untitled199

September 8, 2024

[1]:
```python
import numpy as np
```

[2]:
```python
import pandas as pd
```

[3]:
```python
import matplotlib .pyplot as plt
```

[28]:
```python
import seaborn as sns
```

[5]:
```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

[7]:
```python
data =pd.read_csv("mall_customers.csv")
```

[8]:
```python
data
```

[8]:

	customer_id	gender	age	annual_income	spending_score
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
..	…	…	…	…	…
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

[200 rows x 5 columns]

[10]:
```python
print(data.head(10))
```

```
   customer_id  gender  age  annual_income  spending_score
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
```

```
5             6  Female   22              17              76
6             7  Female   35              18               6
7             8  Female   23              18              94
8             9    Male   64              19               3
9            10  Female   30              19              72
```

[11]:
```
data.describe()
```

[11]:

	customer_id	age	annual_income	spending_score
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

[12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   customer_id     200 non-null    int64
 1   gender          200 non-null    object
 2   age             200 non-null    int64
 3   annual_income   200 non-null    int64
 4   spending_score  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

[21]:
```
# Print the column names to check for discrepancies
print(data.columns)
```

```
Index(['customer_id', 'gender', 'age', 'annual_income', 'spending_score'],
dtype='object')
```

[23]:
```
# Select features for clustering
X = data[['annual_income', 'spending_score']]

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(X_scaled[:5])  # Preview the scaled features
```

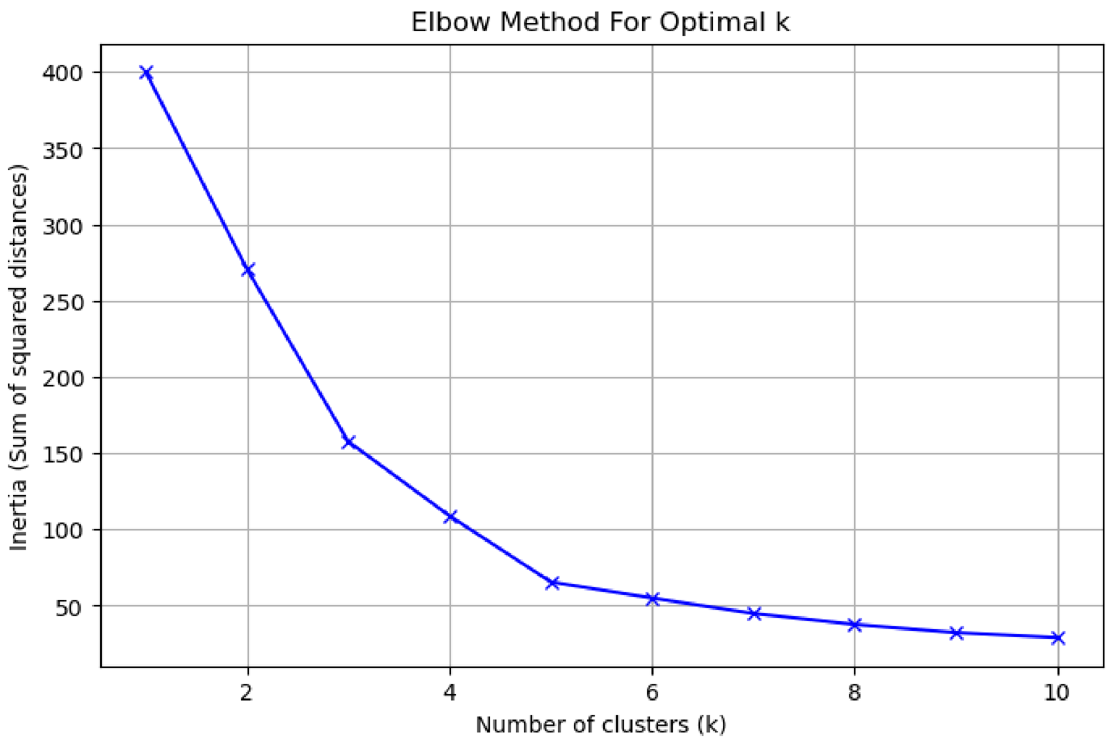
```
[[-1.73899919 -0.43480148]
 [-1.73899919  1.19570407]
 [-1.70082976 -1.71591298]
 [-1.70082976  1.04041783]
 [-1.66266033 -0.39597992]]
```

[25]:
```python
# Elbow Method to find the optimal number of clusters
inertia = []
K = range(1, 11)

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

# Plot the elbow curve
plt.figure(figsize=(8, 5))
plt.plot(K, inertia, 'bx-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (Sum of squared distances)')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



[26]:
```
# Apply K-Means with optimal clusters (e.g., k=5)
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X_scaled)

# Get the cluster labels and add them to the original dataset
data['Cluster'] = kmeans.labels_

# Print centroids
centroids = kmeans.cluster_centers_
print(f"Centroids:\n {centroids}")
```

```
Centroids:
 [[-0.20091257 -0.02645617]
 [ 1.05500302 -1.28443907]
 [-1.30751869 -1.13696536]
 [-1.32954532  1.13217788]
 [ 0.99158305  1.23950275]]
```

[30]:
```
# Visualizing the clusters
plt.figure(figsize=(10, 7))

# Scatter plot for each cluster
```

```
sns.scatterplot(x='annual_income', y='spending_score', hue='Cluster',␣
↪data=data, palette='Set1', s=100)

# Plot the centroids
plt.scatter(centroids[:, 0] * scaler.scale_[0] + scaler.mean_[0],  #␣
↪Denormalize the centroids
            centroids[:, 1] * scaler.scale_[1] + scaler.mean_[1],
            s=300, c='black', marker='X', label='Centroids')

plt.title('Customer Segments based on Income and Spending Score')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
```

[31]:
```
# Group the data by cluster and calculate the mean of each feature
cluster_summary = data.groupby('Cluster').mean()
print(cluster_summary)
```

```
         customer_id        age   annual_income   spending_score
```

```
Cluster
0          86.320988  42.716049      55.296296       49.518519
1         164.371429  41.114286      88.200000       17.114286
2          23.000000  45.217391      26.304348       20.913043
3          23.090909  25.272727      25.727273       79.363636
4         162.000000  32.692308      86.538462       82.128205
```

[ ]: