Name :- Shreyash Ashok Pachpol

Reg No :- 21BCE0896

VIT Vellore

Assignment 5

Market Basket Magic: Extracting Insights for Retail Success

Customer segmentation is a crucial aspect of retail and marketing strategy. Mall Customer Segmentation is a common data analysis project that involves categorizing mall customers into distinct groups or segments based on various characteristics and behaviors. This segmentation is valuable for tailoring marketing efforts, optimizing store layouts, and enhancing customer experiences.

Dataset link: Here

Task:

- Understand the data
- Data Preprocessing
- Machine Learning approach with clustering algorithm

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore

# Load the dataset
dataset_url = "/content/Mall_Customers.csv"
df = pd.read_csv(dataset_url)

# Explore the dataset
print(df.shape)
```

```python
print(df.info())
print(df.head())
```

```
(200, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
   CustomerID  Gender  Age  Annual Income (k$)  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
```

```python
# Understand Columns
print(df.describe())
print(df['Annual Income (k$)'].value_counts())
```

```
       CustomerID         Age  Annual Income (k$)  Spending Score (1-
100)
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
54     12
78     12
48      6
71      6
```

```
63      6
       ..
58      2
59      2
16      2
64      2
137     2
Name: Annual Income (k$), Length: 64, dtype: int64

# Handle Missing Data
df.dropna(inplace=True)

# Data Cleaning
# Example: Removing duplicates
df.drop_duplicates(inplace=True)

# Feature Engineering
# Example: Creating a new feature based on existing ones
df['column'] = df['Age'] * df['Spending Score (1-100)']

# Normalization/Scaling
# Example: Min-Max Scaling
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df[['Age', 'Spending 
Score (1-100)']]), columns=['scaled_feature1', 'scaled_feature2'])
df = pd.concat([df, df_scaled], axis=1)
df

     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
..          ...     ...  ...                 ...                     .
..
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

     column  scaled_feature1  scaled_feature2  scaled_feature1  \
0       741         0.019231         0.387755         0.019231
1      1701         0.057692         0.816327         0.057692
2       120         0.038462         0.051020         0.038462
3      1771         0.096154         0.775510         0.096154
4      1240         0.250000         0.397959         0.250000
..      ...              ...              ...              ...
195    2765         0.326923         0.795918         0.326923
196    1260         0.519231         0.275510         0.519231
197    2368         0.269231         0.744898         0.269231
198     576         0.269231         0.173469         0.269231
199    2490         0.230769         0.836735         0.230769

     scaled_feature2  scaled_feature1  scaled_feature2
scaled_feature1  \
0           0.387755         0.019231         0.387755
0.019231
1           0.816327         0.057692         0.816327
0.057692
2           0.051020         0.038462         0.051020
0.038462
3           0.775510         0.096154         0.775510
0.096154
4           0.397959         0.250000         0.397959
0.250000
..               ...              ...              ...              ..
.
195         0.795918         0.326923         0.795918
0.326923
196         0.275510         0.519231         0.275510
0.519231
197         0.744898         0.269231         0.744898
0.269231
198         0.173469         0.269231         0.173469
0.269231
199         0.836735         0.230769         0.836735
0.230769

     scaled_feature2
0           0.387755
1           0.816327
2           0.051020
3           0.775510
4           0.397959
..               ...
195         0.795918
```

```
196         0.275510
197         0.744898
198         0.173469
199         0.836735

[200 rows x 14 columns]

df_scaled

     scaled_feature1  scaled_feature2
0           0.019231         0.387755
1           0.057692         0.816327
2           0.038462         0.051020
3           0.096154         0.775510
4           0.250000         0.397959
..               ...              ...
195         0.326923         0.795918
196         0.519231         0.275510
197         0.269231         0.744898
198         0.269231         0.173469
199         0.230769         0.836735

[200 rows x 2 columns]

# Outlier Detection and Handling
# Example: Identify and remove outliers using Z-score
from scipy import stats

z_scores = np.abs(stats.zscore(df[['Age', 'Spending Score (1-100)']]))
df_no_outliers = df[(z_scores < 3).all(axis=1)]
df_no_outliers
z_scores

          Age  Spending Score (1-100)
0    1.424569                0.434801
1    1.281035                1.195704
2    1.352802                1.715913
3    1.137502                1.040418
4    0.563369                0.395980
..        ...                     ...
195  0.276302                1.118061
196  0.441365                0.861839
197  0.491602                0.923953
198  0.491602                1.250054
199  0.635135                1.273347

[200 rows x 2 columns]

df_no_outliers
```

```
     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
..          ...     ...  ...                 ...                       .
..
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

     column  scaled_feature1  scaled_feature2  scaled_feature1  \
0       741         0.019231         0.387755         0.019231
1      1701         0.057692         0.816327         0.057692
2       120         0.038462         0.051020         0.038462
3      1771         0.096154         0.775510         0.096154
4      1240         0.250000         0.397959         0.250000
..      ...              ...              ...              ...
195    2765         0.326923         0.795918         0.326923
196    1260         0.519231         0.275510         0.519231
197    2368         0.269231         0.744898         0.269231
198     576         0.269231         0.173469         0.269231
199    2490         0.230769         0.836735         0.230769

     scaled_feature2  scaled_feature1  scaled_feature2
0           0.387755         0.019231         0.387755
1           0.816327         0.057692         0.816327
2           0.051020         0.038462         0.051020
3           0.775510         0.096154         0.775510
4           0.397959         0.250000         0.397959
..               ...              ...              ...
195         0.795918         0.326923         0.795918
196         0.275510         0.519231         0.275510
197         0.744898         0.269231         0.744898
198         0.173469         0.269231         0.173469
199         0.836735         0.230769         0.836735
```

```
[200 rows x 12 columns]

# Display summary after handling outliers
print("Original data shape:", df.shape)
print("Data shape after handling outliers:", df_no_outliers.shape)

Original data shape: (200, 14)
Data shape after handling outliers: (200, 12)

# Data Cleaning
# Example: Removing rows with certain conditions
# Suppose 'column_name' has unacceptable values, and you want to
remove those rows
df = df[df['CustomerID'] != 'unacceptable_value']
df

     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
..          ...     ...  ...                 ...                    .
..
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

     column  scaled_feature1  scaled_feature2  scaled_feature1  \
0       741         0.019231         0.387755         0.019231
1      1701         0.057692         0.816327         0.057692
2       120         0.038462         0.051020         0.038462
3      1771         0.096154         0.775510         0.096154
4      1240         0.250000         0.397959         0.250000
..      ...              ...              ...              ...
195    2765         0.326923         0.795918         0.326923
```

```
196    1260         0.519231         0.275510         0.519231
197    2368         0.269231         0.744898         0.269231
198     576         0.269231         0.173469         0.269231
199    2490         0.230769         0.836735         0.230769

     scaled_feature2  scaled_feature1  scaled_feature2
scaled_feature1  \
0           0.387755         0.019231         0.387755
0.019231
1           0.816327         0.057692         0.816327
0.057692
2           0.051020         0.038462         0.051020
0.038462
3           0.775510         0.096154         0.775510
0.096154
4           0.397959         0.250000         0.397959
0.250000
..               ...              ...              ...              ..
.
195         0.795918         0.326923         0.795918
0.326923
196         0.275510         0.519231         0.275510
0.519231
197         0.744898         0.269231         0.744898
0.269231
198         0.173469         0.269231         0.173469
0.269231
199         0.836735         0.230769         0.836735
0.230769

     scaled_feature2
0           0.387755
1           0.816327
2           0.051020
3           0.775510
4           0.397959
..               ...
195         0.795918
196         0.275510
197         0.744898
198         0.173469
199         0.836735

[200 rows x 14 columns]

# Feature Engineering
# Example: Creating a binary feature based on a condition
df['new_binary_feature'] = df['Age'] > df['Age'].mean()

# Normalization/Scaling (if needed)
```

```
# Example: Standard Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['Annual Income (k$)', 'Spending Score (1-100)']] =
scaler.fit_transform(df[['Annual Income (k$)', 'Spending Score (1-
100)']])

# Outlier Detection and Handling
# Example: Replace outliers with the mean value
from scipy.stats import zscore

z_scores = zscore(df[['Annual Income (k$)', 'Spending Score (1-
100)']])
outliers = (z_scores > 3) | (z_scores < -3)
df_outliers = df.mean()
df_outliers
```

```
<ipython-input-40-6563b845eefc>:7: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it
will default to False. In addition, specifying 'numeric_only=None' is
deprecated. Select only valid columns or specify the value of
numeric_only to silence this warning.
  df_outliers = df.mean()
```

```
CustomerID                1.005000e+02
Age                       3.885000e+01
Annual Income (k$)       -2.131628e-16
Spending Score (1-100)   -1.465494e-16
column                    1.832820e+03
scaled_feature1           4.009615e-01
scaled_feature2           5.020408e-01
scaled_feature1           4.009615e-01
scaled_feature2           5.020408e-01
scaled_feature1           4.009615e-01
scaled_feature2           5.020408e-01
scaled_feature1           4.009615e-01
scaled_feature2           5.020408e-01
new_binary_feature        4.350000e-01
dtype: float64
```

```
# Display summary after handling outliers
print("Original data shape:", df.shape)
```

