**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 K-means clustering algorithm is, how the algorithm works, along with the Python implementation of k-means clustering.

What is the K-Means Algorithm?

K-Means Clustering is an [Unsupervised Learning algorithm](https://www.javatpoint.com/unsupervised-machine-learning), which groups the unlabeled dataset into different clusters. Here K defines the number of predefined 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 to 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.

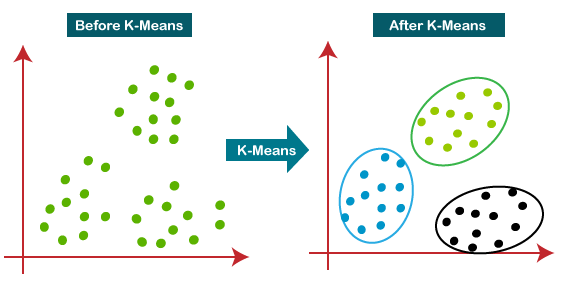
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means [clustering](https://www.javatpoint.com/clustering-in-machine-learning) 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 data points 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 different 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.

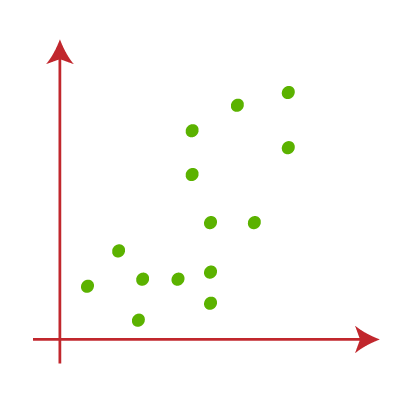
**Step-5:** Repeat the third steps, which means re assign each datapoint to the new closest centroid of each cluster.

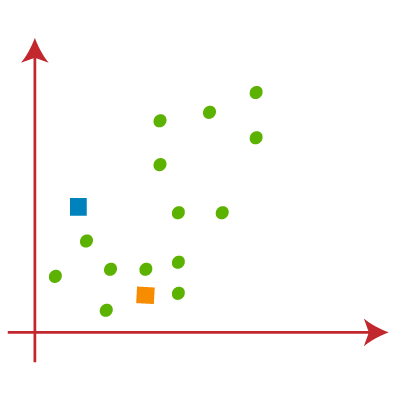
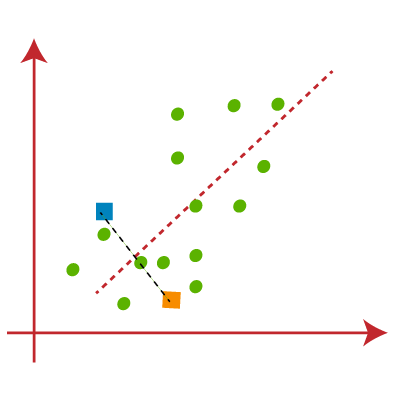
**Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.

**Step-7**: The model is ready.

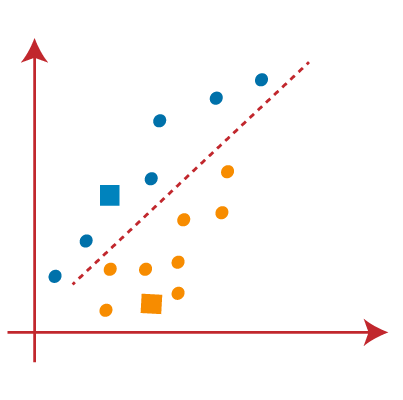
Let's understand the above steps by considering the visual plots:

