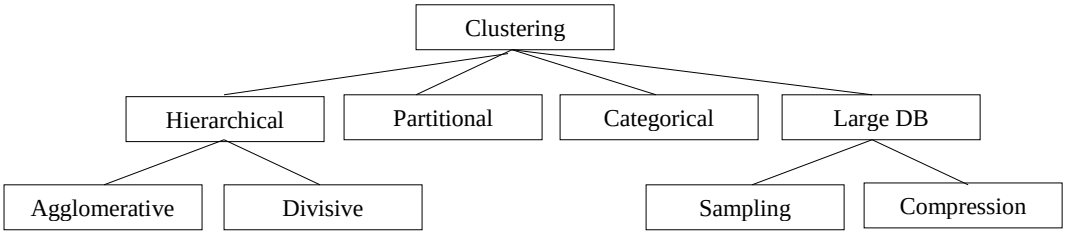


Fig 7.14: Classification of Clusters

Each level in the hierarchy has separate set of clusters. At the lowest level, each item is in its own unique cluster. At the highest level, All Students items belong to the same cluster. With hierarchical clustering, the desired number of clusters is not input. With *partitional* clustering, the algorithm creates only one set of clusters. These approaches use the desired number of clusters to drive how the final set is created. Traditional clustering algorithms tend to be targeted to small numeric databases that fit into memory. These are, however, more recent clustering algorithms that look at categorical data and are targeted to larger, perhaps dynamic, databases. Algorithms targeted to larger databases may adapt to memory constraints by either sampling the database or using data structures, which can be compressed or pruned to fit into memory regardless of the size of the database. Clustering algorithms may also differ based on whether they produce overlapping or non-overlapping clusters. Even though we consider only non-overlapping clusters, it is possible to place an item in multiple clusters. In turn, non-overlapping clusters can be viewed as extrinsic or intrinsic. *Extrinsic* techniques use labeling of the items to assist in the classification process. These algorithms are the traditional classification supervised learning algorithms in which a special input training set is used. *Intrinsic* algorithms do not use any a priori category labels, but depend only on the adjacency matrix containing the distance between objects.

7.7.1 Hierarchical Algorithms

Hierarchical clustering algorithms actually create sets of clusters. Hierarchical algorithms differ in how the sets are created. A tree data structure, called a *dendrogram*, can be sued to illustrate the hierarchical clustering technique and the sets of different clusters. The root in a dedrogeram tree contains one cluster where All Students elements are together. The leaves in the dedrogram each consist of a single element cluster. Internal nodes in the dendrogram represent new clusters formed by merging the clusters that appear as its children in the tree each level in the tree is associated with the distance measure that was used to merge the clusters. All clusters created at a particular level were combined because the children clusters had a distance between them less than the distance value associated with this level in the tree.

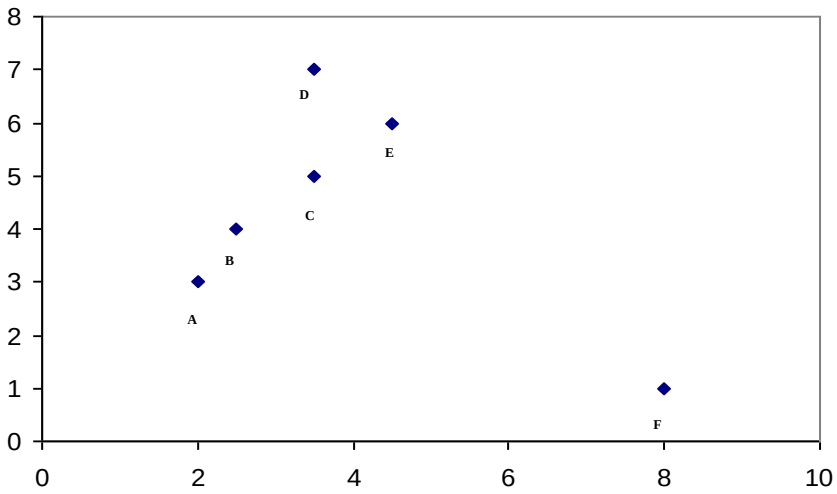


Fig 7.15(a): Six clusters

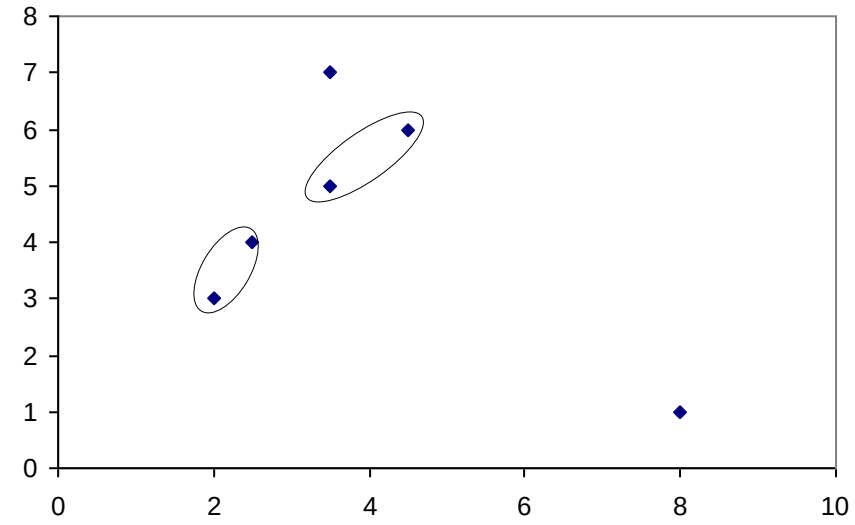


Fig 7.15(b): Four Clusters

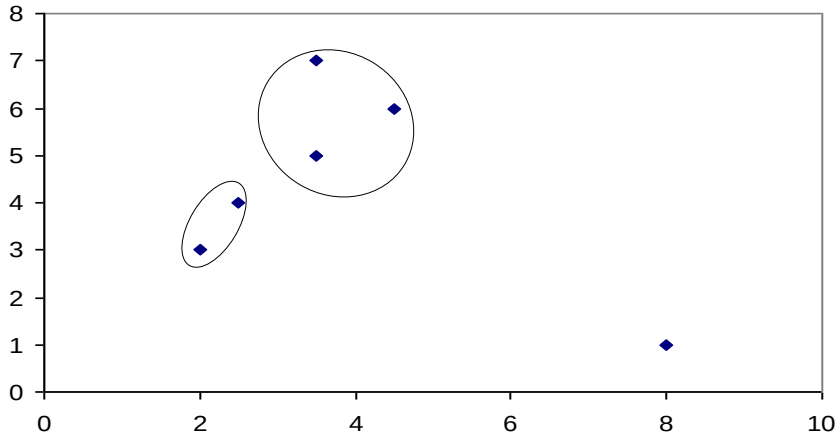


Fig 7.15(c): Three clusters

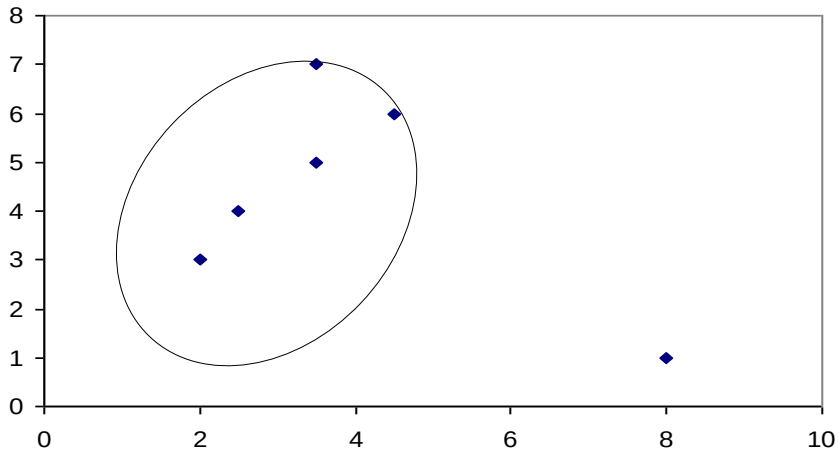


Fig 7.15(d): Two clusters

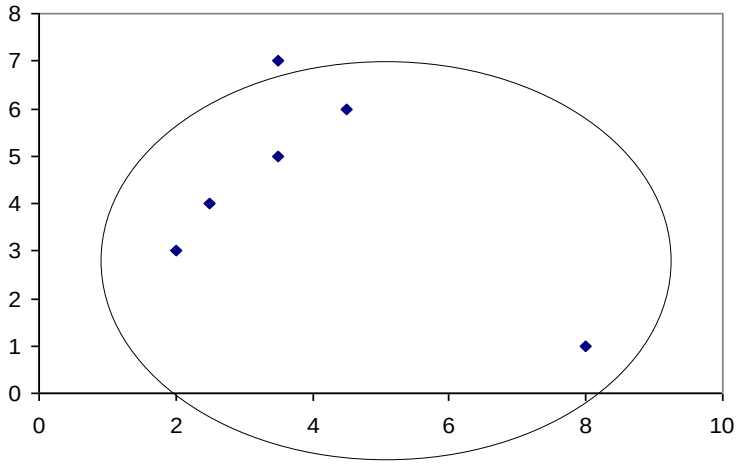


Fig 7.15(e): One cluster

Above figure shows six elements { A, B, C, D, E, F}, to be clustered. Part (a) to (e) of the figure shows five different sets of clusters. In part (a) each cluster is viewed to consist of a single element. Part (b) illustrates four clusters. Here there are two sets of two-element clusters. These clusters are formed at this level because these two elements are closer to each other than any of the other elements. Part (c) shows a new cluster formed by adding a close element to one of the two-element clusters. In part (d) the two-element and three-element clusters are merged to give a five-element cluster. This is done because these two clusters are closer to each other than to the remote element cluster, {F}. At the last stage, part (e), all six elements are merged.

