Chapter 3: ***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?

Raw data can have missing or inconsistent values as well as present a lot of redundant information. The most common problems you can find with raw data can be divided into 3 groups:

- ***Missing data:*** *you can also see this as inaccurate data since the information that isn't there creates gaps that might be relevant to the final analysis. Missing data often appears when there's a problem in the collection phase, such as a glitch that caused a system's downtime, mistakes in data entry, or issues with biometrics use, among others.*
- ***Noisy data:*** *this group encompasses erroneous data and outliers that you can find in the data set but that is just meaningless information. Here you can see noise made of human mistakes, rare exceptions, mislabels, and other issues during data gathering.*
- ***Inconsistent data:*** *inconsistencies happen when you keep files with similar data in different formats and files. Duplicates in different formats, mistakes in codes of names, or the absence of data constraints often lead to inconsistent data, that introduces deviations that you have to deal with before analysis.*

If you didn't take care of those issues, the final output would be plagued with faulty insights. That's especially true for more sensitive analysis that can be more affected by small mistakes, like when it's used in new fields where minimal variations in raw data can lead to wrong assumptions.

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.*

Major Tasks in Data Preprocessing

There are the major steps involved in data preprocessing, namely, data cleaning, data integration, data reduction, and data transformation as follows –

- ***Data Cleaning*** *– Data cleaning routines operate to "clean" the information by filling in missing values, smoothing noisy information, identifying or eliminating outliers, and resolving deviation. If users understand the data are dirty, they are unlikely to trust the results of some data mining that has been used.*

  Moreover, dirty data can make confusion for the mining phase, resulting in unstable output. Some mining routines have some phase for dealing with incomplete or noisy information, they are not always potent. Instead, they can concentrate on preventing overfitting the information to the function being modeled.

- ***Data Integration*** *– Data integration is the procedure of merging data from several disparate sources. While performing data integration, it must work on data redundancy, inconsistency, duplicity, etc. In data mining, data integration is a record preprocessing method that includes merging data from a couple of the heterogeneous data sources into coherent data to retain and provide a unified perspective of the data.*

- *Data integration is especially important in the healthcare industry. Integrated data from multiple patient data and clinics assist clinicians in recognizing medical disorders and diseases by integrating data from multiple systems into an individual perspective of beneficial data from which beneficial insights can be derived.*

- ***Data Reduction*** *– The objective of Data reduction is to define it more compactly. When the data size is smaller, it is simpler to use sophisticated and computationally high-cost algorithms. The reduction of the data can be in terms of the multiple rows (records) or terms of the multiple columns (dimensions).*

- *In dimensionality reduction, data encoding schemes are used so as to acquire a reduced or "compressed" description of the initial data. Examples involve data compression methods (e.g., wavelet transforms and principal components analysis), attribute subset selection (e.g., removing irrelevant attributes), and attribute construction (e.g., where a small set of more beneficial attributes is changed from the initial set).*

- *In numerosity reduction, the data are restored by alternative, smaller description using parametric models such as regression or log-linear models or nonparametric models such as histograms, clusters, sampling, or data aggregation.*

- ***Data transformation*** *– In data transformation, where data are transformed or linked into forms applicable for mining by executing summary or aggregation operations. In Data transformation, it includes –*

- ***Smoothing*** *– It can work to remove noise from the data. Such techniques includes binning, regression, and clustering.*

- ***Aggregation*** *– In aggregation, where summary or aggregation services are used to the data. For instance, the daily sales data can be aggregated to calculate monthly and annual total amounts. This procedure is generally used in developing a data cube for the analysis of the records at several granularities.*

3.2 Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent. *Data cleaning* (or *data cleansing*) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. In this section, you will study basic methods for data cleaning. Section 3.2.1 looks at ways of handling missing values. Section 3.2.2 explains data smoothing techniques. Section 3.2.3 discusses approaches to data cleaning as a process.

3.2.1 Missing Values

Imagine that you need to analyze *AllElectronics* sales and customer data. You note that many tuples have no recorded value for several attributes such as customer *income*. How can you go about filling in the missing values for this attribute? Let's look at the following methods.

1. **Ignore the tuple**: This is usually done when the class label is missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably. By ignoring the tuple, we do not make use of the remaining attributes' values in the tuple. Such data could have been useful to the task at hand.
2. **Fill in the missing value manually**: In general, this approach is time consuming and may not be feasible given a large data set with many missing values.
3. **Use a global constant to fill in the missing value**: Replace all missing attribute 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 foolproof.
4. **Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value**: Chapter 2 discussed measures of central tendency, which indicate the "middle" value of a data distribution. For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median (Section 2.2). For example, suppose that the data distribution regarding the income of *AllElectronics* customers is symmetric and that the mean income is $56,000. Use this value to replace the missing value for *income*.
5. **Use the attribute mean or median for all samples belonging to the same class as the given tuple**: For example, if classifying customers according to *credit_risk*, we may replace the missing value with the mean *income* value for customers in the same credit risk category as that of the given tuple. If the data distribution for a given class is skewed, the median value is a better choice.
6. **Use the most probable value to fill in the missing value**: This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree

Induction.

3.2.2 Noisy data

Noisy data *are data with a large amount of additional meaningless information called noise. This includes data corruption, and the term is often used as a synonym for corrupt data. It also includes any data that a user system cannot understand and interpret correctly.*

Noise is an unavoidable problem that affects the data collection and preparation processes in Data Mining applications, where errors commonly occur. Noise has two main sources, such as:

1. *Implicit errors are introduced by measurement tools, such as different types of sensors.*
2. *And random errors are introduced by batch processes or experts when the data are gathered, such as in a document digitalization process.*

Removing noise from a data set is termed data smoothing. The following ways can be used for Smoothing:

1. Binning

Binning is a technique where we sort the data and then partition the data into equal frequency bins. Then you may either replace the noisy data with the bin mean bin median or the bin boundary. This method is to smooth or handle noisy data. First, the data is sorted then, and then the sorted values are separated and stored in the form of bins. There are three methods for smoothing data in the bin.

- ***Smoothing by bin mean method:*** *In this method, the values in the bin are replaced by the mean value of the bin.*
- ***Smoothing by bin median:*** *In this method, the values in the bin are replaced by the median value.*
- ***Smoothing by bin boundary:*** *In this method, the using minimum and maximum values of the bin values are taken, and the closest boundary value replaces the values.*

2. Regression

This is used to smooth the data and help handle data when unnecessary data is present. For the analysis, purpose regression helps decide the suitable variable. ***Linear regression*** *refers to finding the best line to fit between two variables so that one can be used to predict the other.* ***Multiple linear regression*** *involves more than two variables. Using regression to find a mathematical equation to fit into the data helps to smooth out the noise.*

3. Clustering

This is used for finding the outliers and also in grouping the data. Clustering is generally used in unsupervised learning.

4. Outlier Analysis

Outliers may be detected by clustering, where similar or close values are organized into the same groups or clusters. Thus, values that fall far apart from the cluster may be considered noise or outliers. Outliers are extreme values that deviate from other observations on data. They may indicate variability in measurement, experimental errors, or novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. Outliers can be the following kinds, such as:

- ***Univariate outliers*** *can be found when looking at a distribution of values in a single feature space.*
- ***Multivariate outliers*** *can be found in an n-dimensional space (of n-features). Looking at distributions in n-dimensional spaces can be very difficult for the human brain. That is why we need to train a model to do it for us.*
- ***Point outliers*** *are single data points that lay far from the rest of the distribution.*
- ***Contextual outliers*** *can be noise in data, such as punctuation symbols when realizing text analysis or background noise signal when doing speech recognition.*
- ***Collective outliers*** *can be subsets of novelties in data, such as a signal that may indicate the discovery of new phenomena.*

Data cleaning is an important stage. After all, your results are based on your data. The more the dirt, the more inaccurate your results would prove.

Data Cleaning *eliminates noise and missing values. Data Cleaning is just the first of the many steps for data pre-processing. In addition to the above, data pre-processing includes* ***Aggregation, Feature Construction, Normalization, Discretization, Concept hierarchy generation****, which mostly deal with making the data consistent. Data pre-processing, at times, also comprises 90% of the entire process.*

Advantages and benefits of data cleaning

Having clean data will ultimately increase overall productivity and allow for the highest quality information in your decision-making. Benefits include:

- *Removal of errors when multiple sources of data are at play.*
- *Fewer errors make for happier clients and less-frustrated employees.*
- *Ability to map the different functions and what your data is intended to do.*
- *Monitoring errors and better reporting to see where errors are coming from, making it easier to fix incorrect or corrupt data for future applications.*
- *Using tools for data cleaning will make for more efficient business practices and quicker decision-making.*

Data cleaning as a process

Data integration

Data integration is the process of merging data from several disparate sources. While performing data integration, you must work on data redundancy, inconsistency, duplicity, etc. In data mining, data integration is a record preprocessing method that includes merging data from a couple of the heterogeneous data sources into coherent data to retain and provide a unified perspective of the data.

Issues in Data Integration

When you integrate the data in Data Mining, you may face many issues. There are some of those issues:

- *Entity Identification Problem*

  As you understand, the records are obtained from heterogeneous sources, and how can you 'match the real-world entities from the data'. For example, you were given client data from specialized statistics sites. Customer identity is assigned to an entity from one statistics supply, while a customer range is assigned to an entity from another statistics supply. Analyzing such metadata statistics will prevent you from making errors during schema integration.

  Structural integration is completed by guaranteeing that the functional dependency and referential constraints of a character in the source machine match the functional dependency and referential constraints of the identical character in the target machine. For example, assume that the discount is applied to the entire order in one machine, but in every other machine, the discount is applied to each item in the order. This distinction should be noted before the information from those assets is included in the goal system.

➢ *Redundancy and Correlation Analysis*

One of the major issues in the course of data integration is redundancy. Unimportant data that are no longer required are referred to as redundant data. It may also appear due to attributes created from the use of another property inside the information set. For example, if one truth set contains the patronage and distinct data set as the purchaser's date of the beginning, then age may be a redundant attribute because it can be deduced from the use of the beginning date.

Inconsistencies further increase the level of redundancy within the characteristic. The use of correlation analysis can be used to determine redundancy. The traits are examined to determine their interdependence on each difference, consequently discovering the link between them.

χ^2 Correlation Test for Nominal Data

For nominal data, a correlation relationship between two attributes, A and B, can be discovered by a χ^2 (**chi-square**) test. Suppose A has c distinct values, namely $a_1, a_2, \ldots a_c$. B has r distinct values, namely $b_1, b_2, \ldots b_r$. The data tuples described by A and B can be shown as a **contingency table**, with the c values of A making up the columns and the r values of B making up the rows. Let (A_i, B_j) denote the joint event that attribute A takes on value a_i and attribute B takes on value b_j, that is, where $(A = a_i, B = b_j)$. Each and every possible (A_i, B_j) joint event has its own cell (or slot) in the table. The χ^2 value (also known as the *Pearson* χ^2 *statistic*) is computed as

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \tag{3.1}$$

where o_{ij} is the *observed frequency* (i.e., actual count) of the joint event (A_i, B_j) and e_{ij} is the *expected frequency* of (A_i, B_j), which can be computed as

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}, \tag{3.2}$$

where n is the number of data tuples, $count(A = a_i)$ is the number of tuples having value a_i for A, and $count(B = b_j)$ is the number of tuples having value b_j for B. The sum in Eq. (3.1) is computed over all of the $r \times c$ cells. Note that the cells that contribute the most to the χ^2 value are those for which the actual count is very different from that expected.

