K - Means

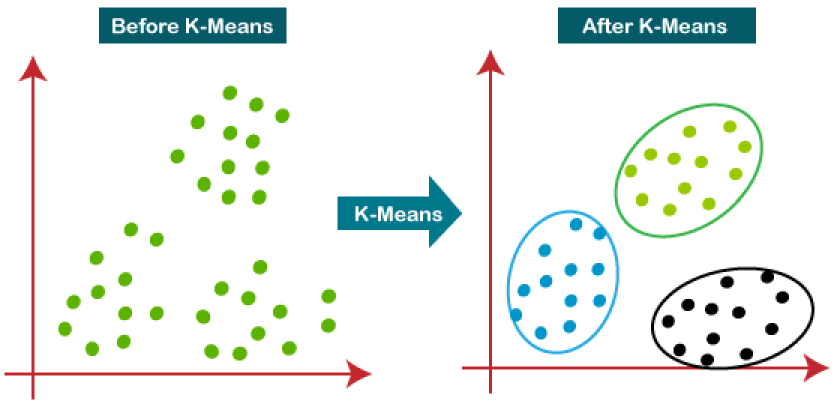
- It is a type of unsupervised algorithm which solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters).
- Data points inside a cluster are homogeneous and heterogeneous to peer groups.
- Remember figuring out shapes from ink blots? K means is somewhat similar this activity. You look at the shape and spread to decipher how many different clusters / population are present!

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.

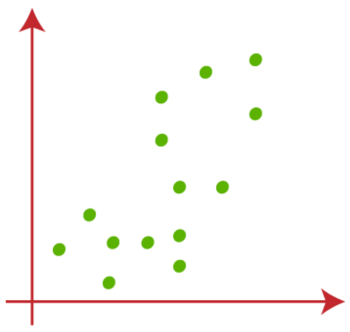
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

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

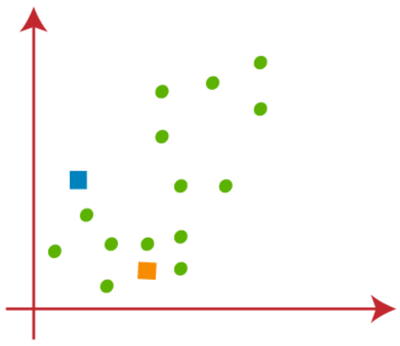
Step-7: The model is ready.

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

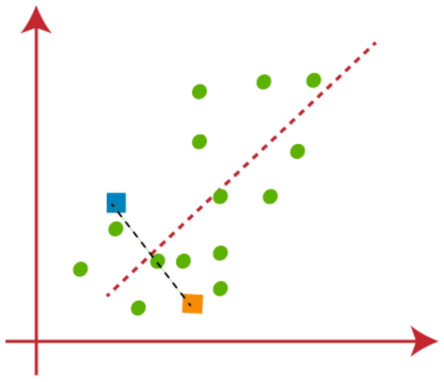
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



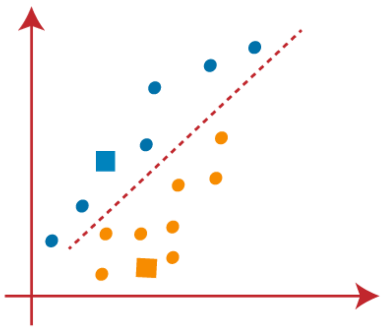
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points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



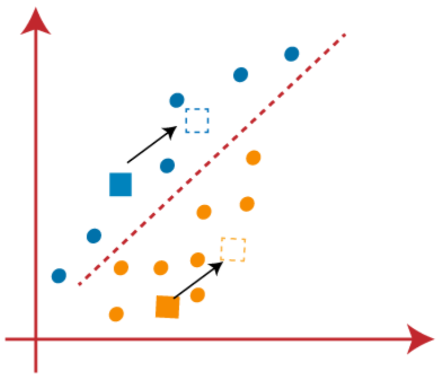
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider below image.



