RAO'S DEGREE &PG COLLEGE::NELLORE

DATA MINING AND WAREHOUSING

FOR

MCA II SEMESTER

(Source of Grasping Subject & Gaining Marks)

"A prestigious COLLEGE for VICTORIES "

→ → → P. Lohith Kumar

UNIT-I

What Is Data Mining?

What Is Data Mining?

Def1: Data mining refers the process or method that extracts or "mines" interesting knowledge or patterns from large amounts of data.

Def2: Data mining is the task of discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses, or other information repositories.

Data mining has improved organizational decision-making through insightful data analyses. The data mining techniques that underpin these analyses can be divided into two main purposes; they can either describe the target dataset or they can predict outcomes through the use of machine learning algorithms.

The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

Data Mining Tasks:

Data mining tasks are majorly categorized into two categories: descriptive and predictive.

1. **Descriptive data mining:**
   Descriptive data mining offers a detailed description of the data, for example- it gives insight into what's going on inside the data without any prior idea.
2. **Predictive Data Mining**:
   This allows users to consider features that are not specifically available. For example, the projection of the market analysis in the next quarters with the output of the previous quarters, In general, the predictive analysis forecasts or infers the features of the data previously available.

Characterization and Discrimination

- **Data Characterization**: The characterization of data is a description of the general characteristics of objects in a target class which creates what are called characteristic rules.

- Data Discrimination: Data discrimination creates a series of rules called discriminate rules that is simply a distinction between the two classes aligned with the goal class and the opposite class of the general characteristics of objects.

Prediction

To detect the inaccessible data, it uses regression analysis and detects the missing numeric values in the data. The prediction of the class mark using the previously developed class model and the prediction of incomplete or incomplete data using prediction analysis are two ways of predicting data.

Classification

Classification is used to create data structures of predefined classes, as the model is used to classify new instances whose classification is not understood. The instances used to produce the model are known as data from preparation.

Association Analysis

The link between the data and the rules that bind them is discovered. And two or more data attributes are associated. It associates qualities that are transacted together regularly. They work out what are called the rules of partnerships that are commonly used in the study of stock baskets.

Outlier Analysis

Data components that cannot be clustered into a given class or cluster are outliers. They are often referred to as anomalies or surprises and are also very important to remember.

Cluster Analysis

Clustering is the arrangement of data in groups. Unlike classification, however, class labels are undefined in clustering and it is up to the clustering algorithm to find suitable classes. Clustering is often called unsupervised classification since provided class labels do not execute the classification.

CHALLENGES OF DATA MINING:-

Data mining has improved organizational decision-making through insightful data analyses. The data mining techniques that underpin these analyses can be divided into two main purposes; they can either describe the target dataset or they can predict outcomes through the use of machine learning algorithms.

Some of the Data mining challenges are given as under:

- Security and Social Challenges
- Noisy and Incomplete Data

- Distributed Data
- Complex Data
- Performance
- Scalability and Efficiency of the Algorithms
- Improvement of Mining Algorithms
- Incorporation of Background Knowledge
- Data Visualization
- Data Privacy and Security
- User Interface
- Mining dependent on Level of Abstraction
- Integration of Background Knowledge
- Mining Methodology Challenges

a)Security and Social Challenges

Dynamic techniques are done through data assortment sharing, so it requires impressive security.

b)Noisy and Incomplete Data

Data Mining is the way toward obtaining information from huge volumes of data. This present reality information is noisy, incomplete, and heterogeneous. Data in huge amounts regularly will be unreliable or inaccurate.

c) Distributed Data

True data is normally put away on various stages in distributed processing conditions. It very well may be on the internet, individual systems, or even on the databases. It is essentially hard to carry all the data to a unified data archive principally because of technical and organizational reasons.

d)Complex Data

True data is truly heterogeneous, and it very well may be media data, including natural language text, time series, spatial data, temporal data, complex data, audio or video, images, etc.

e) Performance

The presentation of the data mining framework basically relies upon the productivity of techniques and algorithms utilized. On the off chance that the techniques and algorithms planned are not sufficient; at that point, it will influence the presentation of the data mining measure unfavorably.

f) Scalability and Efficiency of the Algorithms

The Data Mining algorithm should be scalable and efficient to extricate information from tremendous measures of data in the data set.

g) Improvement of Mining Algorithms
Factors, for example, the difficulty of data mining approaches, the enormous size of the database, and the entire data flow inspire the distribution and creation of parallel data mining algorithms.

h) Incorporation of Background Knowledge
In the event that background knowledge can be consolidated, more accurate and reliable data mining arrangements can be found. Predictive tasks can make more accurate predictions, while descriptive tasks can come up with more useful findings.

i) Data Privacy and Security
Data mining typically prompts significant issues regarding governance, privacy, and data security. For instance, when a retailer investigates the purchase details, it uncovers information about purchasing propensities and choices of customers without their authorization.

j) User Interface
The knowledge is determined utilizing data mining devices is valuable just in the event that it is fascinating or more all reasonable by the client.

k)Mining dependent on Level of Abstraction
Data mining measure should be community-oriented in light of the fact that it permits clients to focus on example optimizing, presenting, and pattern finding for data mining dependent on brought results back.

l)Integration of Background Knowledge
Previous information might be utilized to communicate examples to express discovered patterns and to direct the exploration processes.

m)Mining Methodology Challenges
These difficulties are identified with data mining methods and their limits. Mining methods that cause the issue are the control and handling of noise in data, the dimensionality of the domain, diversity of data available, versatility of the mining method, and so on.

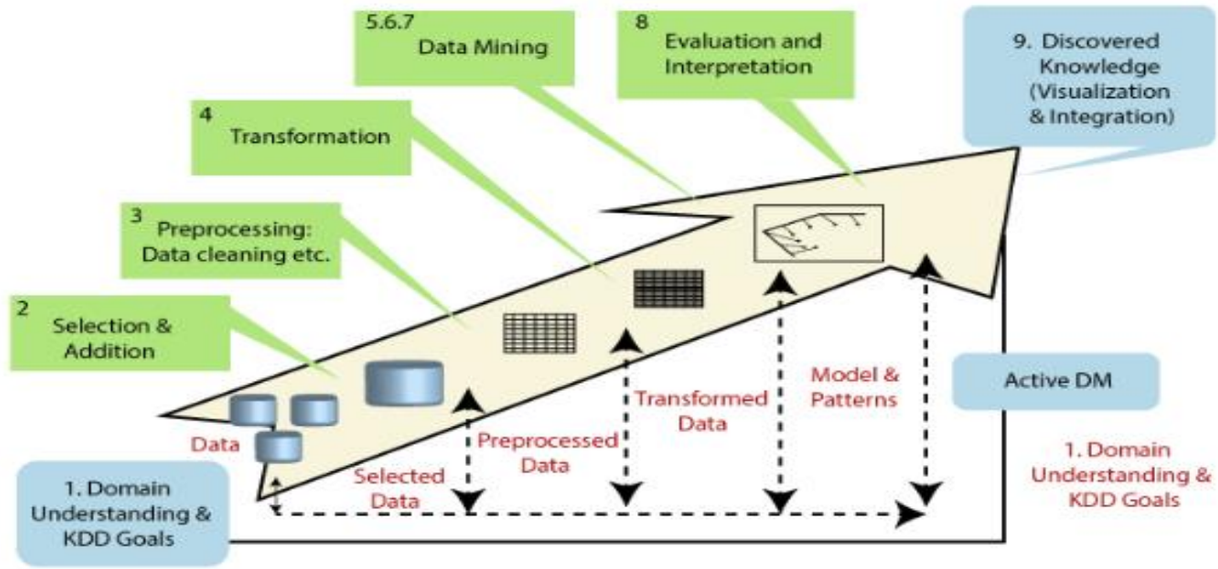
KDD(Knowledge discovery database):

The main objective of the KDD process is to extract information from data in the context of large databases.

The KDD Process

The knowledge discovery process (illustrates in the given figure) is iterative and interactive, comprises of nine steps. The process is iterative at

each stage, implying that moving back to the previous actions might be required. The process has many imaginative aspects in the sense that one cant presents one formula or make a complete scientific categorization for the correct decisions for each step and application type. Thus, it is needed to understand the process and the different requirements and possibilities in each stage. Following is a concise description of the nine-step KDD process, beginning with a managerial step:



1. Building up an understanding of the application domain

This is the initial preliminary step. It develops the scene for understanding what should be done with the various decisions like transformation, algorithms, representation, etc.

2. Choosing and creating a data set on which discovery will be performed

Once defined the objectives, the data that will be utilized for the knowledge discovery process should be determined. This process is important because of Data Mining learns and discovers from the accessible data. This is the evidence base for building the models.

3. Preprocessing and cleansing

In this step, data reliability is improved. It incorporates data clearing, for example, Handling the missing quantities and removal of noise or outliers. It might include complex statistical techniques or use a Data Mining algorithm in this context.

4. Data Transformation

In this stage, the creation of appropriate data for Data Mining is prepared and developed. Techniques here incorporate dimension reduction (for example, feature selection and extraction and record sampling), also attribute transformation(for example, discretization of numerical attributes and

functional transformation). This step can be essential for the success of the entire KDD project, and it is typically very project-specific.

5. Prediction and description

We are now prepared to decide on which kind of Data Mining to use, for example, classification, regression, clustering, etc. This mainly relies on the KDD objectives, and also on the previous steps. There are two significant objectives in Data Mining, the first one is a prediction, and the second one is the description.

6. Selecting the Data Mining algorithm

Having the technique, we now decide on the strategies. This stage incorporates choosing a particular technique to be used for searching patterns that include multiple inducers. For example, considering precision versus understandability, the previous is better with neural networks, while the latter is better with decision trees.

7. Utilizing the Data Mining algorithm

At last, the implementation of the Data Mining algorithm is reached. In this stage, we may need to utilize the algorithm several times until a satisfying outcome is obtained. For example, by turning the algorithms control parameters, such as the minimum number of instances in a single leaf of a decision tree.

8. Evaluation: In this step, we assess and interpret the mined patterns, rules, and reliability to the objective characterized in the first step. Here we consider the preprocessing steps as for their impact on the Data Mining algorithm results.

9. Using the discovered knowledge

Now, we are prepared to include the knowledge into another system for further activity. The knowledge becomes effective in the sense that we may make changes to the system and measure the impacts. The accomplishment of this step decides the effectiveness of the whole KDD process.

Data preprocessing:

Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user.

Data Preprocessing

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Why is Data preprocessing important?

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following

Accuracy: To check whether the data entered is correct or not.

Completeness: To check whether the data is available or not recorded.

Consistency: To check whether the same data is kept in all the places that do or do not match.

Timeliness: The data should be updated correctly.

Believability: The data should be trustable.

Interpretability: The understandability of the data.