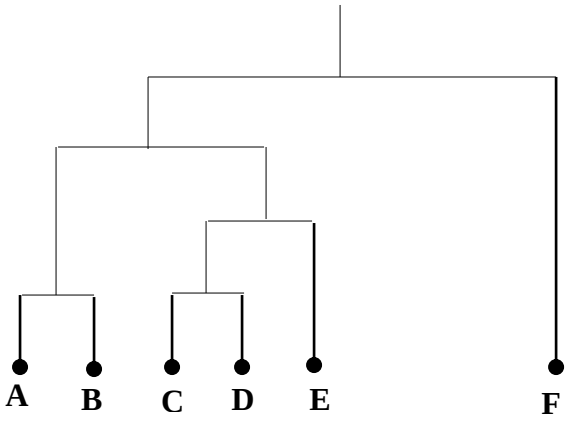


Fig 7.16: Dendrogram for above Example

7.7.2 Agglomerative Algorithm

Agglomerative algorithms start with each individual item in its own cluster and iteratively merge clusters until all item belongs in one cluster. Different agglomerative algorithms differ in how the clusters are merged at each level.

All agglomerative approaches experience excessive time and space constraints.

The space required for the adjacency matrix is $O(n^2)$ where there are n items to cluster. Because of the iterative nature of the algorithm, the matrix (or subset of it) must be accessed multiple times. Let's consider the following algorithm.

(1) Start with n clusters, and a single sample indicates one cluster.
(2) Repeat step 3 until cluster number is what we want or 1.
(3) Find the most similar clusters C_i and C_j, then merge them into one cluster.

What is the most similar cluster pair? Compute distances between each pair of clusters to judge which two clusters have prior opportunity to merge. There are several ways to calculate the distances between cluster C_i and cluster C_j.

7.7.2.1 Single-Linkage Agglomerative Algorithm:

The single link technique is based on the idea of finding maximal connected components in a graph. A *connected component* is a graph in which there exists a path between any to vertices. With the single link approach, two clusters are merged if there is at least one edge that connects the two clusters; that is, if the minimum distance between any two points is less than or equal to the threshold distance being considered. For this reason it is often called the *nearest neighbor* cluster technique.

In single-linkage agglomerative algorithm, defining the distance between two clusters is the shortest distance between a sample in one cluster and a sample in the other cluster.

7.7.2.2 Complete-Linkage Agglomerative Algorithm:

Although the complete link algorithm is similar to the single link algorithm, it looks for cliques rather than connected components. A *clique* is a maximal graph in which there is an edge distance between any clusters so that two clusters are merged if the maximum distance is less than or equal to the distance threshold. In this algorithm, we assume the existence of a procedure, clique, which finds all cliques in a graph. As with the single link algorithm, this is expensive because it is an $O(n^2)$ algorithm.

In the complete-linkage agglomerative algorithm, defining the distance between two clusters is the longest distance between a sample in one cluster and a sample in the other cluster.

$$d(C_i, C_j) = \max_{a \in C_i, b \in C_j} d(a, b)$$

7.7.2.3 Average-Linkage Agglomerative Algorithm:

In the average-linkage agglomerative algorithm, defining the distance between two clusters is the average distance between a sample in one cluster and a sample in the other cluster (n_{c_i} is the number of the cluster C_i).

$$d(C_i, C_j) = \frac{1}{n_{c_i} n_{c_j}} \sum_{a \in C_i, b \in C_j} d(a, b)$$

7.7.3 Divisive Clustering

With divisive clustering, all items are initially placed in one cluster and clusters are repeatedly split in two until all items are in their own cluster. The idea is to split up clusters where some elements are not sufficiently close to other elements.

One simple example of a divisive algorithm is based on the Missing Spanning Tree (MST) version of the single link algorithm. Here, however, we cut out edges from MST from the largest to the smallest. Let's see following algorithm.

(1) Start with just only one cluster. That is, all samples in this one cluster.

(2) Repeat step 3, 4, 5, 6 until cluster number is the number of samples or what we want.

(3) Calculate diameter of each cluster. Diameter is the maximal distance between samples in the cluster. Choose one cluster C having maximal diameter of all clusters to split.

(4) Find the most dissimilar sample x from cluster C. Let x depart from the original cluster C to form a new independent cluster N (now cluster C doesn't include sample x). Assign all members of cluster C to M_C.

(5) Repeat 6 until members of cluster C and N don't change.

(6) Calculate similarities from each member of M_C to cluster C and N, and let the member owning the highest similarities in M_C move to its similar cluster C or N. Update members of C and N.

Here we take a simple example to describe the method above. First, the distance matrix D of 5 samples x_1, x_2, x_3, x_4, x_5 is

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} - & 2 & 6 & 10 & 9 \\ 2 & - & 5 & 9 & 8 \\ 6 & 5 & - & 4 & 5 \\ 10 & 9 & 4 & - & 3 \\ 9 & 8 & 5 & 3 & - \end{bmatrix} \end{array} .$$

Our processing steps are as follows:

1. Because there is only one cluster, this cluster has maximal diameter. For a start, we split this cluster.

2. Calculate average distances from one sample to the others. For example, the average distance from x_1 to x_2, x_3, x_4 and x_5 is $(2+6+10+9)/4 = 6.75$, and the others:

$$x_2 : (2+5+9+8)/4 = 6,$$
$$x_3 : (6+5+4+5)/4 = 5,$$
$$x_4 : (10+9+4+3)/4 = 6.5,$$
$$x_5 : (9+8+5+3)/4 = 6.25.$$

Sample x_1 has maximal average distance, so extract x_1 from the cluster. Now we have 2 clusters: $\{x_2, x_3, x_4, x_5\}$ and $\{x_1\}$.

3. Find average distances from x_2, x_3, x_4 and x_5 to clusters $\{x_2, x_3, x_4, x_5\}$ and $\{x_1\}$.

$$\begin{array}{c} \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cc} \{x_2, x_3, x_4, x_5\} & \{x_1\} \end{array} \\ \begin{bmatrix} 7.33 & 2 \\ 4.67 & 6 \\ 5.33 & 10 \\ 5.33 & 9 \end{bmatrix} \end{array}$$

The distance from x_2 to cluster $\{x_1\}$ is minimum, so put x_2 into cluster $\{x_1\}$. Now clusters are updated to $\{x_3, x_4, x_5\}$ and $\{x_1, x_2\}$. Repeat step 6 of the algorithm to check if members of each cluster are updated.

$$\begin{array}{c} \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cc} \{x_3, x_4, x_5\} & \{x_1, x_2\} \end{array} \\ \begin{bmatrix} 7.33 & 2 \\ 4.5 & 5.5 \\ 3.5 & 9.8 \\ 4 & 8.5 \end{bmatrix} \end{array}$$

The distance from x_2 to cluster $\{x_1, x_2\}$ is also minimum and cluster members don't change again. Go to step 3 of the algorithm. Now there are 2 clusters $\{x_1, x_2\}$ and $\{x_3, x_4, x_5\}$.

4. The diameter of the cluster $\{x_1, x_2\}$ is:

$$diameter(\{x_1, x_2\}) = \max(|x_1 - x_2|) = 2.$$

The diameter of cluster $\{x_3, x_4, x_5\}$ is

$$diameter(\{x_3, x_4, x_5\}) = \max(|x_3 - x_4|, |x_3 - x_5|, |x_4 - x_5|) = 5.$$

We choose the cluster $\{x_3, x_4, x_5\}$ to split (has maximal diameter of all clusters).

5. Calculate average distances from one sample to the others in cluster $\{x_3, x_4, x_5\}$.

$$x_3 : (4+5)/2 = 4.5$$
$$x_4 : (4+3)/2 = 3.5$$
$$x_5 : (5+3)/2 = 4$$

So split $\{x_3, x_4, x_5\}$ into $\{x_3\}$ and $\{x_4, x_5\}$. The average distances from x_4 and x_5 to clusters $\{x_4, x_5\}$ and $\{x_3\}$ are:

$$\begin{array}{c} \\ x_4 \\ x_5 \end{array} \begin{array}{c} \{x_4, x_5\} \quad x_3 \\ \begin{bmatrix} 3 & 4 \\ 3 & 5 \end{bmatrix} \end{array}$$

Because minimum distance is 3, cluster members of each cluster don't update. Go to step 3 of the algorithm.

6. Now we have 3 clusters $\{x_1, x_2\}$, $\{x_3\}$, and $\{x_4, x_5\}$. Their diameters are 2, 0, and 3. Because there is only one sample in cluster $\{x_3\}$, don't think about this cluster. We decide split the cluster $\{x_1, x_2\}$.
7. Split $\{x_1, x_2\}$ into $\{x_1\}$ and $\{x_2\}$. Because cluster members of each cluster don't update, go to step 3.
8. Now we have 4 clusters $\{x_1\}, \{x_2\}$, $\{x_3\}$ and $\{x_4, x_5\}$. Only the cluster $\{x_4, x_5\}$ has more than one sample and have maximal diameter, so split $\{x_4, x_5\}$
9. Split $\{x_4, x_5\}$ into $\{x_4\}$ and $\{x_5\}$. Each sample represents one cluster; so stop (see following figure).

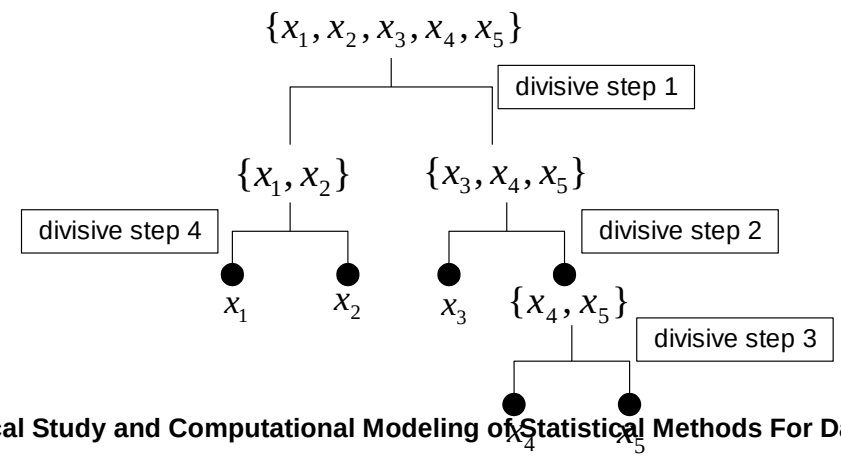


Fig 7.17: An example for hierarchical divisive algorithm

7.7.4 Partitional Algorithm

Nonhierarchical or partitional clustering creates the clusters in one step as opposed to several steps. Only one set of clusters is created, although several different sets of clusters may be created internally within the various algorithms. Since only one set of clusters is output, the user must input the desired number, *k*, of clusters. In addition, some metric or criterion function is used to determine the goodness of any proposed solution. This measure of quality could be average distance between clusters or some other metric. The solution with the best value for the criterion functions is the clustering solution used. One common measure is a squared error metric, which measures the squared distance from each point to the centroid to the associated cluster:

$$\sum_{m=1}^{k} \sum_{t_{mi} \in K_m} \text{dis}(C_m, t_{mi})$$

A problem with partitional algorithms is that they suffer from a combinatorial explosion due to the number of possible solutions. Clearly, searching all possible clustering alternatives usually would not be feasible. For example, given a measurement criteria, a naïve approach could look at all possible sets of *k* clusters. There are *S*(*n, k*) possible combinations to examine.

$$S(n, k) = \frac{\sum_{m=1}^{k} (-1)^{k-1} \, {}^{k}C_i \, (i)^n}{K!}$$

7.7.5 *K* – Means Clustering

K – means is an iterative clustering algorithm in which items are moved among sets of clusters until the desired set is reached. As such, it may be viewed as a type of squared error algorithm, although the convergence criteria need not be defined based on the square error. A high degree of similarity among elements in clusters is obtained, while a high degree of dissimilarity among elements in different clusters is achieved simultaneously. The *cluster mean* of $K_i = \{ t_{i1}, t_{i2}, t_{i3}, \ldots , t_{im}\}$ is defined as

$$m_i = \frac{\sum_{i=1}^{m} t_{ij}}{m}$$

This definition assumes that each tuple has only one numeric value as opposed to a tuple with many attribute values. The K-means algorithm requires that some definition of cluster mean exists, but it does not have to be this particular one. Here the means is defined identically to our earlier definition of centroid. This algorithm assumes that the desired number of clusters, *k*, is an input parameter.

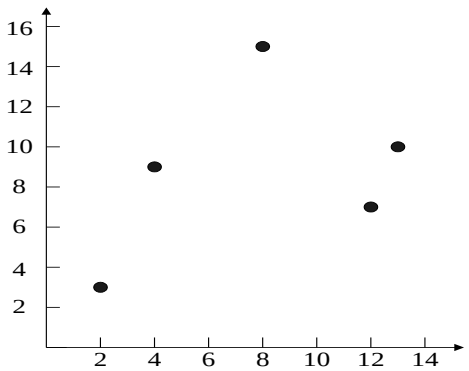
(1) Initialize k cluster centers to be seed points (These centers can be randomly produced or use other ways to generate).

(2) For each sample, find the nearest cluster center, put the sample in this cluster and re-compute centers of the altered cluster (Repeat n times).

(3) Exam all samples again and put each one in the cluster identified with the nearest center (don't re-compute any cluster centers). If members of each cluster haven't been changed, stop. If changed, go to step 2.

Let's consider following example.

$$x_1 = (2, 3), \quad x_2 = (4, 9), \quad x_3 = (8, 15), \quad x_4 = (12, 7), \quad x_5 = (13, 10)$$



Samples in 2-D

We want to have 2 clusters of the data. Our steps are:

1. Set initial points. Because $k = 2$, we select 2 points, $C_1 = x_1$ and $C_2 = x_5$, as center points.
2. x_2 is near C_1, so put x_2 into cluster 1. Now 2 clusters are $\{x_1, x_2\}$ and $\{x_5\}$. Others are as follows:

	C_1 and C_2	Near	New 2 Cluster
x_2 (*step I*)	(2,3), (13,10)	C_1	$\{x_1, x_2\}$ $\{x_5\}$
x_3 (*step II*)	(3,6), (13,10)	C_2	$\{x_1, x_2\}$ $\{x_3, x_5\}$
x_4 (*step III*)	(3,6), (10.5,12.5)	C_2	$\{x_1, x_2\}$ $\{x_3, x_4, x_5\}$

3. Now new 2 centers are $(3, 6)$ and $(11, 10.67)$. For each sample, find its nearest center (don't re-compute the centers). Sample x_1 and x_2 are near $(3, 6)$. Sample x_3, x_4 and x_5 are near $(11, 10.67)$. Members of each cluster don't change. So stop.

K-means algorithm has advantages such as easy to implement and converge in finite iterations. But when samples have outliers, which are sufficiently far removed from the rest of the data (see following figure), they will have influences on the results.

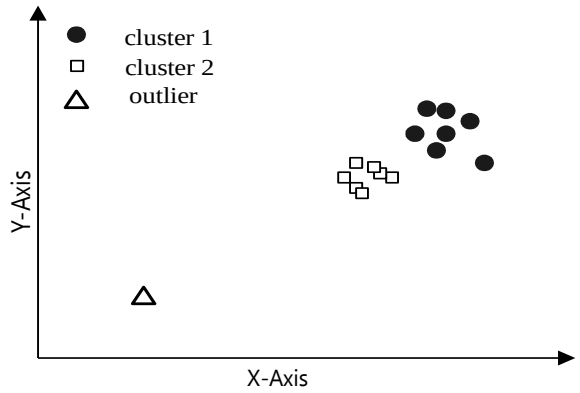


Fig 7.18: An outlier of samples

Another important problem of K-means algorithm is selecting initial seed points because clustering results always depend on initial seed points and partitions. To prevent this problem, one way we can run many conditions of

different initial points to decide which condition is the best. For a fixed number of clusters, the goal of K-means algorithm is to minimize the square error E, so that stop condition can switch to finding minimum of square error instead of observing the change of members.

7.7.6 Nearest Neighbor Algorithm

An algorithm similar to the single link technique is called the *nearest neighbor algorithm.* With this serial algorithm, items are iteratively merged into the existing clusters that are closest. In this algorithm a threshold, *t*, is used to determine if items will be added to existing clusters or if a new cluster is created. The nearest neighbor prediction algorithm simply stated is: Objects that are “near” to each other will have similar prediction values as well. Thus if you know the prediction value of one of the objects you can predict it for its nearest neighbors.

7.7.7 Clustering of large database

The clustering algorithms presented in the preceding sections are some of the classic clustering techniques. When clustering is used with dynamic databases, these algorithms may not be appropriate. First, they all assume that sufficient main memory exists to hold the data to be clustered and the data structures needed to support them. With large databases containing thousands of items (or more), these assumptions are not realistic. In addition, performing I/Os continuously through the multiple iterations of an algorithm is too expensive. Because of these main memory restrictions, the algorithms do not scale up to large databases. Another issue is that some assume that the data are present all at once. These techniques are not appropriate for dynamic databases. Clustering techniques should be able to adapt as the database changes.

Recent research at Microsoft has examined how to efficiently perform the clustering algorithms with large databases. The basic idea of this scaling approach is as follows:

1. Read a subset of the database into main memory.
2. Apply clustering technique to data in memory.

3. Combine results with those from prior samples.
4. The in-memory data are then divided into three different types" those items that will always be needed even when the next sample is brought in, those that can be discarded with appropriate updates to data being kept in order to answer the problem, and those that will be saved in a compressed format. Based on the type, each data item is then kept, deleted, or compressed in memory
5. If termination criteria are not met, then repeat from step 1.

This approach has been applied to the K-means algorithm and has been shown to be effective.

7.7.8 BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

BIRCH is designed for clustering a large amount of metric data. It assumes that there may be a limited amount of main memory and achieves a linear I/O time requiring only one database scan. It is incremental and hierarchical, and it uses an outlier handling technique. Here points that are found in sparsely populated areas are removed. The basic idea of the algorithm is that a tree is built that captures needed information to perform clustering. The clustering is then performed n the tree itself, where labeling of nodes in the tree contain the needed information to calculate distance values. A major characteristics of the BIRCH algorithm is the use of the *clustering feature,* which is a triple that contains information about a cluster. The clustering feature provides a summary of the information about one cluster. By this definition it is clear that BIRCH applies only to numeric data.

7.7.9 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

The approach used by DBSCAN is to create clusters with a minimum size and density. Density is defined as a minimum number of points within a certain distance of each other. This handles the outlier problem by ensuring that an outlier will not create a cluster. One input parameter, *Min Pts,* indicates the minimum number of points in any cluster. In addition, for each point in a cluster there must be another point in the cluster whose distance from it is less than a threshold input value, *Eps*. The *Eps – neighborhood* or

neighborhood of a point is the set of points within a distance of *Eps*. The desired number of clusters, *k*, is not input but rather is determined by the algorithm itself.

7.7.10 CURE (Clustering Using Representatives) Algorithm

One objective for the CURE clustering algorithm is to handle outlier well. It has both hierarchical component and a partitioning component. First, a constant number of points, *c*, are chosen from each cluster. These well-scattered points are the shrunk toward the cluster's centroid by applying a shrinkage factor, α. When α is 1, all points are shrunk to just one – the centroid. These points represent the cluster better than a single point could. With multiple representative points, clusters of unusual shapes (not just a sphere) can be better represented. CURE then uses a hierarchical clustering algorithm. At each step in the agglomerative algorithm, clusters with the closest pair of representative points are chosen to be merged. The distance between them is defined as the minimum distance between any pair of points in the representative sets from the two clusters.

CURE handles limited main memory by obtaining a random sample to find initial clusters. The random sample is partitioned, and each partition is then partially clustered 1. These resulting clusters are then completely clustered to second pass. The sampling and partitioning are done solely to ensure that the data (regardless of database size) can fit into available main memory. When clustering of the sample is complete, the labeling of data on disk is performed. A data item is assigned to the cluster with the closest representative points.

8. WEKA Software (Data Mining Software in Java) 

An exciting and potentially far-reaching development in computer science is the invention and application of methods of machine learning. These enable a computer program to automatically analyze a large body of data and decide what information is most relevant. This crystallized information can then be used to automatically make predictions or to help people make decisions faster and more accurately.

