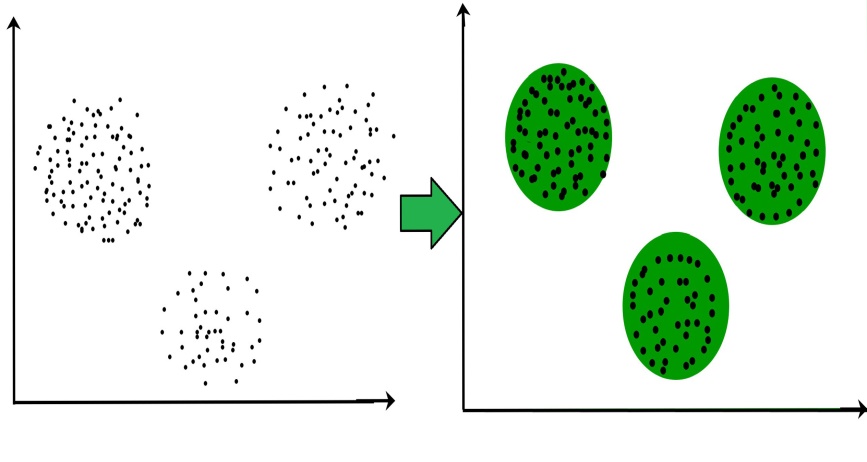
**1) What is clustering? Explain in detail.**

The task of grouping data points based on their similarity with each other is called Clustering or Cluster Analysis. This method is defined under the branch of [Unsupervised Learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/), which aims at gaining insights from unlabelled data points, that is, unlike [supervised learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) we don’t have a target variable.

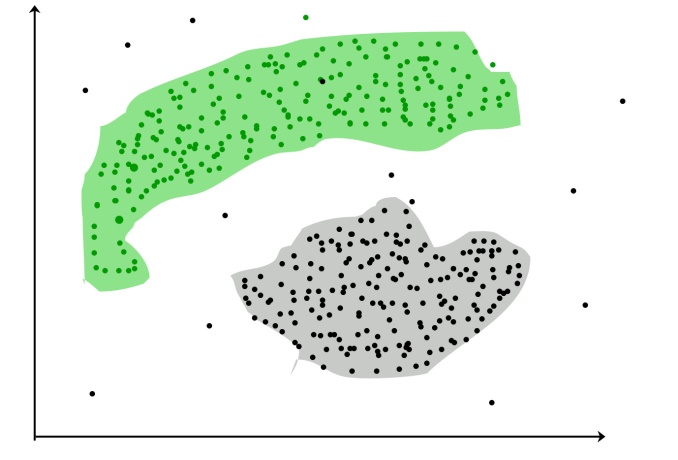
Clustering aims at forming groups of homogeneous data points from a heterogeneous dataset. It evaluates the similarity based on a metric like Euclidean distance, Cosine similarity, Manhattan distance, etc. and then group the points with highest similarity score together.

For Example, In the graph given below, we can clearly see that there are 3 circular clusters forming on the basis of distance.



Now it is not necessary that the clusters formed must be circular in shape. The shape of clusters can be arbitrary. There are many algortihms that work well with detecting arbitrary shaped clusters.

For example, In the below given graph we can see that the clusters formed are not circular in shape.



**Types of Clustering**

Broadly speaking, there are 2 types of clustering that can be performed to group similar data points:

* **Hard Clustering:**In this type of clustering, each data point belongs to a cluster completely or not. For example, Let’s say there are 4 data point and we have to cluster them into 2 clusters. So each data point will either belong to cluster 1 or cluster 2.

| **Data Points** | **Clusters** |
| --- | --- |
| A | C1 |
| B | C2 |
| C | C2 |
| D | C1 |

* **Soft Clustering:**In this type of clustering, instead of assigning each data point into a separate cluster, a probability or likelihood of that point being that cluster is evaluated. For example, Let’s say there are 4 data point and we have to cluster them into 2 clusters. So we will be evaluating a probability of a data point belonging to both clusters. This probability is calculated for all data points.

| Data Points | Probability of C1 | Probability of C2 |
| --- | --- | --- |
| A | 0.91 | 0.09 |
| B | 0.3 | 0.7 |
| C | 0.17 | 0.83 |
| D | 1 | 0 |

**Uses of Clustering**

Now before we begin with types of clustering algorithms, we will go through the use cases of Clustering algorithms. Clustering algorithms are majorly used for:

* [Market Segmentation](https://www.geeksforgeeks.org/customer-segmentation-using-unsupervised-machine-learning-in-python/) – Businesses use clustering to group their customers and use targeted advertisements to attract more audience.
* [Market Basket Analysis](https://www.geeksforgeeks.org/market-basket-analysis-in-data-mining/) – Shop owners analyze their sales and figure out which items are majorly bought together by the customers. For example, In USA, according to a study diapers and beers were usually bought together by fathers.
* [Social Network Analysis](https://www.geeksforgeeks.org/social-network-analysis-using-r-programming/) – Social media sites use your data to understand your browsing behaviour and provide you with targeted friend recommendations or content recommendations.
* Medical Imaging – Doctors use Clustering to find out diseased areas in diagnostic images like X-rays.
* [Anomaly Detection](https://www.geeksforgeeks.org/machine-learning-for-anomaly-detection/) – To find outliers in a stream of real-time dataset or forecasting fraudulent transactions we can use clustering to identify them.
* Simplify working with large datasets – Each cluster is given a cluster ID after clustering is complete. Now, you may reduce a feature set’s whole feature set into its cluster ID. Clustering is effective when it can represent a complicated case with a straightforward cluster ID. Using the same principle, clustering data can make complex datasets simpler.

There are many more use cases for clustering but there are some of the major and common use cases of clustering. Moving forward we will be discussing Clustering Algorithms that will help you perform the above tasks.

**2)Explain K Means clustering.**

[Unsupervised Machine Learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/)is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine’s job in this case is to organize unsorted data according to parallels, patterns, and variations.

K means clustering, assigns data points to one of the K clusters depending on their distance from the center of the clusters. It starts by randomly assigning the clusters centroid in the space. Then each data point assign to one of the cluster based on its distance from centroid of the cluster. After assigning each point to one of the cluster, new cluster centroids are assigned. This process runs iteratively until it finds good cluster. In the analysis we assume that number of cluster is given in advanced and we have to put points in one of the group.

In some cases, K is not clearly defined, and we have to think about the optimal number of K. K Means clustering performs best data is well separated. When data points overlapped this clustering is not suitable. K Means is faster as compare to other clustering technique. It provides strong coupling between the data points. K Means cluster do not provide clear information regarding the quality of clusters. Different initial assignment of cluster centroid may lead to different clusters. Also, K Means algorithm is sensitive to noise. It maymhave stuck in local minima.

## What is the objective of k-means clustering?

The goal of [clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/) is to divide the population or[set](https://www.geeksforgeeks.org/set-in-cpp-stl/) of data points into a number of groups so that the data points within each group are more[comparable](https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/) to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

## How k-means clustering works?

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm, an unsupervised learning algorithm. ‘K’ in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

(It will help if you think of items as points in an n-dimensional space). The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean, and we update the mean’s coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The “points” mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x, the items have values in [0,3], we will initialize the means with values for x at [0,3]).

The above algorithm in pseudocode is as follows:

Initialize k means with random values  
--> For a given number of iterations:  
   
 --> Iterate through items:  
   
 --> Find the mean closest to the item by calculating   
 the euclidean distance of the item with each of the means  
   
 --> Assign item to mean  
   
 --> Update mean by shifting it to the average of the items in that cluster

## Implementation of K-Means Clustering in Python

### Example 1

#### Import the necessary Libraries

We are importing[Numpy](https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/) for statistical computations,[Matplotlib](https://www.geeksforgeeks.org/matplotlib-tutorial/) to plot the[graph,](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/) and make\_blobs from sklearn.datasets.

3) Explain Elbow Method in K Means clustering

## Introduction To Elbow Method

A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. Since we do not have any predefined number of clusters in unsupervised learning. We tend to use some method that can help us decide the best number of clusters.  In the case of K-Means clustering, we use Elbow Method for defining the best number of clustering

## What Is the Elbow Method in K-Means Clustering

As we know in the k-means clustering algorithm we randomly initialize k clusters and we iteratively adjust these k clusters till these k-centroids riches in an equilibrium state. However, the main thing we do before initializing these clusters is that determine how many clusters we have to use.

For determining  K(numbers of clusters) we use Elbow method. **Elbow Method**is a technique that we use to determine the number of centroids(k) to use in a k-means clustering algorithm.  In this method to determine the k-value we continuously iterate for k=1 to k=n (Here n is the [hyperparameter](https://www.geeksforgeeks.org/hyperparameter-tuning/) that we choose as per our requirement). For every value of k, we calculate the within-cluster sum of squares (WCSS) value.

WCSS - It is defined as the sum of square distances between the centroids and

each points.

Now For determining the best number of clusters(k) we plot a graph of k versus their WCSS value. Surprisingly the graph looks like an elbow (*which we will see later*). Also, When k=1 the WCSS has the highest value but with increasing k value WCSS value starts to decrease. We choose that value of k from where the graph starts to look like a straight line.

**4) . Explain Hierarchical clustering.**

# Hierarchical Clustering in Machine Learning

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

### Why hierarchical clustering?

As we already have other [clustering](https://www.javatpoint.com/clustering-in-machine-learning) algorithms such as [**K-Means Clustering**](https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning), then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

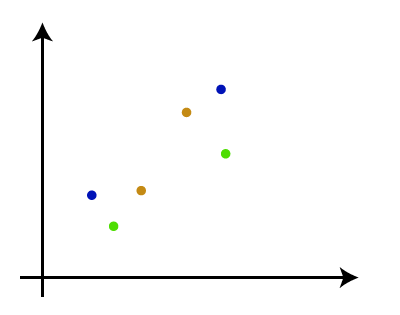
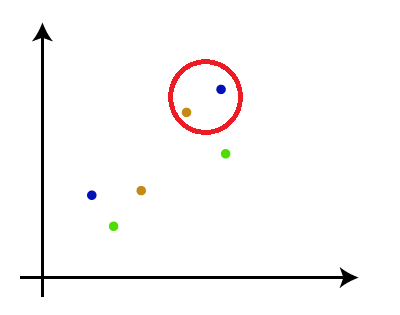
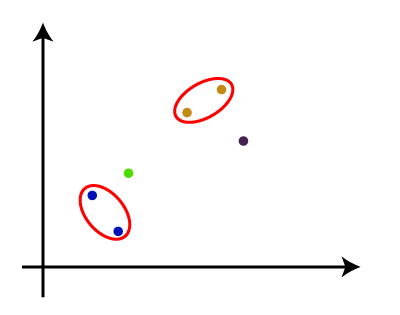
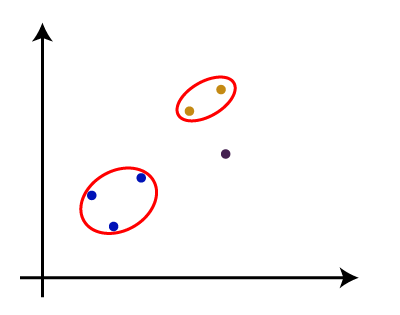
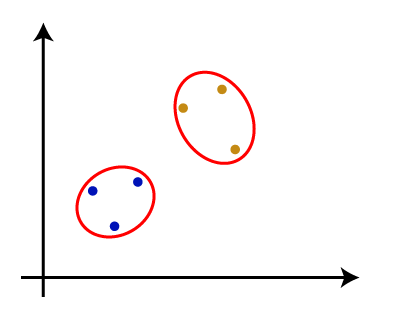
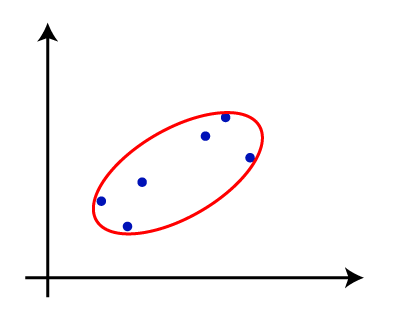
## Agglomerative Hierarchical clustering

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

## How the Agglomerative Hierarchical clustering Work?

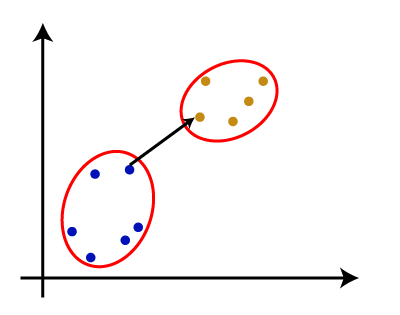
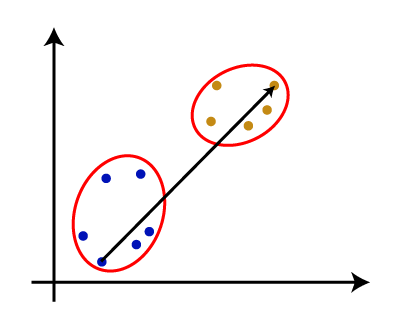
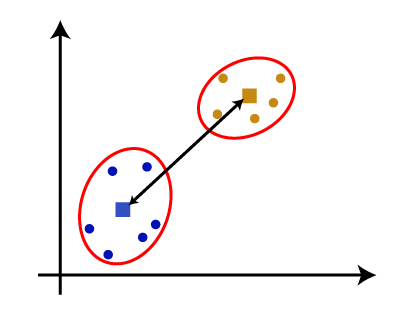
The working of the AHC algorithm can be explained using the below steps:

* **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.  
  
* **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.  
  
* **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.  
  
* **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:  
    
    
  
* **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

#### Note: To better understand hierarchical clustering, it is advised to have a look on k-means clustering

### Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

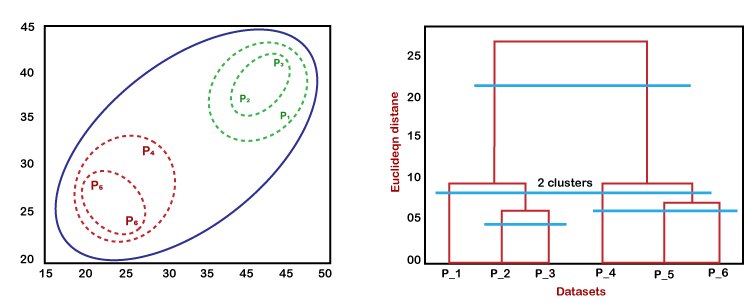
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:  
   
2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.  
   
3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:  
   

From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

### Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

* As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
* In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
* Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
* At last, the final dendrogram is created that combines all the data points together.

We can cut the dendrogram tree structure at any level as per our requirement.

**5) What is Agglomerative Hierarchical clustering?**

### ****Hierarchical Agglomerative Clustering****

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

**Algorithm :**

given a dataset (d1, d2, d3, ....dN) of size N

# compute the distance matrix

for i=1 to N:

# as the distance matrix is symmetric about

# the primary diagonal so we compute only lower

# part of the primary diagonal

for j=1 to i:

dis\_mat[i][j] = distance[di, dj]

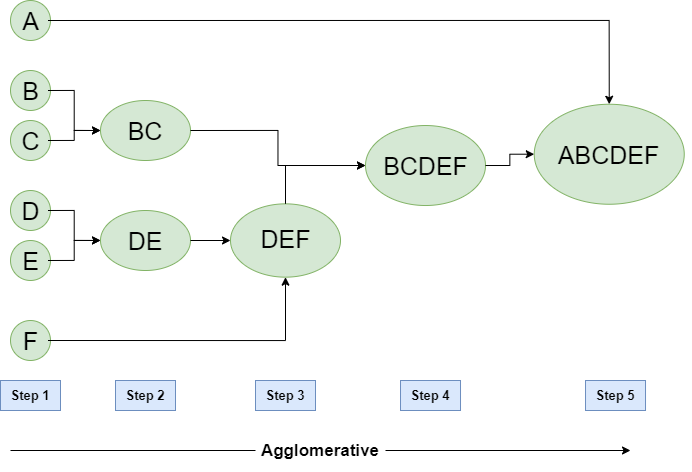
each data point is a singleton cluster

**repeat**

merge the two cluster having minimum distance

update the distance matrix

**until** only a single cluster remains



*Hierarchical Agglomerative Clustering*

**Steps**:

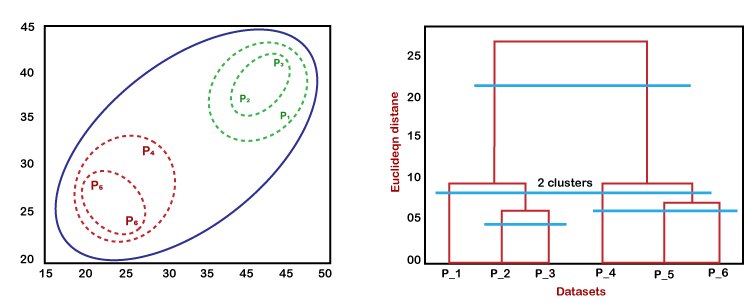
* Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.
* In the second step, comparable clusters are merged together to form a single cluster. Let’s say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly to cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]
* We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]
* Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We’re now left with clusters [(A), (BCDEF)].
* At last, the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

6) Explain working of dendrogram in Hierarchical clustering

### Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

* As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
* In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
* Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
* At last, the final dendrogram is created that combines all the data points together.

