Experiment No.7

Title:Implementation of K-Means Clustering Algorithm on appropriate dataset

Part A:Implementation of K-Means Clustering Algorithm on Synthetic data

Import Libraries

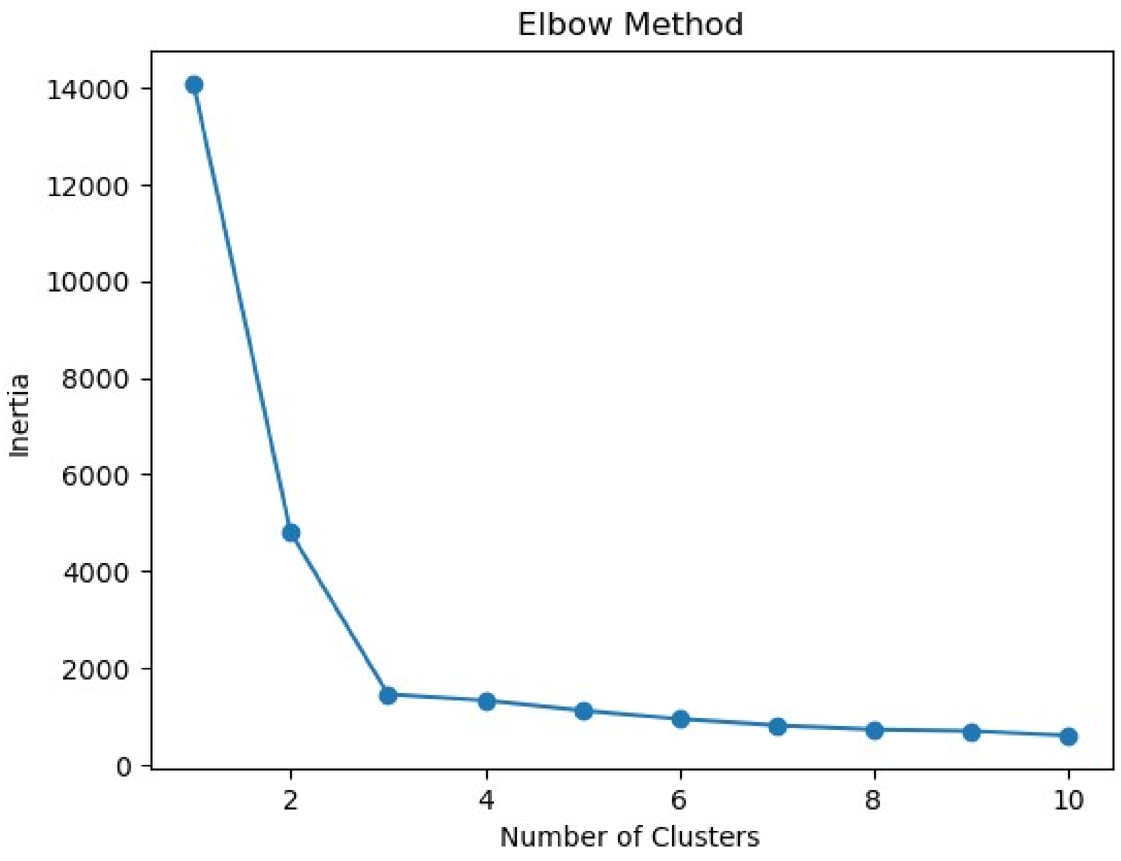
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
```

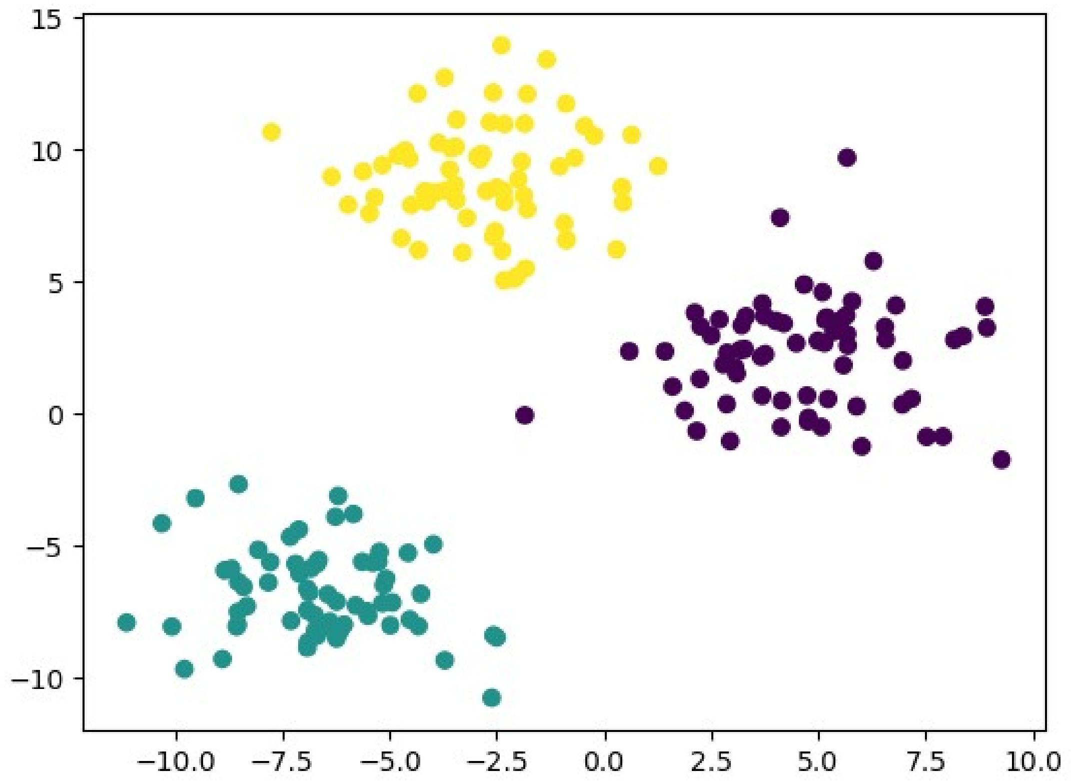
Creating Synthetic data