```
Original data shape: (200, 15)
```

```
# Import necessary libraries for clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

```python
# Feature Selection
selected_features = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
selected_features
```

```
     Age  Annual Income (k$)  Spending Score (1-100)
0     19           -1.738999               -0.434801
1     21           -1.738999                1.195704
2     20           -1.700830               -1.715913
3     23           -1.700830                1.040418
4     31           -1.662660               -0.395980
..   ...                 ...                     ...
195   35            2.268791                1.118061
196   45            2.497807               -0.861839
197   32            2.497807                0.923953
198   32            2.917671               -1.250054
199   30            2.917671                1.273347

[200 rows x 3 columns]
```

```python
# Normalization/Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(selected_features)
scaled_features
```

```
array([[-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],
       [-1.20926872, -1.66266033,  1.00159627],
       [-0.27630176, -1.62449091, -1.71591298],
       [-1.13750203, -1.62449091,  1.70038436],
       [ 1.80493225, -1.58632148, -1.83237767],
       [-0.6351352 , -1.58632148,  0.84631002],
       [ 2.02023231, -1.58632148, -1.4053405 ],
       [-0.27630176, -1.58632148,  1.89449216],
       [ 1.37433211, -1.54815205, -1.36651894],
       [-1.06573534, -1.54815205,  1.04041783],
       [-0.13276838, -1.54815205, -1.44416206],
       [-1.20926872, -1.54815205,  1.11806095],
       [-0.27630176, -1.50998262, -0.59008772],
       [-1.3528021 , -1.50998262,  0.61338066],
       [ 0.94373197, -1.43364376, -0.82301709],
       [-0.27630176, -1.43364376,  1.8556706 ],
       [-0.27630176, -1.39547433, -0.59008772],
       [-0.99396865, -1.39547433,  0.88513158],
       [ 0.51313183, -1.3573049 , -1.75473454],
       [-0.56336851, -1.3573049 ,  0.88513158],
       [ 1.08726535, -1.24279661, -1.4053405 ],
```

```
       [-0.70690189, -1.24279661,  1.23452563],
       [ 0.44136514, -1.24279661, -0.7065524 ],
       [-0.27630176, -1.24279661,  0.41927286],
       [ 0.08253169, -1.20462718, -0.74537397],
       [-1.13750203, -1.20462718,  1.42863343],
       [ 1.51786549, -1.16645776, -1.7935561 ],
       [-1.28103541, -1.16645776,  0.88513158],
       [ 1.01549866, -1.05194947, -1.7935561 ],
       [-1.49633548, -1.05194947,  1.62274124],
       [ 0.7284319 , -1.05194947, -1.4053405 ],
       [-1.28103541, -1.05194947,  1.19570407],
       [ 0.22606507, -1.01378004, -1.28887582],
       [-0.6351352 , -1.01378004,  0.88513158],
       [-0.20453507, -0.89927175, -0.93948177],
       [-1.3528021 , -0.89927175,  0.96277471],
       [ 1.87669894, -0.86110232, -0.59008772],
       [-1.06573534, -0.86110232,  1.62274124],
       [ 0.65666521, -0.82293289, -0.55126616],
       [-0.56336851, -0.82293289,  0.41927286],
       [ 0.7284319 , -0.82293289, -0.86183865],
       [-1.06573534, -0.82293289,  0.5745591 ],
       [ 0.80019859, -0.78476346,  0.18634349],
       [-0.85043527, -0.78476346, -0.12422899],
       [-0.70690189, -0.78476346, -0.3183368 ],
       [-0.56336851, -0.78476346, -0.3183368 ],
       [ 0.7284319 , -0.70842461,  0.06987881],
       [-0.41983513, -0.70842461,  0.38045129],
       [-0.56336851, -0.67025518,  0.14752193],
       [ 1.4460988 , -0.67025518,  0.38045129],
       [ 0.80019859, -0.67025518, -0.20187212],
       [ 0.58489852, -0.67025518, -0.35715836],
       [ 0.87196528, -0.63208575, -0.00776431],
       [ 2.16376569, -0.63208575, -0.16305055],
       [-0.85043527, -0.55574689,  0.03105725],
       [ 1.01549866, -0.55574689, -0.16305055],
       [ 2.23553238, -0.55574689,  0.22516505],
       [-1.42456879, -0.55574689,  0.18634349],
       [ 2.02023231, -0.51757746,  0.06987881],
       [ 1.08726535, -0.51757746,  0.34162973],
       [ 1.73316556, -0.47940803,  0.03105725],
       [-1.49633548, -0.47940803,  0.34162973],
       [ 0.29783176, -0.47940803, -0.00776431],
       [ 2.091999  , -0.47940803, -0.08540743],
       [-1.42456879, -0.47940803,  0.34162973],
       [-0.49160182, -0.47940803, -0.12422899],
       [ 2.23553238, -0.4412386 ,  0.18634349],
       [ 0.58489852, -0.4412386 , -0.3183368 ],
       [ 1.51786549, -0.40306917, -0.04658587],
       [ 1.51786549, -0.40306917,  0.22516505],
```

```
       [ 1.4460988 , -0.25039146, -0.12422899],
       [-0.92220196, -0.25039146,  0.14752193],
       [ 0.44136514, -0.25039146,  0.10870037],
       [ 0.08253169, -0.25039146, -0.08540743],
       [-1.13750203, -0.25039146,  0.06987881],
       [ 0.7284319 , -0.25039146, -0.3183368 ],
       [ 1.30256542, -0.25039146,  0.03105725],
       [-0.06100169, -0.25039146,  0.18634349],
       [ 2.02023231, -0.25039146, -0.35715836],
       [ 0.51313183, -0.25039146, -0.24069368],
       [-1.28103541, -0.25039146,  0.26398661],
       [ 0.65666521, -0.25039146, -0.16305055],
       [ 1.15903204, -0.13588317,  0.30280817],
       [-1.20926872, -0.13588317,  0.18634349],
       [-0.34806844, -0.09771374,  0.38045129],
       [ 0.80019859, -0.09771374, -0.16305055],
       [ 2.091999  , -0.05954431,  0.18634349],
       [-1.49633548, -0.05954431, -0.35715836],
       [ 0.65666521, -0.02137488, -0.04658587],
       [ 0.08253169, -0.02137488, -0.39597992],
       [-0.49160182, -0.02137488, -0.3183368 ],
       [-1.06573534, -0.02137488,  0.06987881],
       [ 0.58489852, -0.02137488, -0.12422899],
       [-0.85043527, -0.02137488, -0.00776431],
       [ 0.65666521,  0.01679455, -0.3183368 ],
       [-1.3528021 ,  0.01679455, -0.04658587],
       [-1.13750203,  0.05496398, -0.35715836],
       [ 0.7284319 ,  0.05496398, -0.08540743],
       [ 2.02023231,  0.05496398,  0.34162973],
       [-0.92220196,  0.05496398,  0.18634349],
       [ 0.7284319 ,  0.05496398,  0.22516505],
       [-1.28103541,  0.05496398, -0.3183368 ],
       [ 1.94846562,  0.09313341, -0.00776431],
       [ 1.08726535,  0.09313341, -0.16305055],
       [ 2.091999  ,  0.09313341, -0.27951524],
       [ 1.94846562,  0.09313341, -0.08540743],
       [ 1.87669894,  0.09313341,  0.06987881],
       [-1.42456879,  0.09313341,  0.14752193],
       [-0.06100169,  0.13130284, -0.3183368 ],
       [-1.42456879,  0.13130284, -0.16305055],
       [-1.49633548,  0.16947227, -0.08540743],
       [-1.42456879,  0.16947227, -0.00776431],
       [ 1.73316556,  0.16947227, -0.27951524],
       [ 0.7284319 ,  0.16947227,  0.34162973],
       [ 0.87196528,  0.24581112, -0.27951524],
       [ 0.80019859,  0.24581112,  0.26398661],
       [-0.85043527,  0.24581112,  0.22516505],
       [-0.06100169,  0.24581112, -0.39597992],
       [ 0.08253169,  0.32214998,  0.30280817],
```

```
       [ 0.010765  ,  0.32214998,  1.58391968],
       [-1.13750203,  0.36031941, -0.82301709],
       [-0.56336851,  0.36031941,  1.04041783],
       [ 0.29783176,  0.39848884, -0.59008772],
       [ 0.08253169,  0.39848884,  1.73920592],
       [ 1.4460988 ,  0.39848884, -1.52180518],
       [-0.06100169,  0.39848884,  0.96277471],
       [ 0.58489852,  0.39848884, -1.5994483 ],
       [ 0.010765  ,  0.39848884,  0.96277471],
       [-0.99396865,  0.43665827, -0.62890928],
       [-0.56336851,  0.43665827,  0.80748846],
       [-1.3528021 ,  0.4748277 , -1.75473454],
       [-0.70690189,  0.4748277 ,  1.46745499],
       [ 0.36959845,  0.4748277 , -1.67709142],
       [-0.49160182,  0.4748277 ,  0.88513158],
       [-1.42456879,  0.51299713, -1.56062674],
       [-0.27630176,  0.51299713,  0.84631002],
       [ 1.30256542,  0.55116656, -1.75473454],
       [-0.49160182,  0.55116656,  1.6615628 ],
       [-0.77866858,  0.58933599, -0.39597992],
       [-0.49160182,  0.58933599,  1.42863343],
       [-0.99396865,  0.62750542, -1.48298362],
       [-0.77866858,  0.62750542,  1.81684904],
       [ 0.65666521,  0.62750542, -0.55126616],
       [-0.49160182,  0.62750542,  0.92395314],
       [-0.34806844,  0.66567484, -1.09476801],
       [-0.34806844,  0.66567484,  1.54509812],
       [ 0.29783176,  0.66567484, -1.28887582],
       [ 0.010765  ,  0.66567484,  1.46745499],
       [ 0.36959845,  0.66567484, -1.17241113],
       [-0.06100169,  0.66567484,  1.00159627],
       [ 0.58489852,  0.66567484, -1.32769738],
       [-0.85043527,  0.66567484,  1.50627656],
       [-0.13276838,  0.66567484, -1.91002079],
       [-0.6351352 ,  0.66567484,  1.07923939],
       [-0.34806844,  0.66567484, -1.91002079],
       [-0.6351352 ,  0.66567484,  0.88513158],
       [ 1.23079873,  0.70384427, -0.59008772],
       [-0.70690189,  0.70384427,  1.27334719],
       [-1.42456879,  0.78018313, -1.75473454],
       [-0.56336851,  0.78018313,  1.6615628 ],
       [ 0.80019859,  0.93286085, -0.93948177],
       [-0.20453507,  0.93286085,  0.96277471],
       [ 0.22606507,  0.97103028, -1.17241113],
       [-0.41983513,  0.97103028,  1.73920592],
       [-0.20453507,  1.00919971, -0.90066021],
       [-0.49160182,  1.00919971,  0.49691598],
       [ 0.08253169,  1.00919971, -1.44416206],
       [-0.77866858,  1.00919971,  0.96277471],
```

```
       [-0.20453507,  1.00919971, -1.56062674],
       [-0.20453507,  1.00919971,  1.62274124],
       [ 0.94373197,  1.04736914, -1.44416206],
       [-0.6351352 ,  1.04736914,  1.38981187],
       [ 1.37433211,  1.04736914, -1.36651894],
       [-0.85043527,  1.04736914,  0.72984534],
       [ 1.4460988 ,  1.23821628, -1.4053405 ],
       [-0.27630176,  1.23821628,  1.54509812],
       [-0.13276838,  1.390894  , -0.7065524 ],
       [-0.49160182,  1.390894  ,  1.38981187],
       [ 0.51313183,  1.42906343, -1.36651894],
       [-0.70690189,  1.42906343,  1.46745499],
       [ 0.15429838,  1.46723286, -0.43480148],
       [-0.6351352 ,  1.46723286,  1.81684904],
       [ 1.08726535,  1.54357172, -1.01712489],
       [-0.77866858,  1.54357172,  0.69102378],
       [ 0.15429838,  1.61991057, -1.28887582],
       [-0.20453507,  1.61991057,  1.35099031],
       [-0.34806844,  1.61991057, -1.05594645],
       [-0.49160182,  1.61991057,  0.72984534],
       [-0.41983513,  2.00160487, -1.63826986],
       [-0.06100169,  2.00160487,  1.58391968],
       [ 0.58489852,  2.26879087, -1.32769738],
       [-0.27630176,  2.26879087,  1.11806095],
       [ 0.44136514,  2.49780745, -0.86183865],
       [-0.49160182,  2.49780745,  0.92395314],
       [-0.49160182,  2.91767117, -1.25005425],
       [-0.6351352 ,  2.91767117,  1.27334719]])
```

```python
# Select a Clustering Algorithm (e.g., K-Means)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans
```

```
KMeans(n_clusters=3, random_state=42)
```

```python
# Train the Model
kmeans.fit(scaled_features)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
```

```
KMeans(n_clusters=3, random_state=42)
```

