Name: Sudhanshu Narsude

Subject: AI/ML

UID: 2019130044

Branch: TE Comps

Batch: C

Experiment 5

Aim: Given a dataset, classify the input into k categories.

Problem Statement: In the given dataset, classify the dataset into k number of categories.

Theory:

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.

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.

Code:

[1]

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn.cluster import KMeans
```

[2]

```
dataset = pd.read_csv('Mall_Customers_data.csv')
```

[3]

```
x = dataset.iloc[:, [3, 4]].values
dataset.head(200)
```

Output:

	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
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

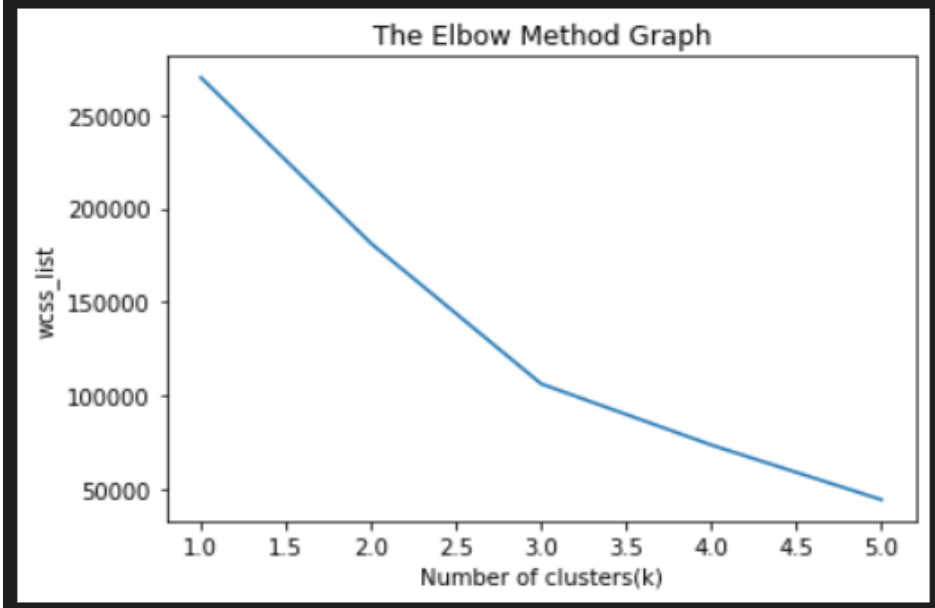
200 rows × 5 columns

[4]

```
#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 1 to 10.
for i in range(1, 6):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 50)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
[print([i+1,wcss_list[i]]) for i in range(5)]
mtp.plot(range(1, 6), wcss_list)
mtp.title('The Elbow Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
```

Output:

```
[1, 269981.28]
[2, 181363.59595959593]
[3, 106348.37306211118]
[4, 73679.78903948836]
[5, 44448.45544793371]
```



[5]

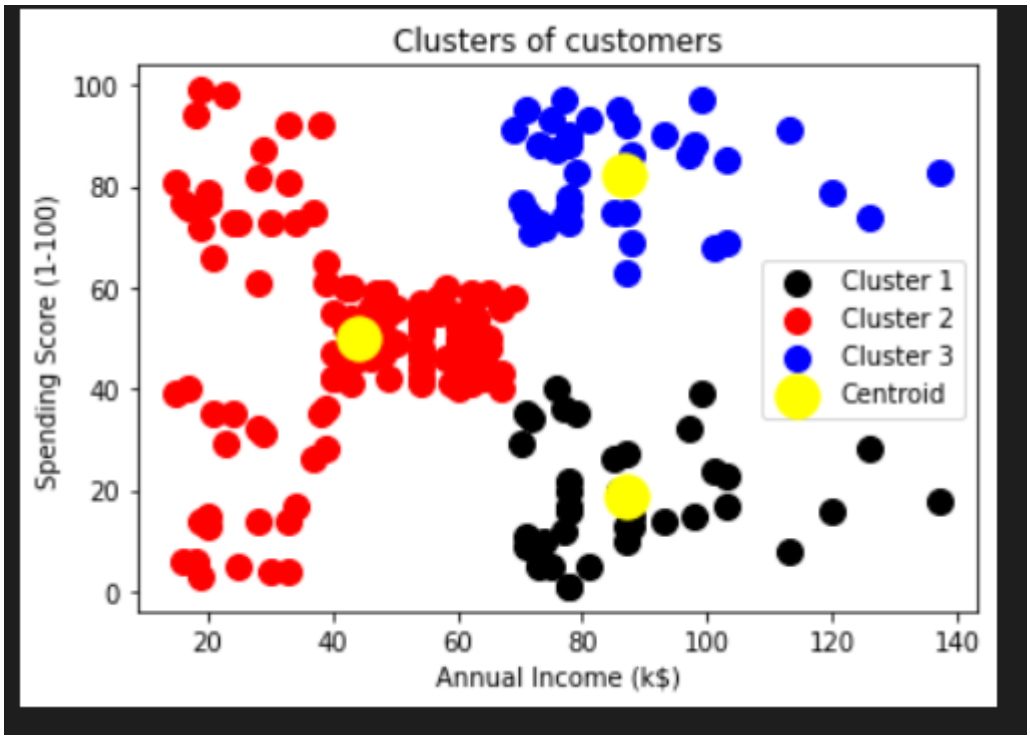
```
#From the above plot, we can see the elbow point is at 3. So the number of clusters here will be 3
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=3, init='k-means++', random_state= 50)
y_predict= kmeans.fit_predict(x)
```

[6]

