Experiment No. 6
Implement clustering algorithm (Kmeans)
Date of Performance: 03/09/25
Date of Submission: 10/09/25
Name: Sumit Metkari
Div/Roll no.: 2 / 32

Aim: To implement K-Means algorithm

Objective: Develop a program to implement K-Means Algorithm

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:

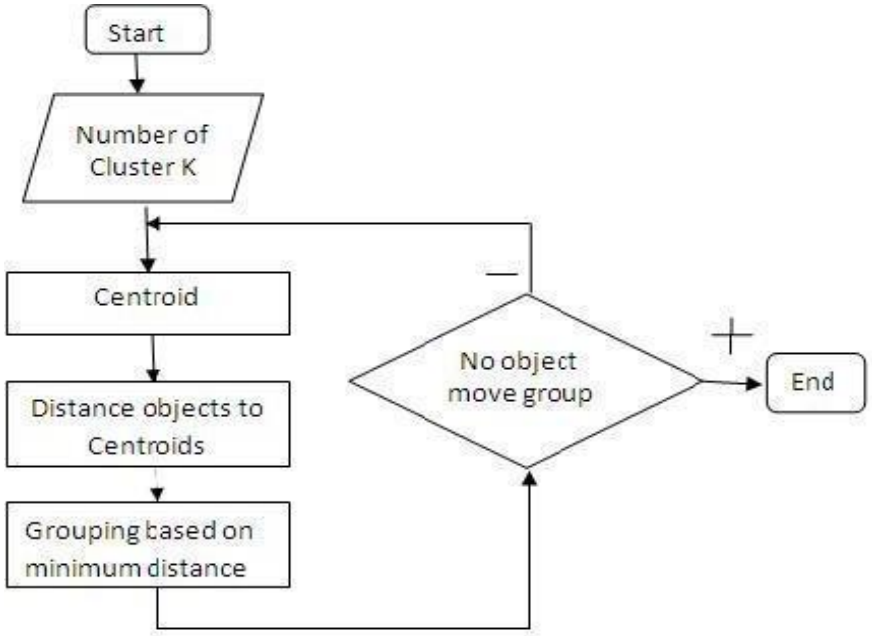
Step 1: Arbitrarily choose k objects from D as the initial cluster centers.

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

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 mean.



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

- Randomly assign mean m1=3 and m2 = 4
- Therefore, k1 = {2,3} and k2 = {4,10,12,20,30,11,25}
- Calculate mean m1=2.5 and m2 = 16 Therefore, k1 = {2,3,4} and k2 = {4,10,12,20,30,11,25}
- Calculate mean m1=3 and m2 = 18
- Therefore, k1 = {2,3,4,10} and k2 = {12,20,30,11,25}
- Calculate mean m1=7 and m2 = 25 Therefore, k1 = {2,3,4,10,11,12} Therefore, k2 = {20,30,25}
- Calculate mean m1=7 and m2 = 25 Therefore, we stop as we are getting same mean values.
- Therefore, Final clusters are : k1 = {2,3,4,10,11,12} Therefore, k2 = {20,30,25}

Code and Output:

[1] ✓ 2s

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[2] ✓ 0s

```
try:
    dataset = pd.read_csv('IRIS.csv')
    print("Dataset loaded successfully!")
except FileNotFoundError:
    print("Error: 'Iris.csv' not found. Please upload the dataset to your Colab environment.")

    exit()
```

```
Dataset loaded successfully!
```

[3] ✓ 0s

```
X = dataset.iloc[:, 0:4].values


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("\nFirst 5 rows of scaled data:")
print(X_scaled[:5])
```

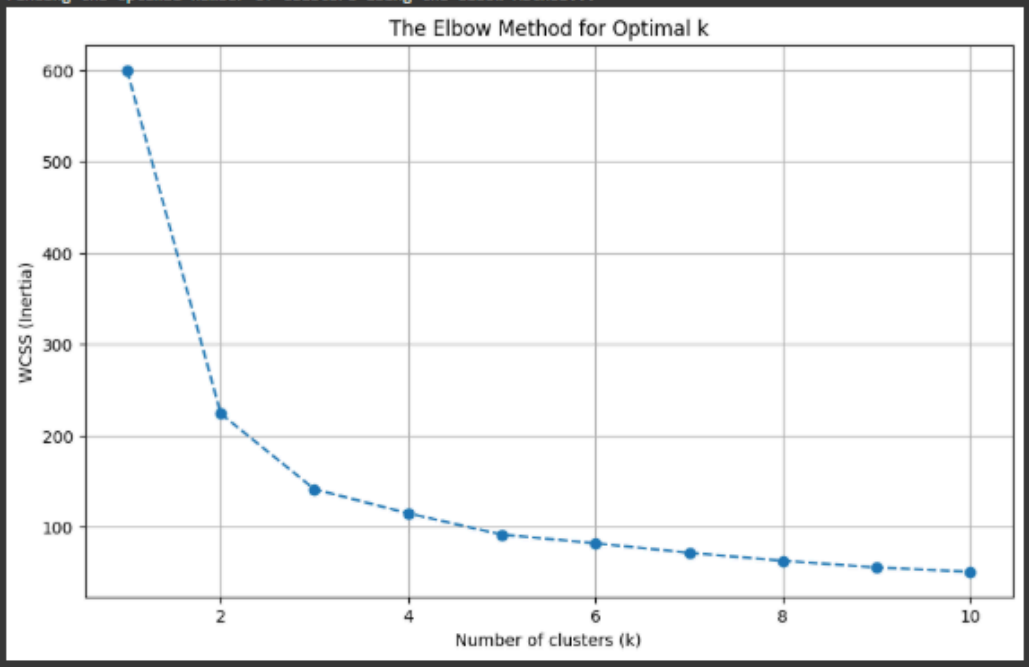
```
First 5 rows of scaled data:
[[-0.90068117  1.03205722 -1.3412724  -1.31297673]
 [-1.14301691 -0.1249576  -1.3412724  -1.31297673]
 [-1.38535265  0.33784833 -1.39813811 -1.31297673]
 [-1.50652052  0.10644536 -1.2844067  -1.31297673]
 [-1.02184904  1.26346019 -1.3412724  -1.31297673]]
```