The χ^2 statistic tests the hypothesis that A and B are *independent*, that is, there is no correlation between them. The test is based on a significance level, with $(r - 1) \times (c - 1)$ degrees of freedom. We illustrate the use of this statistic in Example 3.1. If the hypothesis can be rejected, then we say that A and B are statistically correlated.

Covariance of Numeric Data

In probability theory and statistics, correlation and covariance are two similar measures for assessing how much two attributes change together. Consider two numeric attributes A and B, and a set of n observations $\{(a_1, b_1), \ldots, (a_n, b_n)\}$. The mean values of A and B, respectively, are also known as the **expected values** on A and B, that is,

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n}$$

and

$$E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}.$$

The **covariance** between A and B is defined as

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}. \tag{3.4}$$

If we compare Eq. (3.3) for $r_{A,B}$ (correlation coefficient) with Eq. (3.4) for covariance, we see that

$$r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B}, \tag{3.5}$$

where σ_A and σ_B are the standard deviations of A and B, respectively. It can also be shown that

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}. \tag{3.6}$$

This equation may simplify calculations.

For two attributes A and B that tend to change together, if A is larger than $\bar{A}$ (the expected value of A), then B is likely to be larger than $\bar{B}$ (the expected value of B). Therefore, the covariance between A and B is *positive*. On the other hand, if one of the attributes tends to be above its expected value when the other attribute is below its expected value, then the covariance of A and B is *negative*.

If A and B are *independent* (i.e., they do not have correlation), then $E(A \cdot B) = E(A) \cdot E(B)$. Therefore, the covariance is $Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B} = E(A) \cdot E(B) - \bar{A}\bar{B} = 0$. However, the converse is not true. Some pairs of random variables (attributes) may have a covariance of 0 but are not independent. Only under some additional assumptions

Correlation Coefficient for Numeric Data

For numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the **correlation coefficient** (also known as **Pearson's product moment coefficient**, named after its inventer, Karl Pearson). This is

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B}, \tag{3.3}$$

where n is the number of tuples, a_i and b_i are the respective values of A and B in tuple i, $\bar{A}$ and $\bar{B}$ are the respective mean values of A and B, σ_A and σ_B are the respective standard deviations of A and B (as defined in Section 2.2.2), and $\Sigma(a_i b_i)$ is the sum of the AB cross-product (i.e., for each tuple, the value for A is multiplied by the value for B in that tuple). Note that $-1 \le r_{A,B} \le +1$. If $r_{A,B}$ is greater than 0, then A and B are *positively correlated*, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that A (or B) may be removed as a redundancy.

If the resulting value is equal to 0, then A and B are *independent* and there is no correlation between them. If the resulting value is less than 0, then A and B are *negatively correlated*, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other. Scatter plots can also be used to view correlations between attributes (Section 2.2.3). For example, Figure 2.8's

scatter plots respectively show positively correlated data and negatively correlated data, while Figure 2.9 displays uncorrelated data.

Note that correlation does not imply causality. That is, if A and B are correlated, this does not necessarily imply that A causes B or that B causes A. For example, in analyzing a demographic database, we may find that attributes representing the number of hospitals and the number of car thefts in a region are correlated. This does not mean that one causes the other. Both are actually causally linked to a third attribute, namely, *population*.

➢ *Tuple Duplication*

Information integration has also handled duplicate tuples in addition to redundancy. Duplicate tuples may also appear in the generated information if the denormalized table was utilized as a deliverable for data integration.

➢ *Data warfare Detection and backbone*

The data warfare technique of combining records from several sources is unhealthy. In the same way, that characteristic values can vary, so can statistics units. The disparity may be related to the fact that they are represented differently within the special data units. For example, in one-of-a-kind towns, the price of an inn room might be expressed in a particular currency. This type of issue is recognized and fixed during the data integration process.

Data reduction *techniques ensure the integrity of data while reducing the data. Data reduction is a process that reduces the volume of original data and represents it in a much smaller volume. Data reduction techniques are used to obtain a reduced representation of the dataset that is much smaller in volume by maintaining the integrity of the original data. By reducing the data, the efficiency of the data mining process is improved, which produces the same analytical results.*

Techniques of Data Reduction

Here are the following techniques or methods of data reduction in data mining, such as:

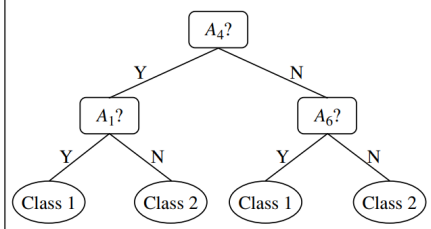
1. Dimensionality Reduction

Whenever we encounter weakly important data, we use the attribute required for our analysis. Dimensionality reduction eliminates the attributes from the data set under consideration, thereby reducing the volume of original data. It reduces data size as it eliminates outdated or redundant features. Here are three methods of dimensionality reduction.

i. ***Wavelet Transform:*** *In the wavelet transform, suppose a data vector A is transformed into a numerically different data vector A' such that both A and A' vectors are of the same length. Then how it is useful in reducing data because the data obtained from the wavelet transform can be truncated. The compressed data is obtained by retaining the smallest fragment of the strongest wavelet coefficients. Wavelet transform can be applied to data cubes, sparse data, or skewed data.*

ii. ***Principal Component Analysis:*** *Suppose we have a data set to be analyzed that has tuples with n attributes. The principal component analysis identifies k independent tuples with n attributes that can represent the data set.*
In this way, the original data can be cast on a much smaller space, and dimensionality reduction can be achieved. Principal component analysis can be applied to sparse and skewed data.

iii. ***Attribute Subset Selection:*** *The large data set has many attributes, some of which are irrelevant to data mining or some are redundant. The core attribute subset selection reduces the data volume and dimensionality. The attribute subset selection reduces the volume of data by eliminating redundant and irrelevant attributes.*
The attribute subset selection ensures that we get a good subset of original attributes even after eliminating the unwanted attributes. The resulting probability of data distribution is as close as possible to the original data distribution using all the attributes.

Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  => Reduced attribute set: $\{A_1, A_4, A_6\}$

1. **Stepwise forward selection**: The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.
2. **Stepwise backward elimination**: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
3. **Combination of forward selection and backward elimination**: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
4. **Decision tree induction**: Decision tree algorithms (e.g., ID3, C4.5, and CART) were originally intended for classification. Decision tree induction constructs a flowchart-like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes.
   When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

The stopping criteria for the methods may vary. The procedure may employ a threshold on the measure used to determine when to stop the attribute selection process.

In some cases, we may want to create new attributes based on others. Such **attribute construction**[6] can help improve accuracy and understanding of structure in high-dimensional data. For example, we may wish to add the attribute *area* based on the attributes *height* and *width*. By combining attributes, attribute construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

2. sNumerosity Reduction

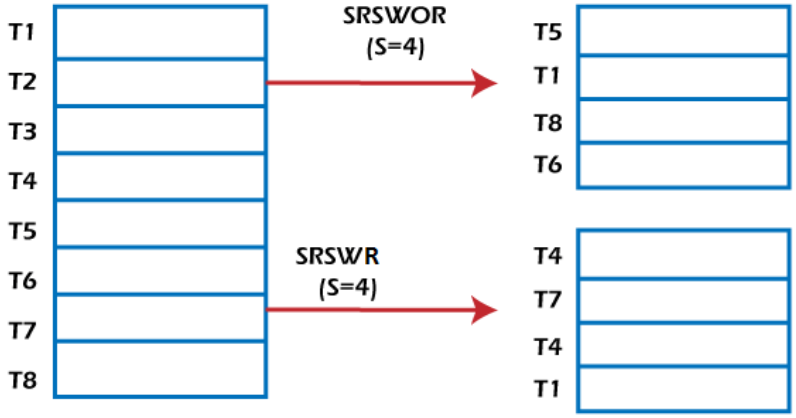
The numerosity reduction reduces the original data volume and represents it in a much smaller form. This technique includes two types parametric and non-parametric numerosity reduction.

i. ***Parametric:*** *Parametric numerosity reduction incorporates storing only data parameters instead of the original data. One method of parametric numerosity reduction is the regression and log-linear method.*
   - ***Regression and Log-Linear:*** *Linear regression models a relationship between the two attributes by modeling a linear equation to the data set. Suppose we need to model a linear function between two attributes.*
     y = wx +b
     Here, y is the response attribute, and x is the predictor attribute. If we discuss in terms of data mining, attribute x and attribute y are the numeric database attributes, whereas w and b are regression coefficients.
     Multiple linear regressions let the response variable y model linear function between two or more predictor variables.
     Log-linear model discovers the relation between two or more discrete attributes in the database. Suppose we have a set of tuples presented in n-dimensional space. Then the log-linear model is used to study the probability of each tuple in a multidimensional space.
     Regression and log-linear methods can be used for sparse data and skewed data.

ii. ***Non-Parametric:*** *A non-parametric numerosity reduction technique does not assume any model. The non-Parametric technique results in a more uniform reduction, irrespective of data size, but it may not achieve a*

high volume of data reduction like the parametric. There are at least four types of Non-Parametric data reduction techniques, Histogram, Clustering, Sampling, Data Cube Aggregation, and Data Compression.

- ***Histogram:*** *A histogram is a graph that represents frequency distribution which describes how often a value appears in the data. Histogram uses the binning method to represent an attribute's data distribution. It uses a disjoint subset which we call bin or buckets.*
  A histogram can represent a dense, sparse, uniform, or skewed data. Instead of only one attribute, the histogram can be implemented for multiple attributes. It can effectively represent up to five attributes.
- ***Clustering:*** *Clustering techniques groups similar objects from the data so that the objects in a cluster are similar to each other, but they are dissimilar to objects in another cluster.*
  How much similar are the objects inside a cluster can be calculated using a distance function. More is the similarity between the objects in a cluster closer they appear in the cluster.
  The quality of the cluster depends on the diameter of the cluster, i.e., the max distance between any two objects in the cluster.
  The cluster representation replaces the original data. This technique is more effective if the present data can be classified into a distinct clustered.
- ***Sampling:*** *One of the methods used for data reduction is sampling, as it can reduce the large data set into a much smaller data sample. Below we will discuss the different methods in which we can sample a large data set D containing N tuples:*
  1. ***Simple random sample without replacement (SRSWOR) of size s:*** *In this s, some tuples are drawn from N tuples such that in the data set D (s<N). The probability of drawing any tuple from the data set D is 1/N. This means all tuples have an equal probability of getting sampled.*
  2. ***Simple random sample with replacement (SRSWR) of size s:*** *It is similar to the SRSWOR, but the tuple is drawn from data set D, is recorded, and then replaced into the data set D so that it can be drawn again.*

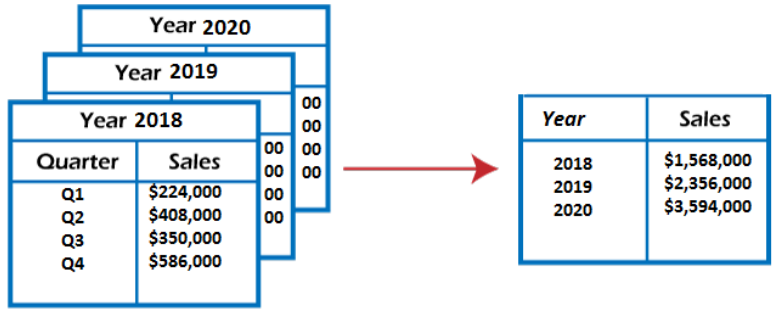


  3. ***Cluster sample:*** *The tuples in data set D are clustered into M mutually disjoint subsets. The data reduction can be applied by implementing SRSWOR on these clusters. A simple random sample of size s could be generated from these clusters where s<M.*
  4. ***Stratified sample:*** *The large data set D is partitioned into mutually disjoint sets called 'strata'. A simple random sample is taken from each stratum to get stratified data. This method is effective for skewed data.*

3. Data Cube Aggregation

This technique is used to aggregate data in a simpler form. Data Cube Aggregation is a multidimensional aggregation that uses aggregation at various levels of a data cube to represent the original data set, thus achieving data reduction.

