Experiment-5

Yash Patel
2019130047
Batch C
TE Comps

Aim:
To train and test machine learning models using K-Means Clustering Algorithm.

Theory:

- K-Means Clustering is an unsupervised learning algorithm used in machine learning and data science to handle clustering problems. It divides the unlabelled data into many clusters. K specifies the number of predetermined clusters that must be produced during the procedure; for example, if K=2, two clusters will be created, and if K=3, three clusters will be created, and so on.
- 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 different 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 assign 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:

```
# %%
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline


# %%
df = pd.read_csv("Mall_Customers.csv")
df


# %%
```

```
plt.scatter(df["Annual Income (k$)"], df["Spending Score (1-100)"])
plt.show()

# %%
X = df.iloc[:, [3, 4]]
X

# %%
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(X[["Annual Income (k$)"]])
X["Annual Income (k$)"] = scaler.transform(X[["Annual Income (k$)"]])

scaler.fit(X[["Spending Score (1-100)"]])
X["Spending Score (1-100)"] = scaler.transform(X[["Spending Score
(1-100)"]])
X

# %%
from sklearn.cluster import KMeans

sse = []
k_rng = range(1, 11)
for k in k_rng:
 k_means = KMeans(n_clusters=k)
 k_means.fit(X)
 sse.append(k_means.inertia_)

sse

# %%
plt.scatter(k_rng, sse)
plt.plot(k_rng, sse)
```

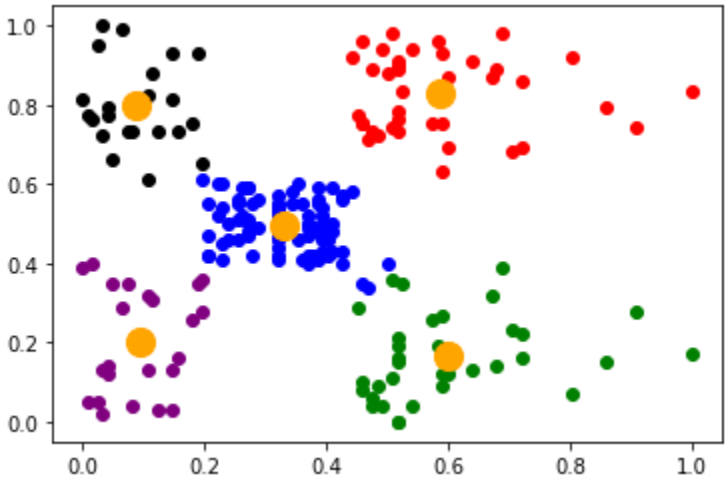
```
# %%
# from the graph we can see that the optimal number of cluster will be 5
K_Means = KMeans(n_clusters=5)
y_predicted = K_Means.fit_predict(X[["Annual Income (k$)", "Spending Score
(1-100)"]])
y_predicted

# %%
X['cluster'] = y_predicted
X

# %%
plt.scatter(X.iloc[y_predicted == 0, 0], X.iloc[y_predicted == 0, 1],
c="red", label="Cluster 1")
plt.scatter(X.iloc[y_predicted == 1, 0], X.iloc[y_predicted == 1, 1],
c="blue", label="Cluster 2")
plt.scatter(X.iloc[y_predicted == 2, 0], X.iloc[y_predicted == 2, 1],
c="black", label="Cluster 3")
plt.scatter(X.iloc[y_predicted == 3, 0], X.iloc[y_predicted == 3, 1],
c="purple", label="Cluster 4")
plt.scatter(X.iloc[y_predicted == 4, 0], X.iloc[y_predicted == 4, 1],
c="green", label="Cluster 5")
plt.scatter(K_Means.cluster_centers_[:, 0], K_Means.cluster_centers_[:,
1], s=200, c="orange", label="Cluster Centers")

# %%
```

Output:



Conclusion:

- I acquired the basics of the K-Means method from the above experiment. It's a centroid-based approach, which means that each cluster has its own centroid.
- The main goal of this technique is to reduce the sum of distances between data points and the clusters that they belong to.
- It uses an iterative procedure to find the best value for K centre points or centroids, and then allocates each data point to the closest k-centre. A cluster is formed by data points that are close to a specific K-center.
- The algorithm takes an unlabeled dataset as input, separates it into k-number of clusters, and continues the procedure until no better clusters are found. In this algorithm, the value of k should be predetermined.
- The algorithm's accuracy varies depending on the number of clusters picked.