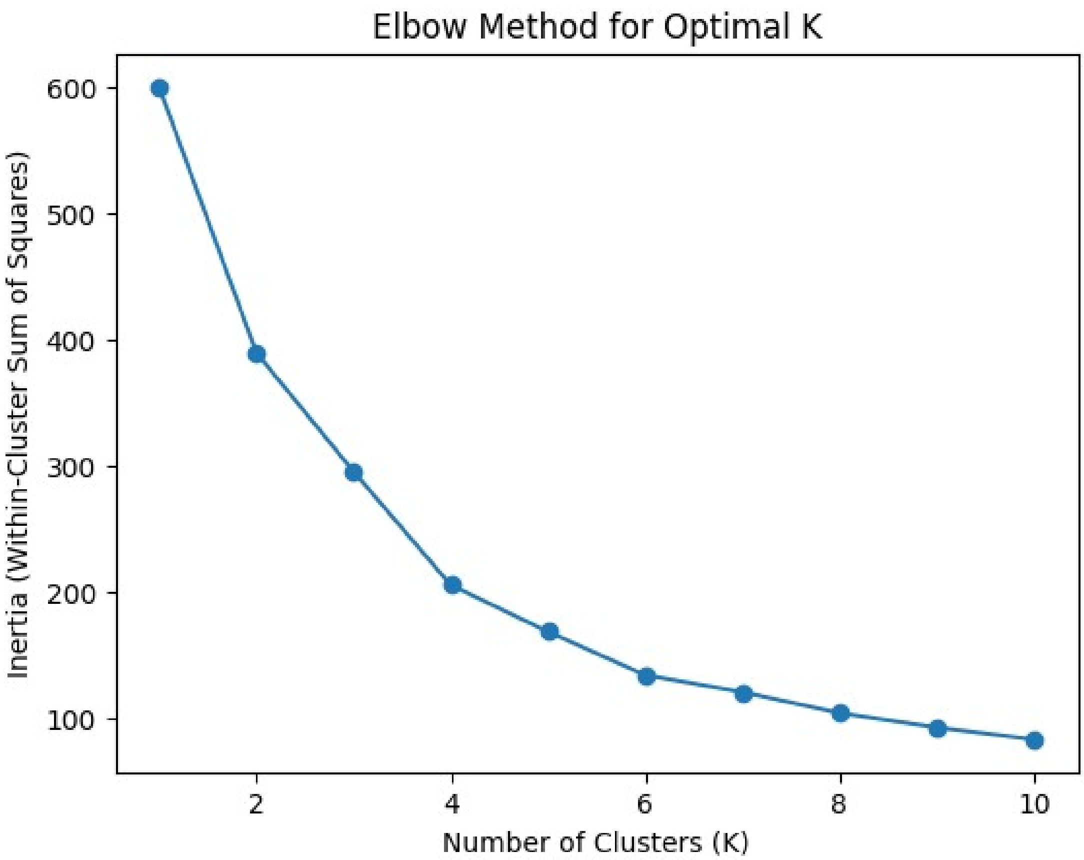
```python
# Optimal Number of Clusters (K)
# Use the elbow method or silhouette analysis to find the optimal K

# Elbow Method
inertia = []
```

```python
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

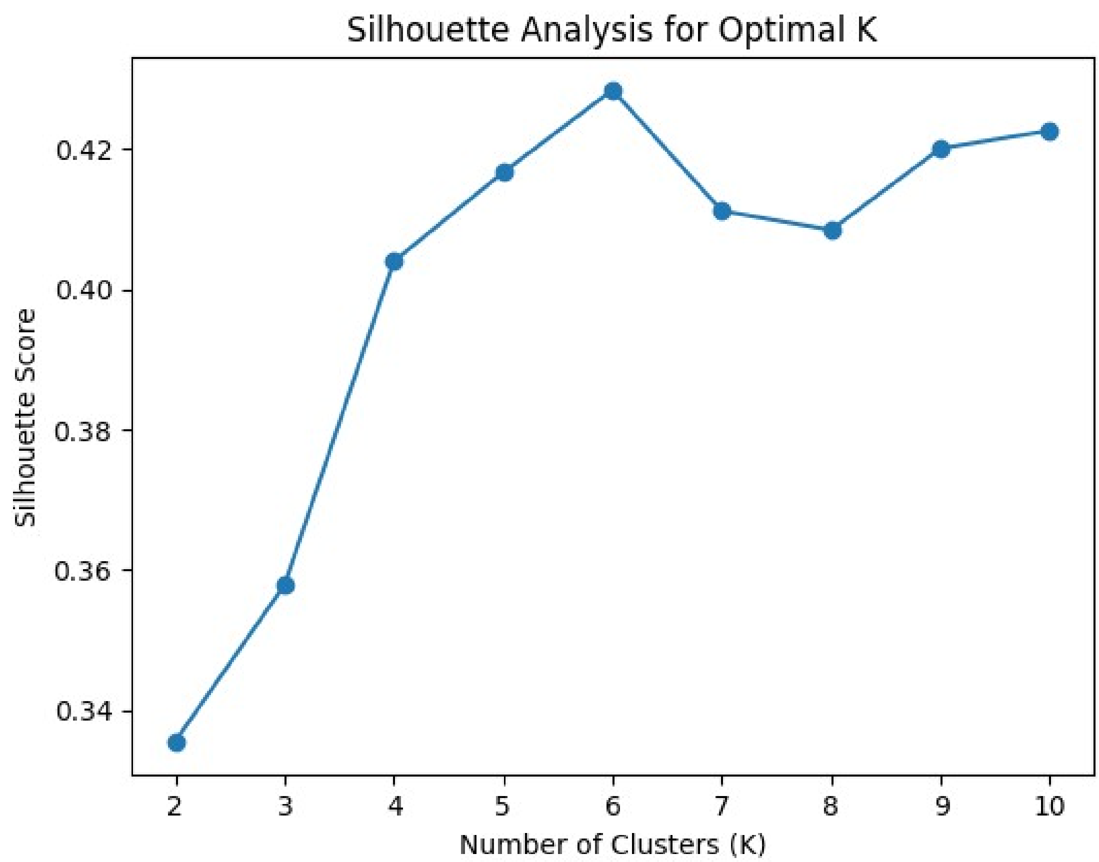
```
# Plot the elbow method
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.show()
```




```
# Silhouette Analysis
sil_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    sil_scores.append(silhouette_score(scaled_features,
kmeans.labels_))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
```