For example, suppose you have the data of All Electronics sales per quarter for the year 2018 to the year 2022. If you want to get the annual sale per year, you just have to aggregate the sales per quarter for each year. In this way, aggregation provides you with the required data, which is much smaller in size, and thereby we achieve data reduction even without losing any data.

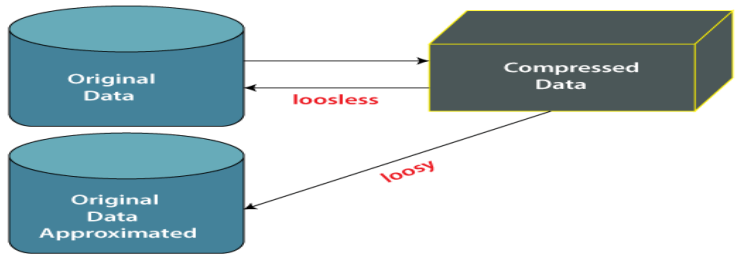


Aggregated Data

The data cube aggregation is a multidimensional aggregation that eases multidimensional analysis. The data cube present precomputed and summarized data which eases the data mining into fast access.

4. Data Compression

Data compression employs modification, encoding, or converting the structure of data in a way that consumes less space. Data compression involves building a compact representation of information by removing redundancy and representing data in binary form. Data that can be restored successfully from its compressed form is called Lossless compression. In contrast, the opposite where it is not possible to restore the original form from the compressed form is Lossy compression. Dimensionality and numerosity reduction method are also used for data compression.



This technique reduces the size of the files using different encoding mechanisms, such as Huffman Encoding and run-length Encoding. We can divide it into two types based on their compression techniques.

i. ***Lossless Compression:*** *Encoding techniques (Run Length Encoding) allow a simple and minimal data size reduction. Lossless data compression uses algorithms to restore the precise original data from the compressed data.*

ii. ***Lossy Compression:*** *In lossy-data compression, the decompressed data may differ from the original data but are useful enough to retrieve information from them. For example, the JPEG image format is a lossy compression, but we can find the meaning equivalent to the original image. Methods such as the Discrete Wavelet transform technique PCA (principal component analysis) are examples of this compression.*

5. Discretization Operation

The data discretization technique is used to divide the attributes of the continuous nature into data with intervals. We replace many constant values of the attributes with labels of small intervals. This means that mining results are shown in a concise and easily understandable way.

i. ***Top-down discretization:*** *If you first consider one or a couple of points (so-called breakpoints or split points) to divide the whole set of attributes and repeat this method up to the end, then the process is known as top-down discretization, also known as splitting.*

ii. ***Bottom-up discretization:*** *If you first consider all the constant values as split-points, some are discarded through a combination of the neighborhood values in the interval. That process is called bottom-up discretization.*

Benefits of Data Reduction

The main benefit of data reduction is simple: the more data you can fit into a terabyte of disk space, the less capacity you will need to purchase. Here are some benefits of data reduction, such as:

- *Data reduction can save energy.*
- *Data reduction can reduce your physical storage costs.*
- *And data reduction can decrease your data center track.*

Data reduction greatly increases the efficiency of a storage system and directly impacts your total spending on capacity.

Data transformation

Data transformation is a technique used to ***convert*** *the raw data into a suitable format that efficiently eases data mining and retrieves strategic information. Data transformation includes data cleaning techniques and a data reduction technique to convert the data into the appropriate form.*

Data Transformation Techniques

There are several data transformation techniques that can help structure and clean up the data before analysis or storage in a data warehouse. Let's study all techniques used for data transformation, some of which we have already studied in data reduction and data cleaning.

1. Data Smoothing

Data smoothing is a process that is used to remove noise from the dataset using some algorithms. It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form.

The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn't see otherwise.

We have seen how the noise is removed from the data using the techniques such as binning, regression, clustering.

- ***Binning:*** *This method splits the sorted data into the number of bins and smoothens the data values in each bin considering the neighborhood values around it.*
- ***Regression:*** *This method identifies the relation among two dependent attributes so that if we have one attribute, it can be used to predict the other attribute.*
- ***Clustering:*** *This method groups similar data values and form a cluster. The values that lie outside a cluster are known as outliers.*

2. Attribute Construction

In the attribute construction method, the new attributes consult the existing attributes to construct a new data set that eases data mining. New attributes are created and applied to assist the mining process from the given attributes. This simplifies the original data and makes the mining more efficient.

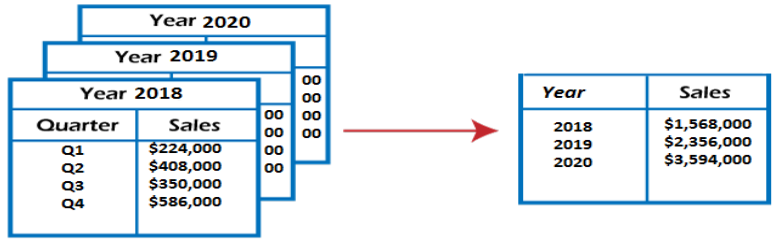
For example, suppose we have a data set referring to measurements of different plots, i.e., we may have the height and width of each plot. So here, we can construct a new attribute 'area' from attributes 'height' and 'weight'. This also helps understand the relations among the attributes in a data set.

3. Data Aggregation

Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used.

Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results. The collection of data is useful for everything from decisions concerning financing or business strategy of the product, pricing, operations, and marketing strategies.

For example, we have a data set of sales reports of an enterprise that has quarterly sales of each year. We can aggregate the data to get the enterprise's annual sales report.



Aggregated Data

4. Data Normalization

Normalizing the data refers to scaling the data values to a much smaller range such as [-1, 1] or [0.0, 1.0]. There are different methods to normalize the data, as discussed below.

Consider that we have a numeric attribute A and we have n number of observed values for attribute A that are V1, V_2, V_3, ...V_n.

- ***Min-max normalization:*** *This method implements a linear transformation on the original data. Let us consider that we have min_A and max_A as the minimum and maximum value observed for attribute A and V_i is the value for attribute A that has to be normalized.*
  The min-max normalization would map V_i to the V'_i in a new smaller range [new_min_A, new_max_A]. The formula for min-max normalization is given below:

$$v'_i = \frac{v_i - min_A}{max_A - min_A}\left(new_{max_A} - new_{min_A}\right) + new\_min_A$$

For example, we have $1200 and $9800 as the minimum, and maximum value for the attribute income, and [0.0, 1.0] is the range in which we have to map a value of $73,600.
The value $73,600 would be transformed using min-max normalization as follows:

$$\frac{73600-1200}{9800-1200}(1.0 - 0.0) + 0.0 = 0.716$$

- ***Z-score normalization:*** *This method normalizes the value for attribute A using the* ***mean****and* ***standard deviation****. The following formula is used for Z-score normalization:*

$$v'_i = \frac{v_i - \bar{A}}{\sigma_A}$$

Here $\bar{A}$ and σ_A are the mean and standard deviation for attribute A, respectively.
For example, we have a mean and standard deviation for attribute A as $54,000 and $16,000. And we have to normalize the value $73.600 using z-score normalization.

$$\frac{73600-5400}{1600} = 1.225$$

- ***Decimal Scaling:*** *This method normalizes the value of attribute A by moving the decimal point in the value. This movement of a decimal point depends on the maximum absolute value of A. The formula for the decimal scaling is given below:*

$$v'_i = \frac{v_i}{10^j}$$

Here j is the smallest integer such that max(|v'_i|)<1
For example, the observed values for attribute A range from -986 to 917, and the maximum absolute value for attribute A is 986. Here, to normalize each value of attribute A using decimal scaling, we have to divide each value of attribute A by 1000, i.e., j=3.
So, the value -986 would be normalized to -0.986, and 917 would be normalized to 0.917.
The normalization parameters such as mean, standard deviation, the maximum absolute value must be preserved to normalize the future data uniformly.

5. 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 data mining task handles a continuous attribute, then its discrete values can be replaced by constant quality attributes. This improves the efficiency of the task.*
- *This method is also called a data reduction mechanism as it transforms a large dataset into a set of categorical data. Discretization also uses decision tree-based algorithms to produce short, compact, and accurate results when using discrete values.*
- *Data discretization can be classified into two types:* ***supervised discretization****, where the class information is used, and* ***unsupervised discretization****, which is based on which direction the process proceeds, i.e., 'top-down splitting strategy' or 'bottom-up merging strategy'.*
- *For example, the values for the age attribute can be replaced by the interval labels such as (0-10, 11-20…) or (kid, youth, adult, senior).*

Advantages of Data Transformation

- ***Better Organization:*** *Transformed data is easier for both humans and computers to use.*
- ***Improved Data Quality:*** *There are many* ***risks and costs associated with bad data****. Data transformation can help your organization eliminate quality issues such as missing values and other inconsistencies.*
- ***Perform Faster Queries:*** *You can quickly and easily retrieve transformed data thanks to it being stored and standardized* ***in a source location.***
- ***Better Data Management:*** *Businesses are constantly generating data from more and more sources. If there are inconsistencies in the metadata, it can be* ***challenging to organize and understand it.*** *Data transformation refines your metadata, so it's easier to organize and understand.*
- ***More Use Out of Data:*** *While businesses may be collecting data constantly, a lot of that data* ***sits around unanalyzed****. Transformation makes it easier to get the most out of your data by standardizing it and making it more usable.*

Disadvantages of Data Transformation

- *Data transformation can be expensive. The cost is dependent on the specific infrastructure, software, and tools used to process data. Expenses may include licensing, computing resources, and hiring necessary personnel.*
- *Data transformation processes can be resource-intensive. Performing transformations in an on-premises data warehouse after loading or transforming data before feeding it into applications can create a computational burden that slows down other operations. If you use a cloud-based data warehouse, you can do the transformations after loading because the platform can scale up to meet demand.*
- *Lack of expertise and carelessness can introduce problems during transformation. Data analysts without appropriate subject matter expertise are less likely to notice incorrect data because they are less familiar with the range of accurate and permissible values.*
- *Enterprises can perform transformations that don't suit their needs. A business might change information to a specific format for one application only to then revert the information to its prior format for a different application.*

Unit -02

Frequent pattern mining

Frequent pattern mining in data mining is the process of identifying patterns or associations within a dataset that occur frequently. This is typically done by analyzing large datasets to find items or sets of items that appear together frequently.