Data Cleaning Process:

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

1. Missing values:

The various methods for handling the problem of missing values in data touples include:

(a) Ignoring the tuple:

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

(b) Manually filling in the missing value:

In general, this approach is time-consuming and may not be a reasonable task for large data sets with many missing values, especially when the value to be filled in is not easily determined.

(c) Using a global constant to fill in the missing value:

(d) Using the attribute mean for quantitative (numeric) values or attribute mode for categorical (nominal) values:

Replace all missing attribute values by the same constant, such as a label like "Unknown," or $-\infty$. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept,

since they all have a value in common — that of "Unknown." Hence, although this method is simple, it is not recommended.

For example, suppose that the average income of All Electronics customers is $28,000. Use this value to replace any missing values for income.

(e) Using the attribute mean for quantitative (numeric) values or attribute mode for categorical (nominal) values, for all samples belonging to the same class as the given tuple:

For example, if classifying customers according to credit risk, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple.

(f) Using the most probable value to fill in the missing value:

This may be determined with regression, inference-based tools using Bayesian formalism, or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for income.

2. Noisy Data:

Noise is a random error or variance in a measured variable. Given a numerical attribute such as, say, price, how can we "smooth" out the data to remove the noise.

Let's look at the following data smoothing techniques:

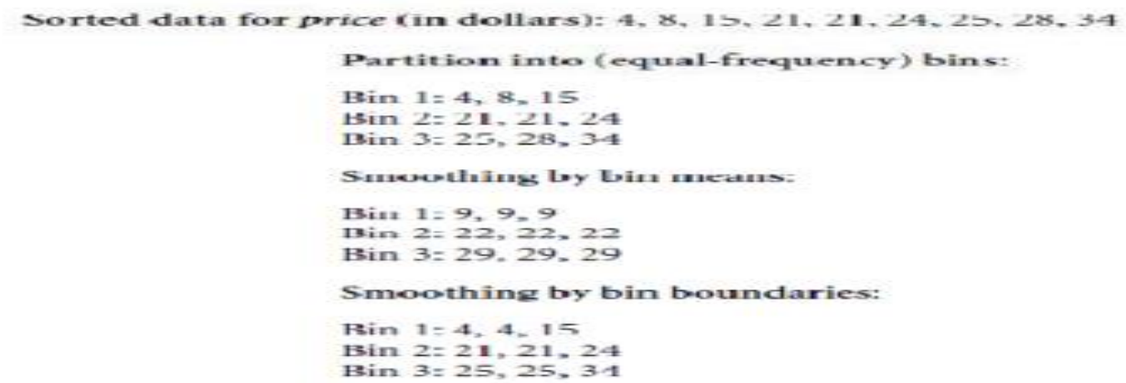
- Binning:
- Regression:
- Clustering:

Binning:

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

Smoothing by bin medians can be employed, in which each bin value is replaced by the bin median.

In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries.



Regression:

Data can be smoothed by fitting the data to a function, such as with regression. Linear regression involves finding the "best" line to fit two attributes (or variables), so that one attribute can be used to predict the other.

Clustering:

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

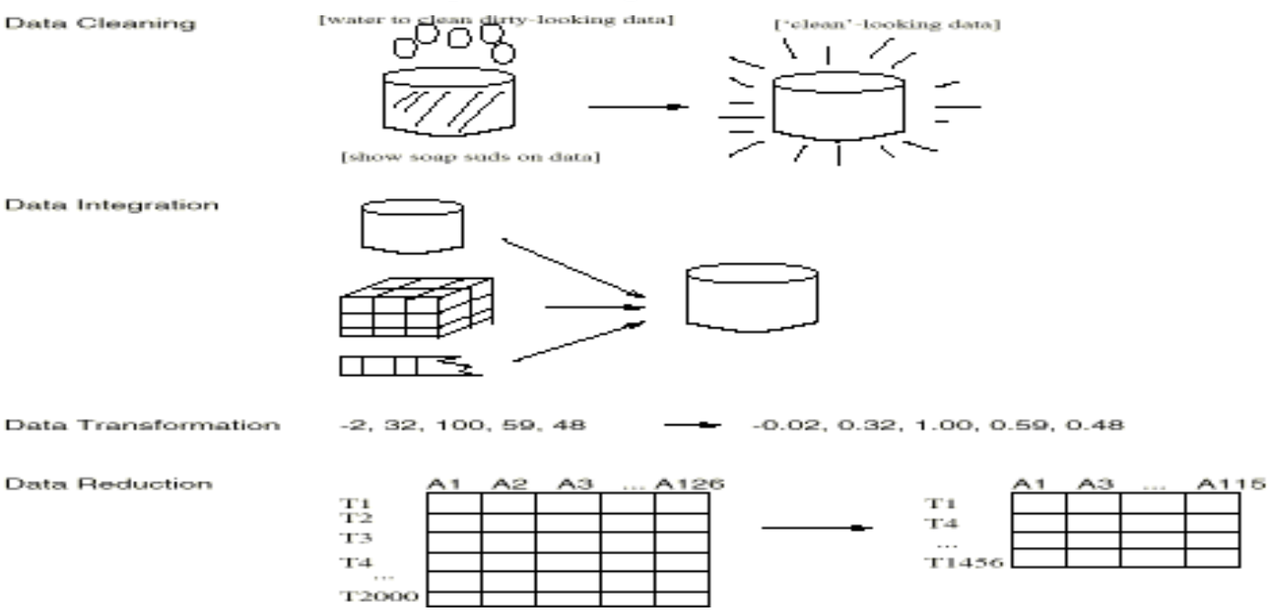
set of clusters may be considered outliers.

Major Tasks in Data Pre-processing

Major tasks in data pre-processing are data cleaning, data integration, data transformation, data reduction and data discretization.

- Data cleaning
  - Data Cleaning includes, filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- Data integration
  - Data Integration includes integration of multiple databases, data cubes, or files.
- Data transformation
  - Data Transformation includes normalization and aggregation.
- Data reduction
  - Data reduction is achieved by obtaining reduced representation of data in volume but produces the same or similar analytical results.
- Data Discretization
  - Data Discretization is part of data reduction but with particular importance, especially for numerical data.

Forms of Data Pre-processing

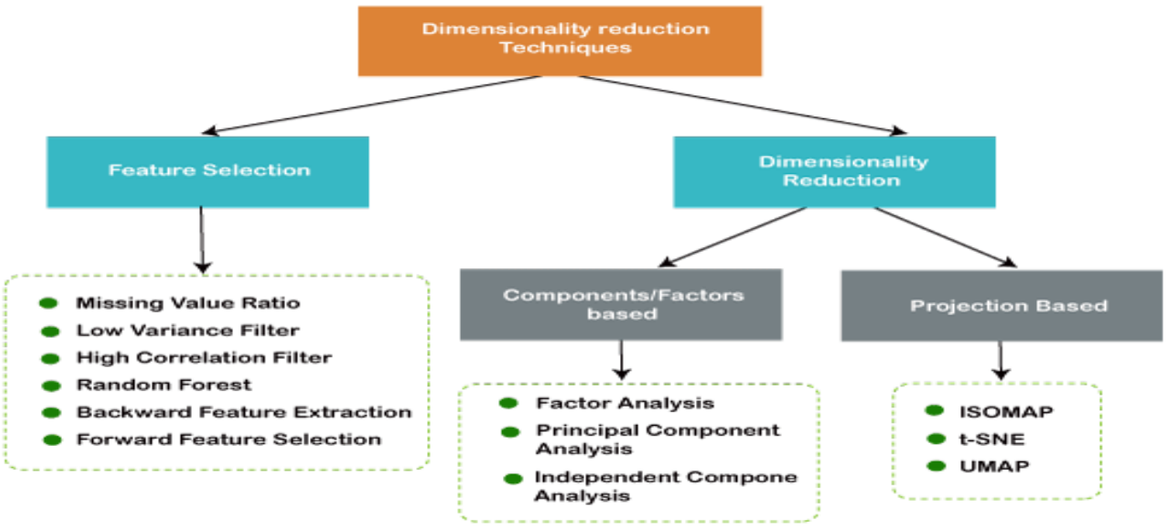


Dimensionality reduction:

What is Dimensionality Reduction?

The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.

Dimensionality reduction technique can be defined as, *"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."* These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.



Benefits of applying Dimensionality Reduction

Some benefits of applying dimensionality reduction technique to the given dataset are given below:

- By reducing the dimensions of the features, the space required to store the dataset also gets reduced.
- Less Computation training time is required for reduced dimensions of features.
- Reduced dimensions of features of the dataset help in visualizing the data quickly.
- **I**t removes the redundant features (if present) by taking care of multicollinearity.

Feature subset selection:

One of the major challenges these days is dealing with large amount of data extracted from the network that needs to be analyzed. Feature Selection plays the very important role in Intrusion Detection System.

Due to availability of large amounts of data from the last few decades, the analysis of data becomes more difficult manually. So the data analysis should be done computerized through Data Mining. Data mining helps in fetching the hidden attributes on the basis of pattern, rules, so on. Data Mining is the only hope for clearing the confusion of patterns.

Feature selection [FS] is the processes that choose a subset of relevant features for building the model. Feature selection is one of the frequently used and most important techniques in data preprocessing for data mining.

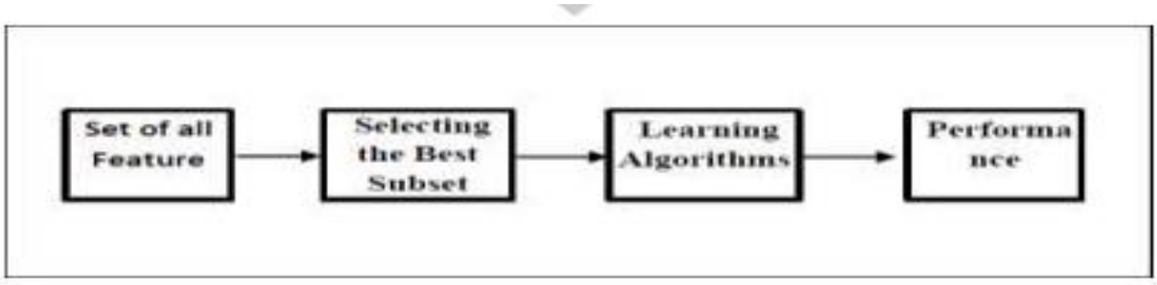
FEATURE SELECTION AND ITS METHODS:

Data holds many features, but all the features may not be related so the feature selection is used so as to eliminate the unrelated features from the data without much loss of the information. Feature selection is also known as attributes selection or variable selection. The feature selection is of three types:

- Filter approach
- Wrapper approach
- Embedded approach

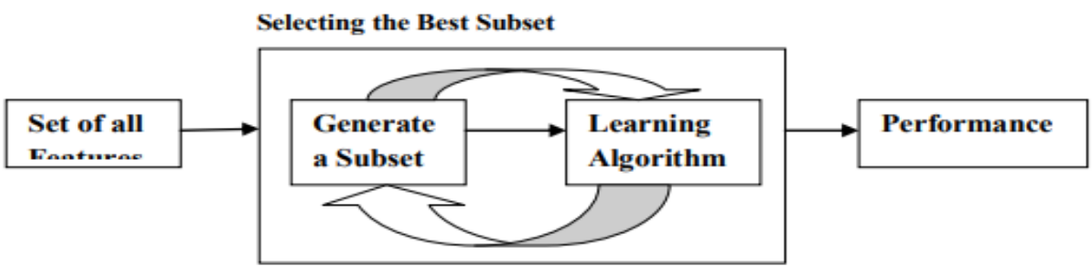
Filter approach

Filter approach or Filter method shown in Figure. This method selects the feature without depending upon the type of classifier used. The advantage of this method is that, it is simple and independent of the type of classifier used so feature selection need to be done only once and drawback of this method is that it ignores the interaction with the classifier, ignores the feature dependencies, and lastly each feature considered separately.