We can cut the dendrogram tree structure at any level as per our requirement.

**7) Explain Association Rule mining.**

# Association Rule Learning

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

The association rule learning is one of the very important concepts of [machine learning](https://www.javatpoint.com/machine-learning), and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together.

For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby. Consider the below diagram:



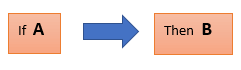
Association rule learning can be divided into three types of algorithms:

1. **Apriori**
2. **Eclat**
3. **F-P Growth Algorithm**

We will understand these algorithms in later chapters.

## How does Association Rule Learning work?

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



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

* **Support**
* **Confidence**
* **Lift**

**Let's understand each of them:**

### Support

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

Association Rule Learning

### Confidence

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

Association Rule Learning

### Lift

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

Association Rule Learning

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

* If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
* **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
* **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

## Types of Association Rule Lerning

Association rule learning can be divided into three algorithms:

### Apriori Algorithm

This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.

It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

### Eclat Algorithm

Eclat algorithm stands for **Equivalence Class Transformation**. This algorithm uses a depth-first search technique to find frequent itemsets in a transaction database. It performs faster execution than Apriori Algorithm.

### F-P Growth Algorithm

The F-P growth algorithm stands for **Frequent Pattern**, and it is the improved version of the Apriori Algorithm. It represents the database in the form of a tree structure that is known as a frequent pattern or tree. The purpose of this frequent tree is to extract the most frequent patterns.

## Applications of Association Rule Learning

It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:

* **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
* **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
* **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
* It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

**8) Explain apriori algorithm.**

# Apriori Algorithm

Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.

The primary objective of the apriori algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions. Let's understand the apriori algorithm with the help of an example; suppose you go to Big Bazar and buy different products. It helps the customers buy their products with ease and increases the sales performance of the Big Bazar. In this tutorial, we will discuss the apriori algorithm with examples.

## Introduction

We take an example to understand the concept better. You must have noticed that the Pizza shop seller makes a pizza, soft drink, and breadstick combo together. He also offers a discount to their customers who buy these combos. Do you ever think why does he do so? He thinks that customers who buy pizza also buy soft drinks and breadsticks. However, by making combos, he makes it easy for the customers. At the same time, he also increases his sales performance.

Similarly, you go to Big Bazar, and you will find biscuits, chips, and Chocolate bundled together. It shows that the shopkeeper makes it comfortable for the customers to buy these products in the same place.

The above two examples are the best examples of Association Rules in [Data Mining](https://www.javatpoint.com/data-mining). It helps us to learn the concept of apriori algorithms.

## 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.

We have considered an easy example to discuss the apriori algorithm in data mining. In reality, you find thousands of such combinations.

**9) Explain Eclat algorithm**

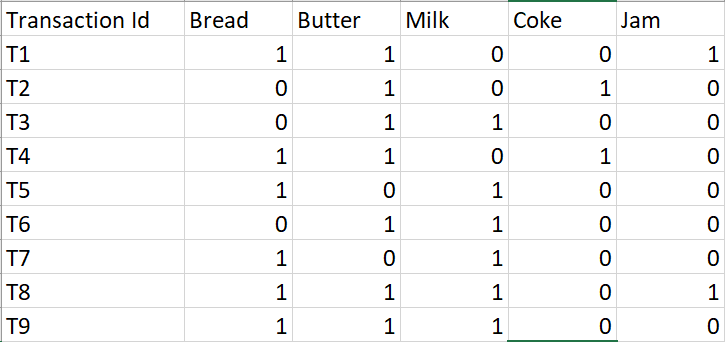
* **Prerequisites:** [Apriori Algorithm](https://www.geeksforgeeks.org/apriori-algorithm/)

The ECLAT algorithm stands for **Equivalence Class Clustering and bottom-up Lattice Traversal**. It is one of the popular methods of [Association Rule mining](https://en.wikipedia.org/wiki/Association_rule_learning). It is a more efficient and scalable version of the Apriori algorithm. While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph. This vertical approach of the ECLAT algorithm makes it a faster algorithm than the Apriori algorithm.