There are several different algorithms used for frequent pattern mining, including:

1. *Apriori algorithm: This is one of the most commonly used algorithms for frequent pattern mining. It uses a "bottom-up" approach to identify frequent itemsets and then generates association rules from those itemsets.*
2. *ECLAT algorithm: This algorithm uses a "depth-first search" approach to identify frequent itemsets. It is particularly efficient for datasets with a large number of items.*
3. *FP-growth algorithm: This algorithm uses a "compression" technique to find frequent patterns efficiently. It is particularly efficient for datasets with a large number of transactions.*
4. *Frequent pattern mining has many applications, such as Market Basket Analysis, Recommender Systems, Fraud Detection, and many more.*

Advantages:

1. *It can find useful information which is not visible in simple data browsing*
2. *It can find interesting association and correlation among data items*

Disadvantages:

1. *It can generate a large number of patterns*
2. *With high dimensionality, the number of patterns can be very large, making it difficult to interpret the results.*

Applications of Frequent Pattern Mining: *basket data analysis, cross-marketing, catalog design, sale campaign analysis, web log analysis, and DNA sequence analysis.*

Issues of frequent pattern mining

- *flexibility and reusability for creating frequent patterns*
- *most of the algorithms used for mining frequent item sets do not offer flexibility for reusing*
- *much research is needed to reduce the size of the derived patterns*
- *Frequent pattern mining has several applications in different areas, including:*
- *Market Basket Analysis: This is the process of analyzing customer purchasing patterns in order to identify items that are frequently bought together. This information can be used to optimize product placement, create targeted marketing campaigns, and make other business decisions.*
- *Recommender Systems: Frequent pattern mining can be used to identify patterns in user behavior and preferences in order to make personalized recommendations.*
- *Fraud Detection: Frequent pattern mining can be used to identify abnormal patterns of behavior that may indicate fraudulent activity.*
- *Network Intrusion Detection: Network administrators can use frequent pattern mining to detect patterns of network activity that may indicate a security threat.*
- *Medical Analysis: Frequent pattern mining can be used to identify patterns in medical data that may indicate a particular disease or condition.*
- *Text Mining: Frequent pattern mining can be used to identify patterns in text data, such as keywords or phrases that appear frequently together in a document.*
- *Web usage mining: Frequent pattern mining can be used to analyze patterns of user behavior on a website, such as which pages are visited most frequently or which links are clicked on most often.*
- *Gene Expression: Frequent pattern mining can be used to analyze patterns of gene expression in order to identify potential biomarkers for different diseases.*

What is Apriori Algorithm?

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. *Support*
2. *Confidence*
3. *Lift*

Let's take an example to understand this concept.

We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.

Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

Support:*Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get*

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)= 400/4000 = 10 percent.

Confidence: *Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.*

Hence,Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400 = 50 percent. It means that 50 percent of customers who bought biscuits bought chocolates also.

Lift*: Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.*

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits) = 50/10 = 5.It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

How does the Apriori Algorithm work in Data Mining?

We will understand this algorithm with the help of an example

Consider a Big Bazar scenario where the product set is P = {Rice, Pulse, Oil, Milk, Apple}. The database comprises six transactions where 1 represents the presence of the product and 0 represents the absence of the product.

Transaction ID	***Rice***	***Pulse***	***Oil Milk***	***Apple***	
t1	*1*	*1*	*1*	*0*	*0*
t2	*0*	*1*	*1*	*1*	*0*
t3	*0*	*0*	*0*	*1*	*1*
t4	*1*	*1*	*0*	*1*	*0*
t5	*1*	*1*	*1*	*0*	*1*
t6	*1*	*1*	*1*	*1*	*1*

The Apriori Algorithm makes the given assumptions

- *All subsets of a frequent itemset must be frequent.*
- *The subsets of an infrequent item set must be infrequent.*
- *Fix a threshold support level. In our case, we have fixed it at 50 percent.*

Step 1*: Make a frequency table of all the products that appear in all the transactions. Now, short the frequency table to add only those products with a threshold support level of over 50 percent. We find the given frequency table.*

Product	***Frequency (Number of transactions)***
Rice (R)	*4*
Pulse(P)	*5*
Oil(O)	*4*
Milk(M)	*4*

The above table indicated the products frequently bought by the customers.

***Step 2**: Create pairs of products such as RP, RO, RM, PO, PM, OM. You will get the given frequency table.*

Itemset	***Frequency (Number of transactions)***
RP	*4*
RO	*3*
RM	*2*
PO	*4*
PM	*3*
OM	*2*

***Step 3:** Implementing the same threshold support of 50 percent and consider the products that are more than 50 percent. In our case, it is more than 3 Thus, we get RP, RO, PO, and PM*

***Step 4**:Now, look for a set of three products that the customers buy together. We get the given combination.*

1. *RP and RO give RPO*
2. *PO and PM give POM*

***Step 5**: Calculate the frequency of the two itemsets, and you will get the given frequency table.*

Itemset	***Frequency (Number of transactions)***
RPO	*4*
POM	*3*

If you implement the threshold assumption, you can figure out that the customers' set of three products is RPO.

Advantages of Apriori Algorithm

- *It is used to calculate large itemsets.*
- *Simple to understand and apply.*

Disadvantages of Apriori Algorithms

- *Apriori algorithm is an expensive method to find support since the calculation has to pass through the whole database.*
- *Sometimes, you need a huge number of candidate rules, so it becomes computationally more expensive.*