The overall goal of our project is to build a state-of-the-art facility for developing machine learning (ML) techniques and to apply them to real-world data mining problems. The team of Weka project incorporated several standard ML techniques into a software "workbench" called WEKA, for Waikato Environment for Knowledge Analysis. With it, a specialist in a particular field is able to use ML to derive useful knowledge from databases that are far too large to be analyzed by hand. WEKA's users are ML researchers and industrial scientists, but it is also widely used for teaching.

The objectives of Weka project are to

- Make ML techniques generally available;
- Apply them to practical problems that matter to New Zealand industry;
- Develop new machine learning algorithms and give them to the world;
- Contribute to a theoretical framework for the field.

This machine-learning package is publically available and presents a collection of algorithms for solving real-world data mining problems. The software is written entirely in Java and includes a uniform interface to a number of standard ML techniques. So we can feel free to browse around.

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License.

8.1 Practically implementation of Data Set into Weka

We have applied our data set into Weka for analysis and found some interesting results and patterns.

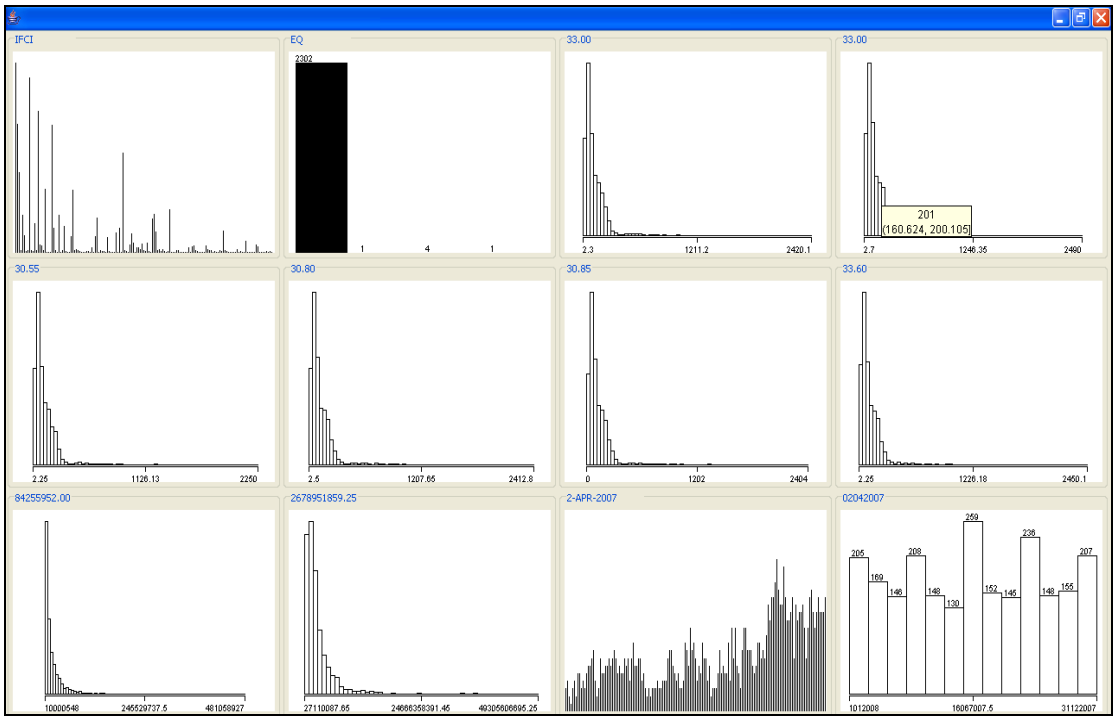


Fig 8.1 : Consolidated Charts

Above figure shows the result of Data Visualization using the WEKA (an open source data mining tool developed using JAVA). For this example the data is taken from our data warehouse. We have not used all the data that we have extracted and transformed because we have experienced that Weka software does not work with very large data set. In our data warehouse we have total 279912 rows of stock market data that is data from1st April 2007 to 16th May 2008. But we have used only those rows whose total trading volume is more than 100000000 shares (100 millions). Total number of rows meeting this criterion is 2308. It means to make 2308 records useful to Weka, it is converted into the ARFF format and then applied to Weka.

By visualizing the result one can easily see that there are 201 scripts whose high rate is ranging from Rs.160.624 to Rs.200.105. Taking individual look of every chart the data can be analyzed as follows.

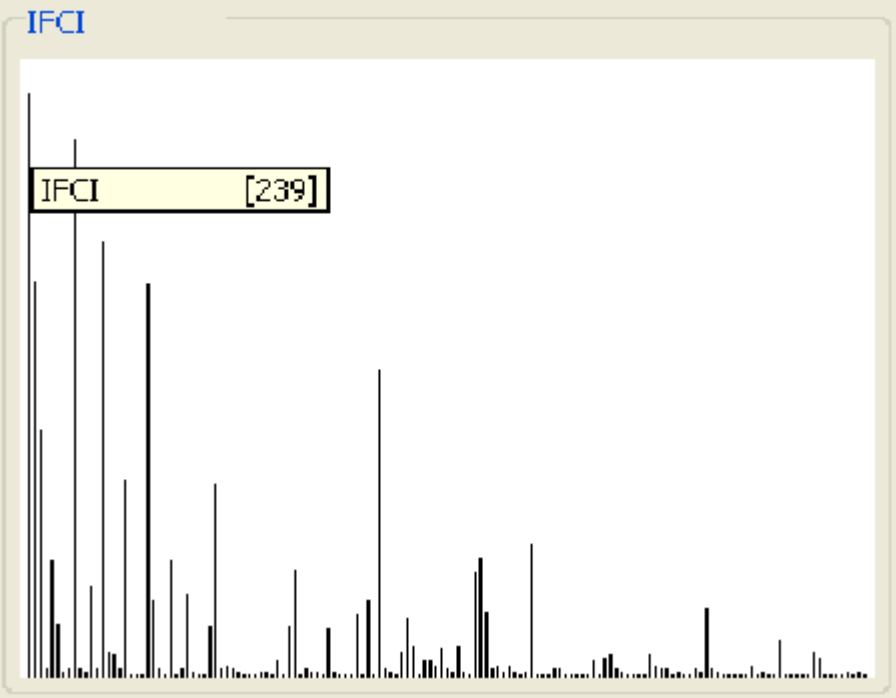


Fig 8.2: Chart of Scripts

Above figure visualizes that out of total 2308 rows, the script IFCI appears 239 times, so it can be said that the script IFCI has 239 times trading volume more than 10 million of shares during the period 1st April 2007 to 16th May 2008.

Above chart displays the number of occurrences for all the scripts whose trading volume is more than 10 million.

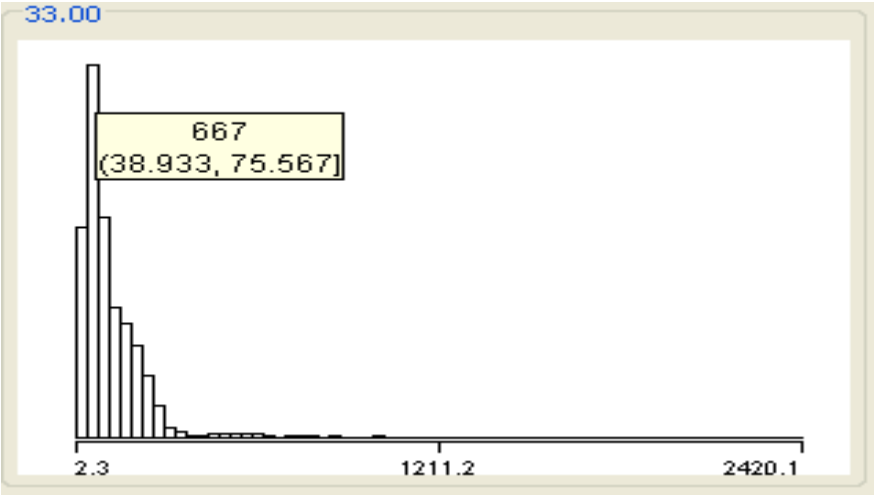


Fig 8.3: Chart of Open rate

Above figure visualize that there are 667 scripts whose trading volume more than 10 million of shares having open rate between 38.933 to 75.567 when share market opened during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 667.

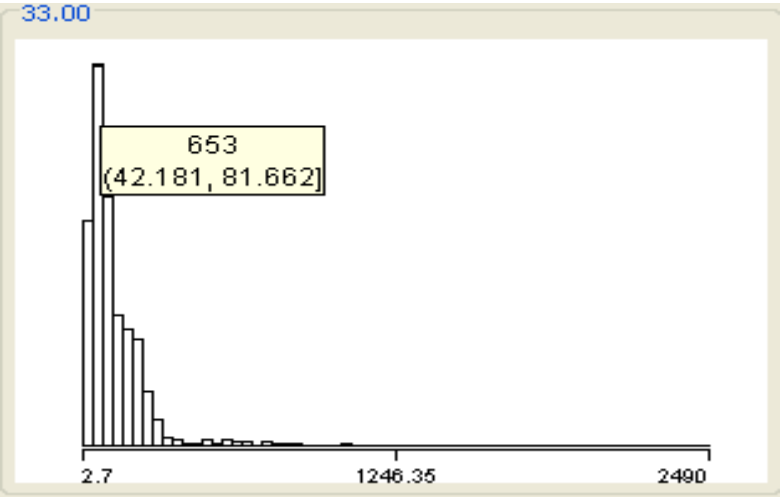


Fig 8.4: Chart of High rate

Above figure visualize that there are 653 scripts whose trading volume more than 10 million of shares having high rate between 42.181 to 81.662 during the day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 653.