```
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(

# Plot silhouette scores
plt.plot(range(2, 11), sil_scores, marker='o')
plt.title('Silhouette Analysis for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.show()
```

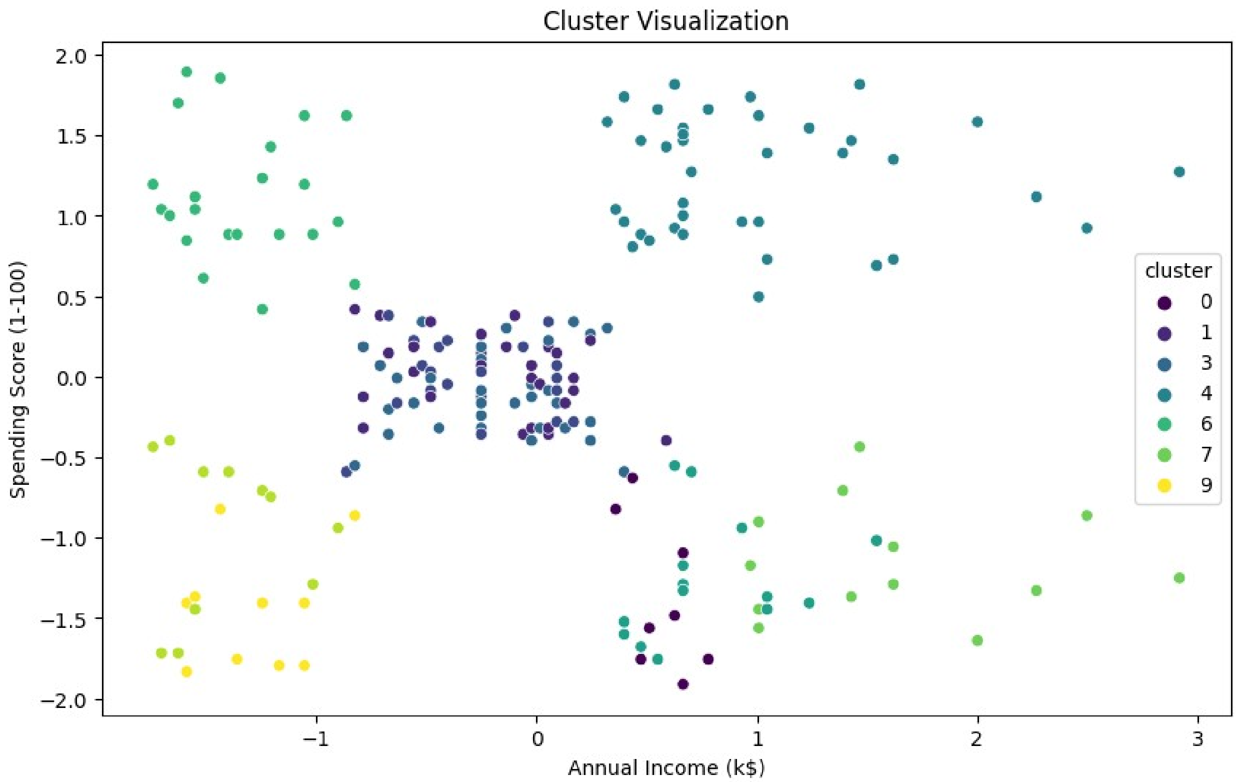



```
# Choose the optimal K and retrain the model
optimal_k = 3  # Choose based on the analysis from the elbow method
and silhouette analysis
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(scaled_features)
optimal_k

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(

3

kmeans

KMeans(n_clusters=3, random_state=42)

kmeans.fit(scaled_features)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
```

```
to suppress the warning
  warnings.warn(

KMeans(n_clusters=3, random_state=42)

# Cluster Assignment
df['cluster'] = kmeans.labels_
df['cluster']

0      8
1      6
2      8
3      6
4      8
      ..
195    4
196    7
197    4
198    7
199    4
Name: cluster, Length: 200, dtype: int32

# Analysis and Interpretation
# Analyze the characteristics of each cluster
cluster_means = df.groupby('cluster').mean()
cluster_means

<ipython-input-64-b52dd425c32c>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the
function.
  cluster_means = df.groupby('cluster').mean()

         CustomerID        Age  Annual Income (k$)  Spending Score (1-
100)  \
cluster

0        145.000000  26.222222            0.576613                -
1.435535
1         85.066667  24.800000           -0.221128
0.019411
2         80.750000  64.850000           -0.282835                -
0.013588
3         86.750000  47.093750           -0.190752                -
0.092686
4        162.000000  32.692308            0.991583
1.239503
5        156.285714  51.285714            0.782910                -
1.261146
6         23.090909  25.272727           -1.329545
```

```
1.132178
7        184.076923  39.230769           1.631655                -
1.154493
8         18.272727  34.090909          -1.412824                -
0.960657
9         24.400000  55.200000          -1.284783                -
1.444162

            column  scaled_feature1  scaled_feature2  scaled_feature1
\
cluster

0       -38.071873         0.158120         0.124717         0.158120

1         0.520468         0.130769         0.507143         0.130769

2        -0.948023         0.900962         0.498469         0.900962

3        -4.056125         0.559495         0.477679         0.559495

4        40.697936         0.282544         0.827839         0.282544

5       -64.775438         0.640110         0.170554         0.640110

6        28.773482         0.139860         0.799629         0.139860

7       -45.110654         0.408284         0.198587         0.408284

8       -32.655991         0.309441         0.249536         0.309441

9       -80.385476         0.715385         0.122449         0.715385


         scaled_feature2  scaled_feature1  scaled_feature2
scaled_feature1  \
cluster

0               0.124717         0.158120         0.124717
0.158120
1               0.507143         0.130769         0.507143
0.130769
2               0.498469         0.900962         0.498469
0.900962
3               0.477679         0.559495         0.477679
0.559495
4               0.827839         0.282544         0.827839
0.282544
5               0.170554         0.640110         0.170554
0.640110
6               0.799629         0.139860         0.799629
0.139860
```

```
7                0.198587          0.408284          0.198587
0.408284
8                0.249536          0.309441          0.249536
0.309441
9                0.122449          0.715385          0.122449
0.715385

         scaled_feature2  new_binary_feature
cluster
0               0.124717            0.000000
1               0.507143            0.000000
2               0.498469            1.000000
3               0.477679            0.906250
4               0.827839            0.102564
5               0.170554            1.000000
6               0.799629            0.000000
7               0.198587            0.538462
8               0.249536            0.272727
9               0.122449            1.000000

# Assuming 'cluster' is the column indicating cluster assignments
for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    cluster_data

# Visualization
# Visualize the clusters using appropriate plots
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
hue='cluster', data=df, palette='viridis')
plt.title('Cluster Visualization')
plt.show()
```




```
# Additional insights and recommendations based on the analysis
for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    cluster_data

# Identify key characteristics or patterns in each cluster
cluster_insights = {
        'Cluster': cluster_id,
        'Mean_Feature1': cluster_data['Age'].mean(),
        'Mean_Feature2': cluster_data['Annual Income (k$)'].mean(),
        'Total_Customers': len(cluster_data),
        # Add more insights as needed
    }
cluster_insights

{'Cluster': 2,
 'Mean_Feature1': 64.85,
 'Mean_Feature2': -0.282835470162833,
 'Total_Customers': 20}

print(f"\nAdditional Insights for Cluster {cluster_id}:")
print(cluster_insights)


Additional Insights for Cluster 2:
```

```
{'Cluster': 2, 'Mean_Feature1': 64.85, 'Mean_Feature2': -
0.282835470162833, 'Total_Customers': 20}
```

```python
# Provide recommendations based on the cluster analysis
recommendations = []
if cluster_id == 0:
        recommendations.append("This cluster represents a group of
high-value customers. Consider targeted marketing to retain and
upsell.")
elif cluster_id == 1:
        recommendations.append("Customers in this cluster show
interest in specific product categories. Tailor promotions
accordingly.")

# Add more recommendations based on your analysis

print("\nRecommendations:")
for recommendation in recommendations:
        print(f"- {recommendation}")
```

```
Recommendations:
```

```python
# General summary or overarching recommendations based on the overall
analysis
overall_insights = {
    'Total_Customers': len(df),
    'Mean_Feature1': df['Age'].mean(),
    'Mean_Feature2': df['Annual Income (k$)'].mean(),
    # Add more overall insights as needed
}

print("\nOverall Insights:")
print(overall_insights)
```

```
Overall Insights:
{'Total_Customers': 200, 'Mean_Feature1': 38.85, 'Mean_Feature2':
7.105427357601002e-17}
```

```python
# General recommendations based on the overall analysis
overall_recommendations = [
    "Explore partnerships with brands/products popular among the
majority of customers.",
    "Consider optimizing store layouts based on common preferences
observed in the clusters.",
    # Add more general recommendations
]
```

```
print("\nOverall Recommendations:")
for recommendation in overall_recommendations:
    print(f"- {recommendation}")
```

```
Overall Recommendations:
- Explore partnerships with brands/products popular among the majority
of customers.
- Consider optimizing store layouts based on common preferences
observed in the clusters.
```

```
# Report
# Summarize findings in a report format
print(f"Optimal number of clusters (K): {optimal_k}")
print("Cluster Means:")
print(cluster_means)
```

```
Optimal number of clusters (K): 3
Cluster Means:
         CustomerID        Age  Annual Income (k$)  Spending Score (1-
100)  \
cluster