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;
```

What is FP Growth Algorithm?

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

This algorithm works as follows:

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

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.

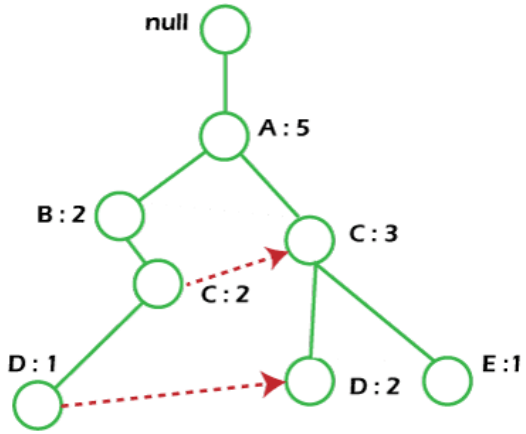
A frequent Pattern Tree is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the item set.

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

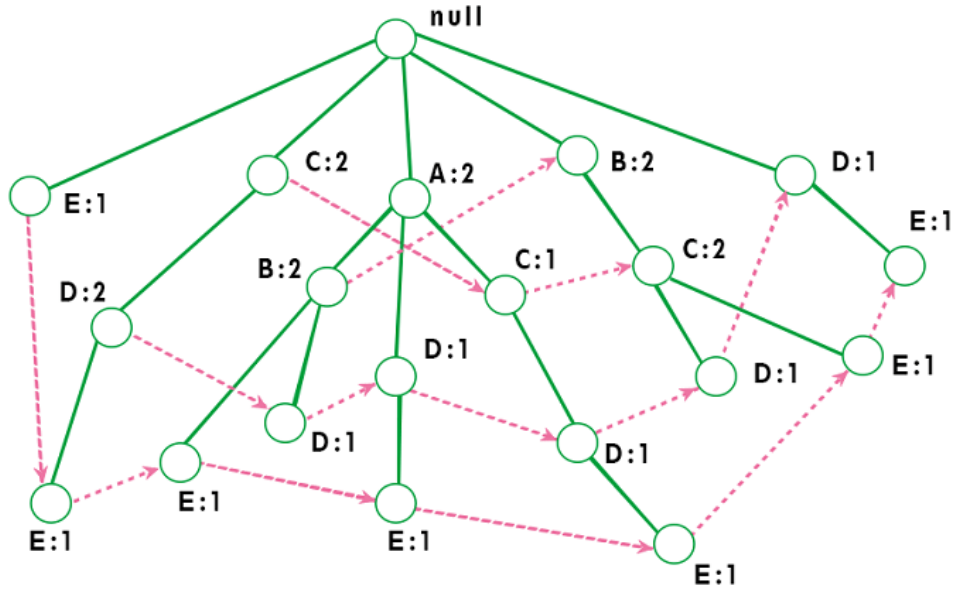
Han defines the FP-tree as the tree structure given below:

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.



Algorithm by Han

The original algorithm to construct the FP-Tree defined by Han is given below:

Algorithm 1: FP-tree construction

Input: A transaction database DB and a minimum support threshold?

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows.

1. *The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets 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, and then the next itemset with the lower count. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.*
4. *The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch, 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. The common node and new node count are 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 is called a conditional pattern base.*
   A 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, formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.*
8. *Frequent Patterns are generated from the Conditional FP Tree.*

Using this algorithm, the FP-tree is constructed in two database scans. The first scan collects and sorts the set of frequent items, and the second constructs the FP-Tree.

Example

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*

Table 2: Count of each item

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

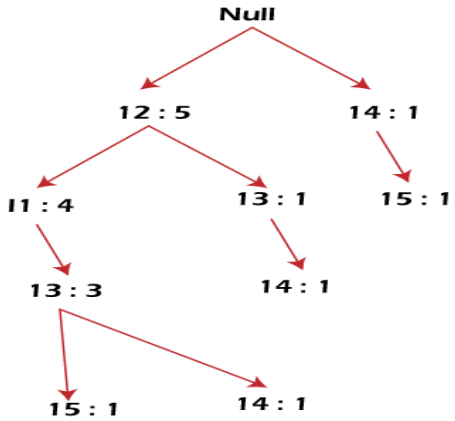
Table 3: Sort the itemset in descending order.

Item	*Count*
I2	*5*
I1	*4*
I3	*4*
I4	*4*

Build FP Tree

Let's build the FP tree in the following steps, such as:

1. *Considering the root node null.*
2. *The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child, I1 is linked to I2 and I3 is linked to I1.*
3. *T2: I2, I3, and I4 contain I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share the I2 node as common as it is already used in T1.*
4. *Increment the count of I2 by 1, and I3 is linked as a child to I2, and I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.*
5. *T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.*
6. *T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node. Hence it will be incremented by 1. Similarly I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.*
7. *T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.*
8. *T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.*



Mining of FP-tree is summarized below:

1. *The lowest node item, I5, is not considered as it does not have a min support count. Hence it is deleted.*
2. *The next lower node is I4. I4 occurs in 2 branches , {I2,I1,I3:,I41},{I2,I3,I4:1}. Therefore considering I4 as suffix the prefix paths will be {I2, I1, I3:1}, {I2, I3: 1} this forms the conditional pattern base.*
3. *The conditional pattern base is considered a transaction database, and an FP tree is constructed. This will contain {I2:2, I3:2}, I1 is not considered as it does not meet the min support count.*
4. *This path will generate all combinations of frequent patterns : {I2,I4:2},{I3,I4:2},{I2,I3,I4:2}*
5. *For I3, the prefix path would be: {I2,I1:3},{I2:1}, this will generate a 2 node FP-tree : {I2:4, I1:3} and frequent patterns are generated: {I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}.*
6. *For I1, the prefix path would be: {I2:4} this will generate a single node FP-tree: {I2:4} and frequent patterns are generated: {I2, I1:4}.*

Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I4	*{I2,I1,I3:1},{I2,I3:1}*	*{I2:2, I3:2}*	*{I2,I4:2},{I3,I4:2},{I2,I3,I4:2}*
I3	*{I2,I1:3},{I2:1}*	*{I2:4, I1:3}*	*{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}*
I1	*{I2:4}*	*{I2:4}*	*{I2,I1:4}*

Advantages of FP Growth Algorithm

- *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

- *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	***FP Growth***
Apriori generates frequent patterns by making the itemsets 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 one database 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 search.*

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

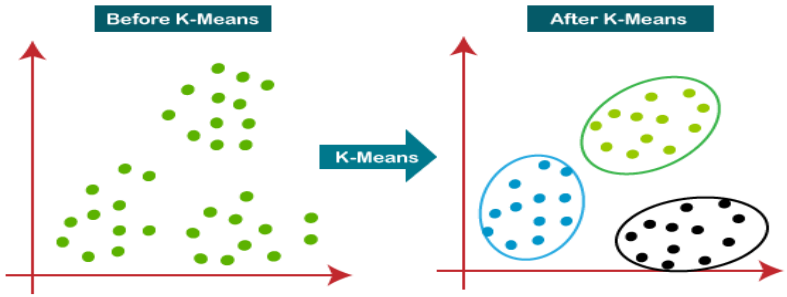
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- *Determines the best value for K center points or centroids by an iterative process.*
- *Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.*

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: *Select the number K to decide the number of clusters.*

Step-2: *Select random K points or centroids. (It can be other from the input dataset).*

Step-3: *Assign each data point to their closest centroid, which will form the predefined K clusters.*

Step-4: *Calculate the variance and place a new centroid of each cluster.*

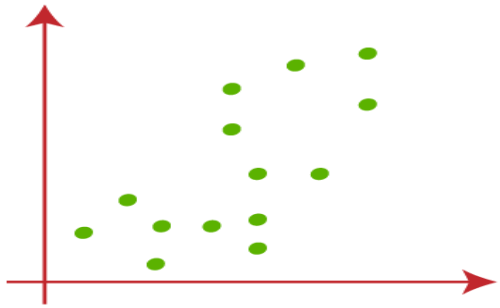
Step-5: *Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.*

Step-6: *If any reassignment occurs, then go to step-4 else go to FINISH.*

Step-7*: The model is ready.*

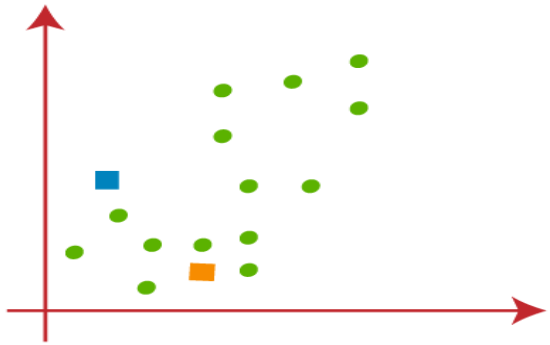
Let's understand the above steps by considering the visual plots:

Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

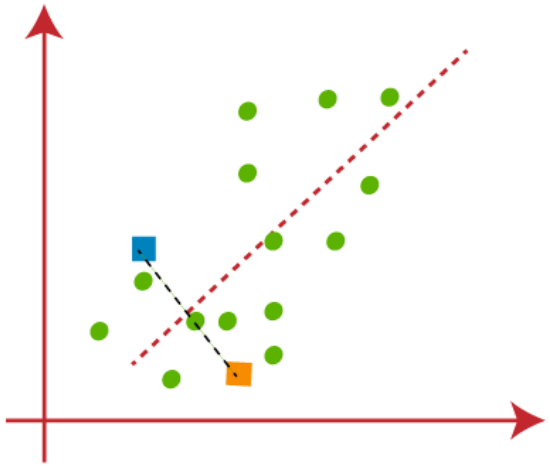


- *Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.*

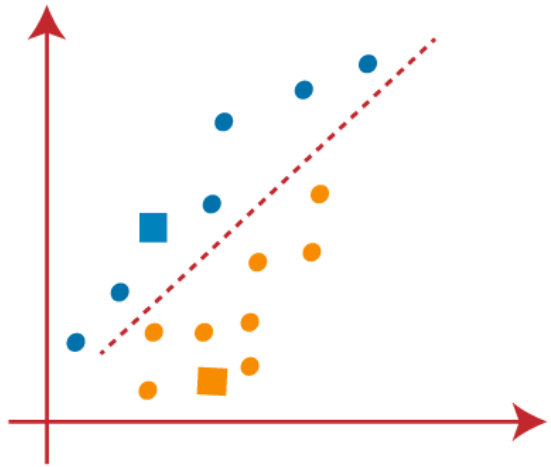
- *We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:*



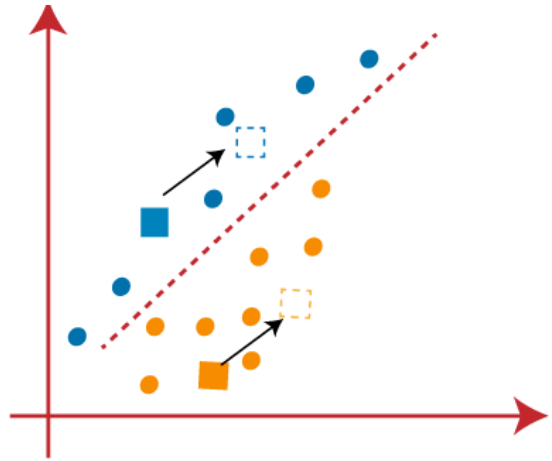
- *Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:*



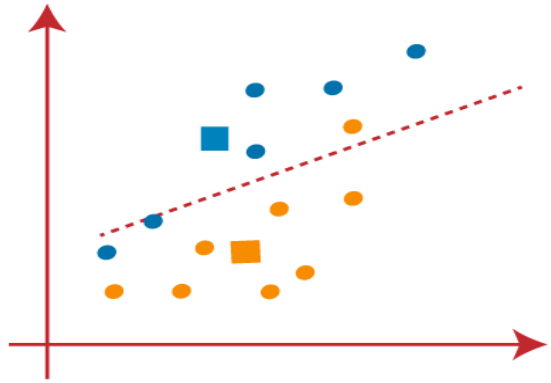
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



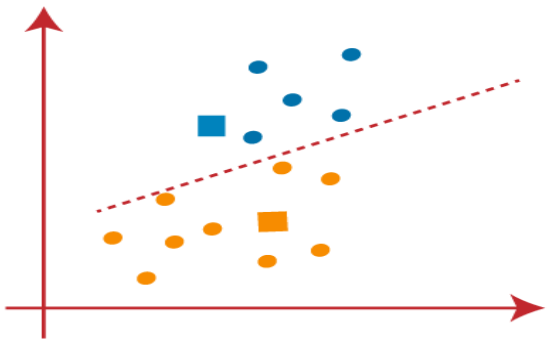
- *As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:*



- *Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:*

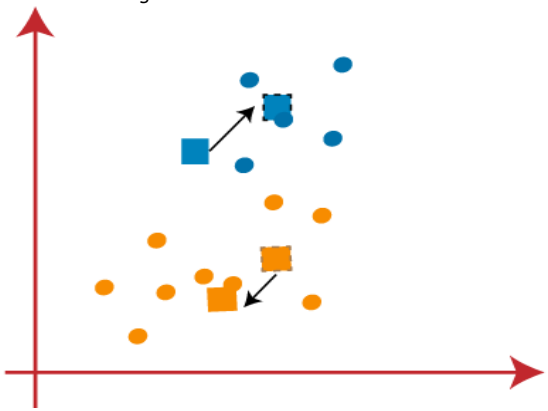


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

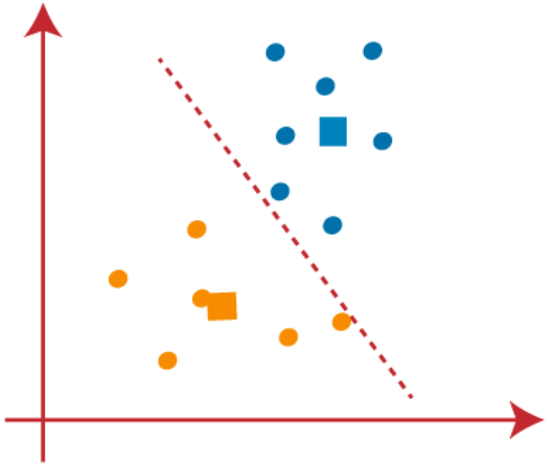


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

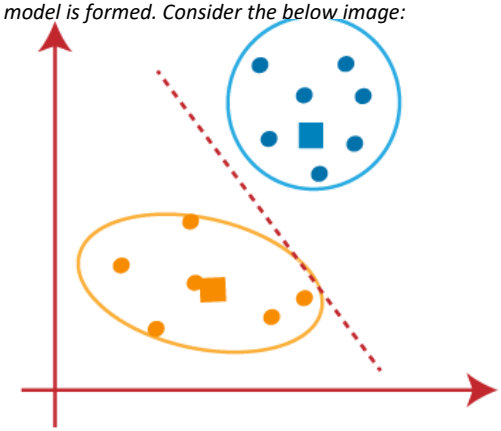
- *We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:*



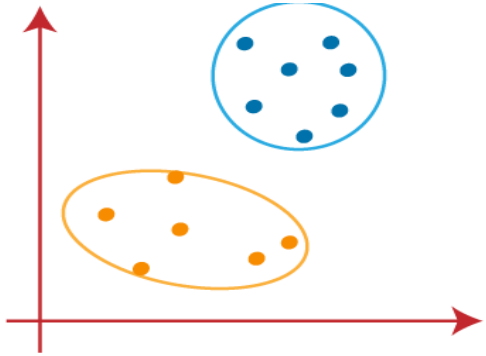
- *As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:*



- *We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:*



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



Clustering Methods

Clustering methods can be classified into the following categories –

- *Partitioning Method*
- *Hierarchical Method*
- *Density-based Method*
- *Grid-Based Method*
- *Model-Based Method*
- *Constraint-based Method*

Partitioning Method

Suppose we are given a database of 'n' objects and the partitioning method constructs 'k' partition of data. Each partition will represent a cluster and k ≤ n. It means that it will classify the data into k groups, which satisfy the following requirements –

- *Each group contains at least one object.*
- *Each object must belong to exactly one group.*

Points to remember –

- *For a given number of partitions (say k), the partitioning method will create an initial partitioning.*
- *Then it uses the iterative relocation technique to improve the partitioning by moving objects from one group to other.*

Hierarchical Methods

This method creates a hierarchical decomposition of the given set of data objects. We can classify hierarchical methods on the basis of how the hierarchical decomposition is formed. There are two approaches here –

- *Agglomerative Approach*
- *Divisive Approach*

Agglomerative Approach

This approach is also known as the bottom-up approach. In this, we start with each object forming a separate group. It keeps on merging the objects or groups that are close to one another. It keep on doing so until all of the groups are merged into one or until the termination condition holds.

Divisive Approach

This approach is also known as the top-down approach. In this, we start with all of the objects in the same cluster. In the continuous iteration, a cluster is split up into smaller clusters. It is down until each object in one cluster or the termination condition holds. This method is rigid, i.e., once a merging or splitting is done, it can never be undone.

Approaches to Improve Quality of Hierarchical Clustering

Here are the two approaches that are used to improve the quality of hierarchical clustering –

- *Perform careful analysis of object linkages at each hierarchical partitioning.*
- *Integrate hierarchical agglomeration by first using a hierarchical agglomerative algorithm to group objects into micro-clusters, and then performing macro-clustering on the micro-clusters.*

Density-based Method

This method is based on the notion of density. The basic idea is to continue growing the given cluster as long as the density in the neighborhood exceeds some threshold, i.e., for each data point within a given cluster, the radius of a given cluster has to contain at least a minimum number of points.

Grid-based Method

In this, the objects together form a grid. The object space is quantized into finite number of cells that form a grid structure.

Advantages

- *The major advantage of this method is fast processing time.*
- *It is dependent only on the number of cells in each dimension in the quantized space.*

Model-based methods

In this method, a model is hypothesized for each cluster to find the best fit of data for a given model. This method locates the clusters by clustering the density function. It reflects spatial distribution of the data points.

This method also provides a way to automatically determine the number of clusters based on standard statistics, taking outlier or noise into account. It therefore yields robust clustering methods.

Constraint-based Method

In this method, the clustering is performed by the incorporation of user or application-oriented constraints. A constraint refers to the user expectation or the properties of desired clustering results. Constraints provide us with an interactive way of communication with the clustering process. Constraints can be specified by the user or the application requirement.

K-Medoids clustering-Theoretical Explanation

K-Medoids and K-Means are two types of clustering mechanisms in Partition Clustering. First, Clustering is the process of breaking down an abstract group of data points/ objects into classes of similar objects such that all the objects in one cluster have similar traits. , a group of n objects is broken down into k number of clusters based on their similarities.