```
#visulaizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'black', label = 'Cluster 1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'red', label = 'Cluster 2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'blue', label = 'Cluster 3') #for third cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
```

```
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output:



[7]

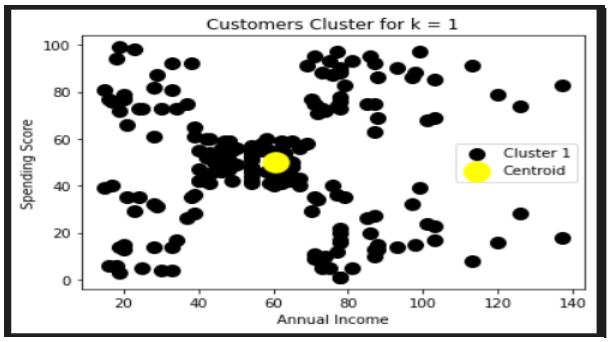
```
def clustering_kmeans(X,k):
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state= 50)
    y= kmeans.fit_predict(X)
    return kmeans,y

# Using the Elbow method the optimal value of cluster(k) is 3 for the given dataset
k_means, y = clustering_kmeans(x, 3)
```

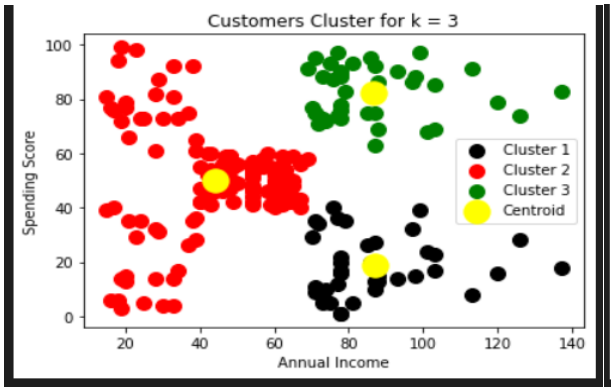
[8]

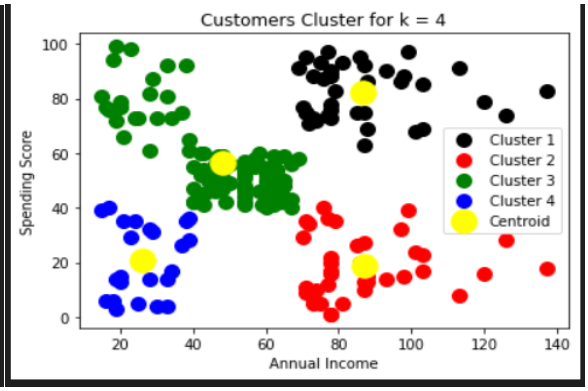
```
for i in range(5) :
    k_means, y = clustering_kmeans(x, i+1)
    mtp.scatter(x[y == 0, 0], x[y == 0, 1], s = 100, c = 'black', label = 'Cluster 1') #for first cluster
    if i>0:
        mtp.scatter(x[y == 1, 0], x[y == 1, 1], s = 100, c = 'red', label = 'Cluster 2') #for second cluster
    if i>1:
        mtp.scatter(x[y== 2, 0], x[y == 2, 1], s = 100, c = 'green', label = 'Cluster 3') #for third cluster
    if i>2:
        mtp.scatter(x[y == 3, 0], x[y == 3, 1], s = 100, c = 'blue', label = 'Cluster 4') #for fourth cluster
    if i>3:
        mtp.scatter(x[y == 4, 0], x[y == 4, 1], s = 100, c = 'orange', label = 'Cluster 5') #for fifth cluster
```

```
    mtp.scatter(k_means.cluster_centers_[:, 0], k_means.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroid')
    mtp.title('Customers Cluster for k = ' + str(i+1))
    mtp.xlabel('Annual Income')
    mtp.ylabel('Spending Score')
    mtp.legend()
    mtp.show()
```

Output:







Conclusion:

In this experiment of AIML lab, I learnt about one of the algorithm used in unsupervised learning i.e. K-means algorithm . I successfully implemented K-means algorithm on 2D dataset.