Wrapper approach

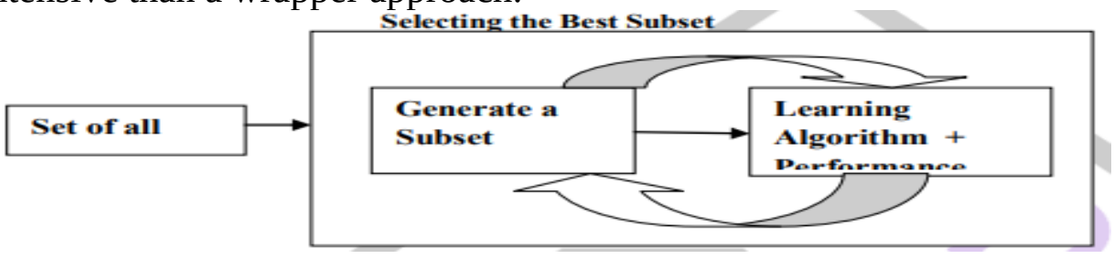
Wrapper approach or Wrapper method is shown in Figure. In this method the feature is dependent upon the classifier used, i.e. it uses the result of the classifier to determine the goodness of the given feature or attribute. The advantage of this method is that it removes the drawback of the filter method, i.e. it includes the interaction with the classifier and also takes the feature dependencies and drawback of this method is that it is slower than the filter method because it takes the dependencies also. The quality of the feature is directly measured by the performance of the classifier.



Embedded approach

The embedded approach or embedded method is shown in Figure. It searches for an optimal subset of features that is built into the classifier

construction. The advantage of this method is that it is less computationally intensive than a wrapper approach.



Data Discretization:-

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss.

Discretization and Binarization:

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss.
There are two forms of data discretization first is supervised discretization, and the second is unsupervised discretization. Supervised discretization refers to a method in which the class data is used.
Unsupervised discretization refers to a method depending upon the way which operation proceeds. It means it works on the top-down splitting strategy and bottom-up merging strategy. Now, we can understand this concept with the help of an example

Suppose we have an attribute of Age with the given values.

Age	1,5,9,4,7,11,14,17,13,18, 19,31,33,36,42,44,46,70,74,78,77

Table before Discretization

Attribute	Age	Age	Age	Age
	1,5,4,9,7	11,14,17,13,18,19	31,33,36,42,44,46	70,74,77,78
After Discretization	Child	Young	Mature	Old

Data Transformation in Data Mining:

Data transformation in data mining is done for combining unstructured data with structured data to analyze it later. It is also important when the data is transferred to a new cloud data warehouse. When the data is homogeneous and well-structured, it is easier to analyze and look for patterns.

For example, a company has acquired another firm and now has to consolidate all the business data. The smaller company may be using a different database than the parent firm. Also, the data in these databases may have unique IDs, keys and values. All this needs to be formatted so that all the records are similar and can be evaluated. This is why data transformation methods are applied. And, they are described below:

Data Smoothing

This method is used for removing the noise from a dataset. Noise is referred to as the distorted and meaningless data within a dataset. Smoothing uses algorithms to highlight the special features in the data. After removing noise, the process can detect any small changes to the data to detect special patterns.

Data Aggregation

Aggregation is the process of collecting data from a variety of sources and storing it in a single format. Here, data is collected, stored, analyzed and presented in a report or summary format. It helps in gathering more information about a particular data cluster. The method helps in collecting vast amounts of data.

Discretization

This is a process of converting continuous data into a set of data intervals. Continuous attribute values are substituted by small interval labels. This makes the data easier to study and analyze. If a continuous attribute is handled by a data mining task, then its discrete values can be replaced by constant quality attributes.

Generalization

In this process, low-level data attributes are transformed into high-level data attributes using concept hierarchies. This conversion from a lower level to a higher conceptual level is useful to get a clearer picture of the data.

Attribute construction

In the attribute construction method, new attributes are created from an existing set of attributes. For example, in a dataset of employee information, the attributes can be employee name, employee ID and address.

Measures of Similarity and Dissimilarity:

Distance or similarity measures are essential in solving many pattern recognition problems such as classification and clustering. Various distance/similarity measures are available in the literature to compare two data distributions. As the names suggest, a similarity measures how close

two distributions are. For multivariate data complex summary methods are developed to answer this question.

Similarity Measure

Numerical measure of how alike two data objects often fall between 0 (no similarity) and 1 (complete similarity)

Dissimilarity Measure

Numerical measure of how different two data objects are range from 0 (objects are alike) to ∞ (objects are different)

Proximity

Refers to a similarity or dissimilarity

Similarity/Dissimilarity for Simple Attributes

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

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

Distance, such as the Euclidean distance, is a dissimilarity measure and has some well-known properties: Common Properties of Dissimilarity Measures

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

Euclidean Distance:

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

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

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

for every pair (i, j) of observations.

The weighted Euclidean distance is:

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

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

Unit-2

ASSOCIATION RULE MINING

Association rules are "if-then" statements, that help to show the probability of relationships between data items, within large data sets in various types of databases. Association rule mining has a number of applications and is widely used to help discover sales correlations in transactional data or in medical data sets.

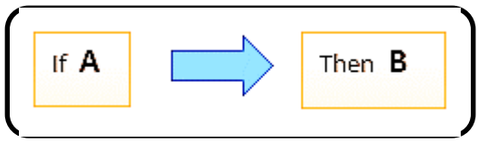
Association rules are created by searching data for frequent if-then patterns and using the criteria *support* and *confidence* to identify the most important relationships. Support is an indication of how frequently the items appear in the data. Confidence indicates the number of times the if-then statements are found true. A third metric, called *lift*, can be used to compare confidence with expected confidence, or how many times an if-then statement is expected to be found true.

Association rules are calculated from *item sets*, which are made up of two or more items. If rules are built from analyzing all the possible item sets, there could be so many rules that the rules hold little meaning. With that, association rules are typically created from rules well-represented in data.

The association rule learning is one of the very important concepts of machine learning.
And it is employed in Market Basket analysis, Web usage mining, continuous production, etc.
Here market basket analysis is a technique used by the various big retailers to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together is put together.

How does Association Rule Learning work?

Association rule learning works on the concept of If and Else Statement, such as if A then B.



Here the If element is called antecedent, and then statement is called as Consequent. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- Support
- Confidence
- Lift

Let's understand each of them:

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the item set X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp(X)} = \frac{Freq(X)}{T}$$

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

Frequent Itemset Generation

In order to generate a frequent itemset list one must avoid using the brute force approach described in the previous section because it can be very expensive to search through the whole data set to find the support count of each itemset. Some of the strategies used to address this problem are:-

1. **Reduce the number of candidates:** use pruning techniques such as the Apriori principle to eliminate some of the candidate item sets without counting their support values
2. **Reduce the number of transactions:** by combining transactions together we can reduce the total number of transactions
3. **Reduce the number of comparisons:** use efficient data structures to store the candidates thereby eliminating the need to match every candidate against every transaction.

The Apriori Principle (Apriori Algorithm)

States that if an itemset is frequent, then all of its subsets must also be frequent.

This principle holds true because of the anti-monotone property of support.

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

This means that when f is anti-monotone (or downward closed) and X is a subset of Y, then the support of X, s(X) must not exceed the support of Y, s(Y).
The converse is also true for when f is monotone (or upward closed), this means that when X is a subset of Y, then the support of Y, s(Y) must not exceed the support of X, s(X).

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(Y) \leq s(X)$$

Apriori Algorithm – Frequent Pattern Algorithms
Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent item sets.

Apriori says:
The probability that item I is not frequent is if:

- P(I) < minimum support threshold, then I is not frequent.
- P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to itemset.
- If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property.

The steps followed in the Apriori Algorithm of data mining are:

1. **Join Step**: This step generates (K+1) itemset from K-item sets by joining each item with itself.
2. **Prune Step**: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate item sets.

Steps In Apriori

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.

#1) In the first iteration of the algorithm, each item is taken as a 1-item sets candidate. The algorithm will count the occurrences of each item.

#2) Let there be some minimum support, min_sup (eg 2). The set of 1 – item sets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

#3) Next, 2-itemset frequent items with min_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.
#4) The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –item sets with min-sup only.

#5) the next iteration will form 3 –item sets using join and prune step. This iteration will follow antimonotone property where the subsets of 3-item sets, that is the 2 –itemset subsets of each group fall in min_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.

#6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.

Example of Apriori: **Support threshold=50%, Confidence= 60%**

TABLE-1

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:
Support threshold=50% => 0.5*6= 3 => min_sup=3

Count Of Each Item

TABLE-2

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

Prune Step: TABLE -2 shows that I5 item does not meet min_sup=3, thus it is deleted, only I1, I2, I3, I4 meet min_sup count.

TABLE-3

Item	Count
I1	4
I2	5
I3	4
I4	4

3. Join Step: Form 2-itemset. From **TABLE-1** find out the occurrences of 2-itemset.

TABLE-4

Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2

Prune Step: TABLE -4 shows that item set {I1, I4} and {I3, I4} does not meet min_sup, thus it is deleted.

TABLE-5

Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

Join and Prune Step: Form 3-itemset. From the **TABLE- 1** find out occurrences of 3-itemset. From **TABLE-5**, find out the 2-itemset subsets which support min_sup.

We can see for itemset {I1, I2, I3} subsets, {I1, I2}, {I1, I3}, {I2, I3} are occurring in **TABLE-5** thus {I1, I2, I3} is frequent.

We can see for itemset {I1, I2, I4} subsets, {I1, I2}, {I1, I4}, {I2, I4}, {I1, I4} is not frequent, as it is not occurring in **TABLE-5** thus {I1, I2, I4} is not frequent, hence it is deleted.

TABLE-6

Item
I1,I2,I3
I1,I2,I4
I1,I3,I4
I2,I3,I4

Only {I1, I2, I3} is frequent.

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

Input:

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

Output: L, frequent itemsets in D.