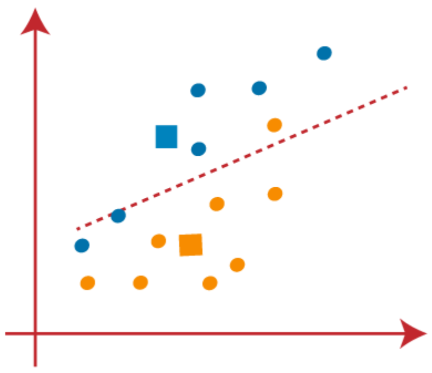
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



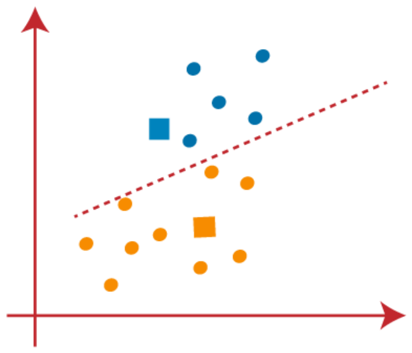
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- 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:

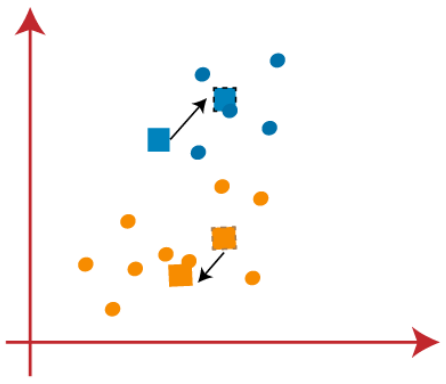


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

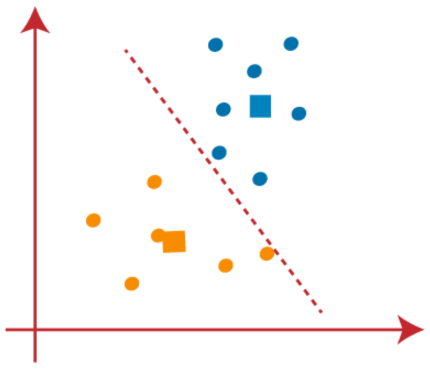


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

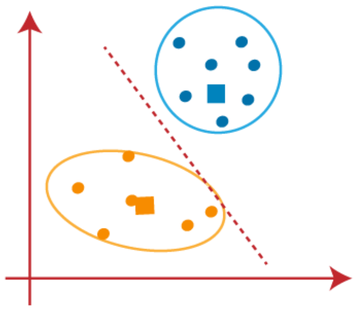
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



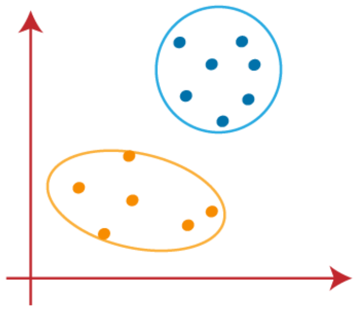
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$\text{WCSS}= \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i\, C_2)^2 + \sum_{P_i \text{ in CLuster3}} \text{distance}(P_i\, C_3)^2$$

In the above formula of WCSS,

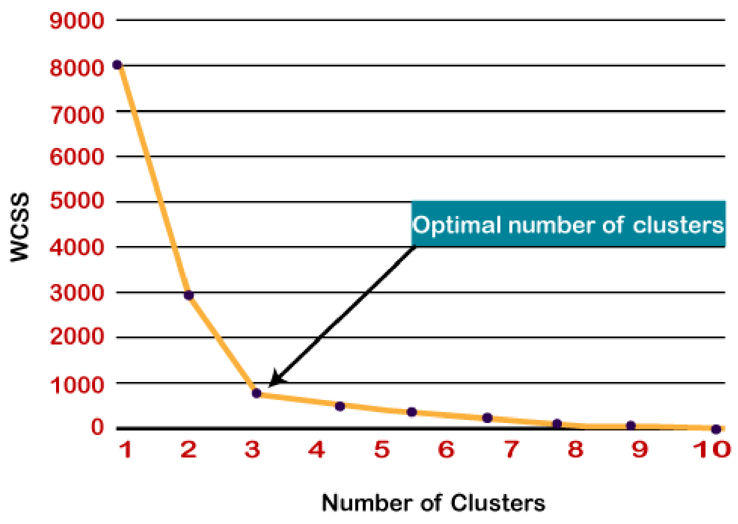
$\sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

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, calculates 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.

