# 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.

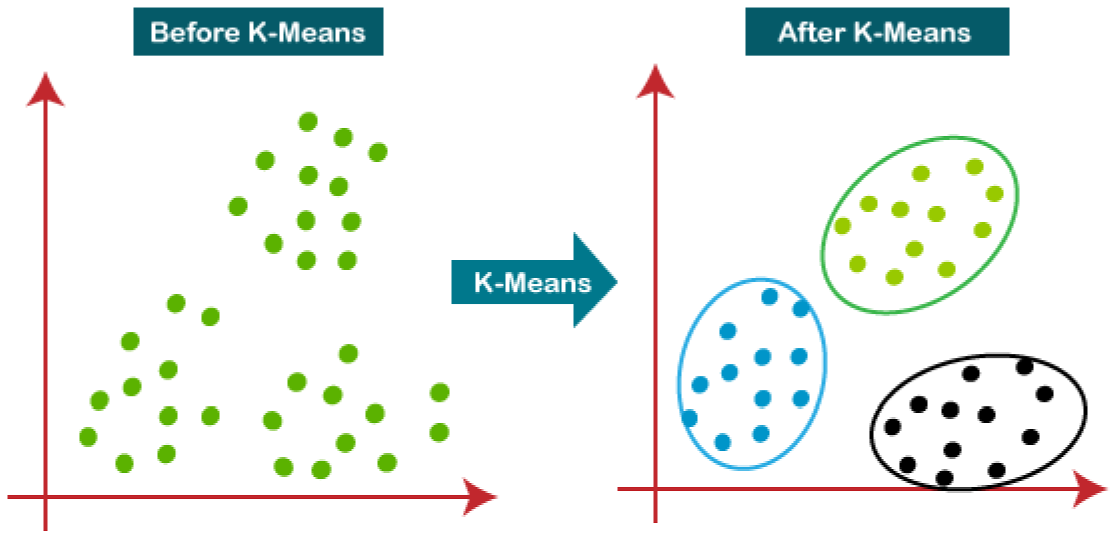
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

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:

### Initialization:

1. **Input**: The input consists of a dataset $X$ containing $n$ data points, each with $d$ features, and the desired number of clusters $k$.
2. **Initialization of Centroids**: Randomly select $k$ data points from the dataset to serve as the initial centroids.

### Iterative Optimization:

3. **Assignment Step**:
   - For each data point $x_i$, calculate its distance to each centroid $c_j$ using a distance metric (commonly Euclidean distance).
   - Assign $x_i$ to the cluster whose centroid is closest: $\arg\min_j \operatorname{dist}(x_i, c_j)$.
   - Repeat this process for all data points, resulting in $k$ clusters.
4. **Update Step**:
   - Recalculate the centroid of each cluster by taking the mean of all data points assigned to that cluster.
   - If a cluster becomes empty after the [illegible]ignment step, randomly reinitialize its centroid.

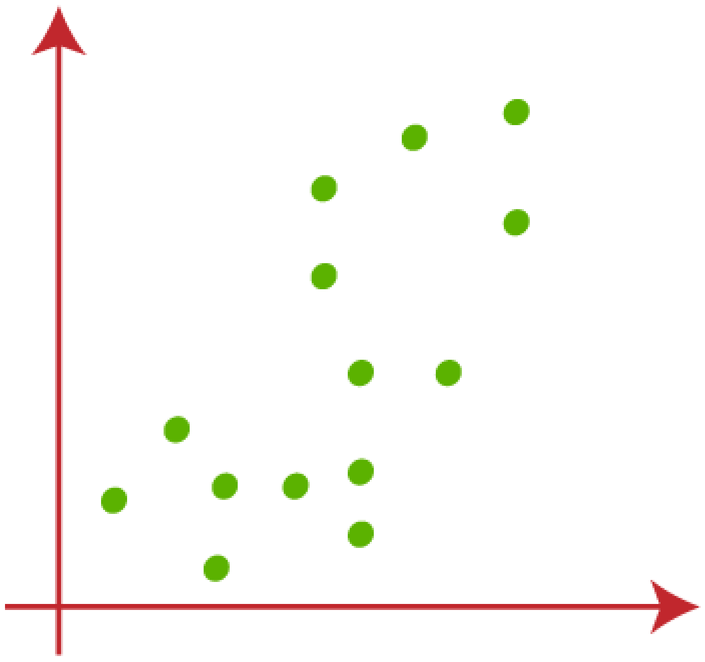
5. **Convergence Check**:
    - Check if the centroids have changed significantly from the previous iteration. This can be measured using a convergence criterion, such as the change in centroids' positions or the change in within-cluster variance.
    - If convergence criterion is not met, repeat steps 3 and 4. Otherwise, proceed to the next step.