0        145.000000  26.222222            0.576613                 -
1.435535
1         85.066667  24.800000           -0.221128
0.019411
2         80.750000  64.850000           -0.282835                 -
0.013588
3         86.750000  47.093750           -0.190752                 -
0.092686
4        162.000000  32.692308            0.991583
1.239503
5        156.285714  51.285714            0.782910                 -
1.261146
6         23.090909  25.272727           -1.329545
1.132178
7        184.076923  39.230769            1.631655                 -
1.154493
8         18.272727  34.090909           -1.412824                 -
0.960657
9         24.400000  55.200000           -1.284783                 -
1.444162

            column  scaled_feature1  scaled_feature2  scaled_feature1
\
cluster

0       -38.071873         0.158120         0.124717         0.158120
```

```
1          0.520468          0.130769          0.507143          0.130769

2         -0.948023          0.900962          0.498469          0.900962

3         -4.056125          0.559495          0.477679          0.559495

4         40.697936          0.282544          0.827839          0.282544

5        -64.775438          0.640110          0.170554          0.640110

6         28.773482          0.139860          0.799629          0.139860

7        -45.110654          0.408284          0.198587          0.408284

8        -32.655991          0.309441          0.249536          0.309441

9        -80.385476          0.715385          0.122449          0.715385


         scaled_feature2  scaled_feature1  scaled_feature2
scaled_feature1  \
cluster

0               0.124717         0.158120         0.124717
0.158120
1               0.507143         0.130769         0.507143
0.130769
2               0.498469         0.900962         0.498469
0.900962
3               0.477679         0.559495         0.477679
0.559495
4               0.827839         0.282544         0.827839
0.282544
5               0.170554         0.640110         0.170554
0.640110
6               0.799629         0.139860         0.799629
0.139860
7               0.198587         0.408284         0.198587
0.408284
8               0.249536         0.309441         0.249536
0.309441
9               0.122449         0.715385         0.122449
0.715385

         scaled_feature2  new_binary_feature
cluster
0               0.124717            0.000000
1               0.507143            0.000000
2               0.498469            1.000000
3               0.477679            0.906250
```

```
4                0.827839           0.102564
5                0.170554           1.000000
6                0.799629           0.000000
7                0.198587           0.538462
8                0.249536           0.272727
9                0.122449           1.000000
```

All the required codes for Assignment

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore

# Load the dataset
dataset_url = "/content/Mall_Customers.csv"
df = pd.read_csv(dataset_url)
df

# Explore the dataset
print(df.shape)
print(df.info())
print(df.head())

# Understand Columns
print(df.describe())
print(df['Age'].value_counts())

# Handle Missing Data
df.dropna(inplace=True)  # or use imputation strategies

# Data Cleaning
# Example: Removing duplicates
df.drop_duplicates(inplace=True)

# Feature Engineering
# Example: Creating a new feature based on existing ones
df['new_feature'] = df['Annual Income (k$)'] * df['Spending Score (1-
100)']

# Normalization/Scaling
# Example: Min-Max Scaling
scaler = StandardScaler()
```

```python
df_scaled = pd.DataFrame(scaler.fit_transform(df[['Annual Income
(k$)', 'Spending Score (1-100)']]), columns=['scaled_income',
'scaled_spending'])
df_scaled
df = pd.concat([df, df_scaled], axis=1)
df

# Outlier Detection and Handling
# Example: Identify and remove outliers using Z-score
z_scores = np.abs(zscore(df[['Annual Income (k$)', 'Spending Score (1-
100)']]))
z_scores
df_no_outliers = df[(z_scores < 3).all(axis=1)]
df_no_outliers

# Clustering Analysis
# Feature Selection
selected_features = df[['Annual Income (k$)', 'Spending Score (1-
100)']]
selected_features

# Normalization/Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(selected_features)
scaled_features

# Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

# Plot the elbow method
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.show()

# Silhouette Analysis
sil_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    sil_score = silhouette_score(scaled_features, kmeans.labels_)
    sil_scores.append(sil_score)

# Plot silhouette scores
plt.plot(range(2, 11), sil_scores, marker='o')
```

```
plt.title('Silhouette Analysis for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.show()

# Choose the optimal K and retrain the model
optimal_k = 3  # Choose based on the analysis from the elbow method
and silhouette analysis
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans
kmeans.fit(scaled_features)

# Cluster Assignment
df['cluster'] = kmeans.labels_
df['cluster']

# Analysis and Interpretation
# Analyze the characteristics of each cluster
cluster_means = df.groupby('cluster').mean()
cluster_means

# Extract insights and patterns from each cluster
for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    cluster_data

    # Display summary statistics for each cluster
    print(f"\nCluster {cluster_id} Summary:")
    print(cluster_data.describe())

    # Explore feature distributions within each cluster
    for column in selected_features.columns:
        plt.figure(figsize=(8, 5))
        sns.histplot(cluster_data[column], bins=20, kde=True)
        plt.title(f'Distribution of {column} in Cluster {cluster_id}')
        plt.show()

# Additional insights and recommendations based on the analysis
for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    cluster_data

    # Identify key characteristics or patterns in each cluster
    cluster_insights = {
        'Cluster': cluster_id,
        'Mean_Annual_Income': cluster_data['Annual Income
(k$)'].mean(),
        'Mean_Spending_Score': cluster_data['Spending Score (1-
100)'].mean(),
        'Total_Customers': len(cluster_data),
```

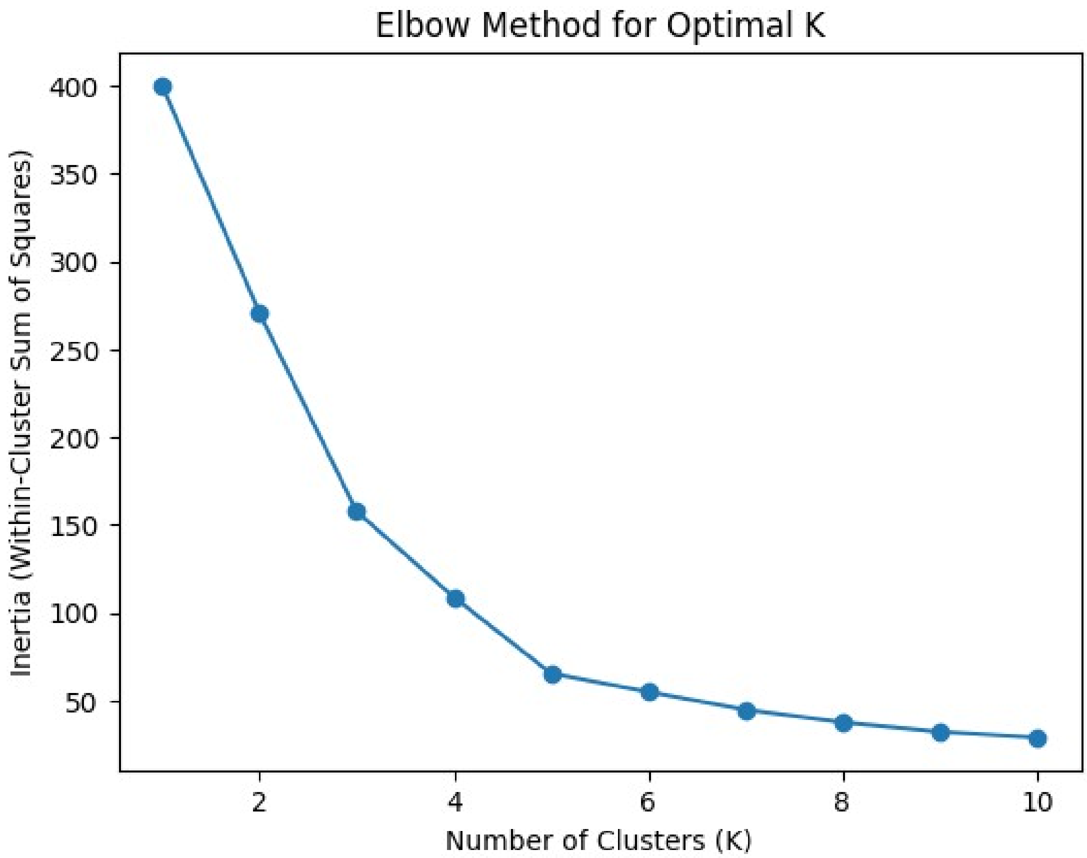
```
        # Add more insights as needed
    }

    print(f"\nAdditional Insights for Cluster {cluster_id}:")
    print(cluster_insights)

    # Provide recommendations based on the cluster analysis
    recommendations = []
    if cluster_id == 0:
        recommendations.append("This cluster represents a group of
high-value customers. Consider targeted marketing to retain and
upsell.")
    elif cluster_id == 1:
        recommendations.append("Customers in this cluster show
interest in specific product categories. Tailor promotions
accordingly.")
    # Add more recommendations based on your analysis

    print("\nRecommendations:")
    for recommendation in recommendations:
        print(f"- {recommendation}")

# General summary or overarching recommendations based on the overall
analysis
overall_insights = {
    'Total_Customers': len(df),
    'Mean_Annual_Income': df['Annual Income (k$)'].mean(),
    'Mean_Spending_Score': df['Spending Score (1-100)'].mean(),
    # Add more overall insights as needed
}

print("\nOverall Insights:")
print(overall_insights)

# General recommendations based on the overall analysis
overall_recommendations = [
    "Explore partnerships with brands/products popular among the
majority of customers.",
    "Consider optimizing store layouts based on common preferences
observed in the clusters.",
    # Add more general recommendations
]

print("\nOverall Recommendations:")
for recommendation in overall_recommendations:
    print(f"- {recommendation}")
```

```
(200, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
```

```
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
   CustomerID  Gender  Age  Annual Income (k$)  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
       CustomerID         Age  Annual Income (k$)  Spending Score (1-
100)
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
32    11
35     9
19     8
31     8
30     7
49     7
40     6
38     6
47     6
27     6
36     6
23     6
34     5
20     5
```