```
df=make_blobs(n_samples=200,n_features=2,centers=3,cluster_std=2,rando
m_state=42)
data=list(zip(df[0][:,0],df[0][:,1]))
```

Finding the optimal no. of clusters for given data

```
inertias=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)
plt.plot(range(1,11),inertias,'-o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()

kmeans=KMeans(n_clusters=3)
kmeans.fit(data)

plt.scatter(df[0][:,0],df[0][:,1],c=kmeans.labels_)
plt.show()
```


Elbow Method
Inertia
Number of Clusters
0
2000
4000
6000
8000
10000
12000
14000
2
4
6
8
10


15
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

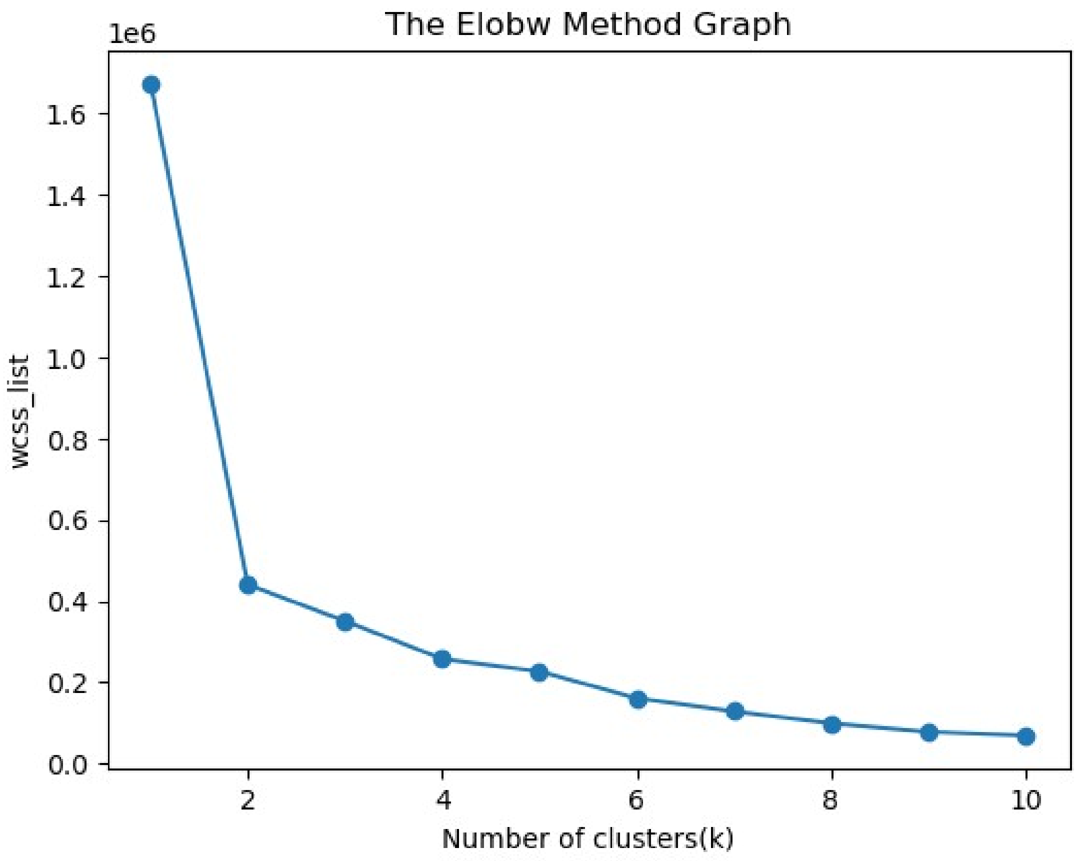
Part B:Implementation of K-Means Clustering Algorithm on a dataset

```
#Importing dataset
df=pd.read_csv('archive/Mall_Customers.csv')
df.head()
```

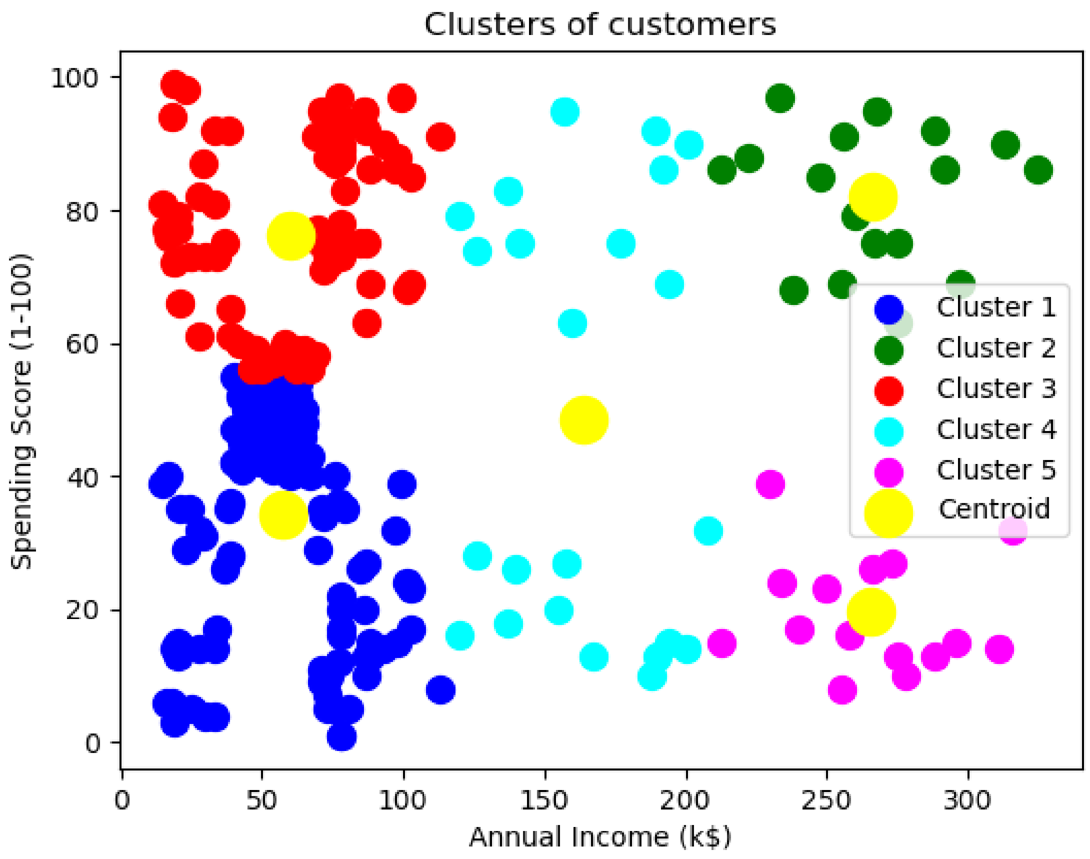
```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

```
x = df.iloc[:, [3, 4]].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 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_)
plt.plot(range(1, 11), wcss_list,'-o')
plt.title('The Elobw Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```