## Termination:

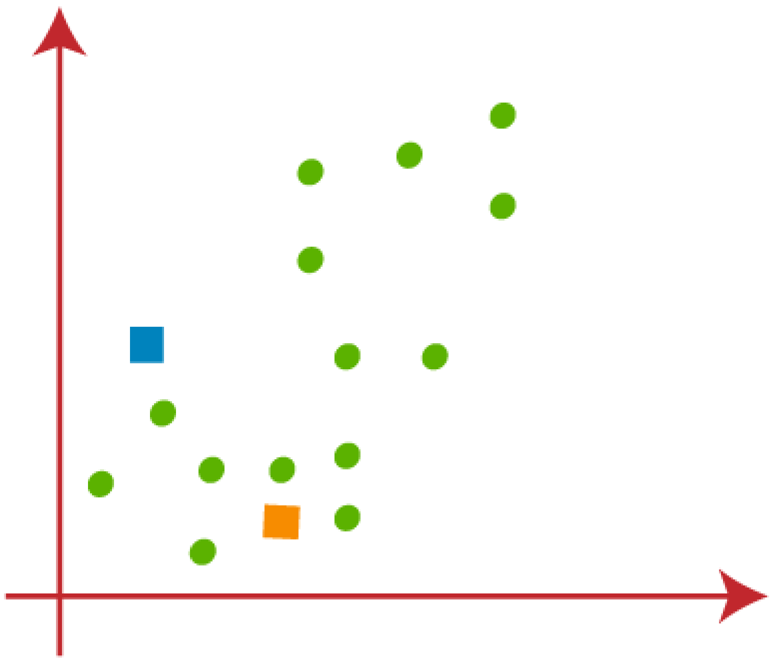
6. **Termination Criteria:**
    - Stop the algorithm when one of the following conditions is met:
        - The centroids no longer change significantly (convergence).
        - A maximum number of iterations is reached.
        - Another stopping criterion specified by the user.

## Output:

7. **Final Clusters:**
    - The algorithm outputs $k$ clusters, each represented by its centroid and containing the data points assigned to it during the final iteration.

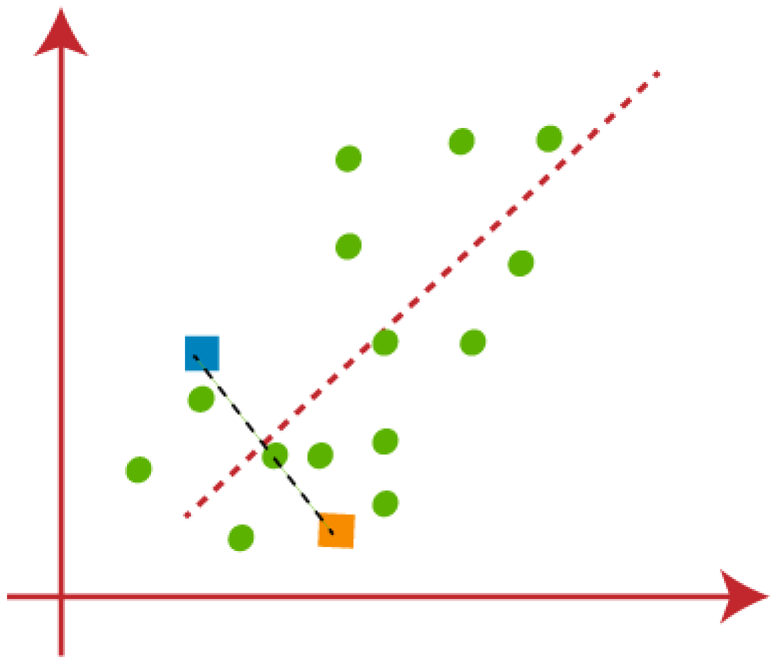


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points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