Method:

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

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

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

Partitioning methods (Algorithms):

1. k-mean algorithm:

Step-1 Take mean value (randomly)
Step-2 Find nearest number of mean and put it in cluster.
Step-3 Repeat step-1&step-2 until we get same mean.
Step-4 k-mean method typically uses the Square error criterion function.

EX: O= {2, 3, 4, 10, 11, 12, 20, 25, 30}; K=2

M1=4 M2=12
C1={2,3,4} C2={10,11,12,20,25,30}
M1=9/3 =3 M2=108/6 =18
C1={2,3,4,10} C2={11,12,20,25,3
M1=19/4 =4.75 M2=19.6
M1= ~5 M2= ~20
C1={2,3,4,10,11,12} C2={20,25,30}
M1=7 M2=25
C1={2,3,4,10,11,12} C2={20,25,30}
M1=7 M2=25

2. K-Medoids algorithm:

The k-mean algorithm is sensitivity to outliers because an object with an extremely large values may substrainly destroy the destroy the distribution of data.

Insetead of taking the mean value of the object in a cluster as a reference point, we can pick actual objects to represent clusters using one representative object per cluster.

Each remaining object is clustered with the representative object to which it is the most similar.

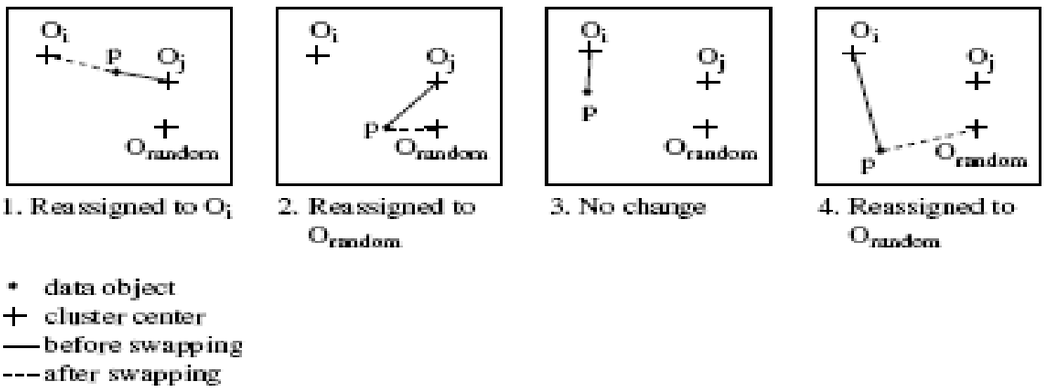
Case-1: 'p' currenty belongs to representative object Oj.If Oj is replaced by O_{randam} as a representative object Oi,i≠j , then p is reassigned to Oi.

• →data object
\+ →cluster center
\- →before swapping
--- →After swapping.

Case-2: 'p' currently belongs to representative object Oj,if Oj is replaced by Orandom as a representative object and p is close to Orandom then p is assigned to Orandom.

Case-3: 'p' currently belongs to representative object Oi,i≠j, if Oj is replaced by Orandom as a representative object and p is still close to Oi, then the assigned doesn't change.

Case-4: 'p' currently belongs to representative object Oi, i≠j, if Oj is replaced by Orandom, then p is reassigned to Orandom.



PAM (partitioning around medoids) algorithm:

1. It is one of the k-medoids Algorithm.
2. It uses a k-medoids to identify the clusters
3. CLARANS [clustering Large Application]:

1. CLARA is a simplifying-based method used to partition Large data base.
2. The idea behind CLARA is instead of taking the whole set of data in to consideration, a small portion of the actual data is choosen.
3. Medoids are then choose from this sample using PAM.
4. CLARA applies PAM on each sample and returns its best clustering as the output.
5. CLARA can deal with larger data sets than PAM. The complexities of Each iteration now becomes $O(ks^2+k(n-k))$

Where 's' size of the sample
'k' is the no.of clusters
'n' is the total no.of objects

Frequent Pattern Growth Algorithm

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.

This tree structure will maintain the association between the item sets. The database is fragmented using one frequent item. This fragmented part is called "pattern fragment". The item sets of these fragmented patterns are analyzed. Thus with this method, the search for frequent item sets is reduced comparatively.

FP Tree

Frequent Pattern Tree is a tree-like structure that is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the itemset.

The root node represents null while the lower nodes represent the item sets. The association of the nodes with the lower nodes that is the item sets with the other item sets are maintained while forming the tree.

Frequent Pattern Algorithm Steps
The frequent pattern growth method lets us find the frequent pattern without candidate generation.

Let us see the steps followed to mine the frequent pattern using frequent pattern growth algorithm:

#1) The first step is to scan the database to find the occurrences of the item sets in the database. This step is the same as the first step of Apriori. The count of 1-item sets in the database is called support count or frequency of 1-itemset.

#2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

#3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction item sets in descending order of count.

#4) The next transaction in the database is examined. The item sets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.
This means that the common itemset is linked to the new node of another itemset in this transaction.

#5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.

#6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base.

Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

#7) Construct a Conditional FP Tree, which is formed by a count of item sets in the path. The item sets meeting the threshold support are considered in the Conditional FP Tree.

#8) Frequent Patterns are generated from the Conditional FP Tree.

Example Of FP-Growth Algorithm

Support threshold=50%, Confidence= 60%

Table 1

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:
Support threshold=50% => 0.5*6= 3 => min_sup=3

1. Count of each item
Table 2

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

2. Sort the itemset in descending order.
Table 3

Item	Count
I2	5
I1	4
I3	4
I4	4

Compact Representation of Frequent Item sets

Introduction

What happens when you have a large market basket data with over a hundred items?
The number of frequent item sets grows exponentially and this in turn creates an issue with storage and it is for this purpose that alternative representations have been derived which reduce the initial set but can be used to generate all other frequent item sets. The Maximal and Closed Frequent Item sets are two such representations that are subsets of the larger frequent item set that will be discussed in this section.

Maximal Frequent Itemset

Definition: **It is a frequent item set for which none of its immediate supersets are frequent.**

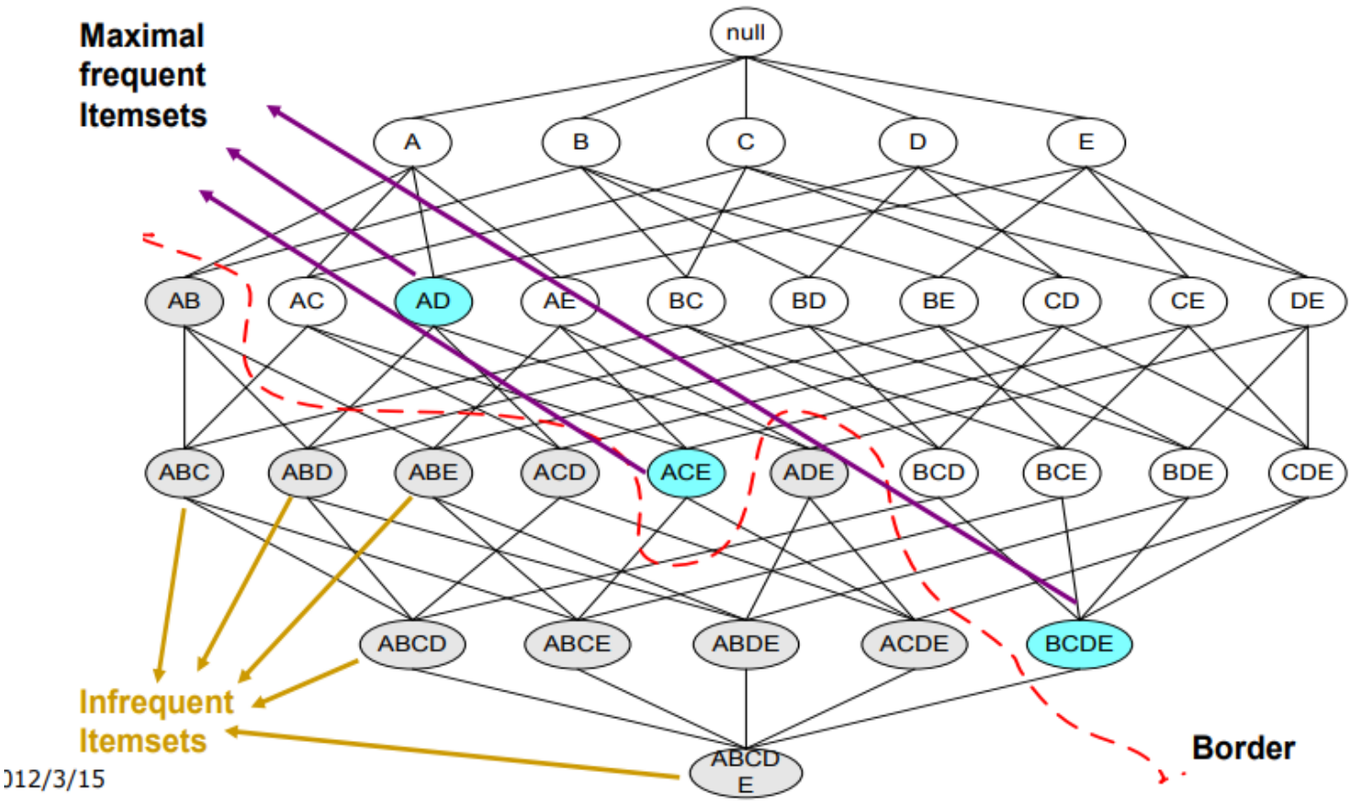
Identification

1. Examine the frequent item sets that appear at the border between the infrequent and frequent item sets.
2. Identify all of its immediate supersets.
3. If none of the immediate supersets are frequent, the item set is maximal frequent.

Illustration

For instance consider the diagram shown below, the lattice is divided into two groups, red dashed line serves as the dermarcation, the item sets above the line that are blank are frequent item sets and the blue ones below the red dashed line are infrequent.

- In order to find the maximal frequent item set, you first identify the frequent item sets at the border namely *d, bc, ad* and *abc*.
- Then identify their immediate supersets,
  the supersets for *d*, *bc* are characterized by the blue dashed line and if you trace the lattice you notice that for d, there are three supersets and one of them, ad is frequent and this can't be maximal frequent,
  for *bc* there are two supersets namely *abc* and *bcd abc* is frequent and so *bc* is NOT maximal frequent.
- The supersets for ad and abc are characterized by a solid orange line, the superset for *abc* is *abcd* and being that it is infrequent, *abcd* is maximal frequent. For ad, there are two supersets *abd* and *acd*, both of them are infrequent and so ***ad*** is also maximal frequent.