Two statisticians, Leonard Kaufman, and Peter J. Rousseeuw came up with this method. This tutorial explains what K-Medoids do, their applications, and the difference between K-Means and K-Medoids.

K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with outlier data sensitivity. Compared to other partitioning algorithms, the algorithm is simple, fast, and easy to implement.

The partitioning will be carried on such that:

1. *Each cluster must have at least one object*
2. *An object must belong to only one cluster*

Here is a small recap on K-Means clustering:

In the K-Means algorithm, given the value of k and unlabelled data:

1. *Choose k number of random points (Data point from the data set or some other points). These points are also called "**Centroids**" or "**Means**".*
2. *Assign all the data points in the data set to the closest centroid by applying any distance formula like **Euclidian distance, Manhattan distance,** etc.*
3. *Now, choose new centroids by calculating the mean of all the data points in the clusters and goto step 2*
4. *Continue step 3 until no data point changes classification between two iterations.*

*The problem with the K-Means algorithm is that the algorithm needs to handle outlier data. An outlier is a point different from the rest of the points. All the outlier data points show up in a different cluster and will attract other clusters to merge with it. Outlier data increases the mean of a cluster by up to 10 units. Hence, **K-Means clustering is highly affected by outlier data.***

K-Medoids:

***Medoid:** A Medoid is a point in the cluster from which the sum of distances to other data points is minimal.*

(or)

A Medoid is a point in the cluster from which dissimilarities with all the other points in the clusters are minimal.

Instead of centroids as reference points in K-Means algorithms, the K-Medoids algorithm takes a Medoid as a reference point.

There are three types of algorithms for K-Medoids Clustering:

1. ***PAM (Partitioning Around Clustering)***
2. ***CLARA (Clustering Large Applications)***
3. ***CLARANS (Randomized Clustering Large Applications)***

PAM is the most powerful algorithm of the three algorithms but has the disadvantage of time complexity. The following K-Medoids are performed using PAM. In the further parts, we'll see what CLARA and CLARANS are.

Algorithm:

Given the value of k and unlabelled data:

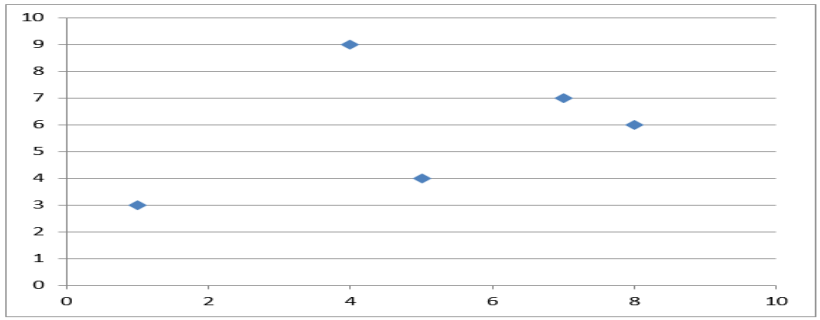
1. *Choose k number of random points from the data and assign these k points to k number of clusters. These are the initial medoids.*
2. *For all the remaining data points, calculate the distance from each medoid and assign it to the cluster with the nearest medoid.*
3. *Calculate the total cost (Sum of all the distances from all the data points to the medoids)*
4. *Select a random point as the new medoid and swap it with the previous medoid. Repeat 2 and 3 steps.*
5. *If the total cost of the new medoid is less than that of the previous medoid, make the new medoid permanent and repeat step 4.*
6. *If the total cost of the new medoid is greater than the cost of the previous medoid, undo the swap and repeat step 4.*
7. *The Repetitions have to continue until no change is encountered with new medoids to classify data points.*

Here is an example to make the theory clear:

Data set:

	x	y
0	5	4
1	7	7
2	1	3
3	8	6
4	4	9

Scatter plot:



If k is given as 2, we need to break down the data points into 2 clusters.

1. ***Initial medoids: M1(1, 3) and M2(4, 9)***
2. *Calculation of distances*

Manhattan Distance: |x1 - x2| + |y1 - y2|

	x<	y	From M1(1, 3)	From M2(4, 9)
0	5	4	5	6
1	7	7	10	5
2	1	3	-	-
3	8	6	10	7
4	4	9	-	-

Cluster 1: *0*

Cluster 2: *1, 3*

1. *Calculation of total cost:*
   (5) + (5 + 7) = 17
2. *Random medoid: (5, 4)*

M1(5, 4) and M2(4, 9):

	x	y	From M1(5, 4)	From M2(4, 9)
0	5	4	-	-
1	7	7	5	5
2	1	3	5	9
3	8	6	5	7
4	4	9	-	-

Cluster 1: *2, 3*

Cluster 2: *1*

1. *Calculation of total cost:*
   (5 + 5) + 5 = 15
   Less than the previous cost
   New medoid: (5, 4).
2. *Random medoid: (7, 7)*

M1(5, 4) and M2(7, 7)

	x	y	From M1(5, 4)	From M2(7, 7)
0	5	4	-	-
1	7	7	-	-
2	1	3	5	10
3	8	6	5	2
4	4	9	6	5

Cluster 1: *2*

Cluster 2: *3, 4*

1. *Calculation of total cost:*
   (5) + (2 + 5) = 12
   Less than the previous cost
   New medoid: (7, 7).
2. *Random medoid: (8, 6)*

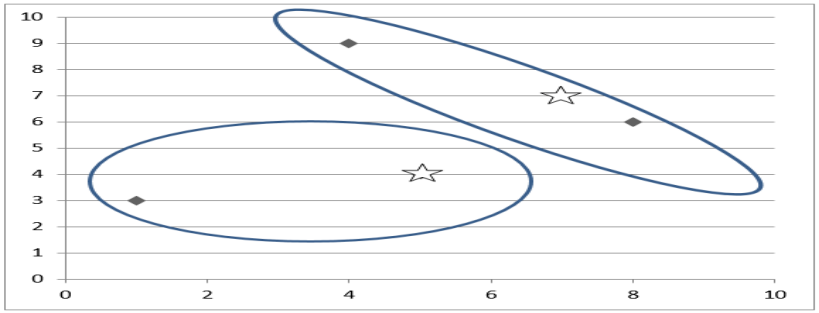
M1(7, 7) and M2(8, 6)

	x	y	From M1(7, 7)	From M2(8, 6)
0	5	4	5	5
1	7	7	-	-
2	1	3	10	10
3	8	6	-	-
4	4	9	5	7

Cluster 1: *4*

Cluster 2: *0, 2*

1. *Calculation of total cost:*
   (5) + (5 + 10) = 20
   Greater than the previous cost
   UNDO
   Hence, the final medoids: ***M1(5, 4) and M2(7, 7)***
   Cluster 1: *2*
   Cluster 2: *3, 4*
   Total cost: 12
   Clusters:



Limitation of PAM:

Time complexity: $O(k * (n - k)^2)$

Possible combinations for every node: k(n - k)*

Cost for each computation: (n - k)

Total cost: $k*(n - k)^2$

Hence, PAM is suitable and recommended to be used for small data sets.

CLARA:

It is an extension to PAM to support Medoid clustering for large data sets. This algorithm selects ***data samples from the data set, applies Pam on each sample, and outputs the best Clustering out of these samples****. This is more effective than PAM. We should ensure that the selected samples aren't biased as they affect the Clustering of the whole data.*

CLARANS:

This algorithm selects a sample of neighbors to examine instead of selecting samples from the data set. In every step, it examines the neighbors of every node. The time complexity of this algorithm is $O(n^2)$, and this is the best and most efficient Medoids algorithm of all.

Advantages of using K-Medoids:

1. *Deals with noise and outlier data effectively*
2. *Easily implementable and simple to understand*
3. *Faster compared to other partitioning algorithms*

Disadvantages:

1. *Not suitable for Clustering arbitrarily shaped groups of data points.*
2. *As the initial medoids are chosen randomly, the results might vary based on the choice in different runs.*

K-Means and K-Medoids:

K-Means	***K-Medoids***
Both methods are types of Partition Clustering.	
Unsupervised iterative algorithms	
Have to deal with unlabelled data	
Both algorithms group n objects into k clusters based on similar traits where k is pre-defined.	
Inputs: *Unlabelled data and the value of k*	
Metric of similarity: *Euclidian Distance*	***Metric of similarity:*** *Manhattan Distance*
Clustering is done based on distance from ***centroids.***	*Clustering is done based on distance from* ***medoids.***
A centroid can be a data point or some other point in the cluster	*A medoid is always a data point in the cluster.*
Can't cope with outlier data	*Can manage outlier data too*
Sometimes, outlier sensitivity can turn out to be useful	*Tendency to ignore meaningful clusters in outlier data*

Hierarchical clustering in data mining

Hierarchical clustering refers to an unsupervised learning procedure that determines successive clusters based on previously defined clusters. It works via grouping data into a tree of clusters. Hierarchical clustering stats by treating each data points as an individual cluster. The endpoint refers to a different set of clusters, where each cluster is different from the other cluster, and the objects within each cluster are the same as one another.

There are two types of hierarchical clustering

- *Agglomerative Hierarchical Clustering*
- *Divisive Clustering*

Agglomerative hierarchical clustering (AGNES)

Agglomerative clustering is one of the most common types of hierarchical clustering used to group similar objects in clusters. Agglomerative clustering is also known as AGNES (Agglomerative Nesting). In agglomerative clustering, each data point act as an individual cluster and at each step, data objects are grouped in a bottom-up method. Initially, each data object is in its cluster. At each iteration, the clusters are combined with different clusters until one cluster is formed.

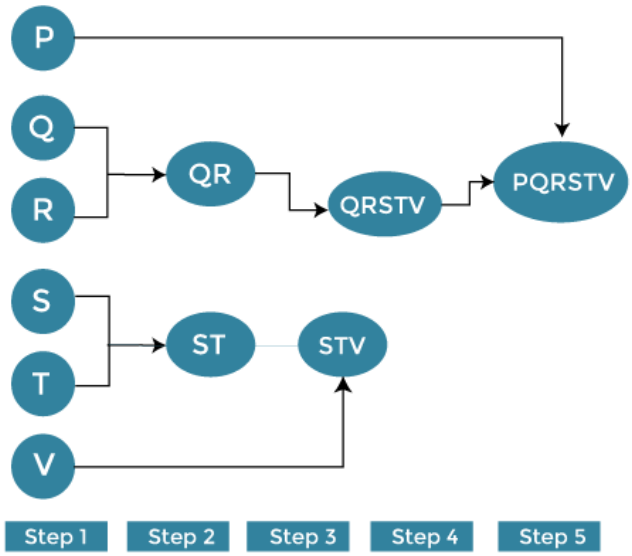
Agglomerative hierarchical clustering algorithm

1. *Determine the similarity between individuals and all other clusters. (Find proximity matrix).*
2. *Consider each data point as an individual cluster.*
3. *Combine similar clusters.*
4. *Recalculate the proximity matrix for each cluster.*
5. *Repeat step 3 and step 4 until you get a single cluster.*

Let's understand this concept with the help of graphical representation using a dendrogram.

With the help of given demonstration, we can understand that how the actual algorithm work. Here no calculation has been done below all the proximity among the clusters are assumed.

Let's suppose we have six different data points P, Q, R, S, T, V.



Step 1:

Consider each alphabet (P, Q, R, S, T, V) as an individual cluster and find the distance between the individual cluster from all other clusters.

Step 2:

Now, merge the comparable clusters in a single cluster. Let's say cluster Q and Cluster R are similar to each other so that we can merge them in the second step. Finally, we get the clusters [(P), (QR), (ST), (V)]

Step 3:

Here, we recalculate the proximity as per the algorithm and combine the two closest clusters [(ST), (V)] together to form new clusters as [(P), (QR), (STV)]

Step 4:

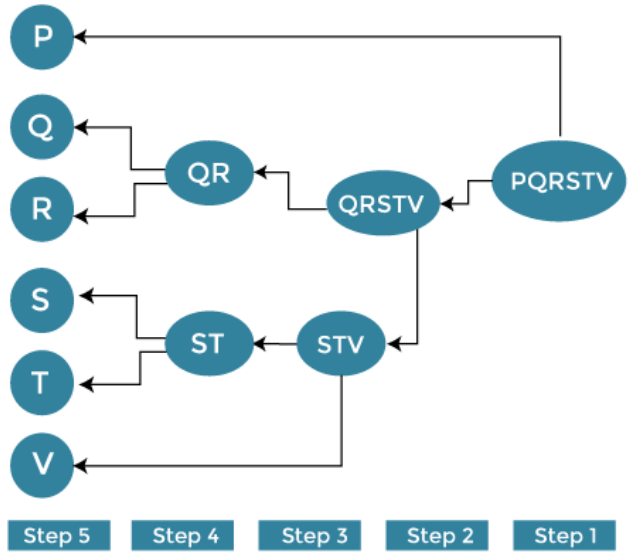
Repeat the same process. The clusters STV and PQ are comparable and combined together to form a new cluster. Now we have [(P), (QQRSTV)].

Step 5:

Finally, the remaining two clusters are merged together to form a single cluster [(PQRSTV)]

Divisive Hierarchical Clustering(DIANA)

Divisive hierarchical clustering is exactly the opposite of Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, all the data points are considered an individual cluster, and in every iteration, the data points that are not similar are separated from the cluster. The separated data points are treated as an individual cluster. Finally, we are left with N clusters.



Advantages of Hierarchical clustering

- *It is simple to implement and gives the best output in some cases.*
- *It is easy and results in a hierarchy, a structure that contains more information.*
- *It does not need us to pre-specify the number of clusters.*

Disadvantages of hierarchical clustering

- *It breaks the large clusters.*
- *It is Difficult to handle different sized clusters and convex shapes.*
- *It is sensitive to noise and outliers.*
- *The algorithm can never be changed or deleted once it was done previously.*

Density-based clustering in data minin

Density-based clustering refers to a method that is based on local cluster criterion, such as density connected points. In this tutorial, we will discuss density-based clustering with examples.

What is Density-based clustering?