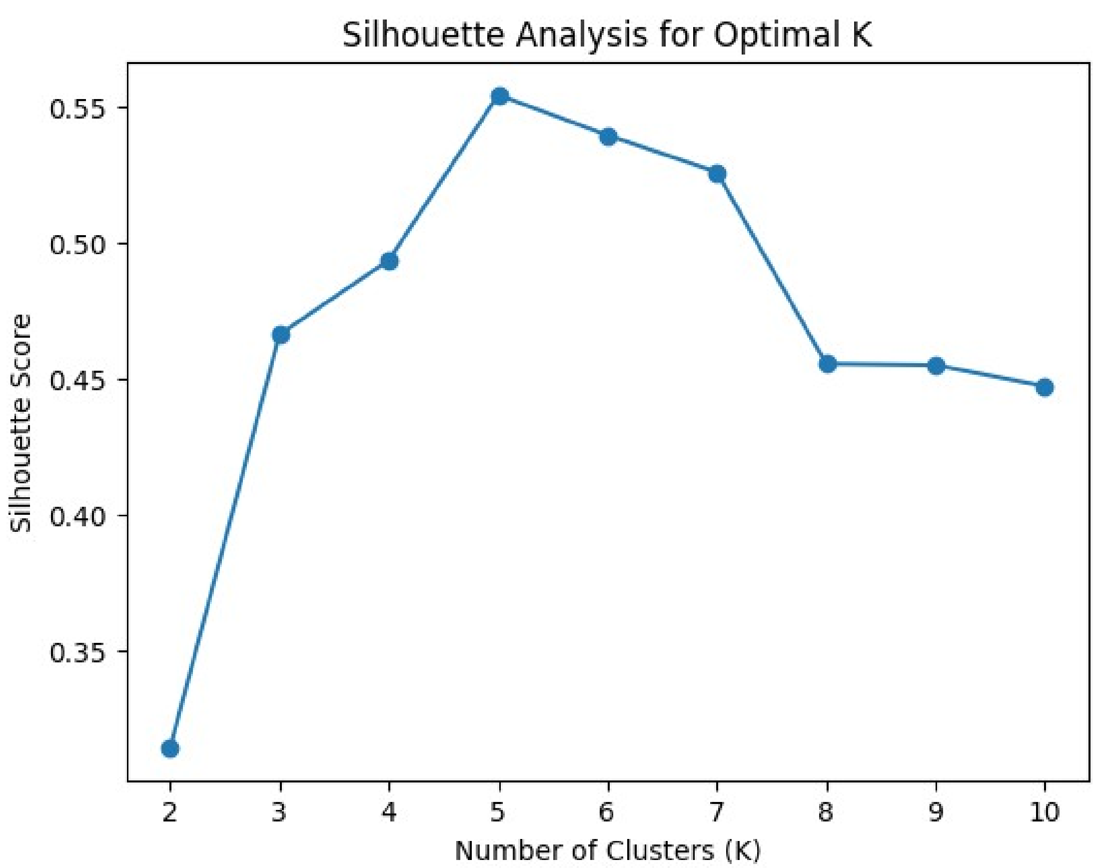
```
29    5
50    5
48    5
21    5
24    4
18    4
28    4
67    4
59    4
54    4
43    3
60    3
45    3
39    3
33    3
37    3
22    3
25    3
46    3
68    3
52    2
44    2
66    2
57    2
26    2
53    2
42    2
63    2
70    2
51    2
58    2
65    2
41    2
55    1
69    1
64    1
56    1
Name: Age, dtype: int64

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
```

```
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

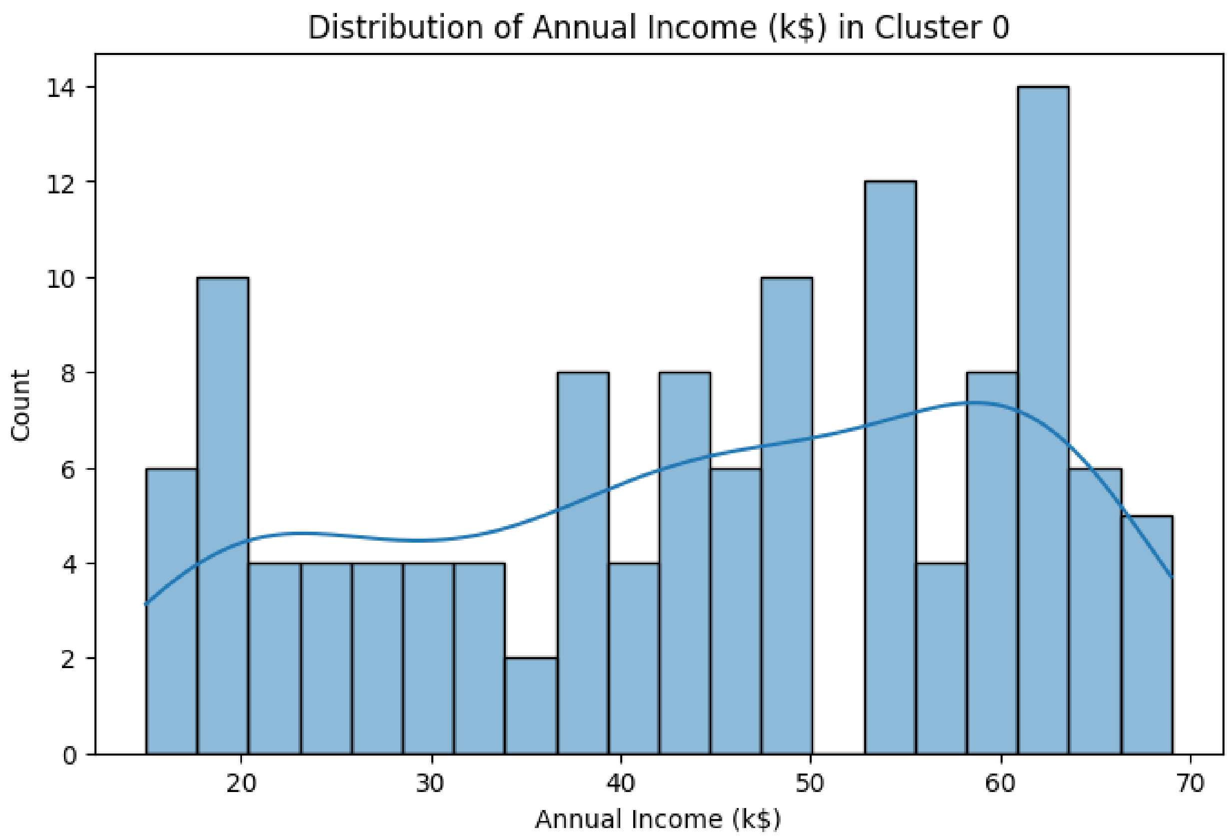



```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
```

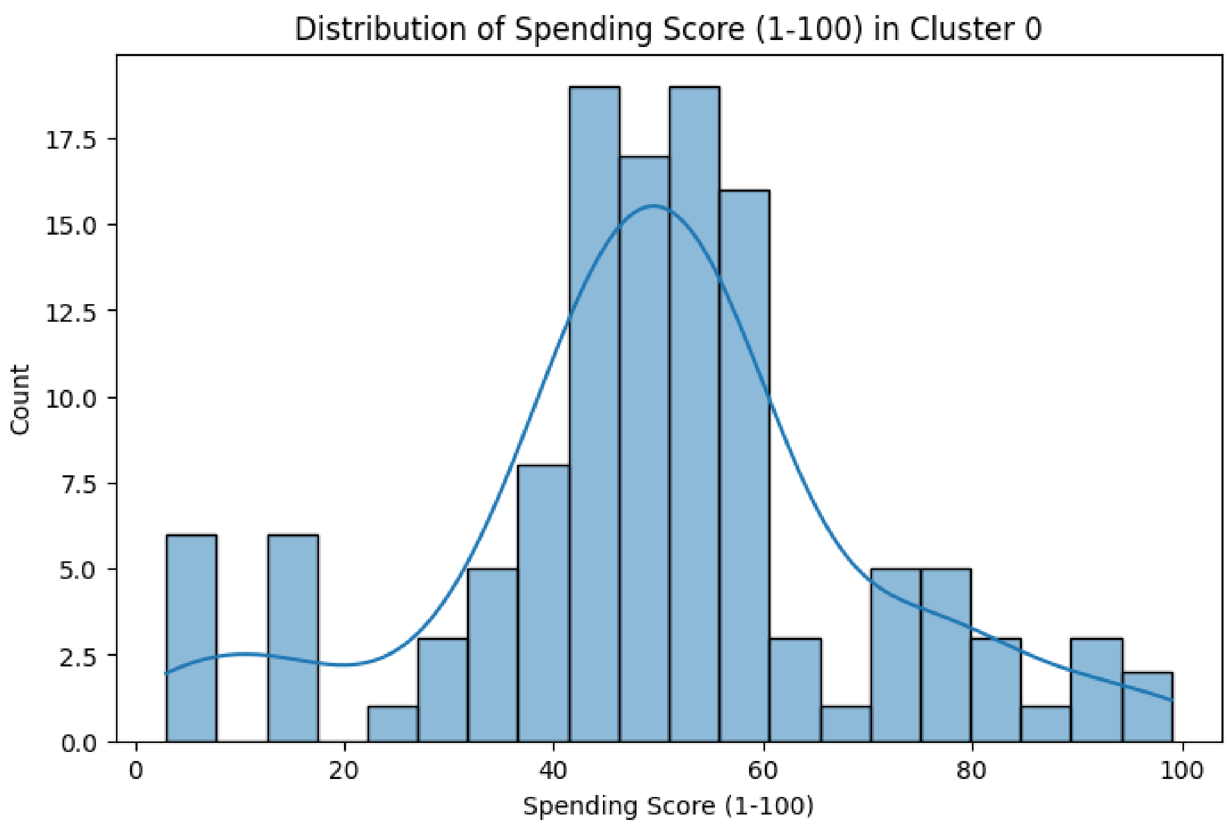
```
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
: FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
<ipython-input-101-e6001476550b>:102: FutureWarning: The default value
of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the
function.
  cluster_means = df.groupby('cluster').mean()