How K-means forms cluster

- K-means picks k number of points for each cluster known as centroids.
- Each data point forms a cluster based on existing cluster members. Here we have new centroids.
- As we have new centroids, repeat step 2 and 3. Find the closest distance for each data point from new centroids and get associated with new k-clusters.
- Repeat this process until convergence occours i.e. centroids does not change.

How to determine value of K

- In k-means, we have clusters and each cluster has its own centroid. Sum of square of difference between centroid and the data points within a cluster constitutes within sum of square value for that cluster, Also, when the sum of square values for all the cluster are added, it becomes total within sum of square value for the cluster solution.
- We know that as the number of cluster increases, this value keeps on decreasing but if you plot the result you may see that the sum of squared distance decreases sharply up to some value of k, and then much more slowly after that. Here, we can find the optimum number of cluster.

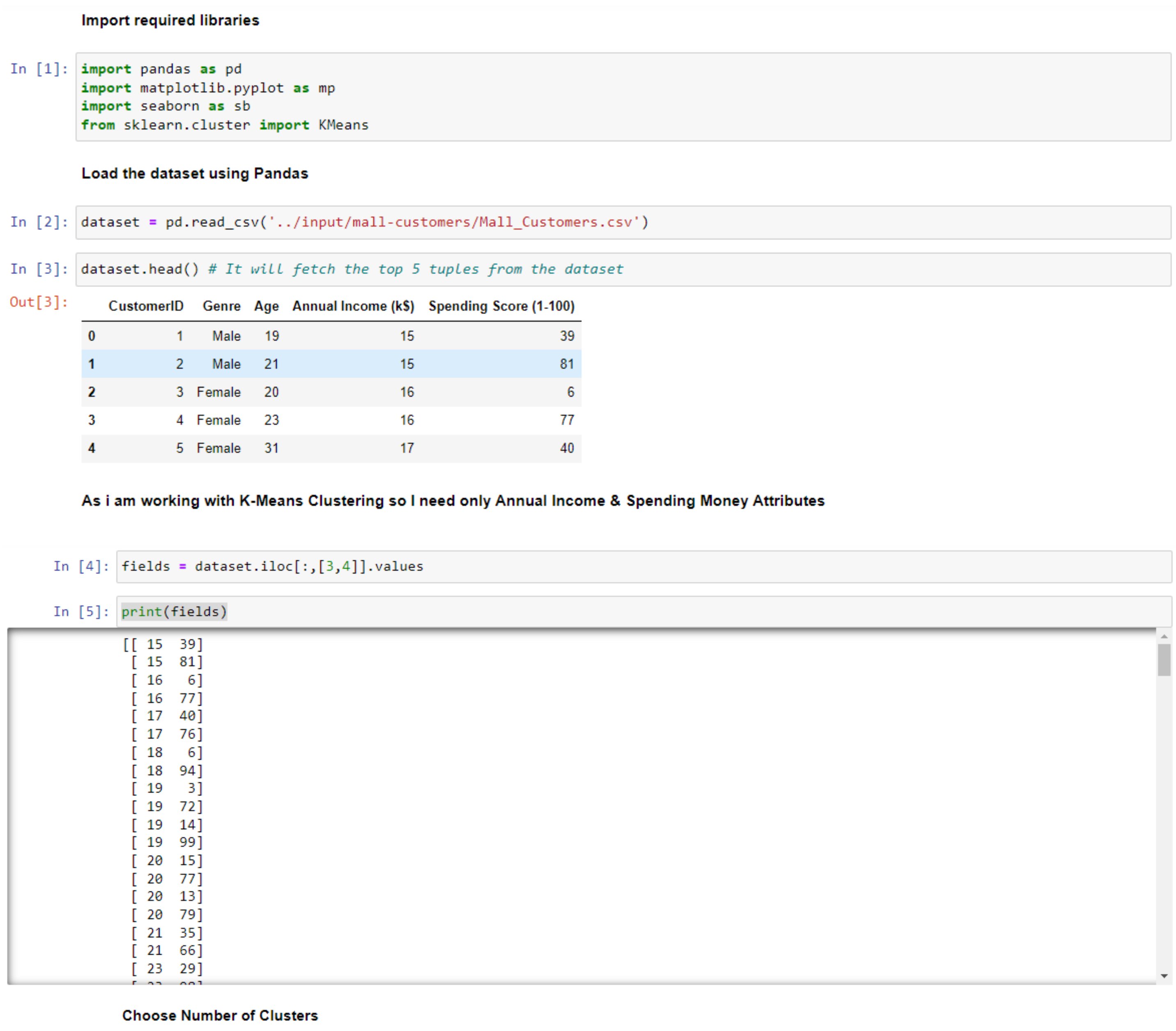
Advantages of K-means clustring

- Easy to implement.
- With a large number of variables, k-means may be computational faster than hierarchical clustring (if k is small).
- K-means may produce higher clusters than hierarchical clustering.

Disadvatnages of k-means clustring

- Difficult to predict the number of clusters (k-value).
- Initial seeds have a string impact on the final results.

Python Implementation of K-means Clustering Algorithm



Import required libraries

```
In [1]: import pandas as pd
        import matplotlib.pyplot as mp
        import seaborn as sb
        from sklearn.cluster import KMeans
```

Load the dataset using Pandas

```
In [2]: dataset = pd.read_csv('../input/mall-customers/Mall_Customers.csv')
```

```
In [3]: dataset.head() # It will fetch the top 5 tuples from the dataset
```

Out[3]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

As i am working with K-Means Clustering so I need only Annual Income & Spending Money Attributes

```
In [4]: fields = dataset.iloc[:,[3,4]].values
```

```
In [5]: print(fields)
```

```
[[ 15  39]
 [ 15  81]
 [ 16   6]
 [ 16  77]
 [ 17  40]
 [ 17  76]
 [ 18   6]
 [ 18  94]
 [ 19   3]
 [ 19  72]
 [ 19  14]
 [ 19  99]
 [ 20  15]
 [ 20  77]
 [ 20  13]
 [ 20  79]
 [ 21  35]
 [ 21  66]
 [ 23  29]
```

Choose Number of Clusters

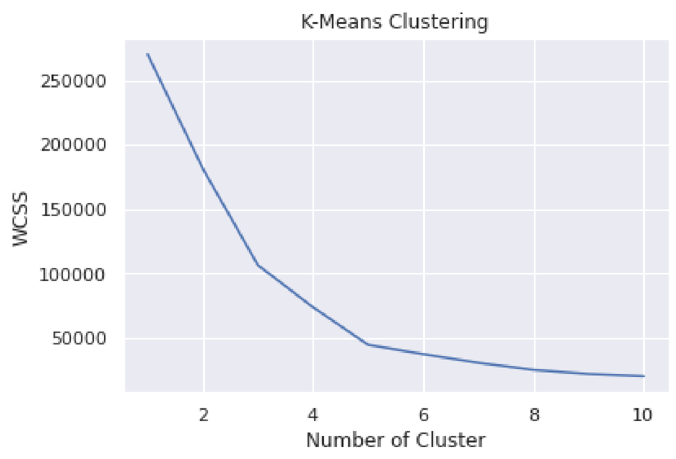
Finding the optimal number of clusters is an important part of this algorithm. A commonly used method for finding optimal K value is Elbow Method.

In the Elbow method, we are actually varying the number of clusters (K) from 1 – 10. For each value of K, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease. WCSS value is largest when K = 1. When we analyze the graph we can see that the graph will rapidly change at a point and thus creating an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value corresponding to this point is the optimal K value or an optimal number of clusters.

```
In [6]: wcss = [] # empty list
        for index in range(1,11): # range 1 to 10 implies that between this range any no. of clusters can be formed
            kmeans = KMeans(n_clusters = index, init = "k-means++", random_state = 2)
            kmeans.fit(fields)

            wcss.append(kmeans.inertia_)
```

**Let's plot the Elbow Method so that i can make out how many clusters will be formed **

```
In [7]: sb.set()
        mp.plot(range(1,11), wcss)
        mp.title("K-Means Clustering")
        mp.xlabel("Number of Cluster")
        mp.ylabel("WCSS")

Out[7]: Text(0, 0.5, 'WCSS')
```