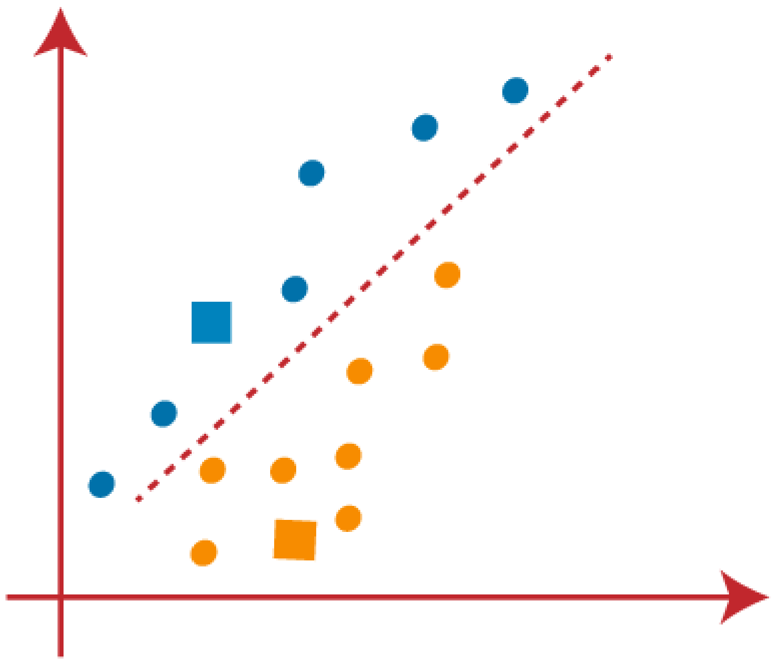


- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids.

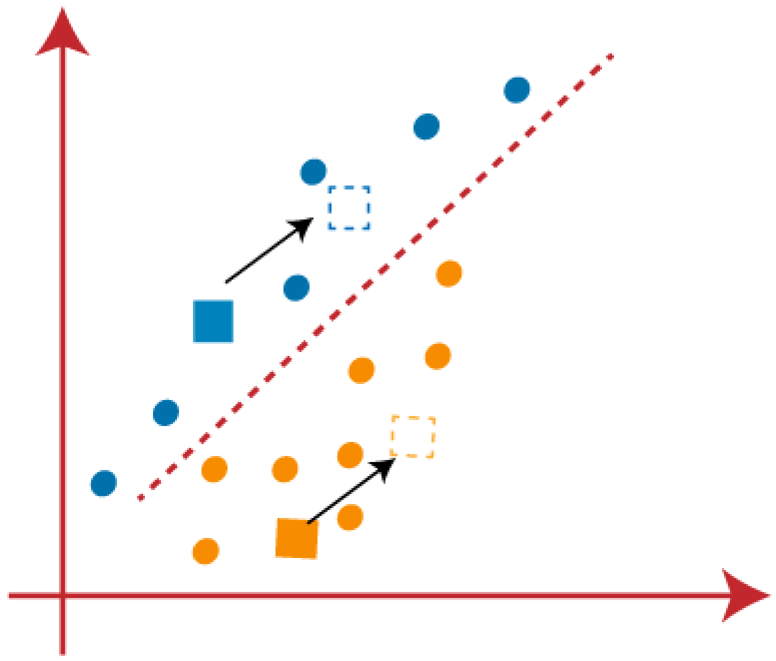
Consider the below image:



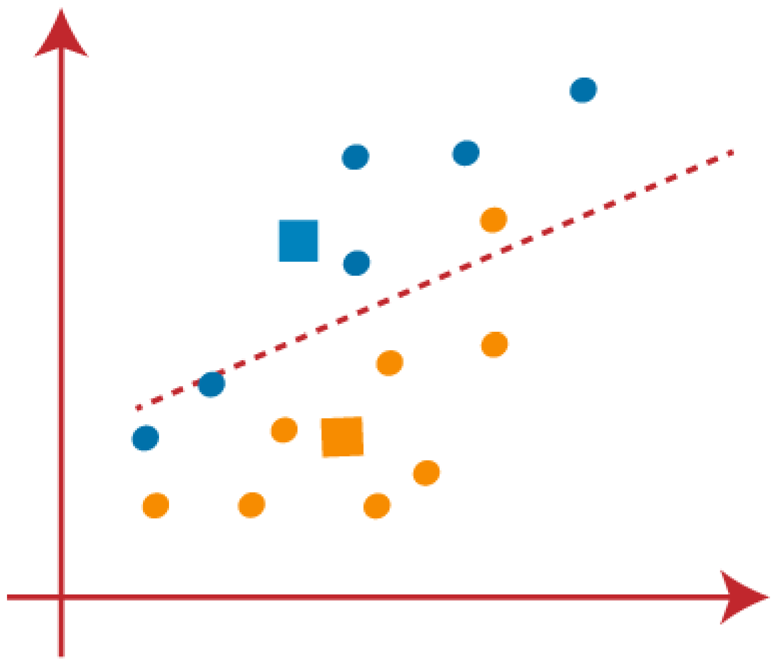
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