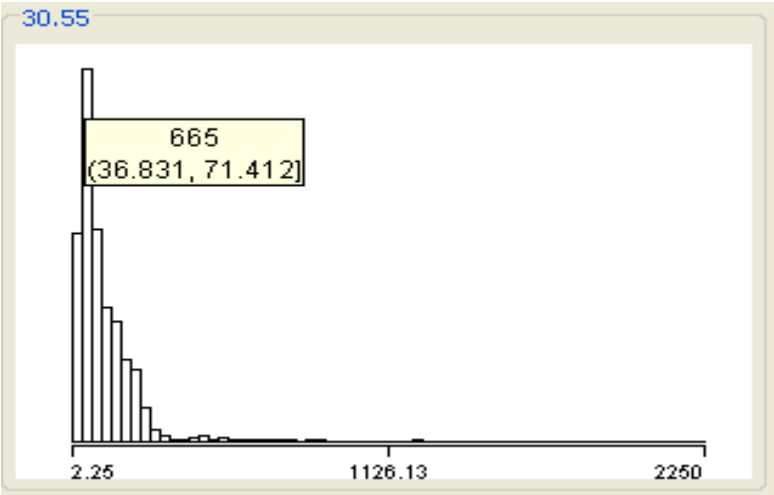


Fig 8.5: Chart of Low rate

Above figure visualize that there are 665 scripts whose trading volume more than 10 million of shares having low rate between 36.831 to 71.412 during the

day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 665.

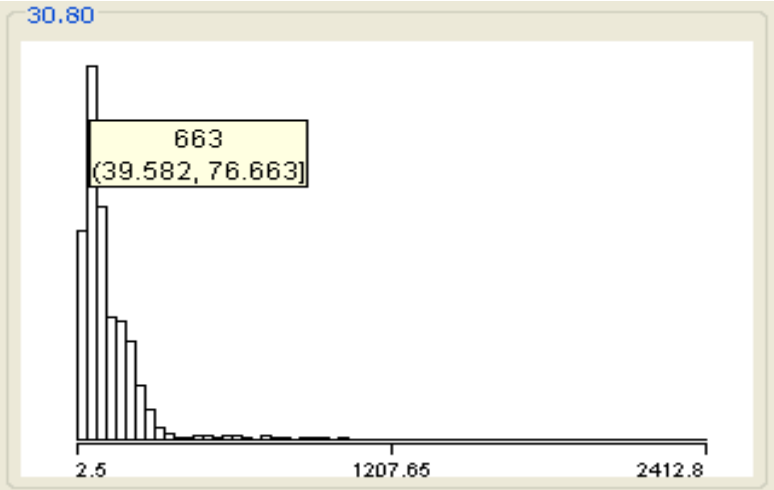


Fig 8.6: Chart of Close rate

Above figure visualize that there are 663 scripts whose trading volume more than 10 million of shares having closing rate between 39.582 to 76.663 during the day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 663.

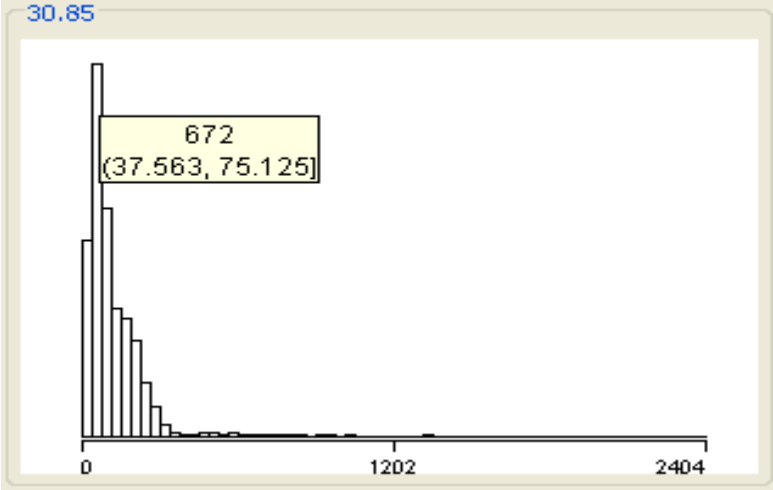


Fig 8.7: Chart of Last rate

Above figure visualize that there are 672 scripts whose trading volume more than 10 million of shares having last rate between 37.563 to 75.125 during the

day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 672.

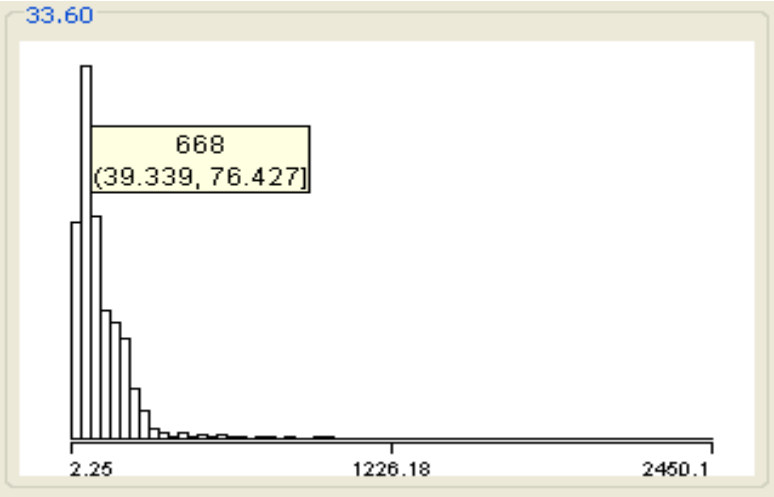


Fig 8.8: Chart of Previous Close rate

Above figure visualize that there are 668 scripts whose trading volume more than 10 million of shares having previous closing rate between 39.339 to 76.427 during the day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 668.

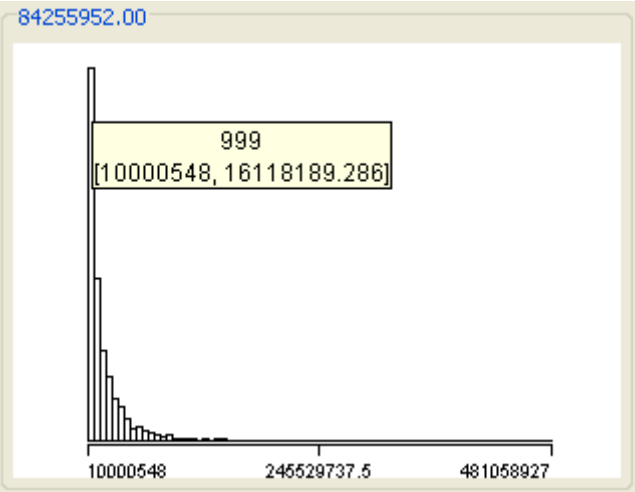


Fig 8.9: Chart of Volume

Above figure visualize that there are 999 scripts whose trading volume more than 10 million of shares having trading quantity between 10000548 to 16118189.286 during the day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 999.

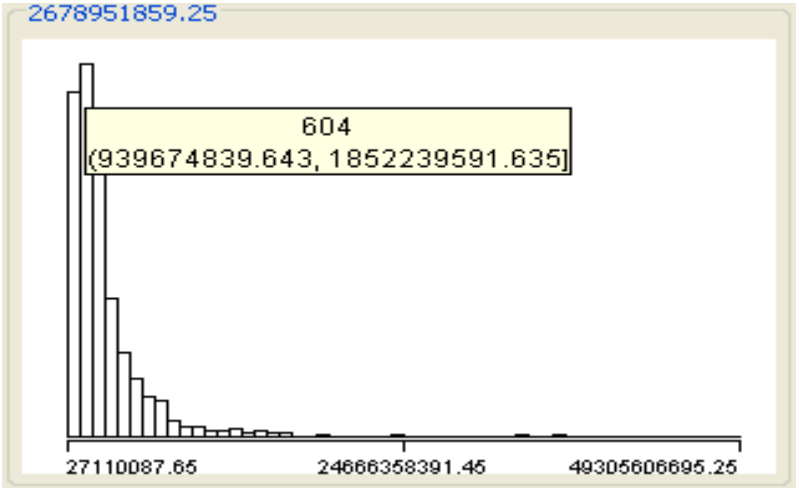


Fig 8.10: Chart of Trading Value

Above figure visualize that there are 604 scripts whose trading volume more than 10 million of shares having trading value between 939674839.643 to 1852239591.635 during the day, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 604.

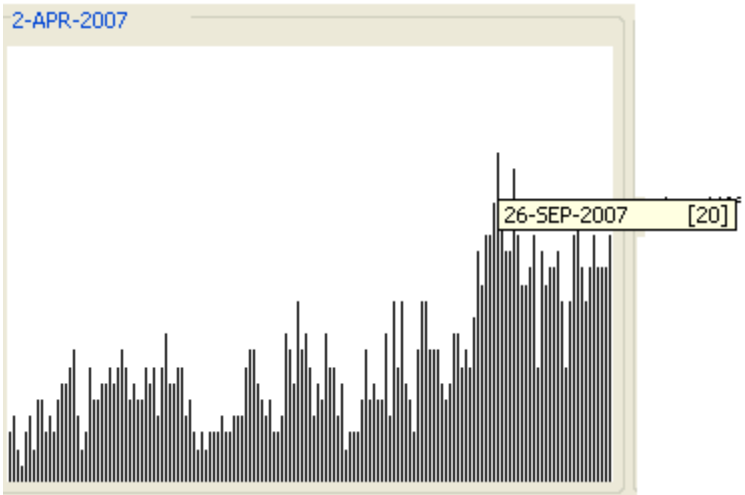


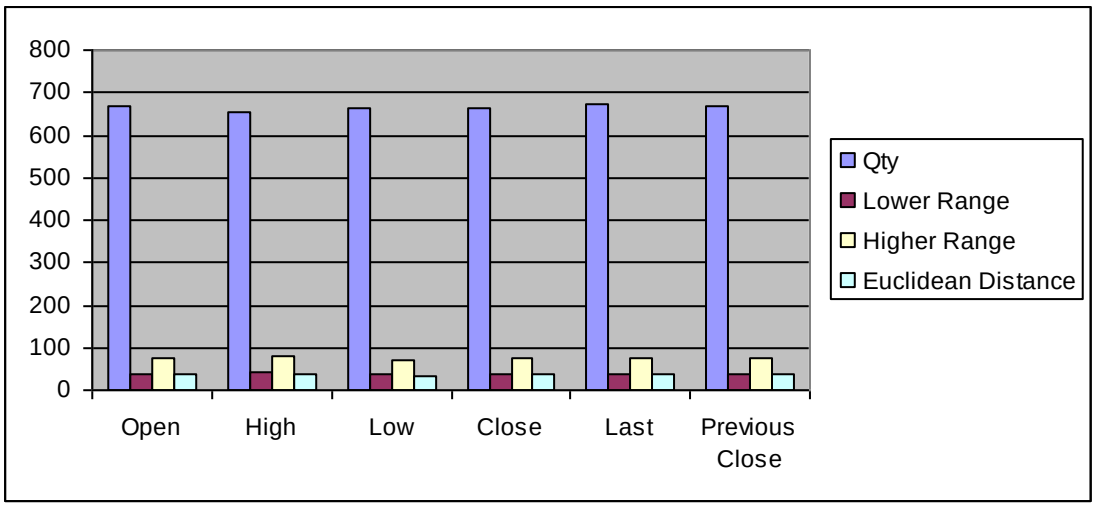
Fig 8.11: Chart of Number of Scripts

Above figure visualize that there are 20 scripts whose trading volume more than 10 million of shares on 26th September 2007, during the period 1st April 2007 to 16th May 2008. Here it is visualized that there are maximum number of scripts are 20.

