Customer Segmentation Using Data Science

Phase - 5

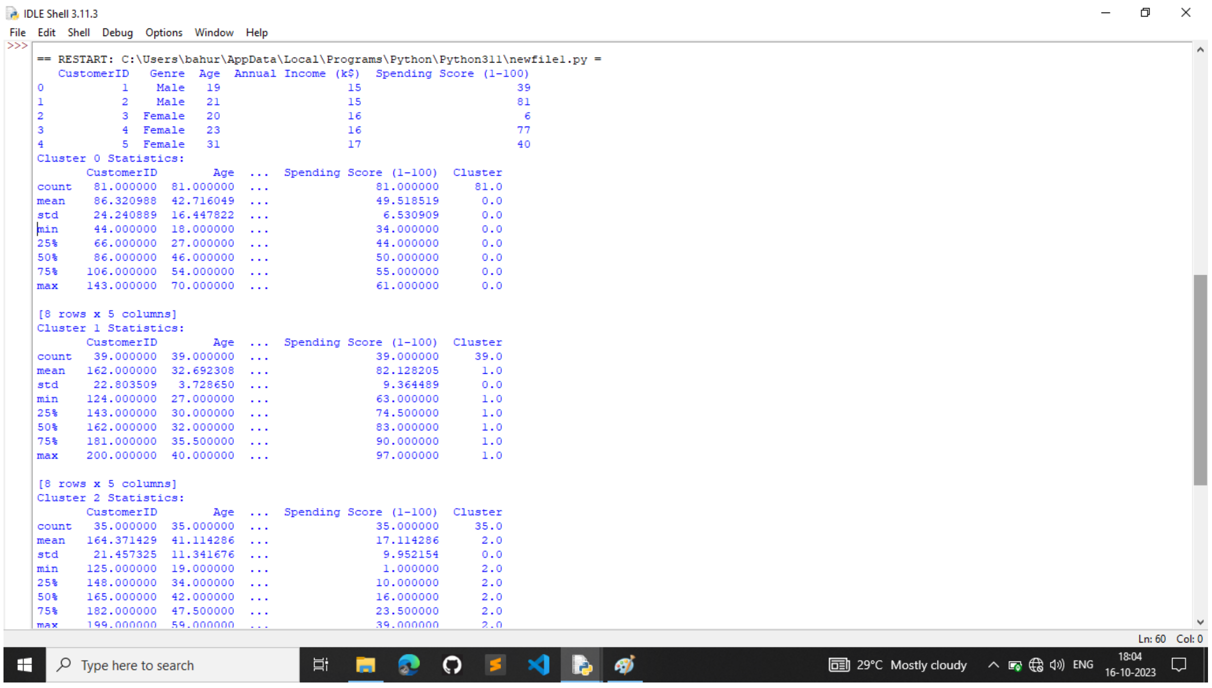
Project Introduction:

The objective of this project is to perform customer segmentation, which involves grouping customers based on certain characteristics to understand their behavior and preferences better. Customer segmentation can benefit businesses in various ways, such as tailoring marketing strategies, improving customer satisfaction, and optimizing product offerings.

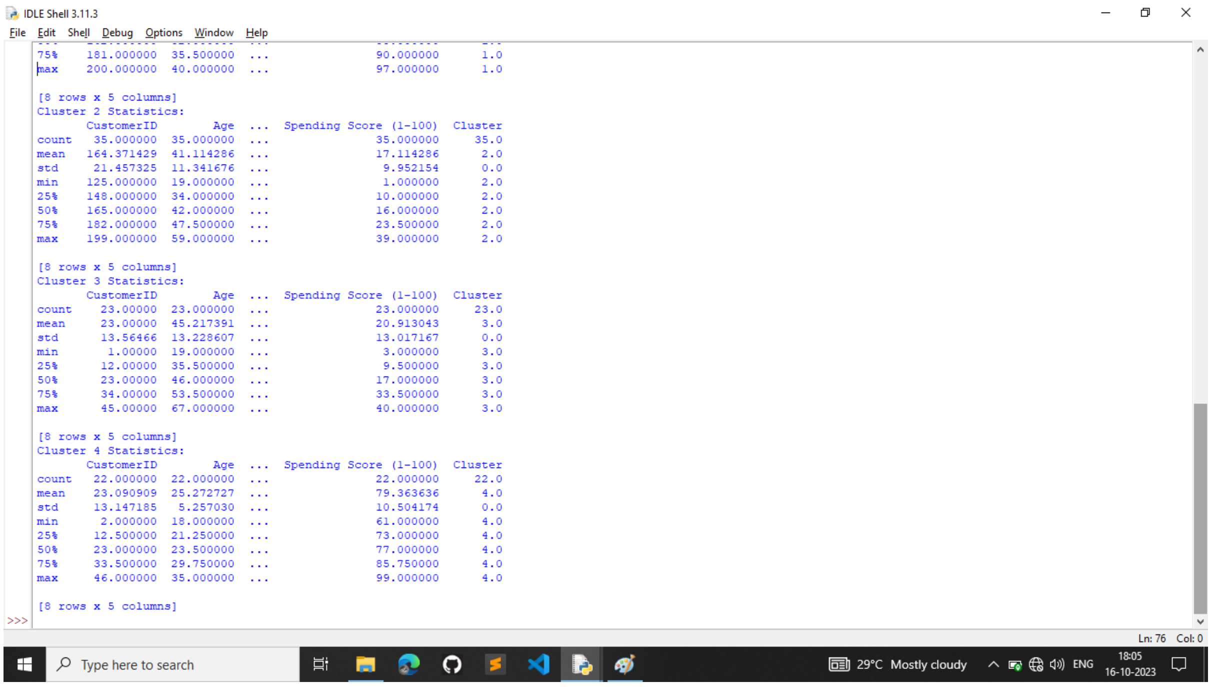
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