Cluster 0 Summary:
       CustomerID         Age  Annual Income (k$)  Spending Score (1-
100)  \
count  123.000000  123.000000          123.000000
123.000000
mean    62.000000   40.325203           44.154472
49.829268
std     35.651087   16.113580           16.037882
19.694265
min      1.000000   18.000000           15.000000
3.000000
25%     31.500000   24.500000           30.000000
42.000000
50%     62.000000   38.000000           46.000000
50.000000
75%     92.500000   51.500000           59.500000
58.500000
max    123.000000   70.000000           69.000000
99.000000

       new_feature  scaled_income  scaled_spending  cluster
count   123.000000     123.000000       123.000000    123.0
mean   2197.357724      -0.626190        -0.014392      0.0
std     972.066821       0.612157         0.764562      0.0
min      57.000000      -1.738999        -1.832378      0.0
25%    1680.000000      -1.166458        -0.318337      0.0
50%    2448.000000      -0.555747        -0.007764      0.0
75%    2847.000000      -0.040460         0.322219      0.0
max    4002.000000       0.322150         1.894492      0.0
```

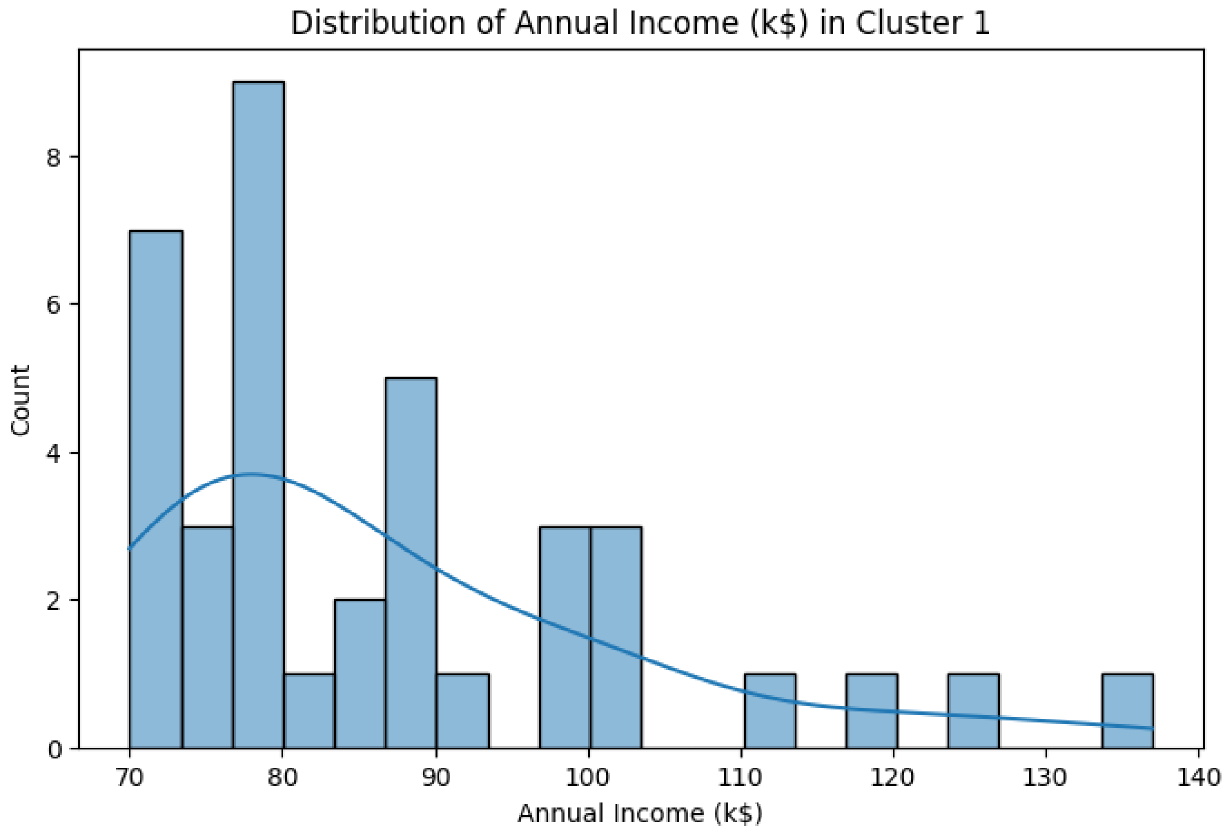


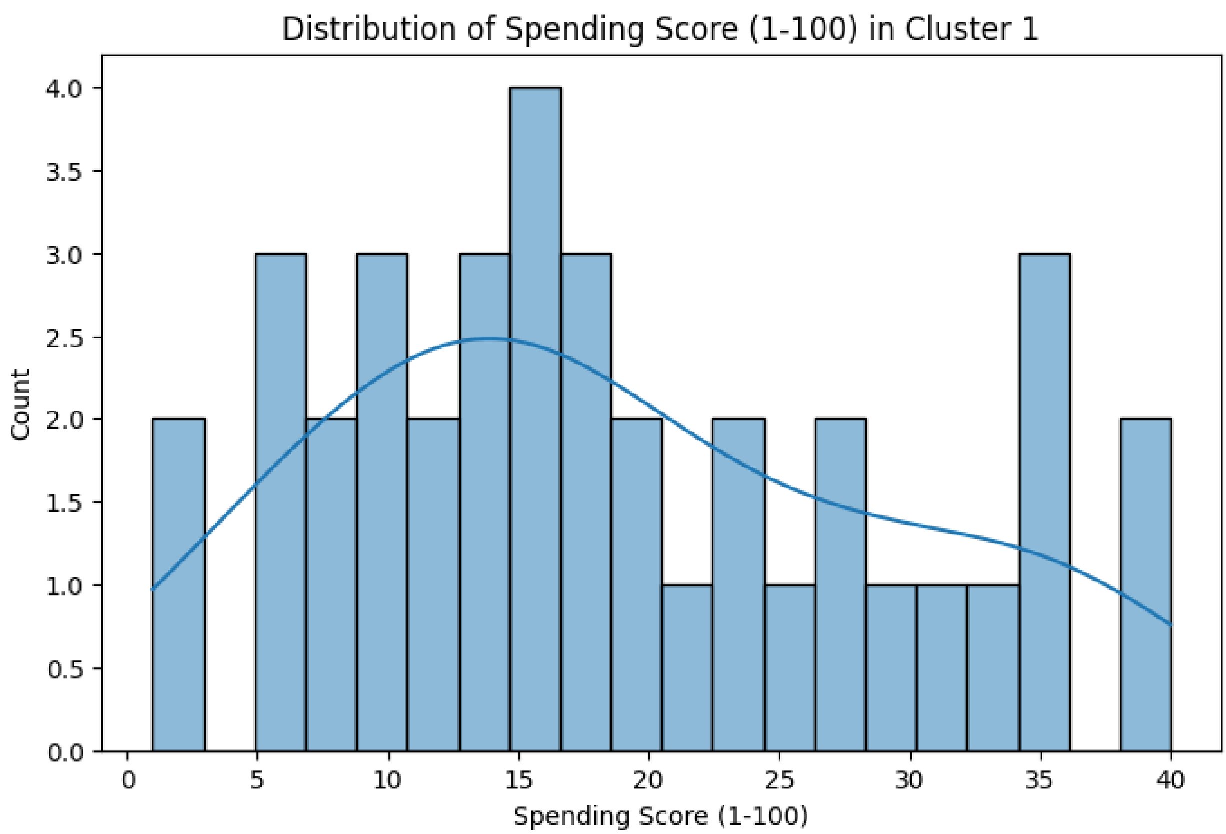
Distribution of Annual Income (k$) in Cluster 0
Count
Annual Income (k$)
0
2
4
6
8
10
12
14
20
30
40
50
60
70



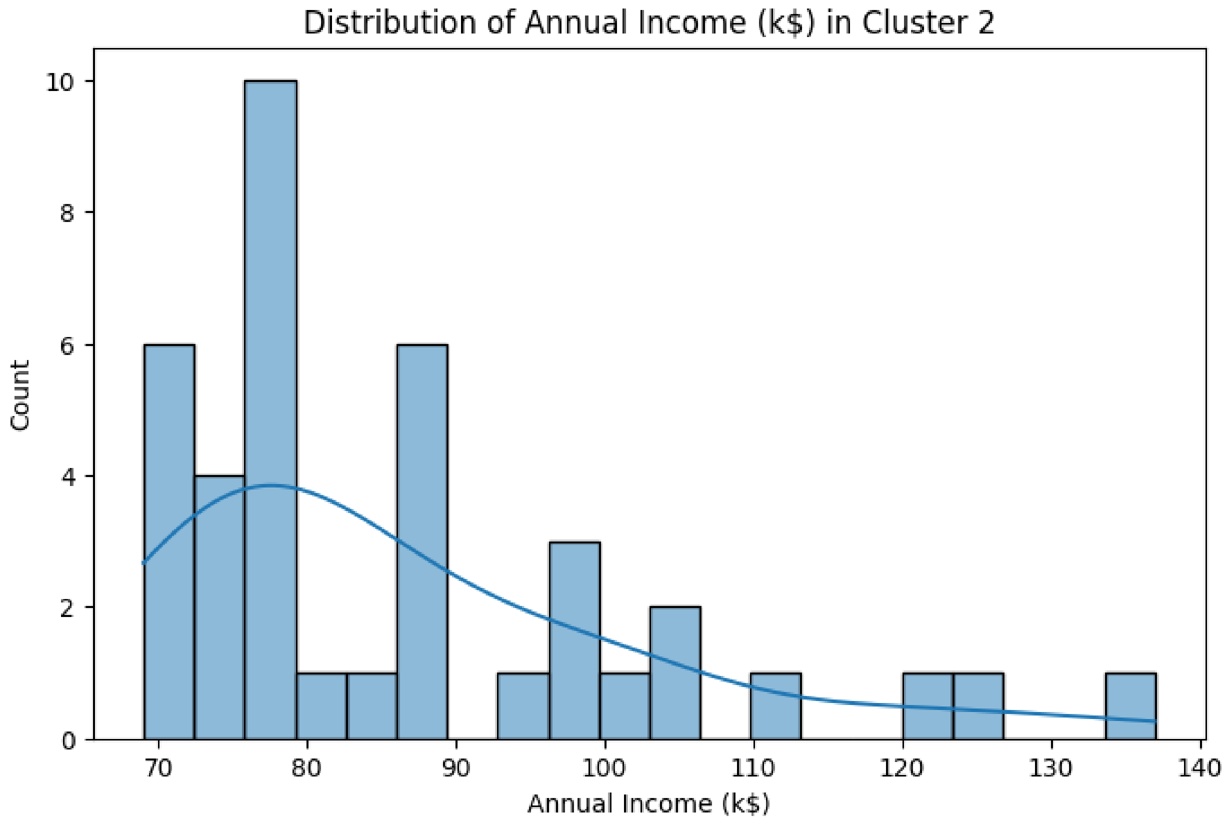
Distribution of Spending Score (1-100) in Cluster 0
Count
Spending Score (1-100)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
20
40
60
80
100