**How the algorithm work? :**  
The basic idea is to use Transaction Id Sets(tidsets) intersections to compute the support value of a candidate and avoiding the generation of subsets which do not exist in the prefix tree. In the first call of the function, all single items are used along with their tidsets. Then the function is called recursively and in each recursive call, each item-tidset pair is verified and combined with other item-tidset pairs. This process is continued until no candidate item-tidset pairs can be combined.

Let us now understand the above stated working with an example:-

Consider the following transactions record:-



The above-given data is a boolean matrix where for each cell (i, j), the value denotes whether the j’th item is included in the i’th transaction or not. 1 means true while 0 means false.

We now call the function for the first time and arrange each item with it’s tidset in a tabular fashion:-

**k = 1, minimum support = 2**

| **Item** | **Tidset** |
| --- | --- |
| Bread | {T1, T4, T5, T7, T8, T9} |
| Butter | {T1, T2, T3, T4, T6, T8, T9} |
| Milk | {T3, T5, T6, T7, T8, T9} |
| Coke | {T2, T4} |
| Jam | {T1, T8} |

We now recursively call the function till no more item-tidset pairs can be combined:-

**k = 2**

| **Item** | **Tidset** |
| --- | --- |
| {Bread, Butter} | {T1, T4, T8, T9} |
| {Bread, Milk} | {T5, T7, T8, T9} |
| {Bread, Coke} | {T4} |
| {Bread, Jam} | {T1, T8} |
| {Butter, Milk} | {T3, T6, T8, T9} |
| {Butter, Coke} | {T2, T4} |
| {Butter, Jam} | {T1, T8} |
| {Milk, Jam} | {T8} |

**k = 3**

| **Item** | **Tidset** |
| --- | --- |
| {Bread, Butter, Milk} | {T8, T9} |
| {Bread, Butter, Jam} | {T1, T8} |

**k = 4**

| **Item** | **Tidset** |
| --- | --- |
| {Bread, Butter, Milk, Jam} | {T8} |

We stop at k = 4 because there are no more item-tidset pairs to combine.

Since minimum support = 2, we conclude the following rules from the given dataset:-

| **Items Bought** | **Recommended Products** |
| --- | --- |
| Bread | Butter |
| Bread | Milk |
| Bread | Jam |
| Butter | Milk |
| Butter | Coke |
| Butter | Jam |
| Bread and Butter | Milk |
| Bread and Butter | Jam |

**Advantages over Apriori algorithm:-**

1. **Memory Requirements:** Since the ECLAT algorithm uses a Depth-First Search approach, it uses less memory than Apriori algorithm.
2. **Speed:** The ECLAT algorithm is typically faster than the Apriori algorithm.
3. **Number of Computations:** The ECLAT algorithm does not involve the repeated scanning of the data to compute the individual support values.

**10) Explain F-P Growth algorithm**

# FP Growth Algorithm in Data Mining

In Data Mining, finding frequent patterns in large databases is very important and has been studied on a large scale in the past few years. Unfortunately, this task is computationally expensive, especially when many patterns exist.

The FP-Growth Algorithm proposed by **Han in**. This is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the TreeProjection. In some later works, it was proved that FP-Growth performs better than other methods, including **Eclat** and **Relim**. The popularity and efficiency of the FP-Growth Algorithm contribute to many studies that propose variations to improve its performance.

### 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:**

ADVERTISEMENT

ADVERTISEMENT

* 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.

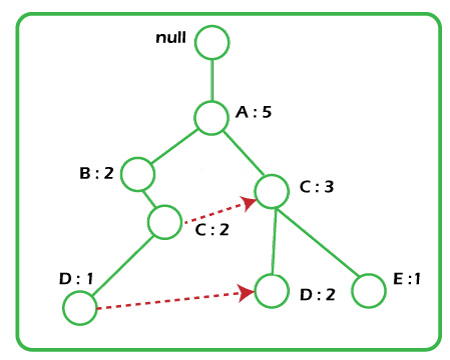
A frequent Pattern Tree is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the item set.

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.