Closed Frequent Itemset

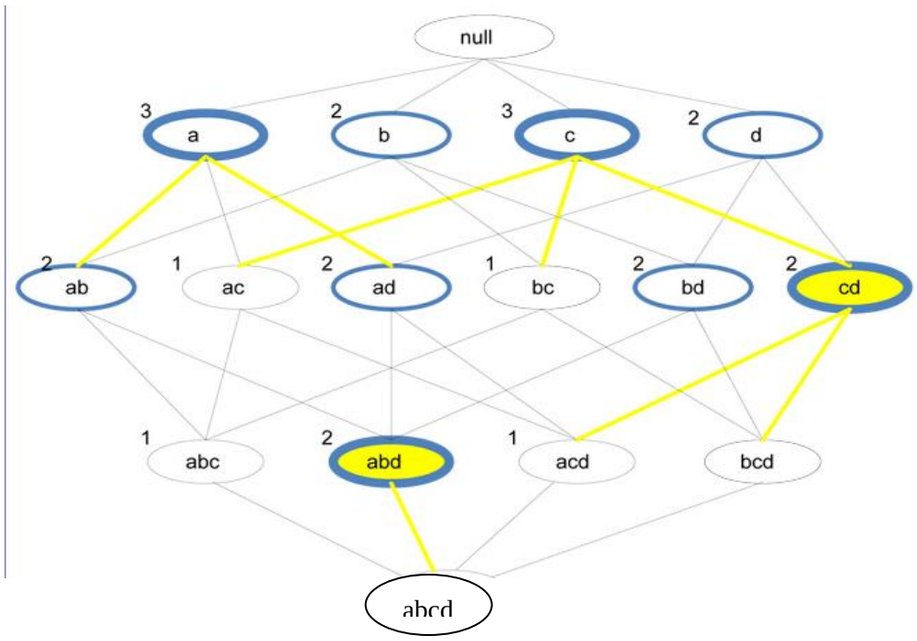
Definition:

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

Identification

1. First identify all frequent item sets.

Then from this group find those that are closed by checking to see if there exists a superset that has the same support as the frequent itemset, if there is, the itemset is disqualified, but if none can be found, the itemset is closed.
An alternative method is to first identify the closed item sets and then use the minsup to determine which ones are frequent.
The lattice diagram above shows the maximal, closed and frequent item sets. The item sets that are circled with blue are the frequent item sets. The item sets that are circled with the thick blue are the closed frequent item sets. The item sets that are circled with the thick blue and have the yellow fill are the maximal frequent item sets. In order to determine which of the frequent item sets are closed, all you have to do is check to see if they have the same support as their supersets, if they do they are not closed.



Finding Frequent Item sets Using Candidate Generation:TheApriori Algorithm

- Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent item sets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties.
- Apriori employs an iterative approach known as a *level-wise* search, where *k*-item sets are used to explore (*k*+1)-item sets.
- First, the set of frequent 1-item sets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted *L*1.Next, *L*1 is used to find *L*2, the set of frequent 2-item sets, which is used to find *L*3, and so on, until no more frequent *k*-item sets can be found.
- The finding of each *Lk*requires one full scan of the database.
- A two-step process is followed in Apriori consisting of join and prune action.

This algorithm works as follows:

- First, it compresses the input database creating an FP-tree instance to represent frequent items.

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

Input:

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

Output: L, frequent itemsets in D.

Method:

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

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

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

- After this first step, it divides the compressed database into a set of conditional databases, each associated with one frequent pattern.
- Finally, each such database is mined separately.

Using this strategy, the FP-Growth reduces the search costs by recursively looking for short patterns and then concatenating them into the long frequent patterns.

In large databases, holding the FP tree in the main memory is impossible. A strategy to cope with this problem is to partition the database into a set of smaller databases (called projected databases) and then construct an FP-tree from each of these smaller databases.

10.5.4 Example of FP-tree

Ex. 10.5.1 : Transactions consist of a set of items I = {a, b, c, ...} , min support = 3

TID	Items Bought
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

Soln. :

Step 1 : Find the minimum support of each item.

Item	Sup.
a	3
b	3
c	4
d	1
e	1
f	4
g	1
h	1
i	1
j	1
k	1
l	2
m	3
n	1
o	2
p	3

Consider items with min support = 3 (given)

Item	Sup.
a	3
b	3
c	4
f	4
m	3
p	3

Consider items with min support = 3 (given)

Item	Sup.
a	3
b	3
c	4
f	4
m	3
p	3

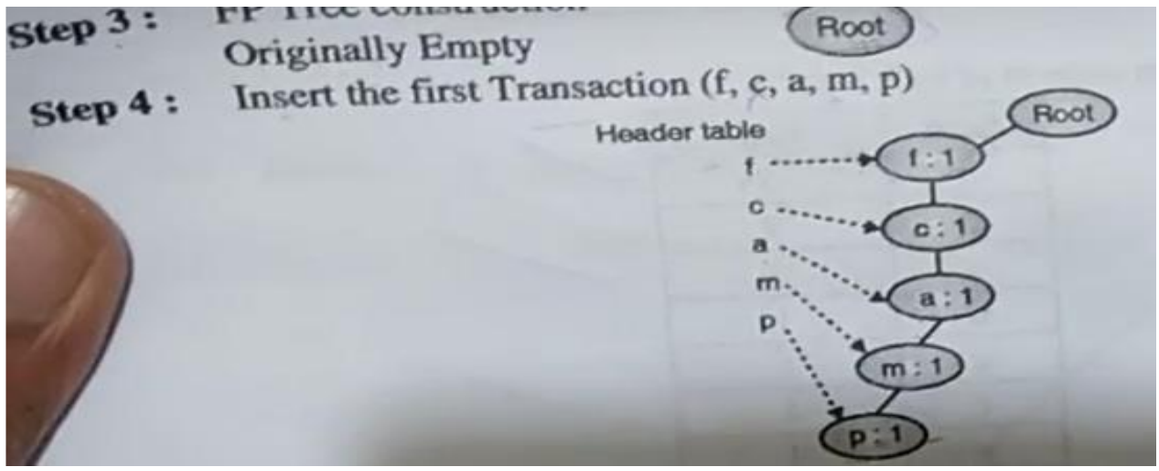
Step 2 : Order all items in itemset in frequency descending order (min support=3)
(**Note:**Consider only items with min support = 3)

TID	Items Bought	(Ordered frequent items)
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, l, m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

(f:4, c:4, a:3, b:3, m:3, p:3)

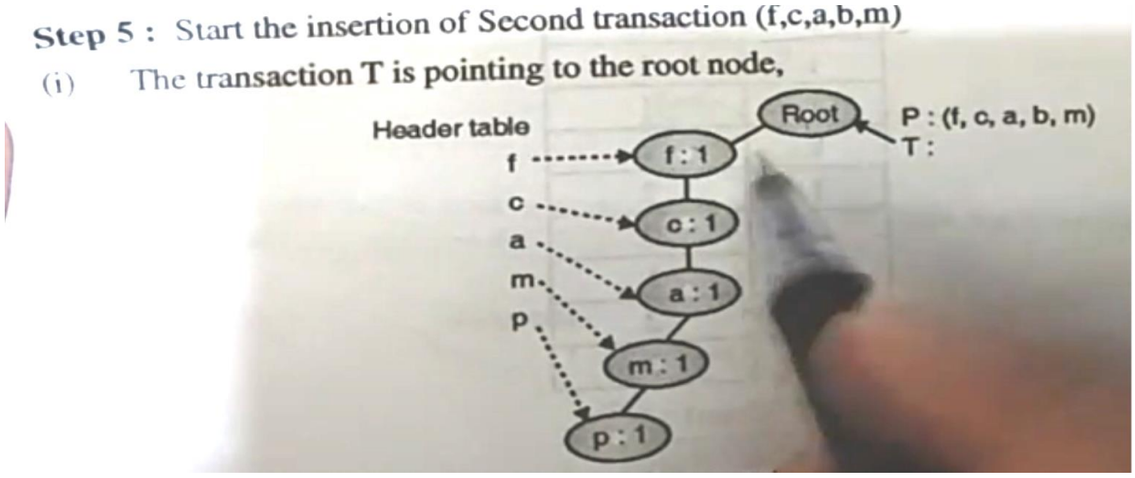
Step 3 : Originally Empty

Step 4 : Insert the first Transaction (f, c, a, m, p)



Step 5 : Start the insertion of Second transaction (f,c,a,b,m)

(i) The transaction T is pointing to the root node,



FP-Tree

The frequent-pattern tree (FP-tree) is a compact data structure that stores quantitative information about frequent patterns in a database. Each transaction is read and then mapped onto a path in the FP-tree. This is done

until all transactions have been read. Different transactions with common subsets allow the tree to remain compact because their paths overlap.

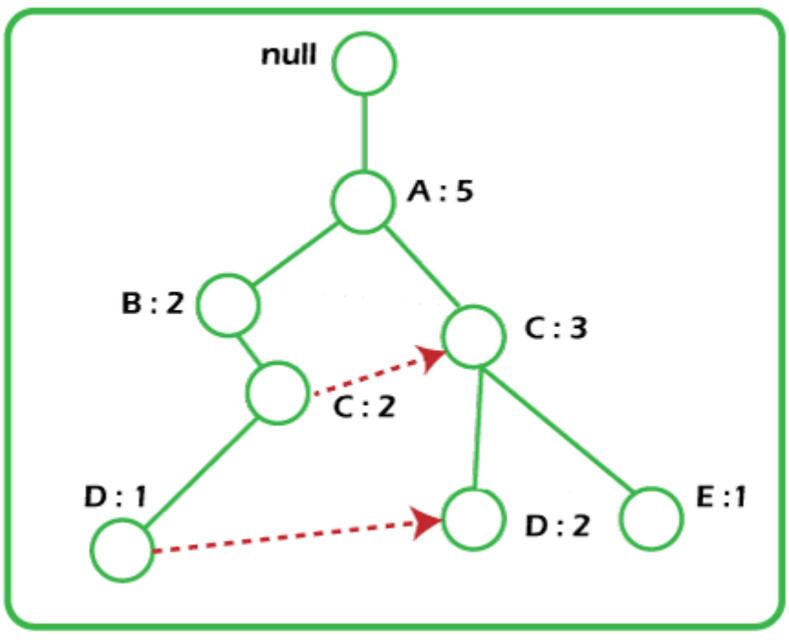
A frequent Pattern Tree is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the item set.

The root node represents null, while the lower nodes represent the item sets. The associations of the s, are maintained while forming the tree.

Han defines the nodes with the lower nodes, that is, the item sets with the other item setFP-tree as the tree structure given below:

1. One root is labelled as "null" with a set of item-prefix subtrees as children and a frequent-item-header table.
2. Each node in the item-prefix subtree consists of three fields:
    - Item-name: registers which item is represented by the node;
    - Count: the number of transactions represented by the portion of the path reaching the node;
    - Node-link: links to the next node in the FP-tree carrying the same item name or null if there is none.
3. Each entry in the frequent-item-header table consists of two fields:
    - Item-name: as the same to the node;
    - Head of node-link: a pointer to the first node in the FP-tree carrying the item name.

Additionally, the frequent-item-header table can have the count support for an item. The below diagram is an example of a best-case scenario that occurs when all transactions have the same itemset; the size of the FP-tree will be only a single branch of nodes.