From Fig 5.28 following data are found, from these data we found some interesting pattern.

Action	Qty	Lower Range	Higher Range	Euclidean Distance
Open	667	38.933	75.567	36.634
High	653	42.181	81.662	39.481
Low	665	36.831	71.412	34.581
Close	663	39.582	76.663	37.081
Last	672	37.563	75.125	37.562
Previous Close	668	39.339	76.427	37.088

From above table following consolidated chart can be generated, it visualized the data graphically.



From above data and chart it can be concluded that for the period 1st April 2007 to 16th May 2008, quantity of scripts for the open rate, high rate, low rate, close rate, last rate and previous close rate does not fluctuated very much. The variation between them is not remarkable; it is proved from Euclidean distance also.

8.2 Rules generated using chart

We can say for total trading quantity of any script is more than 10 million shares per day, maximum number of scripts for open rate, high rate, low rate, close rate, last rate and previous close rate is ranging from 653 to 672, lower range is ranging from 36.831 to 42.181, higher range is ranging from 71.412 to 81.662 and Euclidian distance is ranging from 34.581 to 39.481 also.
It can be concluded that for maximum number of scripts does not have major variations in their rate.

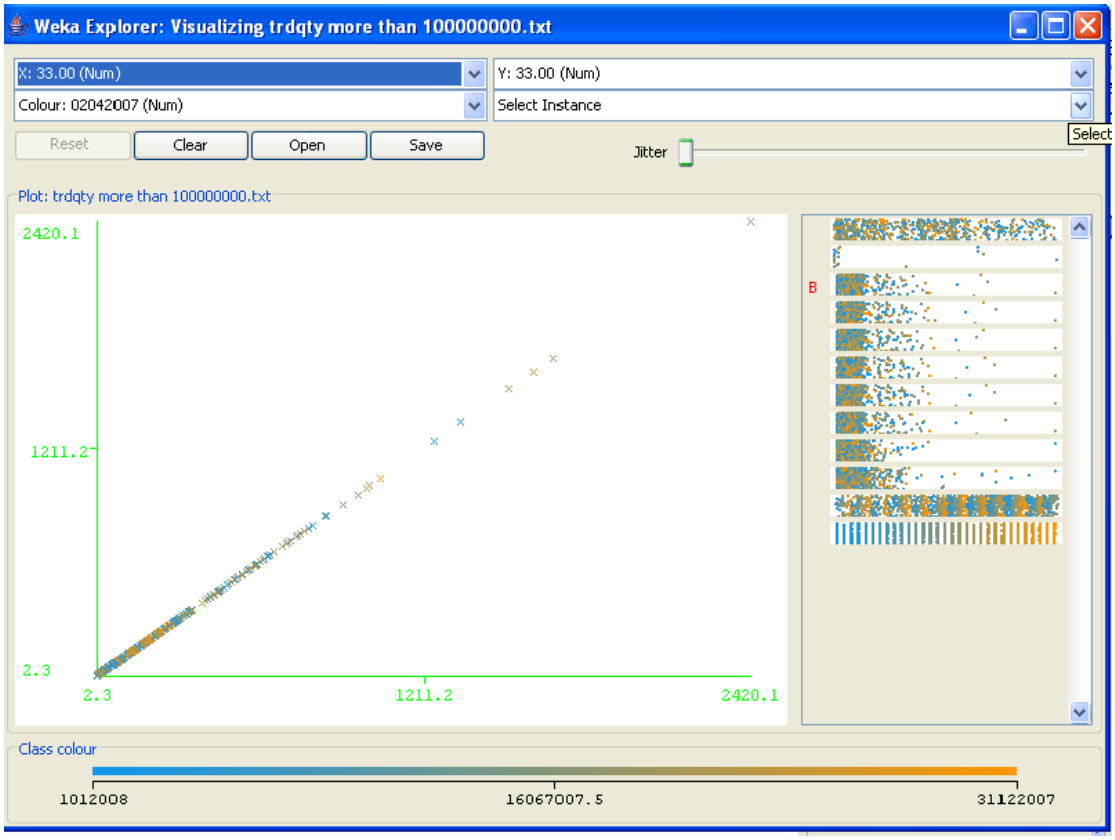


Fig 8.12: Chart of Open Rate and High Rate

Above figure shows the chart for Open rate and High Rate of scripts. On X-axis the data for Open rate is plotted and on Y-axis the data of High Rate is plotted. Below is the data, which is indicated one of the point of the chart. Using this instance information one can generate the rule too.

Following are the examples of results generated by Weka:

Plot: Master Plot

Instance: 192

SCRIPT: MIC
EQ: EQ
OPEN: 262.5
HIGH: 368.0
LOW: 253.0
CLOSE: 338.15
LAST: 337.3
PREVCLOSE: 150.0
TOTALTRQTY: 2.0186521E7
TOTALTRVALUE: 6.8326596157E9
TIMESTAMP: 30-MAY-2007

Plot: Master Plot

Instance: 1040

SCRIPT: SAIL
EQ: EQ
OPEN: 266.0
HIGH: 279.6
LOW: 2262.0
CLOSE: 3276.4
LAST: 275.35
PREVCLOSE: 1262.9
TOTALTRQTY: 1.7033345E7
TOTALTRVALUE: 4.6128225416E9
TIMESTAMP: 29-OCT-2007

From these results we found Euclidean Distance for the various scripts.

Instance	Script	Date	Open	High	Euclidean Distance
192	MIC	30-May-07	362.50	368.00	5.50
1040	SAIL	29-Oct-07	266.00	279.60	13.60
1067	SAIL	31-Oct-07	261.00	267.40	6.40
1081	SAIL	1-Nov-07	264.50	272.00	7.50
1107	RPL	5-Nov-07	271.00	281.80	10.80
1285	SAIL	20-Nov-07	260.00	273.00	13.00
1360	SAIL	27-Nov-07	266.00	272.45	6.45
1434	SAIL	4-Dec-07	264.95	286.85	21.90
1440	GMRINFRA	5-Dec-07	261.25	266.30	5.05
1463	GMRINFRA	6-Dec-07	265.00	269.80	4.80
1484	GMRINFRA	7-Dec-07	259.75	260.00	0.25
1526	SAIL	11-Dec-07	271.00	281.50	10.50
1575	ESSAROIL	18-Dec-07	261.00	298.90	37.90
1613	SAIL	20-Dec-07	263.00	271.35	8.35
1622	SAIL	24-Dec-07	269.70	271.50	1.80
1782	NTPC	8-Jan-08	270.00	272.85	2.85
1795	NTPC	9-Jan-08	266.25	279.40	13.15
1819	NTPC	11-Jan-08	267.10	274.45	7.35
1856	NTPC	16-Jan-08	270.15	270.50	0.35
1866	NTPC	17-Jan-08	259.95	263.25	3.30
1883	ESSAROIL	21-Jan-08	271.00	271.00	0.00
2076	ESSAROIL	16-Apr-08	264.00	274.45	10.45
2189	JPASSOCIAT	30-Apr-08	267.00	274.90	7.90

From above data table it is shown that high rate of the script is always greater than open rate. We can also observe that the script appearing less number of times having higher Euclidean distance in most of cases, these scripts are highlighted.

Script	Number of Occurrence	Variance of Euclidean Distance	Variance of Open rate	Variance of High rate	Euclidean Distance of variance of Open rate and High rate
ESSAROIL	3	383.1858	26.33333	231.3525	205.0192
GMRINFRA	3	7.300833	7.3125	24.66333	17.35083
NTPC	5	25.07	17.17925	35.06425	17.885
SAIL	9	33.1959	13.14944	38.32757	25.17812

Here we can see that Euclidean Distance of variance of open rate and variance of high rate is increasing with the number of occurrence of script, except in the case of ESSAROIL. This is because of zero Euclidean distance of ESSAROIL on 21st Jan 2008. On this date this script may be suspended or there may be some other reason of this. So this script should be excluded from our data set.

Correlation coefficient between variance of open rate and variance of high rate is 0.796727; it means when variance of open rate is increasing; with it variance of high rate is also increasing and number of occurrences of scripts are also increasing.