```
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)

#visulaizing the clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c =
'blue', label = 'Cluster 1') #for first cluster
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c =
'green', label = 'Cluster 2') #for second cluster
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c =
'red', label = 'Cluster 3') #for third cluster
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c =
'cyan', label = 'Cluster 4') #for fourth cluster
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c =
'magenta', label = 'Cluster 5') #for fifth cluster
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 300, c = 'yellow', label = 'Centroid')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



Conclusion

Merits and Demerits of K-Means Clustering Algorithm

Merits of K-Means

1. **Simplicity and Ease of Implementation**:
   - K-Means is straightforward to understand and implement, making it a popular choice for clustering tasks.
2. **Efficiency**:
   - The algorithm is computationally efficient, especially with a small number of clusters, as its time complexity is generally O(n * k * i), where n is the number of data points, k is the number of clusters, and i is the number of iterations.
3. **Scalability**:
   - K-Means can handle large datasets efficiently and can be easily scaled to a large number of samples.
4. **Works Well with Spherical Clusters**:
   - K-Means performs well when the clusters are spherical and equally sized, making it effective for certain types of data distributions.
5. **Easy to Interpret**:
   - The results of K-Means are easy to interpret, as the algorithm assigns cluster labels that can be directly analyzed.

6. **Online Clustering**:
   - Variants of K-Means allow for online clustering, which can update cluster centers incrementally as new data arrives.

Demerits of K-Means

1. **Choosing the Number of Clusters (k)**:
   - The requirement to specify the number of clusters in advance can be challenging, and the wrong choice can lead to poor clustering results.
2. **Sensitivity to Initial Conditions**:
   - K-Means can converge to different solutions based on the initial placement of centroids, leading to potential inconsistencies in clustering results. This issue can be mitigated using techniques like K-Means++ for better initialization.
3. **Assumption of Spherical Clusters**:
   - The algorithm assumes that clusters are spherical and of similar size, which can lead to poor performance on datasets with non-spherical or varying-sized clusters.
4. **Outliers and Noisy Data**:
   - K-Means is sensitive to outliers and noise, which can distort the placement of centroids and negatively impact clustering performance.
5. **Limited to Numerical Data**:
   - K-Means works primarily with numerical data and cannot handle categorical features directly, requiring additional preprocessing steps.
6. **Local Optima**:
   - The algorithm can converge to a local minimum, meaning it might not find the best overall solution. Running K-Means multiple times with different initializations can help address this issue.

K-Means clustering is a widely used algorithm due to its simplicity and efficiency. However, users should be aware of its limitations, particularly regarding the choice of the number of clusters, sensitivity to initial conditions, and its assumptions about data distribution. Proper preprocessing and experimentation can help mitigate some of these issues, making K-Means a valuable tool for exploratory data analysis and clustering tasks.