The worst-case scenario occurs when every transaction has a unique item set. So the space needed to store the tree is greater than the space used to store the original data set because the FP-tree requires additional space to store pointers between nodes and the counters for each item. The diagram below shows how a worst-case scenario FP-tree might appear. As you can see, the tree's complexity grows with each transaction's uniqueness.

Advantages of FP Growth Algorithm

Here are the following advantages of the FP growth algorithm, such as:

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

Disadvantages of FP-Growth Algorithm

This algorithm also has some disadvantages, such as:

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

Difference between Apriori and FP Growth Algorithm

Apriori and FP-Growth algorithms are the most basic FIM algorithms. There are some basic differences between these algorithms, such as:

Apriori	**FP Growth**
Apriori generates frequent patterns by making the item sets using pairings such as single item set, double itemset, and triple itemset.	FP Growth generates an FP-Tree for making frequent patterns.
Apriori uses candidate generation where frequent subsets are extended one item at a time.	FP-growth generates a conditional FP-Tree for every item in the data.
Since apriori scans the database in each step, it becomes time-consuming for data where the number of items is larger.	FP-tree requires only onedatabase scan in its beginning steps, so it consumes less time.
A converted version of the database is saved in the memory	A set of conditional FP-tree for every item is saved in the memory
It uses a breadth-first search	It uses a depth-first s earch.

UNIT-3

Introduction:

What Is Classification?

A bank loans officer needs analysis of her data to learn which loan applicants are "safe" and which are "risky" for the bank. A marketing manager at *All Electronics* needs data
analysis to help guess whether a customer with a given profile will buy a new computer. A medical researcher wants to analyze breast cancer data to predict which one of three specific treatments a patient should receive. In each of these examples, the data analysis task is **classification**,

8.1.1 General Approach to Classification and Algorithm :

"How does classification work?" **Data classification** is a two-step process, consisting of a *learning step* (where a classification model is constructed) and a *classification step* (where the model is used to predict class labels for given data). The process is shown for the loan application data of Figure 8.1. (The data are simplified for illustrative purposes. In reality, we may expect many more attributes to be considered.

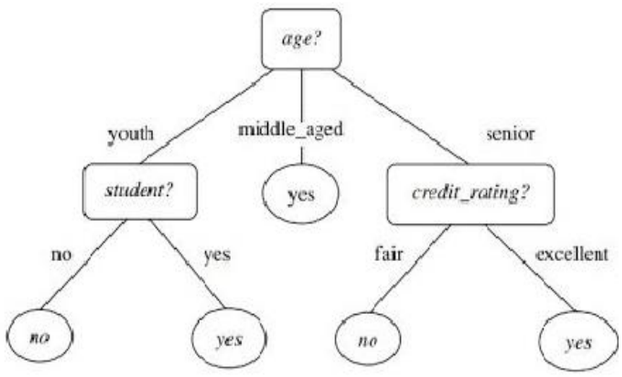
8.1.2 In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels. A tuple, *X*, is represented by an *n*-dimensional attribute General Approach to Classification and Algorithm:

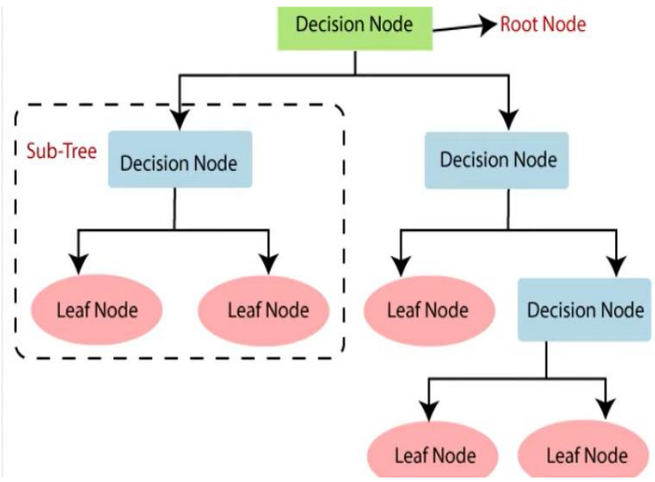
"How does classification work?" **Data classification** is a two-step process, consisting of a *learning step* (where a classification model is constructed) and a *classification step* (where the model is used to predict class labels for given data). The process is shown for the loan application data of Figure 8.1. (The data are simplified for illustrative purposes. In reality, we may expect many more attributes to be considered.

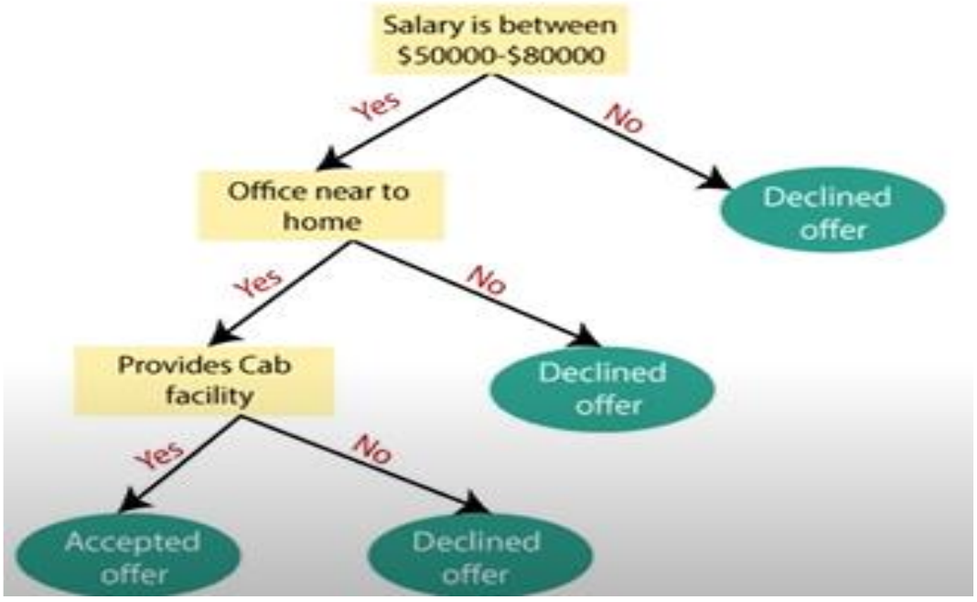
In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the **learning step** (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a **training set** made up of database tuples and their associated class labels. A tuple, ***X***, is represented by an *n*-dimensional **attribute vector**, ***X*** $(x_1, x_2, \ldots, x_n)$, depicting *n* measurements made on the tuple from *n* database attributes, respectively, $A_1, A_2, \ldots, A_n$.[1] Each tuple, ***X***, is assumed to belong to a predefined class as determined by another database attribute called the **class label attribute**. The class label attribute is discrete-valued and unordered. It is *categor- ical* (or nominal) in that each value serves as a category or class.

Decision Tree Induction

During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).







Decision tree Algorithm:

The decision tree algorithm may appear long, but it is quite simply the basis algorithm techniques is as follows:

The **algorithm** is based on three parameters**: D, attribute_list, and Attribute _selection_method.**

Generally, we refer to **D** as a **data partition.**

Initially, **D** is the entire set of **training tuples** and their related **class levels** (input training data).

The parameter **attribute_list** is a set of **attributes** defining the tuples.

Attribute_selection_method specifies a **heuristic process** for choosing the attribute that "best" discriminates the given tuples according to **class**.

Attribute_selection_method process applies an **attribute selection measure**.

Advantages of using decision trees:

A decision tree does not need scaling of information.

Missing values in data also do not influence the process of building a choice tree to any considerable extent.

A decision tree model is automatic and simple to explain to the technical team as well as stakeholders.

Compared to other algorithms, decision trees need less exertion for data preparation during pre-processing.

A decision tree does not require a standardization of

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

(1) create a node N ;

(2) if tuples in D are all of the same class, C, then

(3) return N as a leaf node labeled with the class C;

(4) if attribute list is empty then

(5) return N as a leaf node labeled with the majority class in D; // majority voting

(6) apply Attribute selection method(D, attribute list) to find the “best” splitting criterion;

(7)

(8) label node N with splitting criterion;

(9) if splitting attribute is discrete-valued and

multiway splits allowed then // not restricted to binary trees

(10) attribute list attribute list splitting attribute; // remove splitting attribute

(11) for each outcome j of splitting criterion

// partition the tuples and grow subtrees for each partition

(12) let Dj be the set of data tuples in D satisfying outcome j; // a partition

(13) if Dj is empty then

(14) attach a leaf labeled with the majority class in D to node N ;

(15) else attach the node returned by Generate decision tree(Dj , attribute list) to node N ;
end for

(16) return N ;

UNIT -IV

Cluster Analysis:

- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.

Applications:

- Cluster analysis has been widely used in numerous applications, including market research, pattern recognition, data analysis, and image processing.
- In business, clustering can help marketers discover distinct groups in their customer bases and characterize customer groups based on purchasing patterns.
- In biology, it can be used to derive plant and animal taxonomies, categorize genes with similar functionality, and gain insight into structures inherent in populations.

Typical Requirements Of Clustering In Data Mining:

- **Scalability:**

  Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions of objects.

>.Ability to deal with different types of attributes:

Many algorithms are designed to cluster interval-based (numerical)

data. However, applications may require clustering other types of data, such as binary, categorical (nominal), and ordinal data, or mixtures of these data types.

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

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

- **Minimal requirements for domain knowledge to determine input parameters:**

  Many clustering algorithms require users to input certain parameters in cluster analysis (such as the number of desired clusters). The clustering results can be quite sensitive to input parameters. Parameters are often difficult to determine, especially for data sets containing high-dimensional objects.

Ability to deal with noisy data:

Most real-world databases contain outliers or missing, unknown, or erroneous data.

Some clustering algorithms are sensitive to such data and may lead to clusters of poor quality.

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

  Some clustering algorithms cannot incorporate newly inserted data (i.e., database updates) into existing clustering structures and, instead, must determine a new clustering from scratch. Some clustering algorithms are sensitive to the order of input data. That is, given a set of data objects,

High dimensionality:

A database or a data warehouse can contain several dimensions or attributes. Many clustering algorithms are good at handling low-

dimensional data, involving only two to three dimensions. Human eyes are good at judging the quality of clustering for up to three dimensions.

- **Constraint-based clustering:**

  Real-world applications may need to perform clustering under various kinds of constraints. Suppose that your job is to choose the locations for a given number of new automatic banking machines (ATMs) in a city.

- **Interpretability and usability:**

  Users expect clustering results to be interpretable, comprehensible, and usable. That is, clustering may need to be tied to specific semantic interpretations and applications

Major Clustering Methods:

- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Model-Based Methods

Partitioning Methods:

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

- Each group must contain at least one object, and

Hierarchical Methods:

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

- The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one or until a termination condition holds.
- The divisive approach, also called the top-down approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

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

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

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

Density-based methods:

- Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.
- Other clustering methods have been developed based on the notion of density. Their general idea is to continue growing the given cluster as long as the density in the neighborhood exceeds some threshold; that is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers) and discover clusters

of arbitrary shape.

