**Name:- Khalfe Aynas Roll No:- 14CO12**

**Exp No:- 05 Batch No:- 01**

**Aim:-** Implementation of Clustering algorithms K-means/ Agglomerative.

**Theory:-**

## **What is Clustering?**

Clustering is dividing data points into homogeneous classes or clusters:

* Points in the same group are as similar as possible.
* Points in different group are as dissimilar as possible.

When a collection of objects is given, we put objects into group based on similarity.

## **Clustering Algorithms:**

A Clustering Algorithm tries to analyse natural groups of data on the basis of some similarity. It locates the centroid of the group of data points. To carry out effective clustering, the algorithm evaluates the distance between each point from the centroid of the cluster.

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data.

## **means Clustering Method:**

## K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

## 

**Algorithm:-**

If k is given, the K-means algorithm can be executed in the following steps:

* Partition of objects into k non-empty subsets
* Identifying the cluster centroids (mean point) of the current partition.
* Assigning each point to a specific cluster
* Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
* After re-allotting the points, find the centroid of the new cluster formed.

**Program:-**

**#means\_clustering.py**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#importing the dataset

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

#using the elbow method to find the optimal number of clusters

from sklearn.cluster import KMeans

wcss =[]

for i in range(1, 11):

kmeans=KMeans(n\_clusters = i, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11),wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

#applying kmeans to all dataset

kmeans=KMeans(n\_clusters = 5, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0)

y\_kmeans = kmeans.fit\_predict(X)

#visualising the clusters

plt.scatter(X[y\_kmeans == 0, 0],X[y\_kmeans == 0,1],s = 100, c = 'red', label = 'Careful')

plt.scatter(X[y\_kmeans == 1, 0],X[y\_kmeans == 1,1],s = 100, c = 'blue', label = 'Standard')

plt.scatter(X[y\_kmeans == 2, 0],X[y\_kmeans == 2,1],s = 100, c = 'green', label = 'Target')

plt.scatter(X[y\_kmeans == 3, 0],X[y\_kmeans == 3,1],s = 100, c = 'cyan', label = 'Careless')

plt.scatter(X[y\_kmeans == 4, 0],X[y\_kmeans == 4,1],s = 100, c = 'magenta', label = 'Sensible')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:,1],s = 300, c = 'yellow',label = 'Centroids')

plt.title('Clusters of Clients')

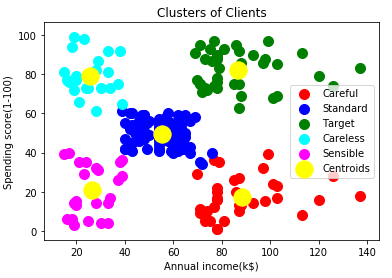
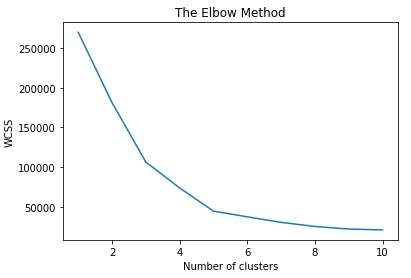
plt.xlabel('Annual income(k$)')

plt.ylabel('Spending score(1-100)')

plt.legend()

plt.show()

**Output:-**



**Conclusion:-**

In this experiment, we have seen how k-means algorithm works.