Aim: To Study and Implement K Means algorithm

Objective:- Understand the working of K Means algorithm and it's implemention using python.

Theory:

In statistics and machine learning, k means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Input

K:-number of clusters

D:- data set containing n objects

Output

A set of k clusters

Given k , the k-means algorithm is implemented in 5 steps:

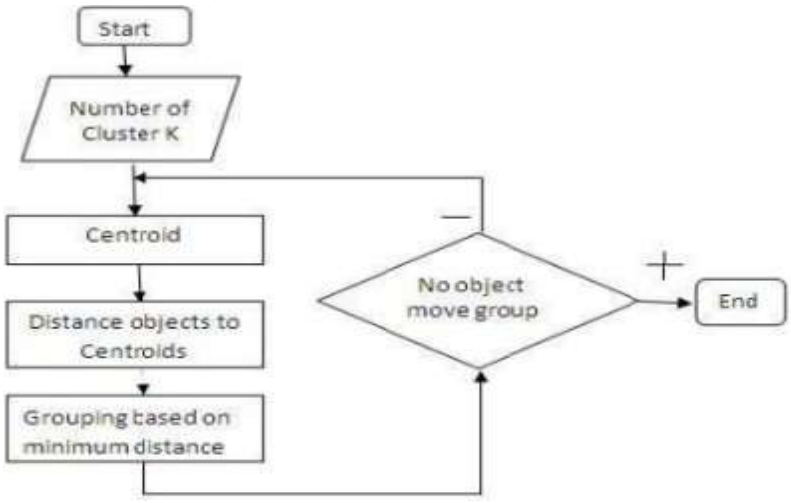
Step 1: Arbitrarily choose k objects from D as the initial cluster centers.

Step 2: Find the distance from each and every object in the dataset with respect to cluster centers

Step 3: Assign each object to the cluster with the nearest seed point based on the mean value of the objects in the cluster.

Step 4: Update the cluster means i,e calculate the mean value of the objects for each cluster.

Step 5: Repeat the procedure, until there is no change in meaning.



Example: d= {2,4,10,12,3,20,30,11,25} k =2

1. Randomly assign mean m1=3 and m2 = 4

Therefore, k1 = {2,3} Therefore, k1 = {4,10,12,20,30,11,25}

2. Randomly assign mean m1=2.5 and m2 = 16

Therefore, k1 = {2,3,4} Therefore, k1 =

{4,10,12,20,30,11,25}

3. Randomly assign mean m1=3 and m2 = 18

Therefore, k1 = {2,3,4,10} Therefore, k1 = {12,20,30,11,25}

4. Randomly assign mean m1=7 and m2 = 25

Therefore, k1 = {2,3,4,10,11,12} Therefore, k1 =

{20,30,25}

5. Randomly assign mean m1=7 and m2 = 25

Therefore, we stop as we are getting same mean

values.

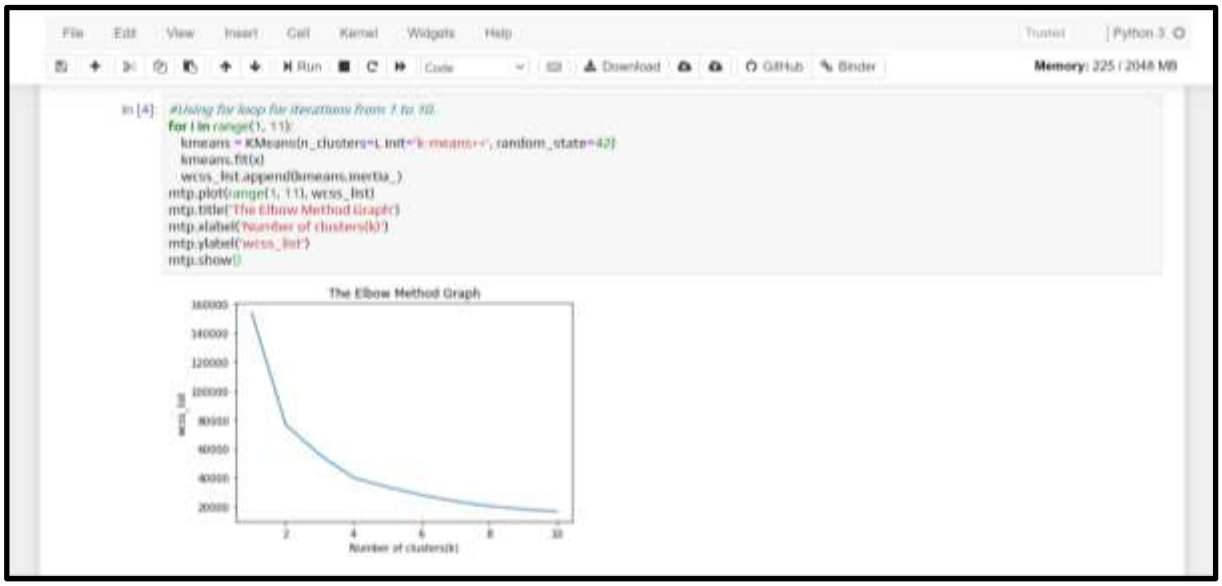
6. Therefore, Final clusters are : k1 = {2,3,4,10,11,12} Therefore, k1 = {20,30,25}

