ALBUKHARY INTERNATIONAL UNIVERSITY

Artificial Intelligence

LAB ELEVEN

K-Means Clustering 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 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.

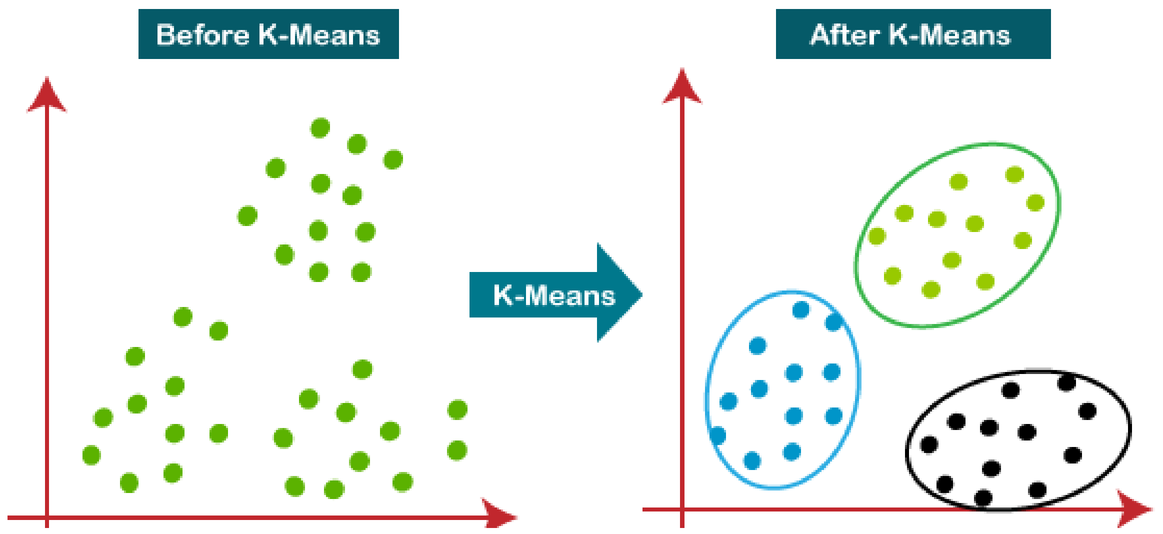
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The 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:

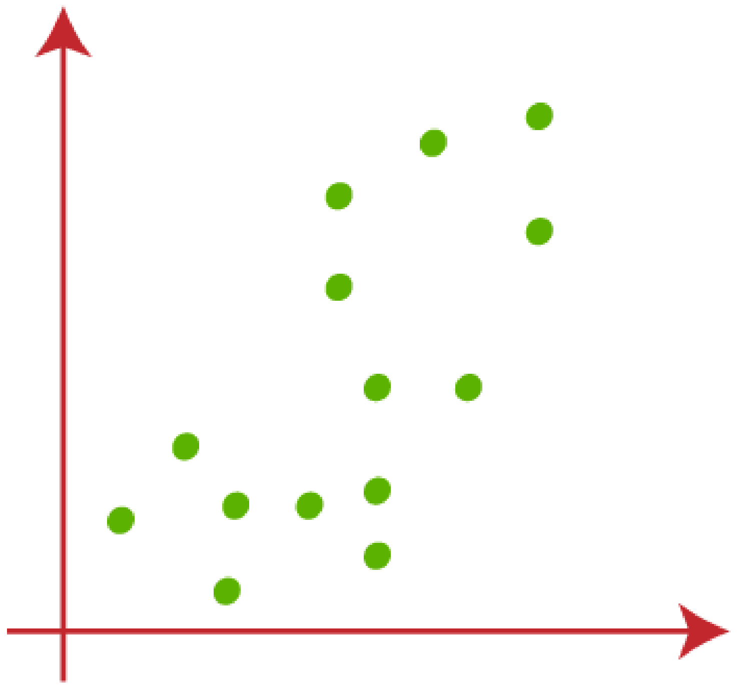


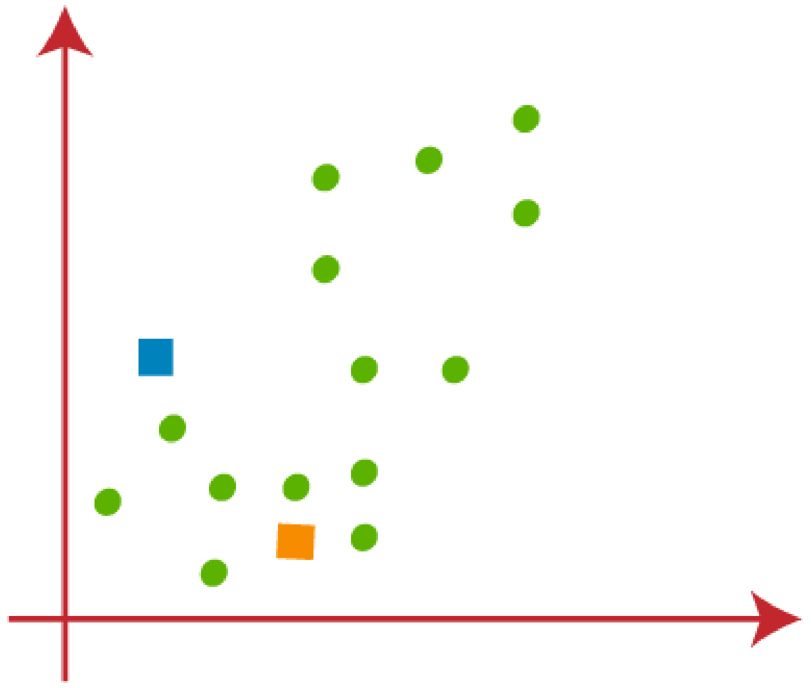
How does the K-Means Algorithm Work?

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

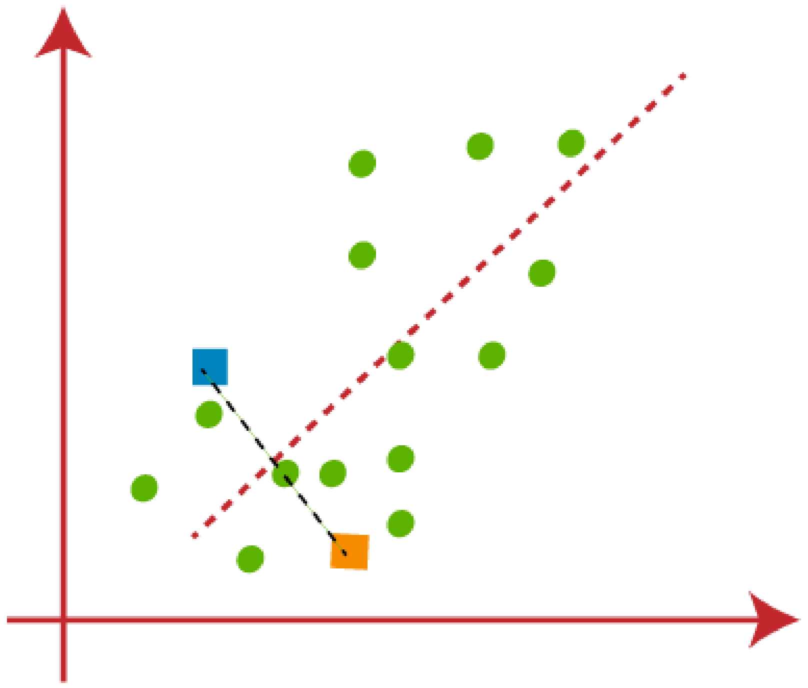
- Select the number K to decide the number of clusters.
- Select random K points or centroids. (It can be other from the input dataset).
- Assign each data point to their closest centroid, which will form the predefined K clusters.
- Calculate the variance and place a new centroid of each cluster.
- Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
- If any reassignment occurs, then go to step-4 else go to FINISH.
- 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:

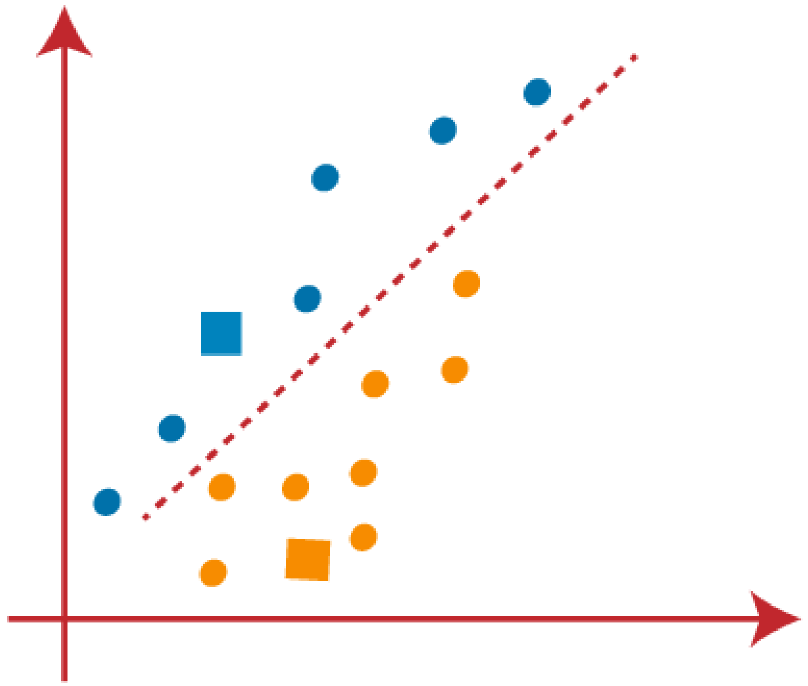




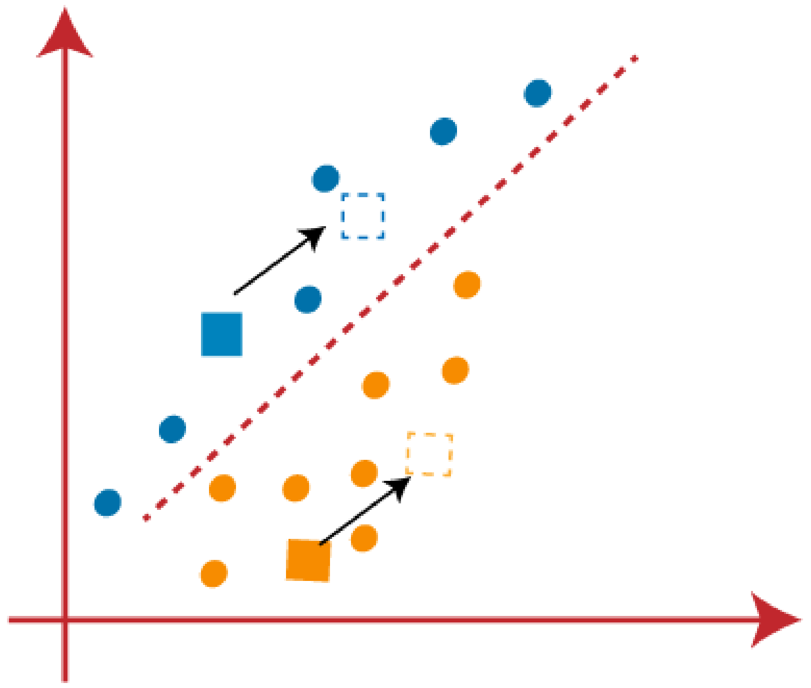
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



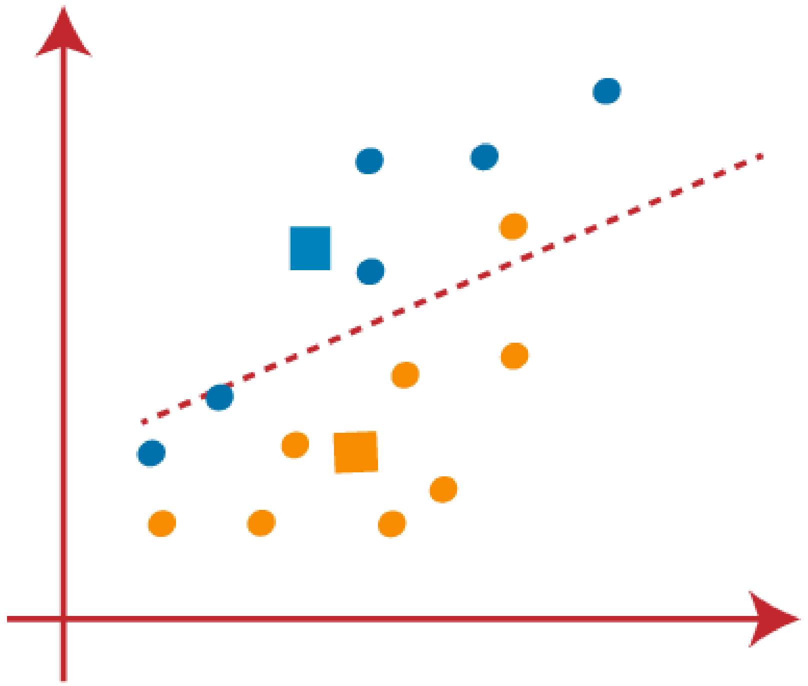
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