[4] ✓ 0s

```
print("\nFinding the optimal number of clusters using the Elbow Method...")
wcss = []
for i in range(1, 11):

    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)


plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('The Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS (Inertia)')
plt.grid(True)
```

```
optimal_k = 3
print(f"\nThe optimal number of clusters is {optimal_k}.")
```

```
Finding the optimal number of clusters using the Elbow Method...
```




```
The optimal number of clusters is 3.
```

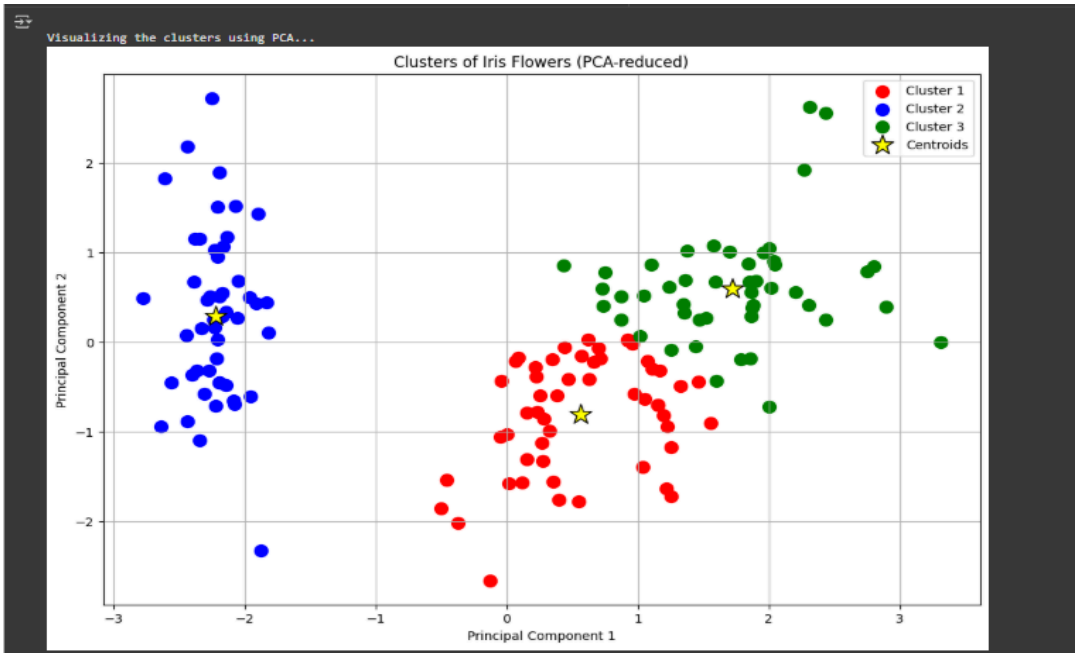
[5]
✓ 0s

```
print(f"Training K-Means model with {optimal_k} clusters...")
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42, n_init=10)
y_kmeans = kmeans.fit_predict(X_scaled)


print("\nCluster assignments for the first 10 data points:")
print(y_kmeans[:10])
```

```
Training K-Means model with 3 clusters...

Cluster assignments for the first 10 data points:
[1 1 1 1 1 1 1 1 1 1]
```

[6]
✓ 0s

```
print("\nVisualizing the clusters using PCA...")
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)


centroids = kmeans.cluster_centers_
centroids_pca = pca.transform(centroids)

plt.figure(figsize=(12, 8))
colors = ['red', 'blue', 'green', 'cyan', 'magenta', 'orange']

for i in range(optimal_k):
    plt.scatter(X_pca[y_kmeans == i, 0], X_pca[y_kmeans == i, 1],
                s=100, c=colors[i], label=f'Cluster {i + 1}')


plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1],
            s=300, c='yellow', marker='*', edgecolor='black', label='Centroids')

plt.title('Clusters of Iris Flowers (PCA-reduced)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid(True)
plt.show()
```



Conclusion:

The clustering algorithms successfully group data points into distinct clusters based on shared characteristics. Each cluster shows internal cohesion, with data points being similar to one another. Differences between clusters highlight underlying variations in the data. Assessing the quality of clusters involves evaluating cohesion, separation, and identifying outliers for better understanding.