Ambertask

March 29, 2024

0.1 Importing Libraries

[ ]:

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd
```

0.2 Loading Dataset And Cleaning it

[ ]:

```
# Load the dataset
df = pd.read_csv('Mall_Customers.csv')

# Drop duplicates
df.drop_duplicates(inplace=True)

# Encode the categorical column 'Gender'
encoder = LabelEncoder()
gender_encoded = encoder.fit_transform(df['Gender'])
df['Gender'] = gender_encoded

# Replace missing values with the mean of the column
imputer = SimpleImputer(strategy='mean')
df = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

0.3 Normalization

[ ]:

```
# Normalize the data
scaler = StandardScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df[['Age', 'Annual Income␣
 ↪(k$)', 'Spending Score (1-100)']]), columns=['Age', 'Annual Income (k$)',␣
 ↪'Spending Score (1-100)'])
```

0.4 Extracting Features

[ ]:

```python
# Select the relevant features
X = df_normalized[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

# K-means clustering
kmeans = KMeans(n_clusters=3, n_init=10, max_iter=300, random_state=0)
kmeans.fit(X)
identified_clusters = kmeans.predict(X)

# Centroids
centroids = kmeans.cluster_centers_
print("Centroids:")
print(centroids)

# Getting the labels assigned to each data point
labels = kmeans.labels_
print("\nLabels:")
print(labels)
```

0.5 Visualization

[ ]:

```python
# Visualize the resulting clusters
plt.scatter(df['CustomerID'], labels, c=labels, cmap='viridis')
plt.xlabel('Customer ID')
plt.ylabel('Cluster Label')
plt.title('K-means Clustering of Customer Segments')
plt.show()

# Elbow Method to determine the optimal number of clusters
# sse = []
# for k in range(1, 11):
#     kmeans = KMeans(n_clusters=k,n_init=10, max_iter=300, random_state=0)
#     kmeans.fit(X)
#     sse.append(kmeans.inertia_)

# plt.plot(range(1, 11), sse)
# plt.title('Elbow Method')
# plt.xlabel('Number of Clusters')
# plt.ylabel('SSE')
# plt.show()
```

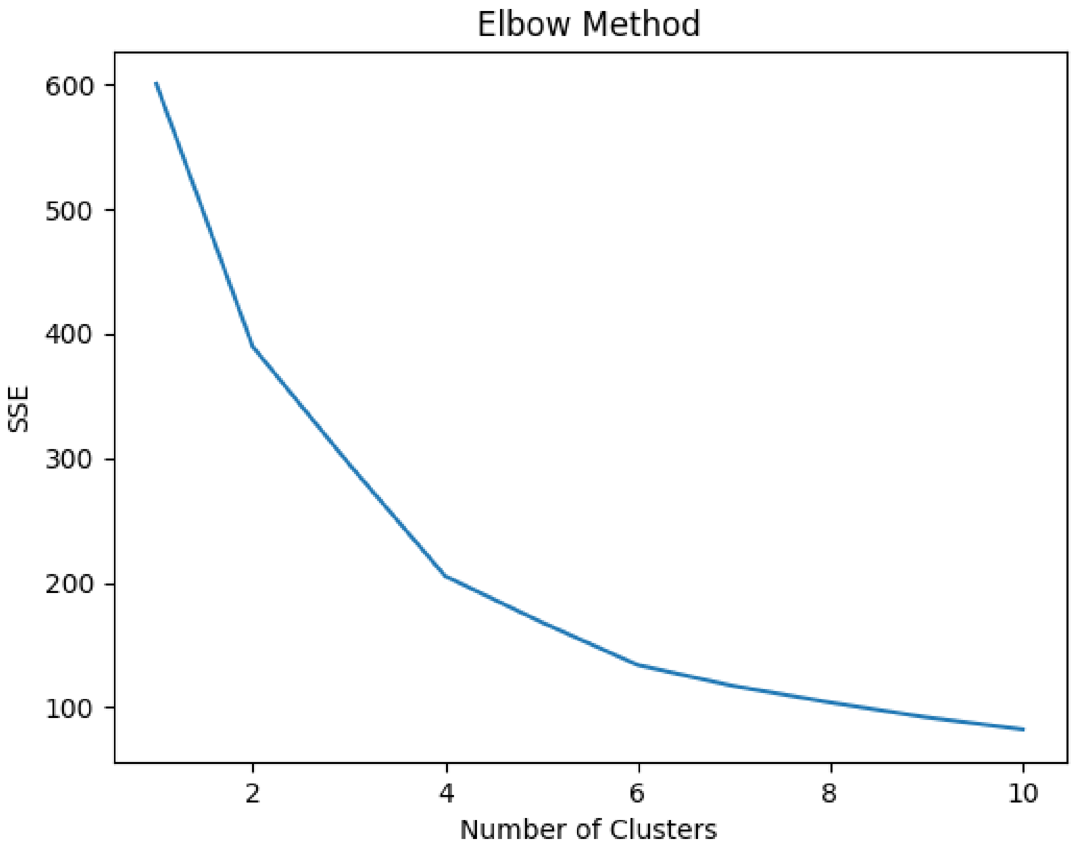
1 Question 2

[ ]:

```python
# Elbow Method to determine the optimal number of clusters
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k,n_init=10, max_iter=300, random_state=0)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)

plt.plot(range(1, 11), sse)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.show()

# Assuming the optimum number of clusters is 3 and you have new data 'new_data'
new_data = X
kmeans = KMeans(n_clusters=3, max_iter=300, n_init=10, random_state=0)
kmeans.fit(X)
labels = kmeans.predict(new_data)
print(labels)
```



```
[2 2 2 2 2 2 2 2 0 2 0 2 0 2 2 2 2 2 0 2 2 2 0 2 0 2 0 2 2 2 0 2 0 2 0 2 0
 2 2 2 0 2 0 2 0 2 0 2 2 2 0 2 2 0 0 0 0 0 2 0 0 2 0 0 0 2 0 0 2 2 0 0 0 0
 0 2 0 0 2 0 0 2 0 0 2 0 0 2 2 0 0 2 0 0 2 2 0 2 0 2 2 0 0 2 0 2 0 0 0 0 0
 2 0 2 2 2 0 0 0 0 2 0 1 1 2 1 0 1 0 1 0 1 2 1 2 1 0 1 2 1 0 1 2 1 2 1 0 1
 0 1 0 1 0 1 0 1 0 1 0 1 0 1 2 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1]
```

data_with_clusters = data.copy() data_with_clusters[‘Clusters’] = identified_clusters plt.scatter(data_with_clusters[‘Longitude’],data_with_clusters[‘Latitude’],c=data_with_clusters[‘Clusters’],cma