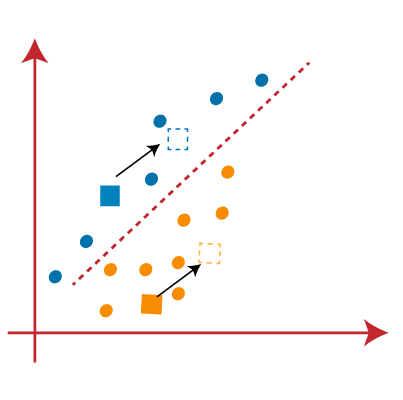
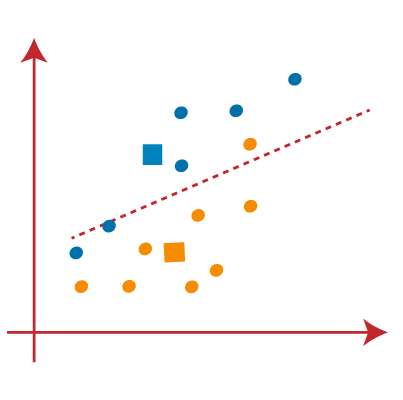
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



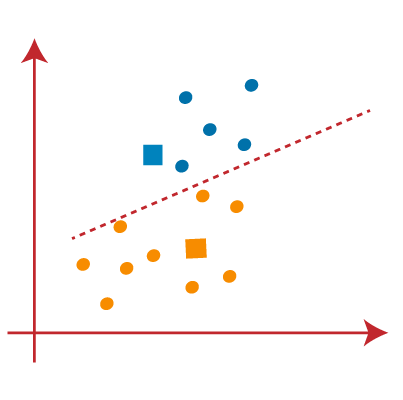
* Let's take the number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
* We need to choose some random k points or centroid to form the cluster. These points can be either the points 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 on the left side of the line are 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.

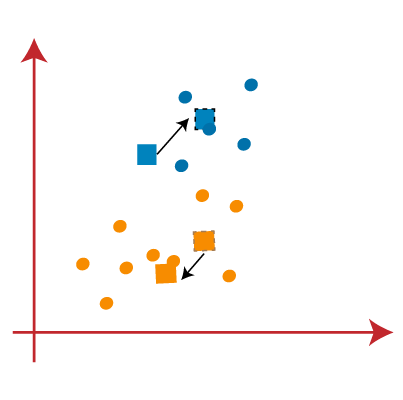
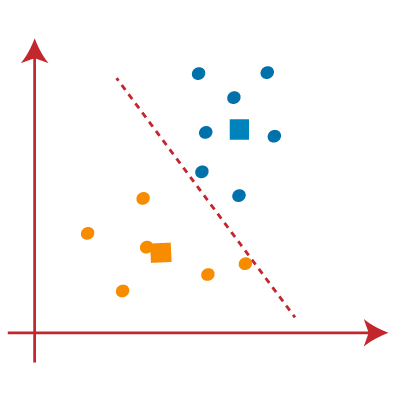
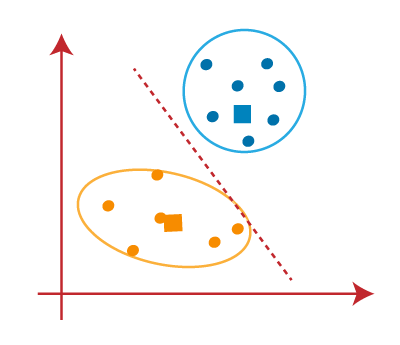


* As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:  
  
* Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:  
  

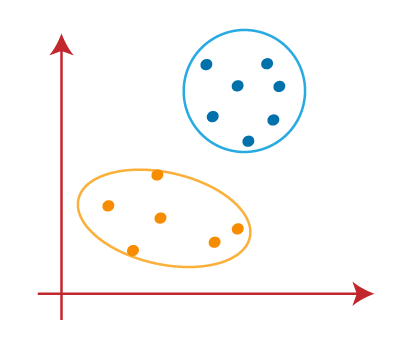
From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.



As reassignment has taken place, we will again go to 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:

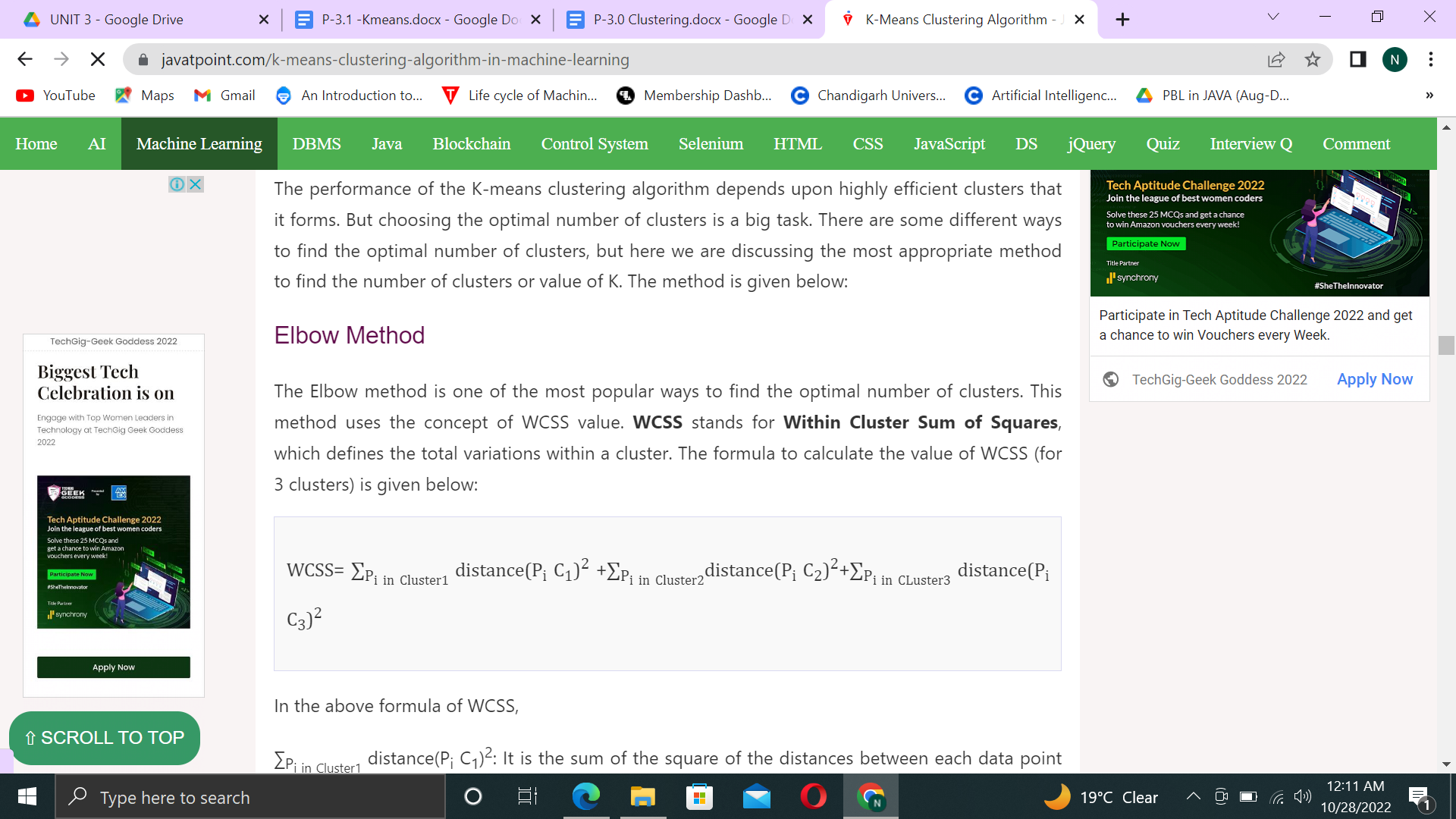


How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below



In the above formula of WCSS,

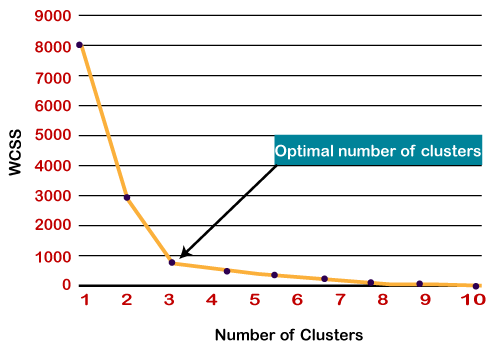
∑Pi in Cluster1 distance(Pi C1)2: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

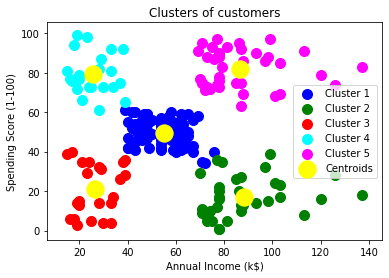
To find the optimal value of clusters, the elbow method follows the below steps:

* It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
* For each value of K, calculate the WCSS value.
* Plots a curve between calculated WCSS values and the number of clusters K.
* The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Note: We can choose the number of clusters equal to the given data points. If we choose the number of clusters equal to the data points, then the value of WCSS becomes zero, and that will be the endpoint of the plot.



The output image is clearly showing the five different clusters with different colors. The clusters are formed between two parameters of the dataset; Annual income of customer and Spending. We can change the colors and labels as per the requirement or choice. We can also observe some points from the above patterns, which are given below:

* **Cluster1** shows the customers with average salary and average spending.
* Cluster2 shows the customer has a high income but low spending, so we can categorize them as **careful**.
* Cluster3 shows the low income and also low spending so they can be categorized as sensible.
* Cluster4 shows the customers with low income with very high spending so they can be categorized as **careless**.
* Cluster5 shows the customers with high income and high spending so they can be categorized as target, and these customers can be the most profitable customers for the mall owner.

Apriori Algorithm in Machine Learning

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rules, it determines how strongly or how weakly two objects are connected. This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.

This algorithm was given by the **R. Agrawal** and **Srikant** in the year **1994**. It is mainly used for *market basket analysis* and helps to find those products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

**What is a Frequent Itemset?**

Frequent itemsets are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.

Suppose there are the two transactions: A= {1,2,3,4,5}, and B= {2,3,7}, in these two transactions, 2 and 3 are the frequent item sets.

Note: To better understand the apriori algorithm, and related terms such as support and confidence, it is recommended to understand the association rule learning.

Steps for Apriori Algorithm

Below are the steps for the apriori algorithm:

**Step-1:** Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

**Step-2:** Take all supports in the transaction with higher support value than the minimum or selected support value.

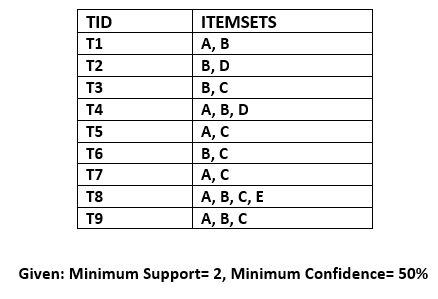
**Step-3:** Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

**Step-4:** Sort the rules as the decreasing order of lift.

Apriori Algorithm Working

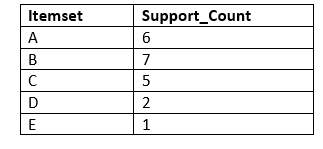
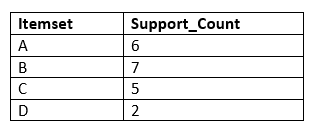
We will understand the apriori algorithm using an example and mathematical calculation:

**Example:** Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm:

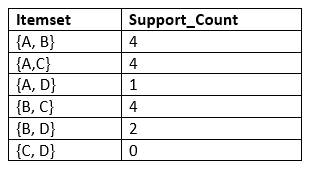
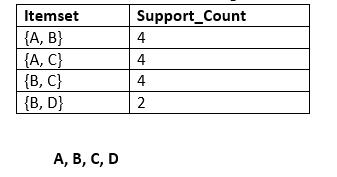


Solution:

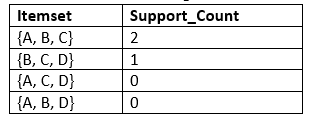
Step-1: Calculating C1 and L1:

* In the first step, we will create a table that contains the support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the **Candidate set or C1.**  
  
* Now, we will take out all the itemsets that have a greater support count than the Minimum Support (2). It will give us the table for the **frequent itemset L1.**  
  Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.  
  