Density-Based Clustering refers to one of the most popular unsupervised learning methodologies used in model building and machine learning algorithms. The data points in the region separated by two clusters of low point density are considered as noise. The surroundings with a radius ε of a given object are known as the ε neighborhood of the object. If the ε neighborhood of the object comprises at least a minimum number, MinPts of objects, then it is called a core object.

Density-Based Clustering - Background

There are two different parameters to calculate the density-based clustering

E_{PS}: It is considered as the maximum radius of the neighborhood.

MinPts: MinPts refers to the minimum number of points in an Eps neighborhood of that point.

NEps (i) : { k belongs to D and dist (i,k) < = Eps}

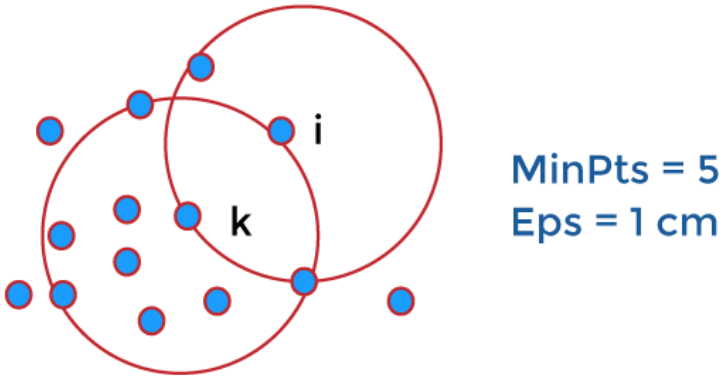
Directly density reachable:

A point i is considered as the directly density reachable from a point k with respect to Eps, MinPts if

i belongs to NEps(k)

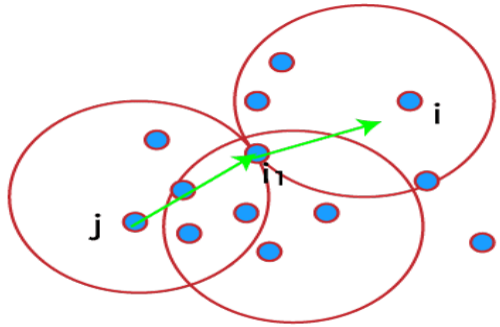
Core point condition:

NEps (k) >= MinPts



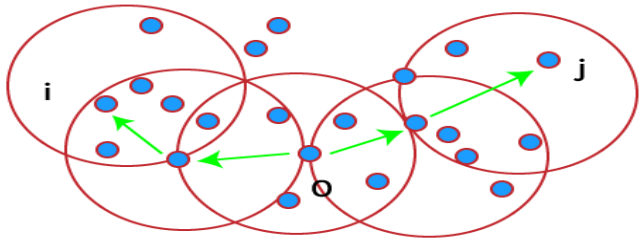
Density reachable:

A point denoted by i is a density reachable from a point j with respect to Eps, MinPts if there is a sequence chain of a point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly density reachable from i_i.



Density connected:

A point i refers to density connected to a point j with respect to Eps, MinPts if there is a point o such that both i and j are considered as density reachable from o with respect to Eps and MinPts.



Working of Density-Based Clustering

Suppose a set of objects is denoted by D', we can say that an object I is directly density reachable form the object j only if it is located within the ε neighborhood of j, and j is a core object.

An object i is density reachable form the object j with respect to ε and MinPts in a given set of objects, D' only if there is a sequence of object chains point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly density reachable from i_i with respect to ε and MinPts.

An object i is density connected object j with respect to ε and MinPts in a given set of objects, D' only if there is an object o belongs to D such that both point i and j are density reachable from o with respect to ε and MinPts.

Major Features of Density-Based Clustering

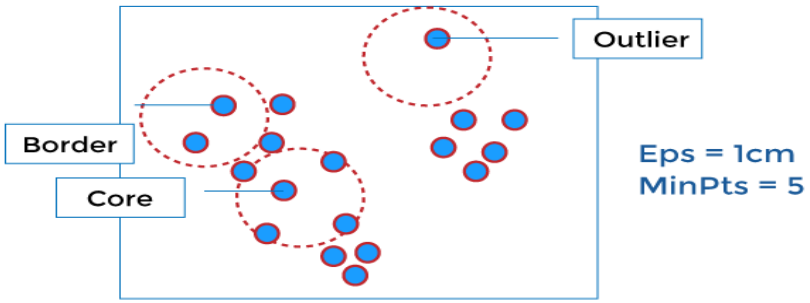
The primary features of Density-based clustering are given below.

- *It is a scan method.*
- *It requires density parameters as a termination condition.*
- *It is used to manage noise in data clusters.*
- *Density-based clustering is used to identify clusters of arbitrary size.*

Density-Based Clustering Methods

DBSCAN

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It depends on a density-based notion of cluster. It also identifies clusters of arbitrary size in the spatial database with outliers.



OPTICS

OPTICS stands for Ordering Points To Identify the Clustering Structure. It gives a significant order of database with respect to its density-based clustering structure. The order of the cluster comprises information equivalent to the density-based clustering related to a long range of parameter settings. OPTICS methods are beneficial for both automatic and interactive cluster analysis, including determining an intrinsic clustering structure.

DENCLUE

Density-based clustering by Hinnebirg and Kiem. It enables a compact mathematical description of arbitrarily shaped clusters in high dimension state of data, and it is good for data sets with a huge amount of noise.

Classification and Predication in Data Mining

There are two forms of data analysis that can be used to extract models describing important classes or predict future data trends. These two forms are as follows:

1. *Classification*
2. *Prediction*

What is Prediction?

Another process of data analysis is prediction. It is used to find a numerical output. Same as in classification, the training dataset contains the inputs and corresponding numerical output values. The algorithm derives the model or a predictor according to the training dataset. The model should find a numerical output when the new data is given. Unlike in classification, this method does not have a class label. The model predicts a continuous-valued function or ordered value.

Regression is generally used for prediction. Predicting the value of a house depending on the facts such as the number of rooms, the total area, etc., is an example for prediction.

For example, suppose the marketing manager needs to predict how much a particular customer will spend at his company during a sale. We are bothered to forecast a numerical value in this case. Therefore, an example of numeric prediction is the data processing activity. In this case, a model or a predictor will be developed that forecasts a continuous or ordered value function.

Classification and Prediction Issues

1. ***Data Cleaning:*** *Data cleaning involves removing the noise and treatment of missing values. The noise is removed by applying smoothing techniques, and the problem of missing values is solved by replacing a missing value with the most commonly occurring value for that attribute.*
2. ***Relevance Analysis:*** *The database may also have irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.*
3. ***Data Transformation and reduction:*** *The data can be transformed by any of the following methods.*
    - ***Normalization:*** *The data is transformed using normalization. Normalization involves scaling all values for a given attribute to make them fall within a small specified range. Normalization is used when the neural networks or the methods involving measurements are used in the learning step.*
    - ***Generalization:*** *The data can also be transformed by generalizing it to the higher concept. For this purpose, we can use the concept hierarchies.*

NOTE: Data can also be reduced by some other methods such as wavelet transformation, binning, histogram analysis, and clustering.

Comparison of Classification and Prediction Methods

- ***Accuracy:*** *The accuracy of the classifier can be referred to as the ability of the classifier to predict the class label correctly, and the accuracy of the predictor can be referred to as how well a given predictor can estimate the unknown value.*
- ***Speed:*** *The speed of the method depends on the computational cost of generating and using the classifier or predictor.*
- ***Robustness:*** *Robustness is the ability to make correct predictions or classifications. In the context of data mining, robustness is the ability of the classifier or predictor to make correct predictions from incoming unknown data.*
- ***Scalability:*** *Scalability refers to an increase or decrease in the performance of the classifier or predictor based on the given data.*
- ***Interpretability:*** *Interpretability is how readily we can understand the reasoning behind predictions or classification made by the predictor or classifier.*

Difference between Classification and Prediction

Classification	***Prediction***
Classification is the process of identifying which category a new observation belongs to based on a training data set containing observations whose category membership is known.	*Predication is the process of identifying the missing or unavailable numerical data for a new observation.*
In classification, the accuracy depends on finding the class label correctly.	*In prediction, the accuracy depends on how well a given predictor can guess the value of a predicated attribute for new data.*
In classification, the model can be known as the classifier.	*In prediction, the model can be known as the predictor.*
A model or the classifier is constructed to find the categorical labels.	*A model or a predictor will be constructed that predicts a continuous-valued function or ordered value.*
For example*, the grouping of patients based on their medical records can be considered a classification.*	***For example****, We can think of prediction as predicting the correct treatment for a particular disease for a person.*

Classification is a task in data mining that involves assigning a class label to each instance in a dataset based on its features. The goal of classification is to build a model that accurately predicts the class labels of new instances based on their features.

There are two main types of classification: binary classification and multi-class classification. Binary classification involves classifying instances into two classes, such as "spam" or "not spam", while multi-class classification involves classifying instances into more than two classes.

The process of building a classification model typically involves the following steps:

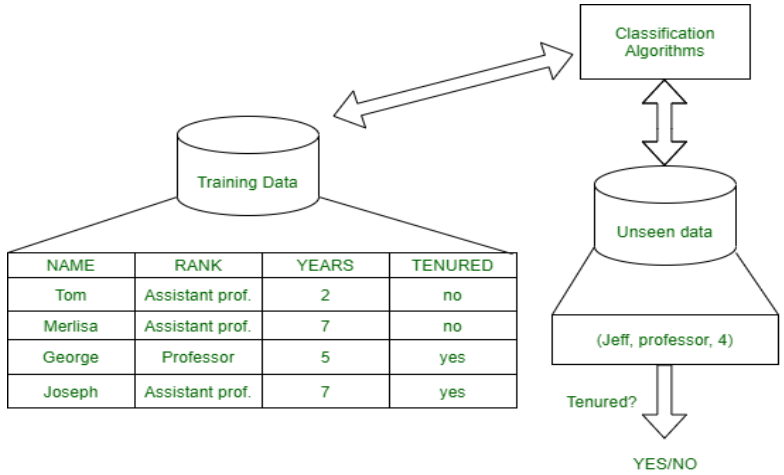
1. *Data preparation: This step involves cleaning and pre-processing the data, such as removing missing values and transforming the data into a format that can be used by the classification algorithm.*
2. *Model selection: This step involves choosing an appropriate classification algorithm based on the characteristics of the data and the desired outcome. Common algorithms include decision trees, k-nearest neighbors, and support vector machines.*
3. *Model training: This step involves using the training data to train the classification algorithm and build the model. The model is trained by adjusting its parameters to minimize the difference between the predicted class labels and the actual class labels.*
4. *Model evaluation: This step involves evaluating the performance of the classification model on a test dataset that is separate from the training data. This can be done by calculating metrics such as accuracy, precision, recall, and F1-score.*
5. *Model deployment: This step involves deploying the classification model in a production environment, where it can be used to make predictions on new instances.*

Classification is a widely used technique in data mining and is applied in a variety of domains, such as email filtering, sentiment analysis, and medical diagnosis.

***Classification**: It is a data analysis task, i.e. the process of finding a model that describes and distinguishes data classes and concepts. Classification is the problem of identifying to which of a set of categories (subpopulations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.*

***Example**: Before starting any project, we need to check its feasibility. In this case, a classifier is required to predict class labels such as 'Safe' and 'Risky' for adopting the Project and to further approve it. It is a two-step process such as:*

1. ***Learning Step (Training Phase)**: Construction of Classification Model*
   Different Algorithms are used to build a classifier by making the model learn using the training set available. The model has to be trained for the prediction of accurate results.
2. ***Classification Step**: Model used to predict class labels and testing the constructed model on test data and hence estimate the accuracy of the classification rules.*



NAME	RANK	YEARS	TENURED
Tom	Assistant prof.	2	no
Merlisa	Assistant prof.	7	no
George	Professor	5	yes
Joseph	Assistant prof.	7	yes

Test data are used to estimate the accuracy of the classification rule

Training and Testing:

Suppose there is a person who is sitting under a fan and the fan starts falling on him, he should get aside in order not to get hurt. So, this is his training part to move away. While Testing if the person sees any heavy object coming towards him or falling on him and moves aside then the system is tested positively and if the person does not move aside then the system is negatively tested.
The same is the case with the data, it should be trained in order to get the accurate and best results.

There are certain data types associated with data mining that actually tells us the format of the file (whether it is in text format or in numerical format).

Attributes – Represents different features of an object. Different types of attributes are:

1. ***Binary**: Possesses only two values i.e. True or False*
   Example: Suppose there is a survey evaluating some products. We need to check whether it's useful or not. So, the Customer has to answer it in Yes or No.
   Product usefulness: Yes / No
   - ***Symmetric**: Both values are equally important in all aspects*
   - ***Asymmetric**: When both the values may not be important.*
2. ***Nominal**: When more than two outcomes are possible. It is in Alphabet form rather than being in Integer form.*
   ***Example**: One needs to choose some material but of different colors. So, the color might be Yellow, Green, Black, Red.*
   Different Colors: Red, Green, Black, Yellow
   - ***Ordinal**: Values that must have some meaningful order.*
     Example: Suppose there are grade sheets of few students which might contain different grades as per their performance such as A, B, C, D
     Grades: A, B, C, D
   - ***Continuous**: May have an infinite number of values, it is in float type*
     Example: Measuring the weight of few Students in a sequence or orderly manner i.e. 50, 51, 52, 53
     Weight: 50, 51, 52, 53
   - ***Discrete**: Finite number of values.*
     Example: Marks of a Student in a few subjects: 65, 70, 75, 80, 90
     Marks: 65, 70, 75, 80, 90

Issues Regarding Classification and Prediction

This section describes issues regarding preprocessing the data of classification and prediction. Criteria for the comparison and evaluation of classification methods are also described.

8.3.1 Preparing the Data for Classification and Prediction

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

Data Cleaning: *This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques) and the treatment of missing values (e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics.) Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning.*

Relevance Analysis: *Many of the attributes in the data may be irrelevant to the classification or prediction task. For example, data recording the day of the week on which a bank loan application was filed is unlikely to be relevant to the success of the application. Furthermore, other attributes may be redundant. Hence, relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process. In machine learning, this step is known as feature selection. Including such attributes may otherwise slow down, and possibly mislead, the learning step.*

Ideally, the time spent on relevance analysis, when added to the time spent on learning from the resulting "reduced" feature subset should be less than the time that would have been spent on learning from the original set of features. Hence, such analysis can help improve classification efficiency and scalability.

Data Transformation: *The data can be generalized to higher – level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous – valued attributes. For example, numeric values for the attribute income may be generalized to discrete ranges such as low, medium, and high. Similarly, nominal – valued attributes like street, can be generalized to higher – level concepts, like city. Since generalization compresses the original training data, fewer input / output operations may be involved during learning.*

The data may also be normalized, particularly when neural networks or methods involving distance measurements are used in the learning step. ***Normalization*** *involves scaling all values for a given attribute so that they fall within a small specified range, such as – 1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurements, for example, this would prevent attributes with initially large ranges (like, say, income) from outweighing attributes with initially smaller ranges (such as binary attributes).*

8.3.2 Comparing Classification Methods

Classification and prediction methods can be compared and evaluated according to the following criteria:

Predictive Accuracy: *This refers to the ability of the model to correctly predict the class label of new or previously unseen data.*

Speed: *This refers to the computation costs involved in generating and using the model.*

Robustness: *This is the ability of the model to make correct predictions given noisy data or data with missing values.*

Scalability: *This refers to the ability to construct the model efficiently given large amount of data.*

Interpretability: *This refers to the level of understanding and insight that is provided by the model.*

Types Of Data Used In Cluster Analysis Are:

- ***Interval-Scaled variables***
- ***Binary variables***
- ***Nominal, Ordinal, and Ratio variables***
- ***Variables of mixed types***

Types Of Data Structures

First of all, let us know what types of data structures are widely used in cluster analysis.

We shall know the types of data that often occur in cluster analysis and how to preprocess them for such analysis.

Suppose that a data set to be clustered contains n objects, which may represent persons, houses, documents, countries, and so on.

Main memory-based clustering algorithms typically operate on either of the following two data structures.

Types of data structures in cluster analysis are

- ***Data Matrix*** *(or object by variable structure)*
- ***Dissimilarity Matrix*** *(or object by object structure)*

Data Matrix

This represents n objects, such as persons, with p variables (also called measurements or attributes), such as age, height, weight, gender, race and so on. The structure is in the form of a relational table, or n-by-p matrix (n objects x p variables)

The Data Matrix is often called a two-mode matrix since the rows and columns of this represent the different entities.

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Dissimilarity Matrix

This stores a collection of proximities that are available for all pairs of n objects. It is often represented by a n – by – n table, where d(i,j) is the measured difference or dissimilarity between objects i and j. In general, d(i,j) is a non-negative number that is close to 0 when objects i and j are higher similar or "near" each other and becomes larger the more they differ. Since d(i,j) = d(j,i) and d(i,i) =0, we have the matrix in figure. This is also called as one mode matrix since the rows and columns of this represent the same entity.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Types Of Data In Cluster Analysis Are:

Interval-Scaled Variables

- *Interval-scaled variables are continuous measurements of a roughly linear scale.*

  Typical examples include weight and height, latitude and longitude coordinates (e.g., when clustering houses), and weather temperature.

  The measurement unit used can affect the clustering analysis. For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to a very different clustering structure.
- *In general, expressing a variable in smaller units will lead to a larger range for that variable, and thus a larger effect on the resulting clustering structure.*

  To help avoid dependence on the choice of measurement units, the data should be standardized. Standardizing measurements attempts to give all variables an equal weight.

  This is especially useful when given no prior knowledge of the data. However, in some applications, users may intentionally want to give more weight to a certain set of variables than to others.

  For example, when clustering basketball player candidates, we may prefer to give more weight to the variable height.

Binary Variables: *A binary variable is a variable that can take only 2 values.For example, generally, gender variables can take 2 variables male and female.*

Contingency Table For Binary Data

Let us consider binary values 0 and 1

	1	0	*sum*
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

Let p=a+b+c+d

Simple matching coefficient *(invariant, if the binary variable is symmetric):*

$$d(i, j) = \frac{b + c}{a + b + c + d}$$

Jaccard coefficient *(noninvariant if the binary variable is asymmetric):*

$$d(i, j) = \frac{b + c}{a + b + c}$$

Nominal or Categorical Variables*: A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green.*

Method 1: Simple matching

The dissimilarity between two objects i and j can be computed based on the simple matching.

m*: Let m be no of matches (i.e., the number of variables for which i and j are in the same state).*

p*: Let p be total no of variables.*

$$d(i, j) = \frac{p - m}{p}$$

Method 2: use a large number of binary variables: *Creating a new binary variable for each of the M nominal states.*

Ordinal Variables: *An ordinal variable can be discrete or continuous. In this order is important, e.g., rank.It can be treated like interval-scaled By replacing xif by their rank,*

$$r_{if} \in \{1, \ldots, M_f\}$$

By mapping the range of each variable onto [0, 1] by replacing the i-th object in the f-th variable by,

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Then compute the dissimilarity using methods for interval-scaled variables.

Ratio-Scaled Intervals: Ratio-scaled variable*: It is a positive measurement on a nonlinear scale, approximately at an exponential scale, such as Ae^Bt or A^e-Bt.*

Methods*:*

- *First, treat them like interval-scaled variables — not a good choice! (why?)*
- *Then apply logarithmic transformation i.e.y = log(x)*
- *Finally, treat them as continuous ordinal data treat their rank as interval-scaled.*

Variables Of Mixed Type

A database may contain all the six types of variables
symmetric binary, asymmetric binary, nominal, ordinal, interval, and ratio.

And those combinedly called as mixed-type variables.