- 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:

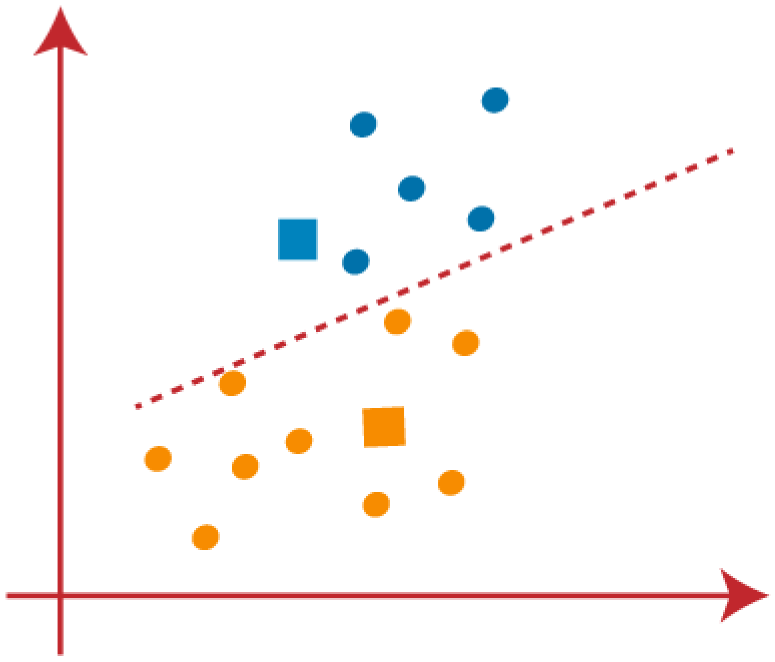


- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



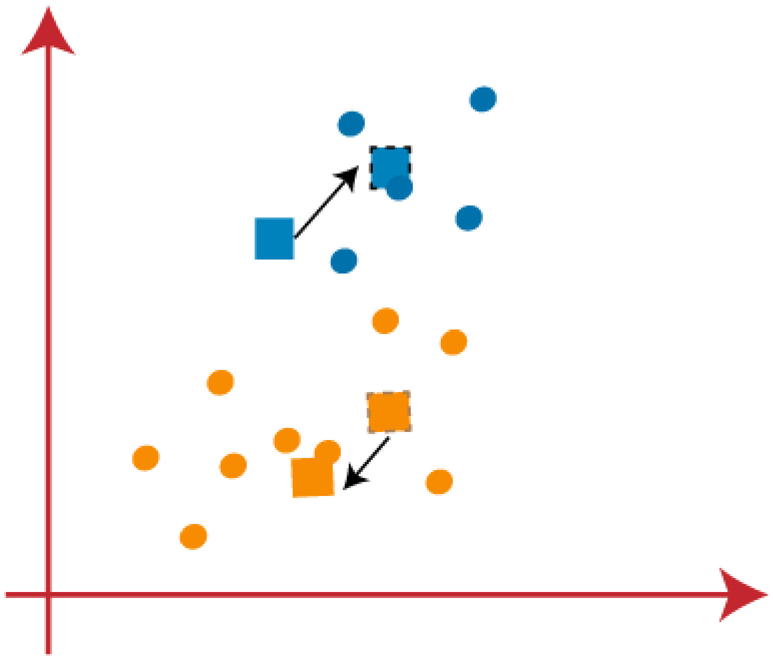
From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