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

Grid-Based Methods:

- Grid-based methods quantize the object space into a finite number of cells that form a grid structure.
- All of the clustering operations are performed on the grid structure i.e., on the quantized space. The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.
- STING is a typical example of a grid-based method. Wave Cluster applies wavelet transformation for clustering analysis and is both grid-based and density-based.

Model-Based Methods:

- Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model.
- A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points.
- It also leads to a way of automatically determining the number of clusters based on standard statistics, taking ―noise‖ or outliers into account and thus yielding robust clustering methods.

Clustering High-Dimensional Data:

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

information on the expression levels of thousands of genes under hundreds of conditions.

- Clustering high-dimensional data is challenging due to the curse of dimensionality.
- Many dimensions may not be relevant. As the number of dimensions increases,

  The data become increasingly sparse so that the distance measurement between pairs of points becomes meaningless and the average density of points anywhere in the data is likely to be low. Therefore, a different clustering methodology needs to be developed for high-dimensional data.
- CLIQUE and PROCLUS are two influential subspace clustering methods, which search for clusters in subspaces of the data, rather than over the entire data space.
- Frequent pattern–based clustering, another clustering methodology, extracts distinct frequent patterns among subsets of dimensions that occur frequently. It uses such patterns to group objects and generate meaningful clusters.

Constraint-Based Clustering:

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

Classical Partitioning Methods:

The most well-known and commonly used partitioning methods are

- The *k*-Means Method
- k-Medoids Method

Centroid-Based Technique: The K-Means Method:

The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity islow.

Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

The *k*-means algorithm proceeds as follows.

- First, it randomly selects *k* of the objects, each of which initially represents a clustermean or center.
- For each of the remaining objects, an object is assigned to the cluster to which it is themost similar, based on the distance between the object and the cluster mean.
- It then computes the new mean for each cluster.
- This process iterates until the criterion function converges.

Typically, the square-error criterion is used, defined as : $$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2,$$

Where E is the sum of the square error for all objects in the dataset p is the point in space representing a given object m_i is the mean of cluster C_i

The k-means partitioning algorithm:

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

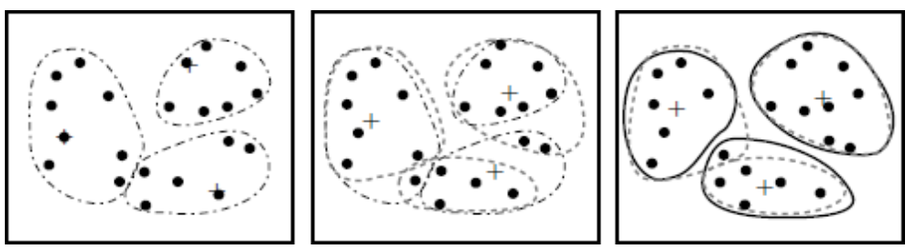
Input:

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

Output: A set of *k* clusters.

Method:

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



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

The *k*-Medoids Method:

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

- That is, an absolute-error criterion is used, defined as: $$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

Where E is the sum of the absolute error for all objects in the data set $\boldsymbol{p}$ is the point in space representing a given object in cluster C_j $\mathbf{o_j}$ is the representative object of C_j.

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

Case 1:

P currently belongs to representative object, o_j. If o_j is replaced by o_{random} as a representative object and p is closest to one of the other representative objects, o_i, I ≠ j, then p is reassigned to o_i.

Case 2:

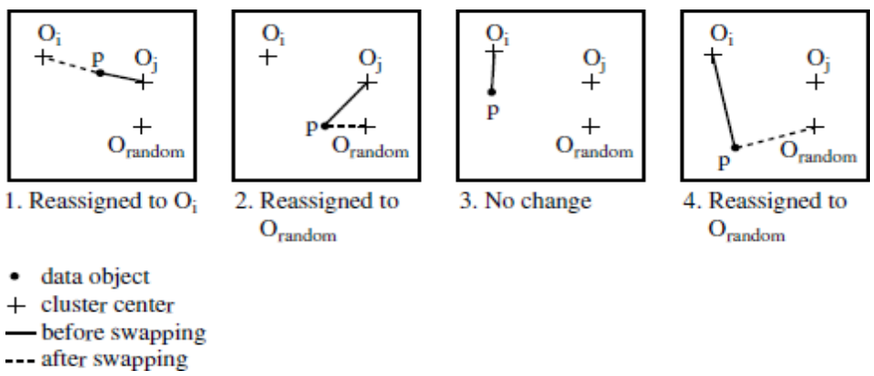
P currently belongs to representative object, o_j. If ojis replaced by o_{random} as a representative object and p is closest to o_{random}, then p is reassigned to o_{random}.

Case 3:

P currently belongs to representative object, o_i, I ≠ j. If ojis replaced by o_{random} as a representative object and p is still closest to o_i, then the assignment does not change.

Case 4:

P currently belongs to representative object, oi, I ≠ j. If o_j is replaced by o_{random} as a representative object and p is closest to o_{random}, then p is reassigned to Orandom.



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

The k-Medoids Algorithm:

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

Input:

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

Output: A set of *k* clusters.

Method:

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

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

Hierarchical Clustering Methods:

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

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

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

Agglomerative hierarchical clustering:

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

Divisive hierarchical clustering:

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

Constraint-Based Cluster Analysis:

Constraint-based clustering finds clusters that satisfy user-specified preferences or constraints. Depending on the nature of the constraints, constraint-based clustering may adopt rather different approaches.
There are a few categories of constraints.

- Constraints on individual objects:

  We can specify constraints on the objects to be clustered. In a real estate application, for example, one may like to spatially cluster only those luxury mansions worth over a million dollars. This constraint confines the set of objects to be clustered. It can easily be handled by preprocessing after which the problem reduces to an instance of

unconstrained clustering.

- Constraints on the selection of clustering parameters:

  A user may like to set a desired range for each clustering parameter. Clustering parameters are usually quite specific to the given clustering algorithm. Examples of parameters include k, the desired number of clusters in a k-means algorithm; or e the radius and the minimum number of points in the DBSCAN algorithm. Although such user-specified parameters may strongly influence the clustering results, they are usually confined to the algorithm itself. Thus, their fine tuning and processing are usually not considered a form of constraint-based clustering.

- Constraints on distance or similarity functions:

  We can specify different distance or similarity functions for specific attributes of the objects to be clustered, or different distance measures for specific pairs of objects. When clustering sportsmen, for example, we may use different weighting schemes for height, body weight, age, and skill level. Although this will likely change the mining results, it may not alter the clustering process per se. However, in some cases, such changes may make the evaluation of the distance function nontrivial, especially when it is tightly intertwined with the clustering process.

- User-specified constraints on the properties of individual clusters:

  A user may like to specify desired characteristics of the resulting clusters, which may strongly influence the clustering process.

- Semi-supervised clustering based on partial supervision:

  The quality of unsupervised clustering can be significantly improved using some weak form of supervision. This may be in the form of pair wise constraints (i.e., pairs of objects labeled as belonging to the same or different cluster). Such a constrained clustering process is called semi-supervised clustering.

Outlier Analysis:

- There exist data objects that do not comply with the general behavior or model of the data. Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called outliers.
- Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information because one person's noise could be another person's signal. In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity. Thus, outlier detection and analysis is an interesting data mining task, referred to as outlier mining.
- It can be used in fraud detection, for example, by detecting unusual usage of credit cards or telecommunication services. In addition, it is useful in customized marketing for identifying the spending behavior of customers with extremely low or extremely high incomes, or in medical analysis for finding unusual responses to various medical treatments.

Outlier mining can be described as follows: Given a set of *n* data points or objects and *k*, the expected number of outliers, find the top *k* objects that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data. The outlier mining problem can be viewed as two sub problems:

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

Types of outlier detection:

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

Statistical Distribution-Based Outlier Detection:

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

A statistical discordancy test examines two hypotheses:

- A working hypothesis.
- An alternative hypothesis.

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

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

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

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

There are different kinds of alternative distributions.

- **Inherent alternative distribution**:

In this case, the working hypothesis that all of the objects come from distribution F is rejected in favor of the alternative hypothesis that all of the objects arise from another distribution, G:

H: oi € G, where i = 1, 2,…, n

F and G may be different distributions or differ only in parameters of the same distribution.

There are constraints on the form of the *G* distribution in that it must have potential to produce outliers. For example, it may have a different mean or dispersion, or a longer tail.

Mixture alternative distribution:

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

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

Slippage alternative distribution:

These alternative states that all of the objects (apart from some prescribed small number) arise independently from the initial model, F, with its given parameters, whereas the remaining objects are independent observations from a modified version of F in which the parameters have been shifted.

There are two basic types of procedures for detecting outliers:

Block procedures:

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

Consecutive procedures:

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

Distance-Based Outlier Detection:

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

distance greater than dmin from o. In other words, rather that relying on statistical tests, we can think of distance-based outliers as those objects that do not have enough neighbors, where neighbors are defined based on distance from the given object. In comparison with statistical-based methods, distance based outlier detection generalizes the ideas behind discordancy testing for various standard distributions. Distance-based outlier detection avoids the excessive computation that can be associated with fitting the observed distribution into some standard distribution and in selecting discordancy tests.

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

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

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

Index-based algorithm:

Given a data set, the index-based algorithm uses multidimensional indexing structures, such as R-trees or k-d trees, to search for neighbors of each object ***o*** within radius *dmin* around that object. Let *M* be the maximum number of objects within the *dmin*-neighborhood of an outlier. Therefore, once*M*+1 neighbors of object ***o*** are found, it is clear that ***o*** is not an outlier. This algorithm has a worst-case complexity of *O*(*n*2*k*), where *n* is the number of objects in the data set and *k* is the dimensionality. The index-based algorithm scales well as *k* increases. However, this complexity evaluation takes only the search time into account, even though the task of building an index in itself can be computationally intensive.

Nested-loop algorithm:

The nested-loop algorithm has the same computational complexity as the index-based algorithm but avoids index structure construction and tries to minimize the number of I/Os. It divides the memory buffer space into two

halves and the data set into several logical blocks. By carefully choosing the order in which blocks are loaded into each half, I/O efficiency can be achieved.

Cell-based algorithm:

To avoid $O(n^2)$ computational complexity, a cell-based algorithm was developed for memory-resident data sets. Its complexity is $O(c^k+n)$, where c is a constant depending on the number of cells and k is the dimensionality.

In this method, the data space is partitioned into cells with a side length equal to $\frac{dmin}{2\sqrt{k}}$. Each cell has two layers surrounding it. The first layer is one cell thick, while the second is $\lceil 2\sqrt{k}-1 \rceil$ cells thick, rounded up to the closest integer. The algorithm counts outliers on a cell-by-cell rather than an object-by-object basis. For a given cell, it accumulates three counts—the number of objects in the cell, in the cell and the first layer together, and in the cell and both layers together. Let's refer to these counts as cell count, cell + 1 layer count, and cell + 2 layers count, respectively.