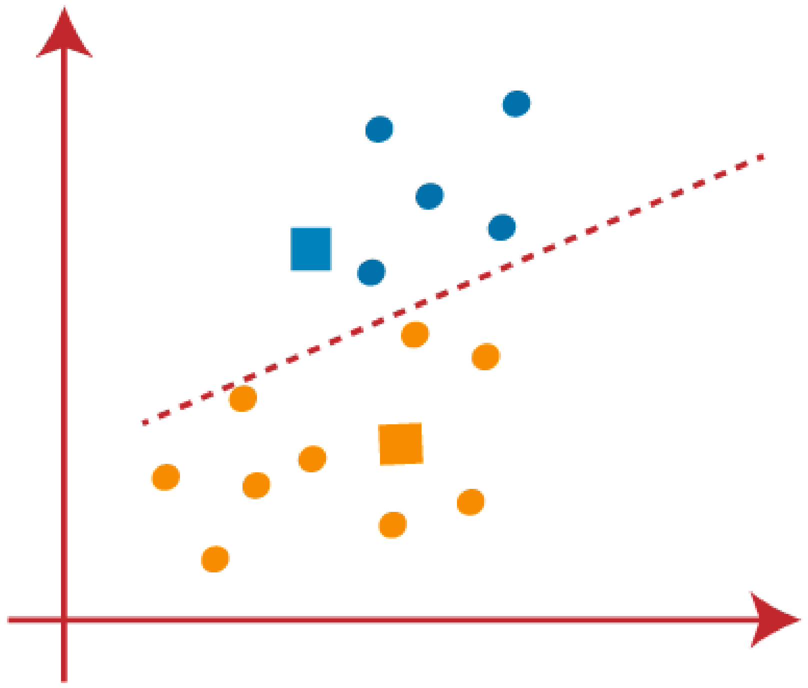
- As we need to find the closest cluster, so we will repeat the process by choosing a new centroid. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- 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:

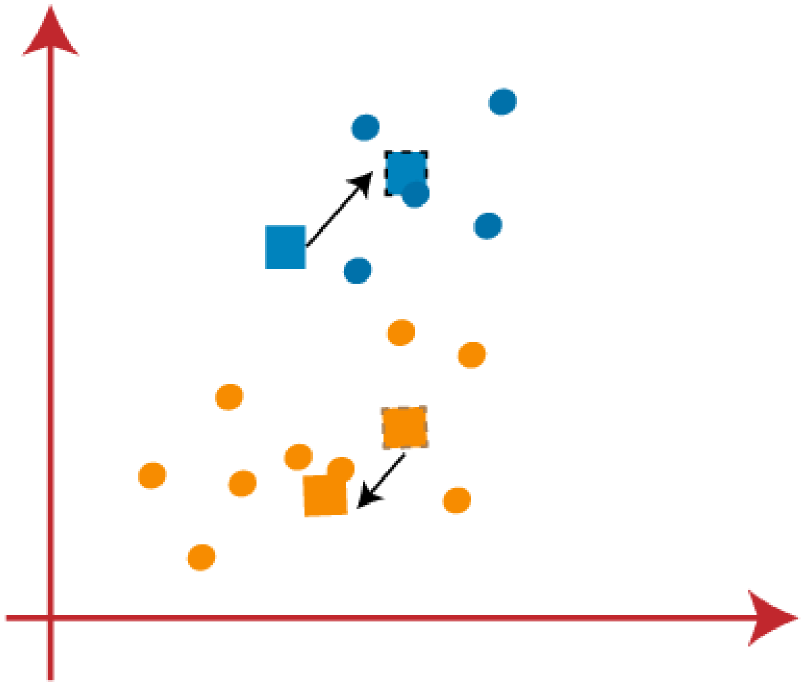


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

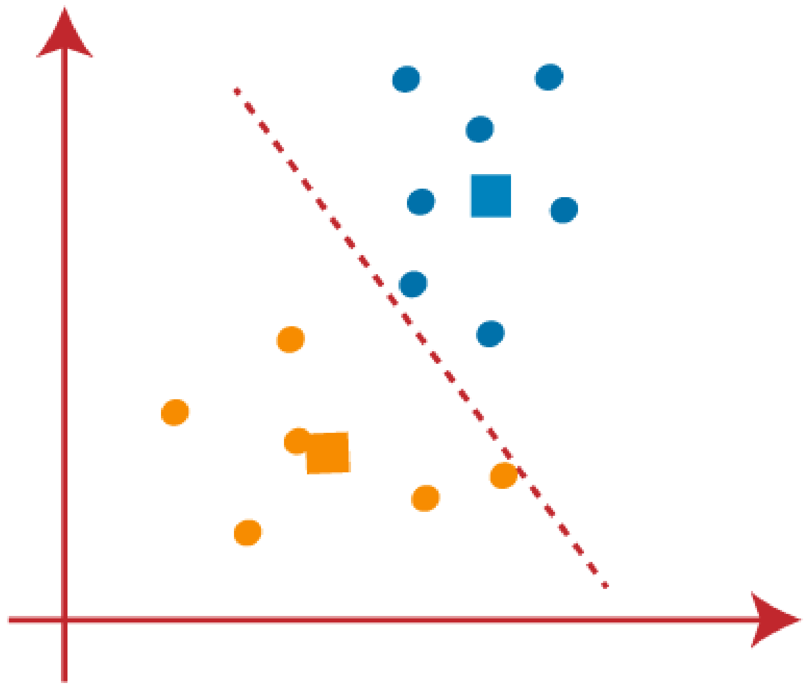


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

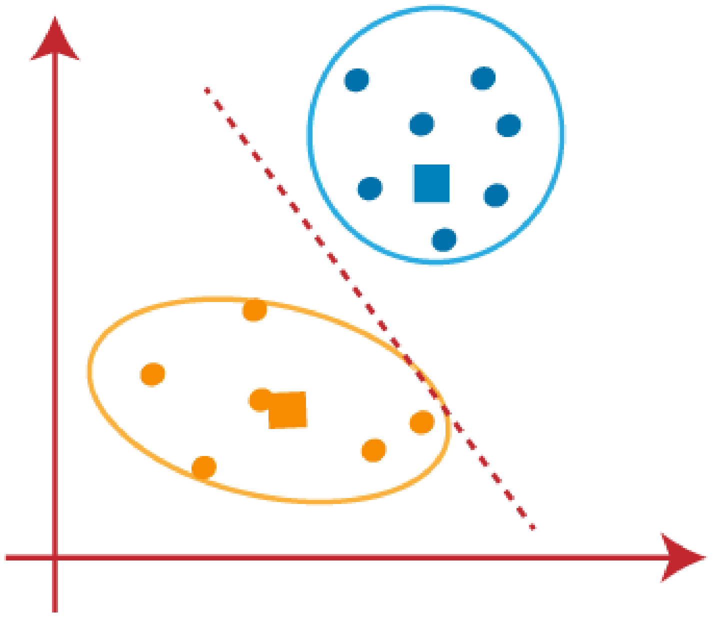
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