ADVERTISEMENT
ADVERTISEMENT

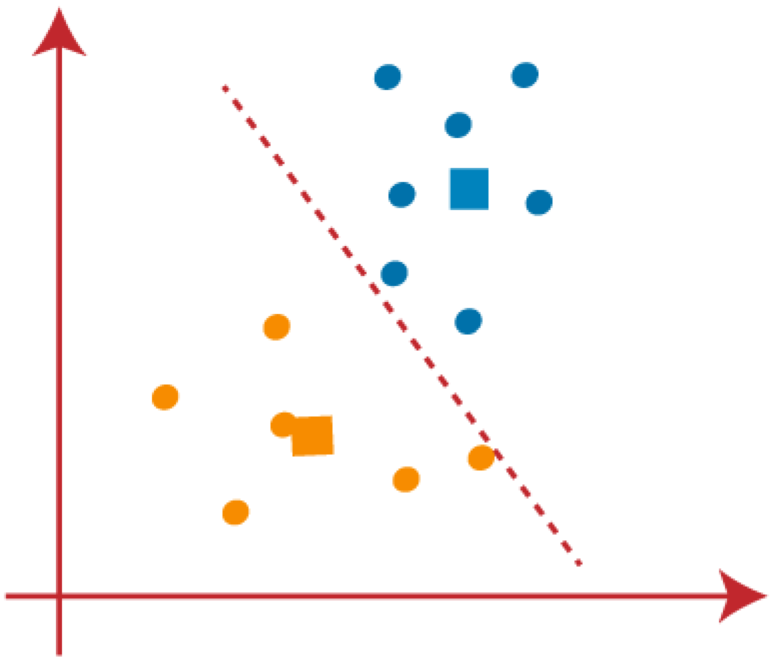


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

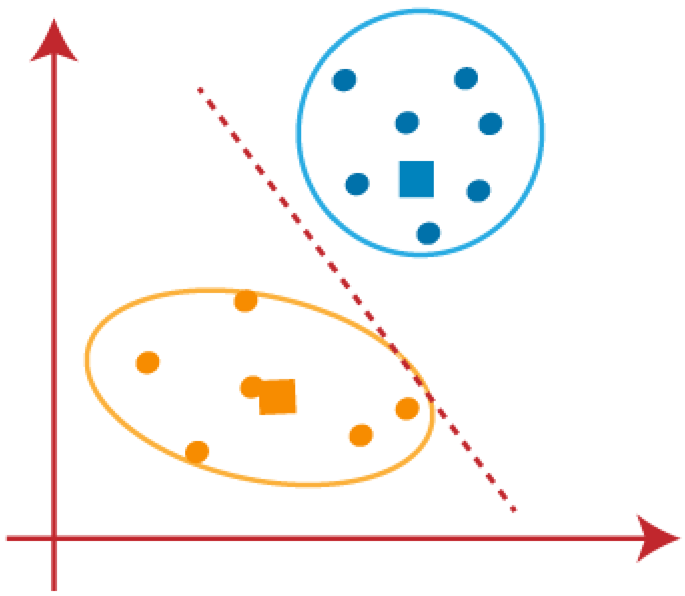
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



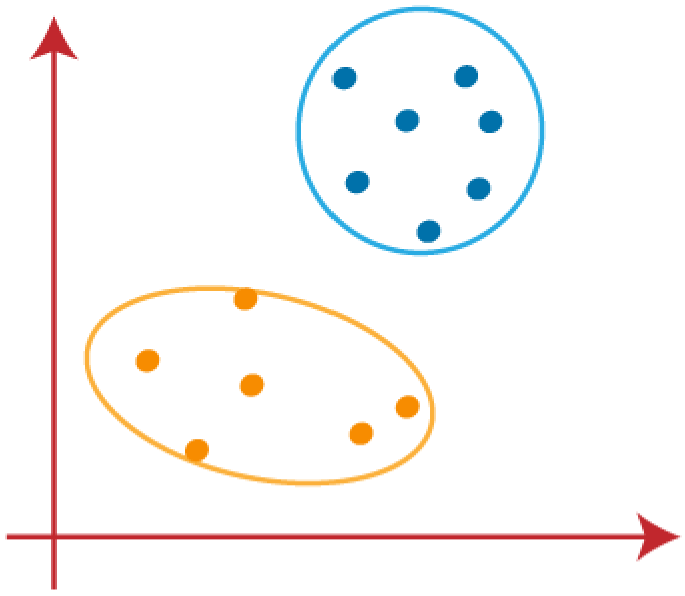
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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:



## Advantages and Disadvantages

## Advantages

The following are some advantages of K-Means clustering algorithms −

- It is very easy to understand and implement.
- If we have large number of variables then, K-means would be faster than Hierarchical clustering.
- On re-computation of centroids, an instance can change the cluster.
- Tighter clusters are formed with K-means as compared to Hierarchical clustering.

## Disadvantages

The following are some disadvantages of K-Means clustering algorithms −

- It is a bit difficult to predict the number of clusters i.e. the value of k.
- Output is strongly impacted by initial inputs like number of clusters (value of k).
- Order of data will have strong impact on the final output.
- It is very sensitive to rescaling. If we will rescale our data by means of normalization or standardization, then the output will completely change.final output.
- It is not good in doing clustering job if the clusters have a complicated geometric shape.