Step-2: Candidate Generation C2, and L2:

* In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.
* After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2:  
  
* Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2  
  

Step-3: Candidate generation C3, and L3:

* For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:  
  
* Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination, i.e., **{A, B, C}.**

Step-4: Finding the association rules for the subsets:

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula **sup( A ^B)/A.** After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Consider the below table:

| **Rules** | **Support** | **Confidence** |
| --- | --- | --- |
| A ^B → C | 2 | Sup{(A ^B) ^C}/sup(A ^B)= 2/4=0.5=50% |
| B^C → A | 2 | Sup{(B^C) ^A}/sup(B ^C)= 2/4=0.5=50% |
| A^C → B | 2 | Sup{(A ^C) ^B}/sup(A ^C)= 2/4=0.5=50% |
| C→ A ^B | 2 | Sup{(C^( A ^B)}/sup(C)= 2/5=0.4=40% |
| A→ B^C | 2 | Sup{(A^( B ^C)}/sup(A)= 2/6=0.33=33.33% |
| B→ B^C | 2 | Sup{(B^( B ^C)}/sup(B)= 2/7=0.28=28% |

As the given threshold or minimum confidence is 50%, so the first three rules **A ^B → C, B^C → A, and A^C → B** can be considered as the strong association rules for the given problem.

Advantages of Apriori Algorithm

* This is easy to understand algorithm
* The join and prune steps of the algorithm can be easily implemented on large datasets.

Disadvantages of Apriori Algorithm

* The apriori algorithm works slow compared to other algorithms.
* The overall performance can be reduced as it scans the database for multiple times.
* The time complexity and space complexity of the apriori algorithm is O(2D), which is very high. Here D represents the horizontal width present in the database.

# K-Medoids Clustering Algorithm

K-Medoids is a clustering algorithm resembling the K-Means clustering technique. It falls under the category of unsupervised machine learning.

K-Medoids is a clustering algorithm resembling the K-Means clustering technique. It falls under the category of [unsupervised machine learning](https://en.wikipedia.org/wiki/Unsupervised_learning). It majorly differs from the K-Means algorithm in terms of the way it selects the clusters’ centres. The former selects the average of a cluster’s points as its centre (which may or may not be one of the data points) while the latter always picks the actual data points from the clusters as their centres (also known as ‘**exemplars**’ or ‘**medoids**’). K-Medoids also differs in this respect from the K-Medians algorithm whic,h is the same as K-means, except that it chooses the medians (instead of means) of the clusters as centres.

Bottom of Form

## **Working of the K-Medoids approach**

The steps followed by the K-Medoids algorithm for clustering are as follows:

1. Randomly choose ‘k’ points from the input data (‘k’ is the number of clusters to be formed). The correctness of the choice of k’s value can be assessed using methods such as [silhouette method](https://en.wikipedia.org/wiki/Silhouette_(clustering)).
2. Each data point gets assigned to the cluster to which its nearest medoid belongs.
3. For each data point of cluster i, its distance from all other data points is computed and added. The point of ith cluster for which the computed sum of distances from other points is minimal is assigned as the medoid for that cluster.
4. Steps (2) and (3) are repeated until convergence is reached i.e. the medoids stop moving.

## **Complexity of K-Medoids algorithm**

The complexity of the K-Medoids algorithm comes to O(N2CT) where N, C and T denote the number of data points, number of clusters and number of iterations respectively. With similar notations, the complexity K-Means algorithm can be given as O(NCT).

## **Advantages of the technique**

Mean of the data points is a measure that gets highly affected by the extreme points. So in K-Means algorithm, the centroid may get shifted to a wrong position and hence result in incorrect clustering if the data has outliers because then other points will move away from  . On the contrary, a medoid in the K-Medoids algorithm is the most central element of the cluster, such that its distance from other points is minimum. Since medoids do not get influenced by extremities, the K-Medoids algorithm is more robust to outliers and noise than K-Means algorithm. The following figure explains how mean’s and medoid’s

K-Medoids algorithm is found useful for practical applications such as face recognition. The medoid can correspond to the typical photo of the individual whose face is to be recognized. But if K-Means algorithm is used instead, some blurred image may get assigned as the centroid, which has mixed features from several photos of the individual and hence makes the face recognition task difficult.