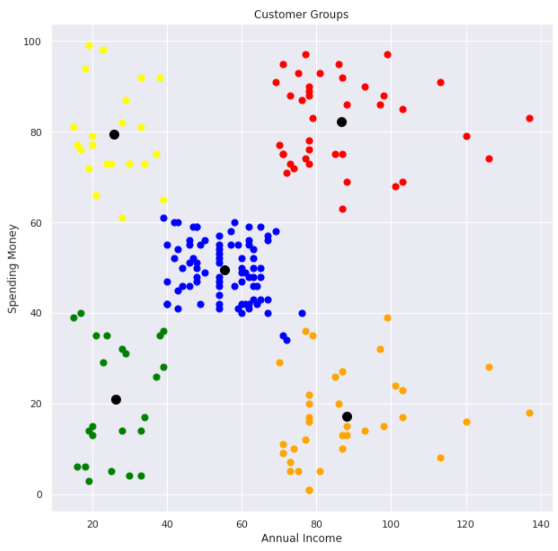
After plotting the Elbow Method , i have make out that 5 clusters will be formed based on the provided datasets

```
In [8]: kmeans = KMeans(n_clusters=5, init="k-means++",random_state=42)

        cluster_values = kmeans.fit_predict(fields)
        print(cluster_values)

        [2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
         3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4
         1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
         4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]
```

Now, Let's Visualize all the clusters

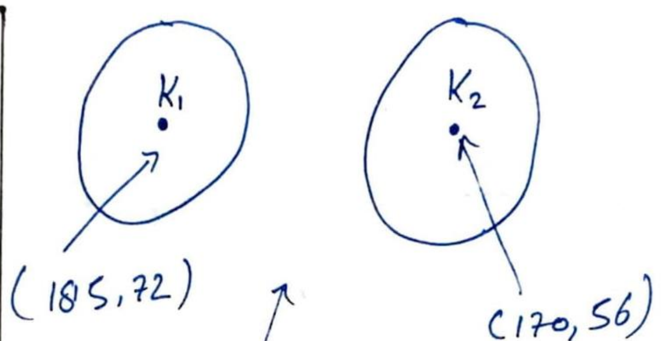
```
In [9]: mp.figure(figsize=(10,10))
        mp.scatter(fields[cluster_values==0,0], fields[cluster_values==0,1], s=50, c='blue', label = "Cluster 1")
        mp.scatter(fields[cluster_values==1,0], fields[cluster_values==1,1], s=50, c='orange', label = "Cluster 1")
        mp.scatter(fields[cluster_values==2,0], fields[cluster_values==2,1], s=50, c='green', label = "Cluster 1")
        mp.scatter(fields[cluster_values==3,0], fields[cluster_values==3,1], s=50, c='yellow', label = "Cluster 1")
        mp.scatter(fields[cluster_values==4,0], fields[cluster_values==4,1], s=50, c='red', label = "Cluster 1")

        mp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], s=100, c="black", label = "Centroids")

        mp.title("Customer Groups")
        mp.xlabel("Annual Income")
        mp.ylabel("Spending Money")
```

```
Out[9]: Text(0, 0.5, 'Spending Money')
```



K-means Algo-

	x Height	y Weight
1.	185	72
2.	170	52
3.	168	60
4.	179	68
5.	182	72
6.	188	77
7.	180	71
8.	180	70
9.	183	84
10.	180	88
11.	180	67
12.	177	76



→ There are centroid value

- And other 10 values are observed value

Euclidean Distance =

$$\sqrt{(x_o - x_c)^2 + (y_o - y_c)^2}$$

$$\boxed{\begin{array}{l} o = \text{observed value} \\ c = \text{centroid value} \end{array}}$$