```
Cluster 1 Summary:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-
100)  \
count   38.000000  38.000000           38.000000
38.000000
mean   162.000000  40.394737           87.000000
18.631579
std     22.226111  11.376931           16.271348
10.915947
min    125.000000  19.000000           70.000000
1.000000
25%    143.500000  34.000000           76.250000
10.250000
50%    162.000000  41.500000           80.000000
16.500000
75%    180.500000  47.000000           96.000000
26.750000
max    199.000000  59.000000          137.000000
40.000000

       new_feature  scaled_income  scaled_spending  cluster
count    38.000000      38.000000        38.000000     38.0
mean   1634.973684       1.009200        -1.225535      1.0
std     980.022738       0.621068         0.423774      0.0
min      78.000000       0.360319        -1.910021      1.0
25%     878.500000       0.598878        -1.550921      1.0
50%    1515.000000       0.742014        -1.308287      1.0
75%    2410.250000       1.352725        -0.910366      1.0
max    3861.000000       2.917671        -0.395980      1.0
```

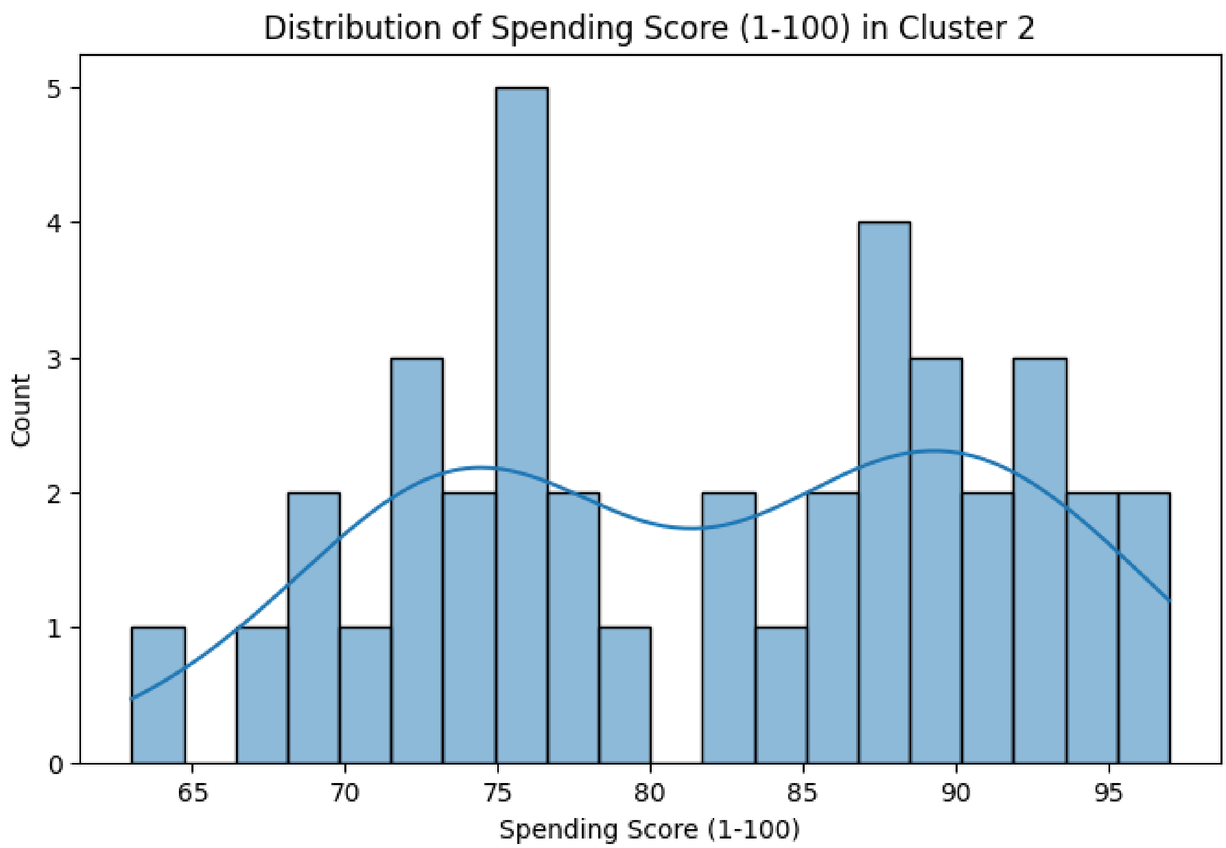

Distribution of Annual Income (k$) in Cluster 1
Count
Annual Income (k$)
0
2
4
6
8
70
80
90
100
110
120
130
140


Distribution of Spending Score (1-100) in Cluster 1
Count
Spending Score (1-100)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
5
10
15
20
25
30
35
40

```
Cluster 2 Summary:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-
100)  \
count   39.000000  39.000000           39.000000
39.000000
mean   162.000000  32.692308           86.538462
82.128205
std     22.803509   3.728650           16.312485
9.364489
min    124.000000  27.000000           69.000000
63.000000
25%    143.000000  30.000000           75.500000
74.500000
50%    162.000000  32.000000           79.000000
83.000000
75%    181.000000  35.500000           95.000000
90.000000
max    200.000000  40.000000          137.000000
97.000000

        new_feature  scaled_income  scaled_spending  cluster
count     39.000000      39.000000        39.000000     39.0
mean    7101.410256       0.991583         1.239503      2.0
std     1523.960210       0.622638         0.363544      0.0
min     5112.000000       0.322150         0.496916      2.0
25%     6000.000000       0.570251         0.943364      2.0
50%     6864.000000       0.703844         1.273347      2.0
75%     8087.000000       1.314555         1.545098      2.0
max    11371.000000       2.917671         1.816849      2.0
```



Distribution of Annual Income (k$) in Cluster 2
Count
Annual Income (k$)
0
2
4
6
8
10
70
80
90
100
110
120
130
140




```
Additional Insights for Cluster 0:
{'Cluster': 0, 'Mean_Annual_Income': 44.15447154471545,
'Mean_Spending_Score': 49.829268292682926, 'Total_Customers': 123}

Recommendations:
- This cluster represents a group of high-value customers. Consider
targeted marketing to retain and upsell.

Additional Insights for Cluster 1:
{'Cluster': 1, 'Mean_Annual_Income': 87.0, 'Mean_Spending_Score':
18.63157894736842, 'Total_Customers': 38}

Recommendations:
- Customers in this cluster show interest in specific product
categories. Tailor promotions accordingly.

Additional Insights for Cluster 2:
{'Cluster': 2, 'Mean_Annual_Income': 86.53846153846153,
'Mean_Spending_Score': 82.12820512820512, 'Total_Customers': 39}

Recommendations:

Overall Insights:
```

```
{'Total_Customers': 200, 'Mean_Annual_Income': 60.56,
'Mean_Spending_Score': 50.2}

Overall Recommendations:
- Explore partnerships with brands/products popular among the majority
of customers.
- Consider optimizing store layouts based on common preferences
observed in the clusters.
```