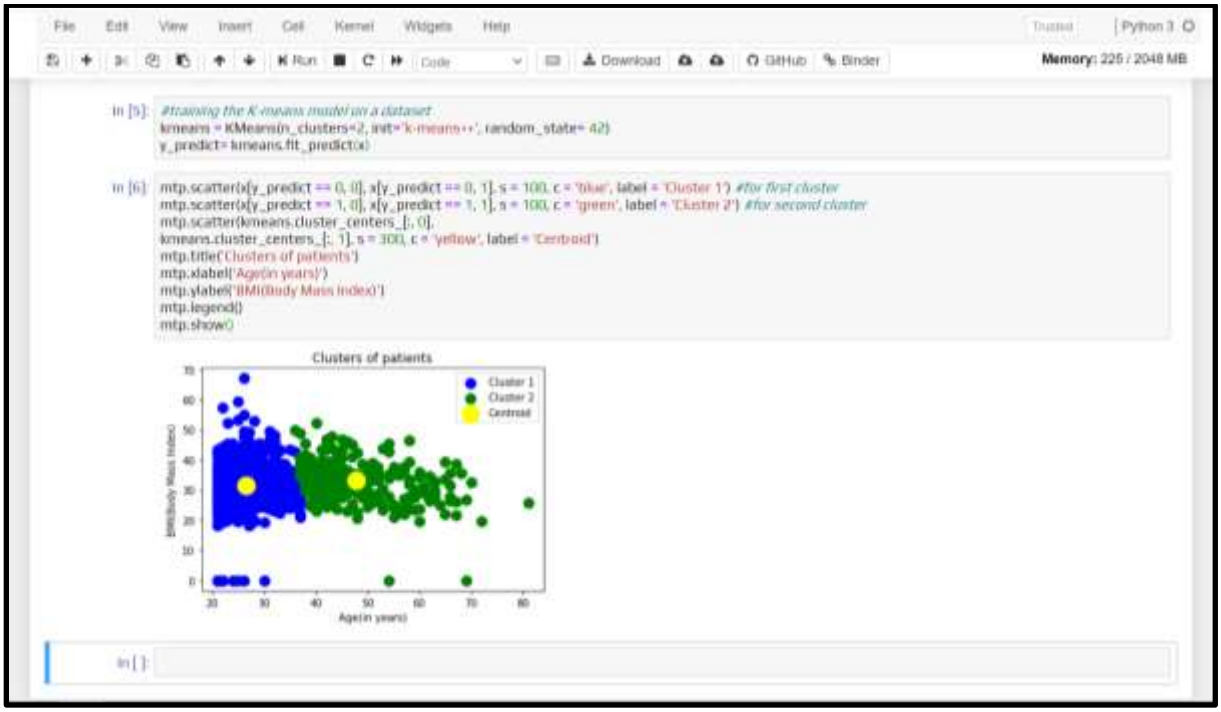
CODE:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('diabetes_csv.csv')
x = dataset.iloc[:, [7, 5]].values

#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 a loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elbow Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()

#training the K-means model on a dataset
kmeans = KMeans(n_clusters=2, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for
first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for
second cluster
mtp.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of patients')
mtp.xlabel('Age(in years)')
mtp.ylabel('BMI(Body Mass Index)')
mtp.legend()
mtp.show()
```

OUTPUT:


File Edit View Insert Cell Kernel Widgets Help
Run Code Download GitHub Binder
Memory: 225 / 2048 MB
In [1]:
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
In [2]:
# importing the dataset
dataset = pd.read_csv('diabetes_csv.csv')
x = dataset.iloc[:, [7, 5]].values
In [3]:
#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= [] #Initializing the list for the values of WCSS


File Edit View Insert Cell Kernel Widgets Help
Run Code Download GitHub Binder
Memory: 325 / 2048 MB
In [4]:
#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
kmeans.fit(x)
wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elbow Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
The Elbow Method Graph
wcss_list
Number of clusters(k)



Sample Dataset

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	tested_negative
764	2	122	70	27	0	36.8	0.340	27	tested_negative
765	5	121	72	23	112	26.2	0.245	30	tested_negative
766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

768 rows × 9 columns

CONCLUSION:

Advantages of K-means clustering:

1. Relatively simple to implement.
2. Scales to large data sets.
3. Guarantees convergence.
4. Can warm-start the positions of centroids.
5. Easily adapts to new examples.
6. Generalizes to clusters of different shapes and sizes, such as elliptical clusters.