```
# !pip install pandas numpy matplotlib seaborn scikit-learn

#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler



#load the dataset
dataset_url="/content/customers.csv"
df = pd.read_csv(dataset_url)

#display the first few rows of the dataset
print("first 5 rows of the dataset:")
print(df.head())

#display basic information about the data set
print("\nMissing values in dataset:")
print(df.isnull().sum())
```

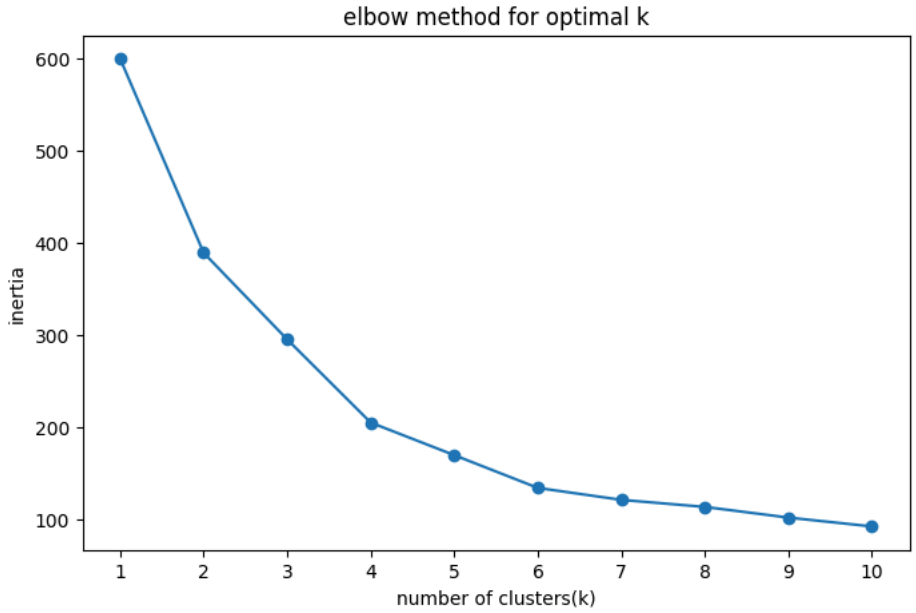
```
first 5 rows of the dataset:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16        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

Missing values in dataset:
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Generated code may be subject to a license | sarathshah/EV_Global-data

```
#select relevant columns(e.g, age, annual income, spending score)
features =df[["Age", "Annual Income (k$)", "Spending Score (1-100)"]]

#standardize the data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

#display the first few rows of the standardized data
print("\nFirst 5 rows of scaled features:")
print(scaled_features[:5])
```

```
First 5 rows of scaled features:
[[-1.42456879 -1.73899919 -0.43480148]
 [-1.28103541 -1.73899919  1.19570407]
 [-1.3528021  -1.70082976 -1.71591298]
 [-1.13750203 -1.70082976  1.04041783]
 [-0.56336851 -1.66266033 -0.39597992]]
```

```
#elbow mwthod to find the optimal number of clusters
inertia = []
k_range = range(1,11)

for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  kmeans.fit(scaled_features)
  inertia.append(kmeans.inertia_)

#plot the elbow method graph
plt.figure(figsize=(8,5))
plt.plot(k_range,inertia,marker='o')
plt.title('elbow method for optimal k')
plt.xlabel('number of clusters(k)')
plt.ylabel('inertia')
plt.xticks(k_range)
plt.show()
```




```
#perform k-means clustering with the optional
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
cluster_labels = kmeans.fit_predict(scaled_features)

#add cluster labels to the original dataset
df['cluster']=cluster_labels

#display the first few rows with cluster labels
print("\nFirst 5 rows with cluster labels:")
print(df.head())
```

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

```
# Visualize clusters (using the first two features for plotting)
plt.figure(figsize=(8, 6))
sns.scatterplot (x=scaled_features[:, 0], y=scaled_features[:, 1], hue=cluster_labels, palette='viridis',s=35)
plt.scatter (kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red', label='Centroids')
plt.title('Customer Segments')
plt.xlabel('Feature 1 (scaled)')
plt.ylabel('Feature 2 (scaled)')
plt.legend()
plt.show()
```


Customer Segments
Feature 2 (scaled)
3
2
1
0
−1
0
1
2
3
4
Centroids