ED for 3 → $K_1 = \sqrt{(168-185)^2 + (60-72)^2}$

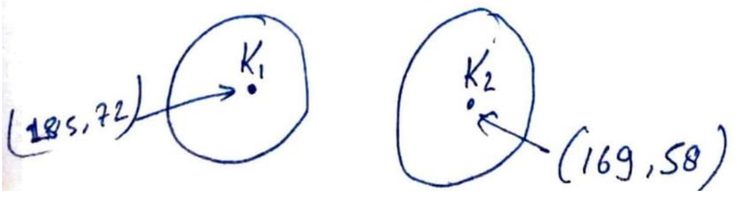
$= 20.80$

→ $K_2 = \sqrt{(168-170)^2 + (60-56)^2}$

$= 4.48$

Next Step → New Centroid Calculation:-

for $K_2 = \left(\frac{170+168}{2}, \frac{60+56}{2}\right) = (169, 58)$



ED for 4 $\rightarrow K_1 = \sqrt{(179-185)^2 + (68-72)^2}$

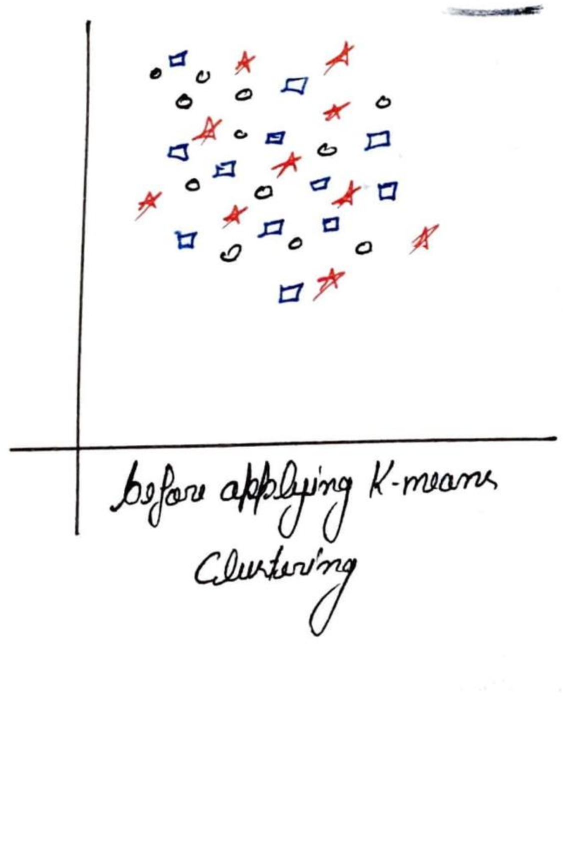
$= (6.32)$

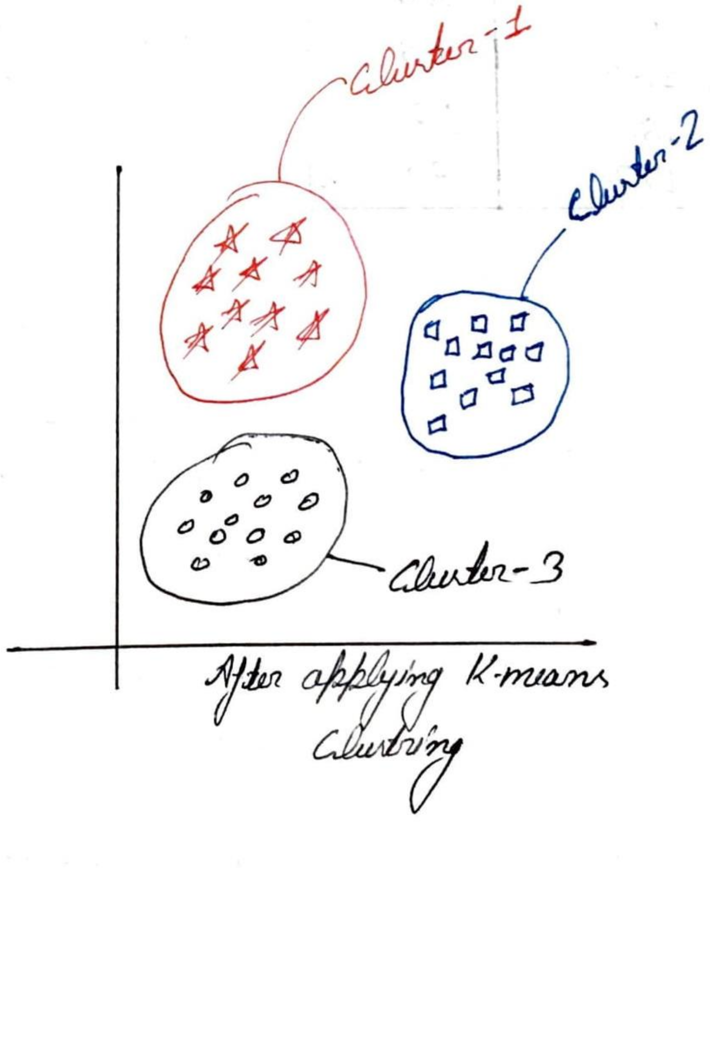
$\rightarrow K_2 = \sqrt{(179-169)^2 + (68-58)^2}$

$= 14.14$

Now, 4th value gone to the K_1, bcoz in K_1 has less distance comparision to K_2.

Explain throw Graph :


before applying K-means Clustering


Cluster-1
Cluster-2
Cluster-3
After applying K-means Clustering