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

- An object, **o**, in the current cell is considered an outlier only if cell + 1 layer count is less than or equal to M. If this condition does not hold, then all of the objects in the cell can be removed from further investigation as they cannot be outliers.
- If cell_+ 2_layers_count is less than or equal to M, then all of the objects in the cell are considered outliers. Otherwise, if this number is more than M, then it is possible that some of the objects in the cell may be outliers. To detect these outliers, object-by-object processing is used where, for each object, **o**, in the cell, objects in the second layer of **o** are examined. For objects in the cell, only those objects having no more than M points in their dmin-neighborhoods are outliers. The dmin-neighborhood of an object consists of the object's cell, all of its first layer, and some of its second layer.

A variation to the algorithm is linear with respect to n and guarantees that no more than three passes over the data set are required. It can be used for

large disk-resident data sets, yet does not scale well for high dimensions.

Density-Based Local Outlier Detection:

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

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

These are defined as follows:

The k-distance of an object p is the maximal distance that p gets from its k-nearest neighbors. This distance is denoted as k-distance(p). It is defined as the distance, d(p, o), between p and an object o 2 D, such that for at least k objects, o_0 2 D, it holds thatd(p, o·)_d(p, o). That is, there are at least k objects in D that are as close as or closer to p thano, and for at most k-1 objects, o00 2 D, it holds that d(p;o'') <d(p, o).

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

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

The k-distance neighborhood of an object p is denoted $N_{kdistance(p)(}p)$, or $N_k(p)$for short. By setting k to MinPts, we get $N_{MinPts}(p)$. It contains the MinPts-nearest neighbors of p. That is, it contains every object whose distance is not greater than theMinPts-distance of p.

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

reach

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

Intuitively, if an object p is far away , then the reachability distance between the

two is simply their actual distance. However, if they are sufficiently close (i.e., where p is within the MinPts-distance neighborhood of o), then the actual distance is replaced by the MinPts- distance of o. This helps to significantly reduce the statistical fluctuations of d(p, o) for all of the p close to o.

The higher the value of MinPts is, the more similar is the reachability distance for objects within the same neighborhood.

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

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

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

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

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

Deviation-Based Outlier Detection:

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

Sequential Exception Technique:

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

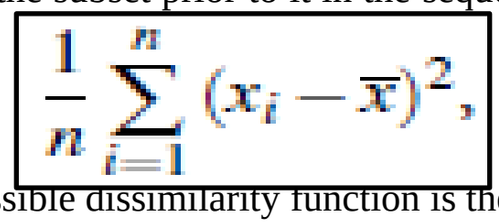
$$D_{j-1} \subset D_j, \quad \text{where } D_j \subseteq D.$$

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

Exception set:

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

Dissimilarity function:
This function does not require a metric distance between the objects. It is any function that, if given a set of objects, returns a low value if the objects are similar to one another. The greater the dissimilarity among the objects, the higher the value returned by the function. The dissimilarity of a subset is incrementally computed based on the subset prior to it in the sequence. Given a subset of *n* numbers, {x_1, …, x_n}, a possible dissimilarity function is the variance of the numbers in the set, that is,



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

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

Smoothing factor:

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

WEB MINING AND TEXT MINING

Web Mining

Web mining is the process which includes various data mining techniques to extract knowledge from web data categorized as web content, web structure and data usage. It includes a process of discovering the useful and unknown information from the web data.
Web mining can be classified based on the following categories:
1. Web Content
2. Web Structure
3. Web Usage

Web Content Mining

Web content mining is defined as the process of converting raw data to useful information using the content of web page of a specified web site.
The process starts with the extraction of structured data or information from web pages and then identifying similar data with integration. Various types of web content include text, audio, video etc. This process is called as text mining.
Text Mining uses Natural Language processing and retrieving information techniques for a specific mining process.

Web Structure Mining

Web graphs include a typical structure which consists of web pages such as nodes and hyperlinks which will be treated as edges connected between web pages. It

includes a process of discovering a specified structure with information from the web.
This category of mining can be performed either at document level or hyperlink level. The research activity which involves hyperlink level is called hyperlink analysis.

Terminologies associated with Web structure

1. Web graph: It is a directed graph which represents the web.
2. Node: Each web page includes a node of the web graph.
3. Link: Hyperlink is a type of directed edge of the web graph.
4. In-degree: In-degree specifies the number of distinct links that point to a specified node.
5. Out-degree: Out-degree specifies the number of distinct lakes originating at a node that points to other nodes.
6. Directed path: Directed path includes a sequence of links starting from a specified node that can be followed to reach another node.
7. Shortest Path: The shortest path will be the shortest length out of all the paths between p and q.
8. Diameter: The maximum of the shortest path between a pair of nodes p and q for all pairs of nodes p and q in the web graph.

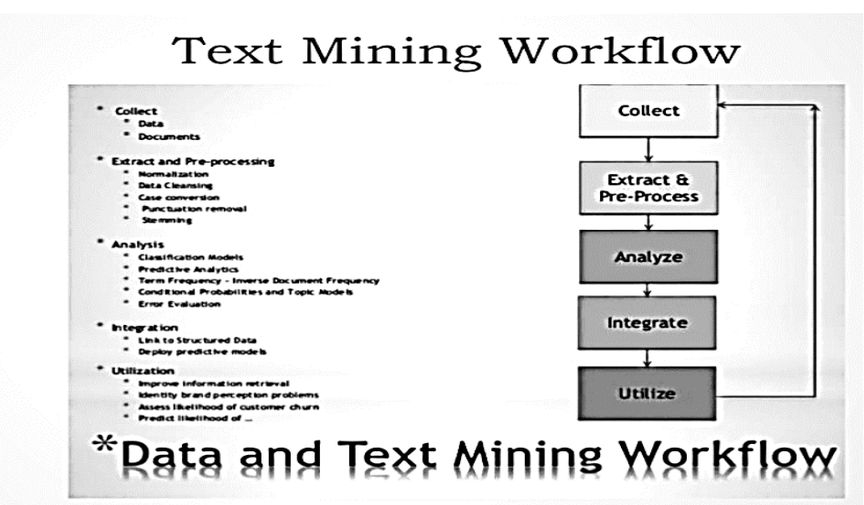
Web Usage Mining: Web includes a collection of interrelated files with one or more web servers. It includes a pattern of discovery of meaningful patterns of data generated by the client-server transaction.
The typical sources of data are mentioned below:
1. Data which is generated automatically is stored in server access logs, referrer logs, agent logs and client-side cookies.
2. Information of user profiles.
3. Metadata which includes page attributes and content attributes.

Web server log

Server logs created by the server record all activities. The page forwarded to the web server includes every single piece of basic information about URL.

Text Mining

The objective of text mining is to exploit information which is included in textual documents in various patterns and trends in association with entities and predictive rules.



The results are manipulated and used for:
1. The analysis of a collection
2. Providing information about intelligent navigation and browsing method.

TEXT CLUSTERING

The amount of text data being generated in the recent years has exploded exponentially. It's essential for organizations to have a structure in place to mine actionable insights from the text being generated. From social media analytics to risk management and cybercrime protection, dealing with textual data has never been more important.

Text clustering is the task of grouping a set of unlabelled texts in such a way that texts in the same cluster are more similar to each other than to those in other clusters. Text clustering algorithms process text and determine if natural clusters (groups) exist in the data.

Any text clustering approach involves broadly the following steps:

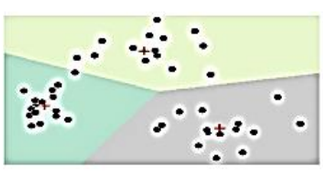
Text pre-processing: Text can be noisy, hiding information between stop words, inflexions and sparse representations. Pre-processing makes the dataset easier to work with.

Feature Extraction: One of the commonly used technique to extract the features from textual data is calculating the frequency of words/tokens in the document/corpus.

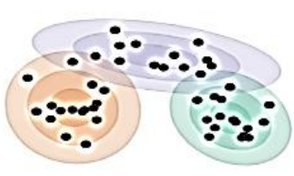
Clustering: We can then cluster different text documents based on the features we have generated.

Text clustering can be document level, sentence level or word level.

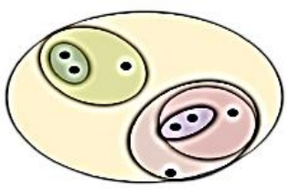
- **Document level**: It serves to regroup documents about the same topic. Document clustering has applications in news articles, emails, search engines, etc.
- **Sentence level**: It's used to cluster sentences derived from different documents. Tweet analysis is an example.
- **Word level**: Word clusters are groups of words based on a common theme. The easiest way to build a cluster is by collecting synonyms for a particular word. For example, WordNet is a lexical database for the English language that groups English words into sets of synonyms called *synsets*.



K-means clustering



Mixture model (Gaussian)



Hierarchical clustering

Graph based clustering

TextClustering Algorithms:

- **Hierarchical**: In the *divisive* approach, we start with one cluster and split that into sub-clusters. Example algorithms include DIANA and MONA. In the *agglomerative* approach, each document starts as its own cluster and then we merge similar ones into bigger clusters. Examples include BIRCH and CURE.

- **Partitioning**: k-means is a popular algorithm but requires the right choice of *k*. Other examples are ISODATA and PAM.
- **Density**: Instead of using a distance measure, we form clusters based on how many data points fall within a given radius. DBSCAN is the most well-known algorithm.
- **Graph**: Some algorithms have made use of knowledge graphs to assess document similarity. This addresses the problem of polysemy (ambiguity) and synonymy (similar meaning).
- **Probabilistic**: A cluster of words belong to a topic and the task is to identify these topics. Words also have probabilities that they belong to a topic. *Topic Modelling* is a separate NLP task but it's similar to soft clustering. pLSA and LDA are example topic models.

Measuring the quality of a clustering algorithm has shown to be as important as the algorithm itself. We can evaluate it in two ways:

- **External quality measure**: External knowledge is required for measuring the external quality. For example, we can conduct surveys of users of the application that includes text clustering.
- **Internal quality measure**: The evaluation of the clustering is compared only with the result itself, that is, the structure of found clusters and their relations to one another. Two main concepts are compactness and separation. *Compactness* measures how closely data points are grouped in a cluster. *Separation* measures how different the found clusters are from each other. More formally, compactness is intra-cluster variance whereas separation is inter-cluster distance.

The challenges include the following:

- Selecting appropriate features of documents that should be used for clustering.
- Selecting an appropriate similarity measure between documents.
- Selecting an appropriate clustering method utilising the above similarity measure.
- Implementing the clustering algorithm in an efficient way that makes it feasible in terms of memory and CPU resources.
- Finding ways of assessing the quality of the performed clustering.