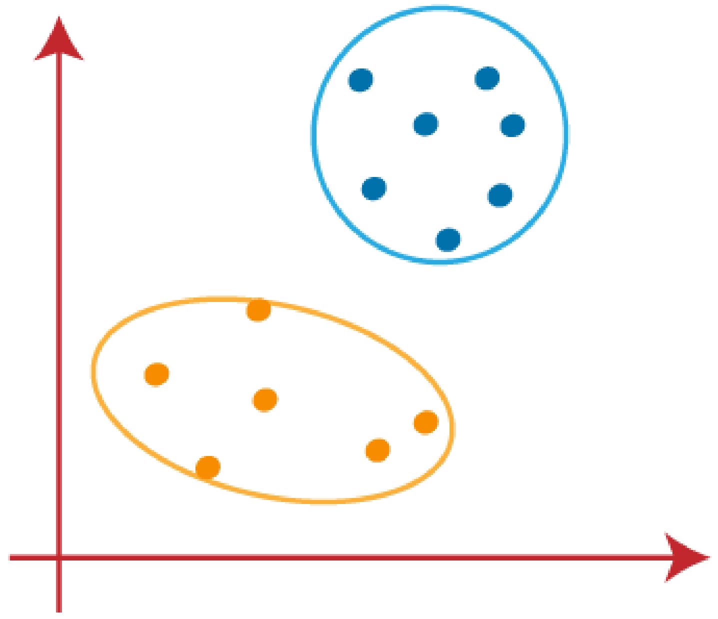
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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:



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.

Python Implementation of K-means Clustering Algorithm

Before implementation, let's understand what type of problem we will solve here. So, we have a dataset of Mall_Customers, which is the data of customers who visit the mall and spend there.

In the given dataset, we have Customer_Id, Gender, Age, Annual Income ($), and Spending Score (which is the calculated value of how much a customer has spent in the mall, the more the value, the more he has spent). From this dataset, we need to calculate some patterns, as it is an unsupervised method, so we don't know what to calculate exactly.

The steps to be followed for the implementation are given below:

- Data Pre-processing
- Finding the optimal number of clusters using the elbow method
- Training the K-means algorithm on the training dataset
- Visualizing the clusters

1. Data pre-processing Step

Importing Libraries

In [ ]:
```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the Dataset

In [ ]:
```
# Importing the dataset
data = pd.read_csv('Mall_Customers.csv')
```

```
In [ ]: data
```

Extracting Independent Variables

Here we don't need any dependent variable for data pre-processing step as it is a clustering problem, and we have no idea about what to determine. So we will just add a line of code for the matrix of features.

```
In [ ]: x = data.iloc[:, [3, 4]].values
```

2. Finding the optimal number of clusters using the elbow method

As we know, the elbow method uses the WCSS concept to draw the plot by plotting WCSS values on the Y-axis and the number of clusters on the X-axis. So we are going to calculate the value for WCSS for different k values ranging from 1 to 10.

```
In [ ]: #finding optimal number of clusters using the elbow method
        from sklearn.cluster import KMeans
        wcss_list= []  #Initializing the list for the values of WCSS

        #Using for loop for iterations from 2 to 9.
        for i in range(2, 10):
            kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
            kmeans.fit(x)
            wcss_list.append(kmeans.inertia_)
        plt.plot(range(2, 10), wcss_list)
        plt.title('The Elobw Method Graph')
        plt.xlabel('Number of clusters(k)')
        plt.ylabel('wcss_list')
        plt.show()
```

As we can see in the above code, we have used the KMeans class of sklearn. cluster library to form the clusters.

Next, we have created the wcss_list variable to initialize an empty list, which is used to contain the value of wcss computed for different values of k ranging from 2 to 9.

After that, we have initialized the for loop for the iteration on a different value of k ranging from 2 to 9; since for loop in Python, exclude the outbound limit, so it is taken as 11 to include 10th value.

The rest part of the code is similar as we did in earlier topics, as we have fitted the model on a matrix of features and then plotted the graph between the number of clusters and WCSS.

In [ ]:
```
wcss_list
```

3. Training the K-means algorithm on the training dataset

As we have got the number of clusters, so we can now train the model on the dataset.

To train the model, we will use the same two lines of code as we have used in the above section, but here instead of using i, we will use 5, as we know there are 5 clusters that need to be formed.

In [ ]:
```
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

The first line is the same as above for creating the object of KMeans class.

In the second line of code, we have created the dependent variable y_predict to train the model.

In [ ]:
```
y_predict
```

4 Visualizing the Clusters

The last step is to visualize the clusters. As we have 5 clusters for our model, so we will visualize each cluster one by one.

In [ ]:
```python
#visulaizing the clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first clu
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second c
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third clust
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cl
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centro
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

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 so we can categorize these customers as
- 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.

In [ ]:
```

```