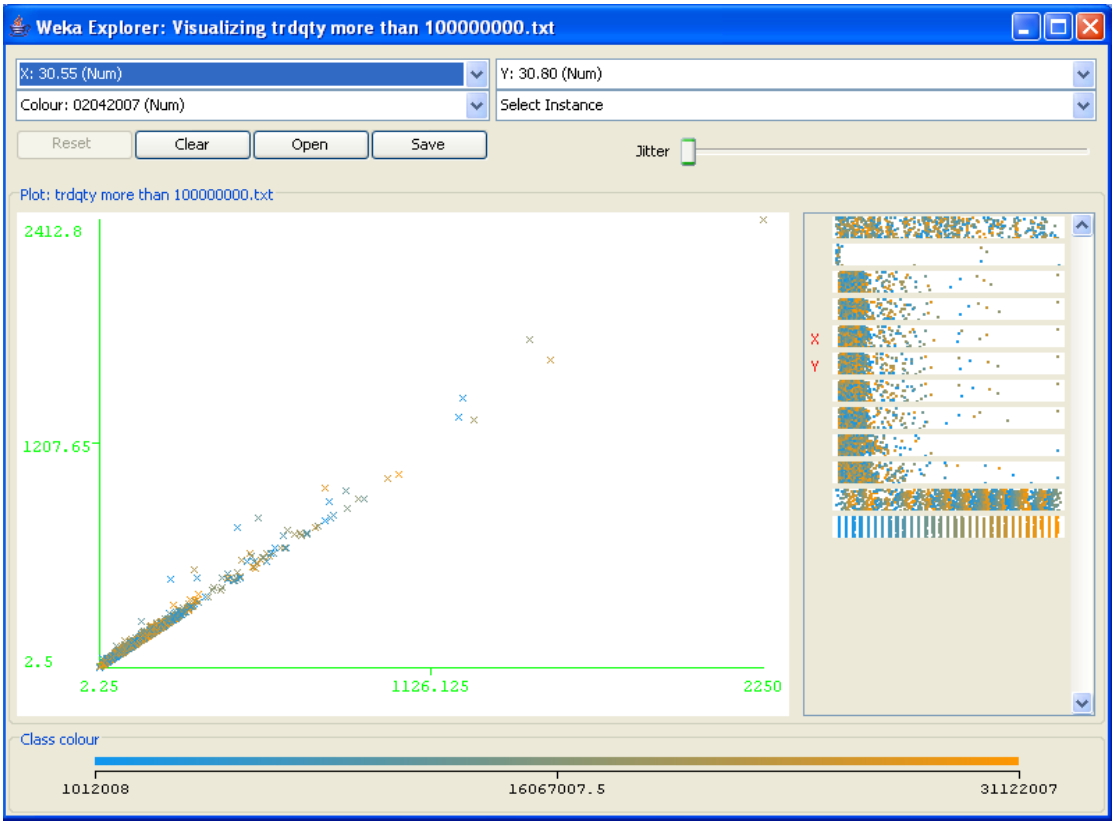


Fig 8.13: Chart of Low Rate and Close Rate

Above figure shows the chart for Low rate and Close Rate of scripts. On X-axis the data for Low rate is plotted and on Y-axis the data of Close Rate is plotted. Below is the data, which is indicated one of the point of the chart. Using this instance information one can generate the rule too.

Following is the example of result generated by Weka:

Plot : Master Plot

Instance: 18

SCRIPT: ORBITCORP
EQ: EQ
OPEN: 113.0
HIGH: 137.6
LOW: 110.15
CLOSE: 128.2
LAST: 129.7
PREVCLOSE: 110.0
TOTALTRQTY: 1.4577737E7
TOTALTRVALUE: 1.8676312227E9
TIMESTAMP: 12-APR-2007

Following table shows the data of above instances generated by Weka.

SCRIPT	LOW	CLOSE	DATE	TRADING QTY
ORBITCORP	119.15	128.20	12-Apr-07	14577737.00
INDOWIND	120.00	130.55	17-Sep-07	14840899.00
TRIVENI	110.55	136.15	19-Sep-07	19126004.00
TORNTPOWER	113.20	128.70	5-Oct-07	10532720.00
IDBI	118.00	136.80	17-Oct-07	10390131.00
POWERGRID	106.20	136.80	17-Oct-07	115242925.00
MRPL	106.20	128.05	15-Nov-07	49015305.00
MRPL	120.00	130.40	16-Nov-07	44919996.00
GMRINFRA	111.10	156.05	22-Jan-08	18947204.00
RPL	107.25	147.00	22-Jan-08	44940240.00

We calculate Euclidean Distance between LOW and CLOSE rate in following table.

SCRIPT	LOW	CLOSE	TRADING QTY	Euclidean Distance of LOW and CLOSE
ORBITCORP	119.15	128.20	14577737.00	9.05
INDOWIND	120.00	130.55	14840899.00	10.55
TRIVENI	110.55	136.15	19126004.00	25.60
TORNTPOWER	113.20	128.70	10532720.00	15.50
IDBI	118.00	136.80	10390131.00	18.80
POWERGRID	106.20	136.80	115242925.00	30.60
MRPL	106.20	128.05	49015305.00	21.85
MRPL	120.00	130.40	44919996.00	10.40
GMRINFRA	111.10	156.05	18947204.00	44.95
RPL	107.25	147.00	44940240.00	39.75

Correlation Coefficient between Trading Quantity and Euclidean Distance between Low and Close is 0.292344. It means there is not much effect on fluctuation of Trading Quantity on Euclidean Distance between Low and Close. This can be said for different scripts for non-consecutive dates.

We can observe the data of 15th Nov. 2007 and 16th Nov. 2007 for the script MRPL. There is major difference in the Euclidean distance of both dates. When trading quantity is higher, Euclidean distance is also higher. So it can be concluded that when fluctuation in low and close rate is more, trading quantity is also increased. This can be said for single script for consecutive dates.

8.3 Analysis of Data using Weka

Total 12 attributes like Script, Equity, Open Rate, High Rate, Low Rate, Close Rate, Last Rate, Previous Close, Total Trading Quantity, Total Revenue, Time Stamps, etc. are used for analysis of data using Weka.

There are various Test modes under various schemes. Examples of some scheme and test modes related to it are given as below:

Scheme: weka.classifiers.rules.ZeroR

Relation: trdqty more than 100000000.txt

Instances: 2308

Attributes: 12

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

ZeroR predicts class value: 1.6170382486568458E7

Time taken to build model: 0.02 seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient	-0.0614
Mean absolute error	7627934.2008
Root mean squared error	8846610.1455
Relative absolute error	100 %
Root relative squared error	100 %
Total Number of Instances	2308

Scheme: weka.classifiers.rules.ZeroR

Relation: trdqty more than 100000000.txt

Instances: 2308

Attributes: 12

Test mode: split 66% train, remainder test

=== Classifier model (full training set) ===

ZeroR predicts class value: 1.6170382486568458E7

Time taken to build model: 0 seconds

=== Evaluation on test split ===

=== Summary ===

Correlation coefficient	0
Mean absolute error	7691482.3211
Root mean squared error	8906977.6853
Relative absolute error	100 %
Root relative squared error	100 %
Total Number of Instances	785

Scheme: weka.classifiers.rules.ZeroR

Relation: trdqty more than 100000000.txt

Instances: 2308

Attributes: 12

Test mode: Evaluate on training data

=== Classifier model (full training set) ===

ZeroR predicts class value: 1.6170382486568458E7

Time taken to build model: 0 seconds

=== Evaluation on training set ===

=== Summary ===

Correlation coefficient	0
Mean absolute error	7623738.3568
Root mean squared error	8842698.919
Relative absolute error	100 %
Root relative squared error	100 %
Total Number of Instances	2308

Followings are algorithms and its related test modes as listed below:

Scheme:weka.classifiers.rules.DecisionTable-X1-S "weka.attributeSelection.BestFirst -D 1 -N 5"

Test mode: 10-fold cross-validation

Test mode: split 66% train, remainder test

Test mode: evaluate on training data

Evaluator: weka.attributeSelection.CfsSubsetEval

Search: weka.attributeSelection.BestFirst -D 1 -N 5

Evaluation mode: evaluate on all training data

Scheme: weka.classifiers.trees.DecisionStump

Test mode: evaluate on training data

Test mode: 10-fold cross-validation

Test mode: split 50% train, remainder test

Scheme: weka.clusterers.SimpleKMeans -N 2 -S 10

Test mode: evaluate on training data

Scheme: weka.classifiers.rules.M5Rules -M 4.0

Test mode: 10-fold cross-validation

Scheme: weka.classifiers.rules.ConjunctiveRule -N 3 -M 2.0 -P -1 -S 1

Test mode: evaluate on training data

Test mode: split 50% train, remainder test

8.4 Decision Tree using Analysis services

A decision tree is a form of classification shown in a tree structure, in which a node in the tree structure represents each question used to further classify data. The various methods used to create decision trees have been used widely for decades, and there is a large body of work describing these statistical techniques.

The algorithm builds a tree that will predict the value of a column based upon the remaining columns in the training set. Therefore, each node in the tree represents a particular case for a column. The decision on where to place this node is made by the algorithm, and a node at a different depth than its siblings may represent different cases of each column.

8.5 Data Cluster Using the Analysis services

Like decision trees, clustering is a well-documented data mining technique. Clustering is the classification of data into groups based on specific criteria. The topic discussing the Microsoft Clustering algorithm goes into greater detail regarding the details of clustering as a data mining technique. The Clustering algorithm is an expectation method that uses iterative refinement techniques to group records into neighborhoods (clusters) that exhibit similar, predictable characteristics. Often, these characteristics may be hidden or non-intuitive.

8.6 Analyzing the Visual Result

As shown in figures, by seeing one can understand the patterns in the data and also you can compare one subject marks with other subject marks in a graphical way and find the particular point (instance) information.

8.7 Weka Classifier Result

Here researcher has applied the different data-mining algorithm for classification using the different test mode.

The test modes are

- **Using training set:** The classifier is evaluated on how well it predicts the class of the instances it was trained on.
- **Supplied test set:**: The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file.
- **Cross-validation:** The classifier is evaluated by cross-validation, using the number of folds.
- **Percentage split:** The classifier is evaluated on how well it predicts a certain percentage of the data, which is held out for testing.

The result table shows the summary, a list of statistics summarizing how accurately the classifier was able to predict the true class of the instances under the chosen test mode.

Correlation coefficient shows the strength of relationship between variables. So

if it is high then it will give a good result.

Absolute error shows difference between a measurement and its true value. So if this value is low, that indicate good algorithm. The low value of root mean square error also indicates the good result.

From the following table we can select the best algorithm based on the given parameter.

8.8 Comparison of Various Algorithms

Algorithm	Test Mode	Correlation Coefficient	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Root Relative Squared Error	Number of Instances
Zero R	10-fold cross-validation	-0.0614	7627934	8846610.1	100%	100%	2308
Zero R	split 66% train, remainder test	0	7691482	8906977.7	100%	100%	785
Zero R	Evaluate on training data	0	7623738	8842698.9	100%	100%	2308
DecisionTable -X 1 -S	10-fold cross-validation	0.9984	20223.27	497863.48	0.27%	5.63%	2308
DecisionTable -X 1 -S	split 66% train, remainder test	0.9903	157944.1	1240456.7	2.05%	13.93%	785
DecisionTable -X 1 -S	Evaluate on training data	1	0	0	0%	0%	2308

DecisionStump	Evaluate on training data	0.1459	7508151	8748108.1	98.48%	98.93%	2308
DecisionStump	10-fold cross-validation	0.08	7589317	8814612.5	99.49%	99.64%	2308
DecisionStump	split 50% train, remainder test	0.1114	7520474	8736410	99.15%	99.38%	1154
M5Rules -M 4.0	10-fold cross-validation	0.9988	2.6281	6.5352	3.42%	5.01%	2308
ConjunctiveRule -N 3 -M 2.0 -P -1 -S 1	Evaluate on training data	0.7428	62.2453	87.3115	81.16%	66.97%	2308
ConjunctiveRule -N 3 -M 2.0 -P -1 -S 2	split 50% train, remainder test	0.7296	62.834	81.7491	82.41%	68.93%	1154

Fig 8.14: Comparison of Algorithms

9. Introduction of various Data Mining and Data Warehousing Tools

9.1 SQL Server 2008 Data Mining Features

Analysis Services

Analysis Services delivers extensions to the scalability, manageability, reliability, availability, and programmability of data warehousing, business intelligence, and line-of-business solutions.

Data Transformation Services (DTS)

A complete redesign of the DTS architecture and tools provides developers and database administrators with increased flexibility and manageability.

Reporting Services

Reporting Services is a new report server and tool set for building, managing, and deploying enterprise reports.

Data Mining

Data mining is enhanced with four new algorithms as well as improved data modeling and manipulation tools.

Create an easy–to–use, extensible, accessible, and flexible business intelligence platform and take the next step in business intelligence with SQL Server data-mining capabilities. Explore your data, discover patterns, and uncover business data to reveal the hidden trends about your products, customer, market, and employees, and better analyze those components that are critical to your organization's success.

Integration

SQL Server Data Mining is part of a family of business intelligence technologies that can be used together to enhance and develop a new breed of intelligent applications. These technologies include the following:

- **SQL Server 2008 Integration Services.** Create a more powerful data pipeline by working with SQL Server 2005 Integration Services, allowing

your organization to flag outliers, separate data, and fill in missing values based on the predictive analytics of the data-mining algorithms.

- **SQL Server 2008 Analysis Services.** Create a richer Unified Dimensional Model by adding data-mining dimensions that slice your data by the hidden patterns within.

- **SQL Server Reporting Services.** Create smarter, insightful reports based on data-mining queries that present the right information to the right audiences.

Architecture

Providing data mining to organizations of any size introduces new challenges. Deployment, scalability, manageability, and security all become important factors. SQL Server Data Mining is part of the SQL Server Analysis Services server that provides all the enterprise-class server features you would expect:

- **Deployment.** SQL Server Data Mining is based on client- server architecture, allowing you to access models from your local area network (LAN), wide area network (WAN), or the Internet. Standard application programming interfaces (APIs) provide access to your models regardless of location or client platform.

- **Scalability.** SQL Server Data Mining is designed from the ground up with a parallel architecture to scale to enterprise-class data sets and thousands of concurrent users, and can respond to millions of queries per day.
- **Manageability.** SQL Server Data Mining is integrated into the new SQL Server Management Studio, providing a one-stop tool for managing all your SQL Server family properties.

- **Security.** SQL Server Data Mining provides fine-grained, role- based security to ensure that your intellectual property will be further protected.

Extensibility

SQL Server Data Mining is fully extensible through Microsoft .NET– stored procedures and plug-in algorithms and viewers that embed seamlessly to take advantage of all the platform abilities and integration. Adopting SQL Server Data Mining as your platform means that you will never be limited by the inherent functionality of your data-mining system because it can always be extended to meet your needs.

9.2 DB2 Intelligent Miner

IBM's data mining capabilities help you detect fraud, segment your customers, and simplify market basket analysis. IBM's in-database mining capabilities integrate with your existing systems to provide scalable, high performing predictive analysis without moving your data into proprietary data mining platforms. Use SQL, Web Services, or Java to access DB2's data mining capabilities from your own applications or business intelligence tools from IBM's business partners.

Tools

- **DB2 Intelligent Miner Modeling**

  Delivers DB2 Extenders for modeling operations

- **DB2 Intelligent Miner for Scoring**

  Provides scoring technology as database extensions: DB2 Extenders and Oracle cartridges

- **DB2 Intelligent Miner Visualization**

  Provides Java visualizes to Interact and graphically present the results of associations

- **DB2 Intelligent Miner for Data**

  Provides new business insights and harvests valuable business intelligence from your enterprise data.

Tool Benefits

- **Using PMML**

  Share data mining data using the industry standard PMML

- **Scalable Mining**

  Intelligent Miner tools integrate seamlessly with DB2 UDB

9.3 Informatica PowerCenter Advanced Edition

One of the biggest challenges facing organizations today is the fragmentation of data across disparate IT systems. Unlocking and deriving business value from these strategic data assets — no matter where they reside — has become a top priority.

Companies are realizing that in order to support their business objectives—such as providing a single view of the customer, migrating away from legacy systems to new technology, or consolidating multiple instances of an ERP system—they must be able to effectively integrate, move and access their data, or its business value will be lost.

Informatica PowerCenter Advanced Edition, the leading independent data integration platform, and addresses this need—delivering the industry's most comprehensive set of capabilities for enterprise-wide data integration. PowerCenter Advanced Edition supports the entire data integration lifecycle—allowing companies to access, discover, and integrate data from the widest variety of enterprise systems and deliver that data to other operational systems, applications, databases and to the business user for decision making.

9.4 Business Object

BusinessObjects XI Release 2 provides performance management, reporting, query and analysis, and data integration in one solution.

- Performance management - Match actions with strategy.

- Reporting - Access, format, and deliver data.
- Query and analysis - Self-serve analysis for users.
- BI platform - Manage BI tools, reports, and applications.
- Data integration - Access, transform, and integrate data.

9.5 Cognos 8 Business Intelligence

Cognos 8 Business Intelligence is the only BI product to deliver the complete range of BI capabilities: reporting, analysis, Scorecarding, dashboards, business event management as well as data integration, on a single, proven architecture.

Easy to integrate, deploy and use, Cognos 8 BI delivers a simplified BI environment that improves user adoption, enables better decision- making, and serves as an enterprise-scale foundation for performance management.

Reporting

Reporting is a key capability within Cognos 8 Business Intelligence, a single product that provides complete BI capabilities on a proven architecture.

Reporting gives you access to a complete list of self-serve report types, is adaptable to any data source, and operates from a single metadata layer for a variety of benefits such as multilingual reporting.

Analysis

Analysis is a key capability within Cognos 8 Business Intelligence, a single product that provides complete BI capabilities on a proven architecture.

Analysis enables the guided exploration and analysis of information that pertains to all dimensions of your business-regardless of where the data is stored. Analyze and report against online analytical processing (OLAP) and dimensionally aware relational sources.

Scorecarding

Scorecarding is a key capability within Cognos 8 Business Intelligence, a single product that provides complete BI capabilities on a proven architecture. Scorecarding helps you align your teams and tactics with strategy; communicate goals consistently, and monitor performance against targets.

Dashboards

Business dashboards communicate complex information quickly. They translate information from your various corporate systems and data into visually rich presentations using gauges, maps, charts, and other graphical elements to show multiple results together.

Business Event Management

Cognos 8 BI business event management tracks significant events that need attention. It monitors these events and uses decision- process and business-process automation to compress the time to action and resolution.

Data Integration

Data integration is a key component within Cognos 8 Business Intelligence, a single product that provides complete BI capabilities on a proven architecture.

Cognos data integration is an enterprise-wide ETL solution designed for high performance business intelligence. It optimizes data merging, extraction, transformation, and dimensional management to deliver data warehouses ready for business reporting and analysis.

9.6 Comparison of Data Mining Tools

Elder Research Inc. is a company engaged with Data Mining and pattern discovery. This group has evaluated and compared various data mining tools.

Product	Company	URL	Version Tested	Our Experience
Clementine	Integral Solutions, Ltd.	http://www.isl.co.uk/clem.html	4	Moderate
Darwin	Thinking Machines, Corp.	http://www.think.com/html/products/products.htm	3.0.1	Moderate
DataCruncher	DataMind	http://www.datamindcorp.com	2.1.1	High
Enterprise Miner	SAS Institute	http://www.sas.com/software/components/miner.html	Beta	Moderate
GainSmarts	Urban Science	http://www.urbanscience.com/main/gainpage.htm	4.0.3	Low
Intelligent Miner	IBM	http://www.software.ibm.com/data/iminer/	2	Low
MineSet	Silicon Graphics, Inc.	http://www.sgi.com/Products/software/MineSet/	2.5	Low
Model 1	Group 1/Unica Technologies	http://www.unica-usa.com/model1.htm	3.1	Moderate
ModelQuest	AbTech Corp.	http://www.abtech.com	1	Moderate
PRW	Unica Technologies, Inc.	http://www.unica-usa.com/prodinfo.htm	2.1	High
CART	Salford Systems	http://www.salford-systems.com	3.5	Moderate
NeuroShell	Ward Systems Group, Inc.	http://www.wardsystems.com/neuroshe.htm	3	Moderate
OLPARS	PAR Government Systems	mailto://olpars@partech.com	8.1	High
Scenario	Cognos	http://www.cognos.com/busintell/products/index.html	2	Moderate
See5	RuleQuest Research	http://www.rulequest.com/see5-info.html	1.07	Moderate
S-Plus	MathSoft	http://www.mathsoft.com/splus/	4	High
WizWhy	WizSoft	http://www.wizsoft.com/why.html	1.1	Moderate

Fig 9.1: Comparison of Various Tools