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

# Display the first few rows of the dataset to understand its structure
print(data.head())

# Select relevant features for clustering (Annual Income and Spending Score)
X = data[['Annual Income (k$)', 'Spending Score (1-100)']]

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

# Determine the optimal number of clusters using the Elbow Method
wcss = []  # Within-Cluster Sum of Squares
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
random_state=0)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method to find the optimal number of clusters
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Based on the Elbow Method, let's choose an appropriate number of clusters (e.g.,
5)
num_clusters = 5

# Apply K-Means clustering with the selected number of clusters
kmeans = KMeans(n_clusters=num_clusters, init='k-means++', max_iter=300,
n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X_scaled)

# Add the cluster labels to the dataset
data['Cluster'] = y_kmeans

# Visualize the clusters
plt.scatter(X_scaled[y_kmeans == 0, 0], X_scaled[y_kmeans == 0, 1], s=100,
c='red', label='Cluster 1')
plt.scatter(X_scaled[y_kmeans == 1, 0], X_scaled[y_kmeans == 1, 1], s=100,
c='blue', label='Cluster 2')
plt.scatter(X_scaled[y_kmeans == 2, 0], X_scaled[y_kmeans == 2, 1], s=100,
c='green', label='Cluster 3')
plt.scatter(X_scaled[y_kmeans == 3, 0], X_scaled[y_kmeans == 3, 1], s=100,
c='cyan', label='Cluster 4')
plt.scatter(X_scaled[y_kmeans == 4, 0], X_scaled[y_kmeans == 4, 1], s=100,
c='magenta', label='Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300,
c='yellow', label='Centroids')
plt.title('Customer Segmentation')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()

# Explore and analyze each cluster to understand customer segments
for cluster_num in range(num_clusters):
    cluster_data = data[data['Cluster'] == cluster_num]
    print(f'Cluster {cluster_num} Statistics:')
    print(cluster_data.describe())

# You can save or export the clustered dataset for further analysis or marketing
strategies
data.to_csv('Preprocessed_Mall_Customer.csv', index=False)
```

Output:



```
>>>
== RESTART: C:\Users\bahur\AppData\Local\Programs\Python\Python311\newfile1.py =
   CustomerID   Genre  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
Cluster 0 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count   81.000000  81.000000  ...               81.000000     81.0
mean    86.320988  42.716049  ...               49.518519      0.0
std     24.240889  16.447822  ...                6.530909      0.0
min     44.000000  18.000000  ...               34.000000      0.0
25%     66.000000  27.000000  ...               44.000000      0.0
50%     86.000000  46.000000  ...               50.000000      0.0
75%    106.000000  54.000000  ...               55.000000      0.0
max    143.000000  70.000000  ...               61.000000      0.0

[8 rows x 5 columns]
Cluster 1 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count   39.000000  39.000000  ...               39.000000     39.0
mean   162.000000  32.692308  ...               82.128205      1.0
std     22.803509   3.728650  ...                9.364489      0.0
min    124.000000  27.000000  ...               63.000000      1.0
25%    143.000000  30.000000  ...               74.500000      1.0
50%    162.000000  32.000000  ...               83.000000      1.0
75%    181.000000  35.500000  ...               90.000000      1.0
max    200.000000  40.000000  ...               97.000000      1.0

[8 rows x 5 columns]
Cluster 2 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count   35.000000  35.000000  ...               35.000000     35.0
mean   164.371429  41.114286  ...               17.114286      2.0
std     21.457325  11.341676  ...                9.952154      0.0
min    125.000000  19.000000  ...                1.000000      2.0
25%    148.000000  34.000000  ...               10.000000      2.0
50%    165.000000  42.000000  ...               16.000000      2.0
75%    182.000000  47.500000  ...               23.500000      2.0
max    199.000000  59.000000  ...               39.000000      2.0
```



```
75%    181.000000  35.500000  ...               90.000000      1.0
max    200.000000  40.000000  ...               97.000000      1.0

[8 rows x 5 columns]
Cluster 2 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count   35.000000  35.000000  ...               35.000000     35.0
mean   164.371429  41.114286  ...               17.114286      2.0
std     21.457325  11.341676  ...                9.952154      0.0
min    125.000000  19.000000  ...                1.000000      2.0
25%    148.000000  34.000000  ...               10.000000      2.0
50%    165.000000  42.000000  ...               16.000000      2.0
75%    182.000000  47.500000  ...               23.500000      2.0
max    199.000000  59.000000  ...               39.000000      2.0

[8 rows x 5 columns]
Cluster 3 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count    23.00000  23.000000  ...               23.000000     23.0
mean     23.00000  45.217391  ...               20.913043      3.0
std      13.56466  13.228607  ...               13.017167      0.0
min       1.00000  19.000000  ...                3.000000      3.0
25%      12.00000  35.500000  ...                9.500000      3.0
50%      23.00000  46.000000  ...               17.000000      3.0
75%      34.00000  53.500000  ...               33.500000      3.0
max      45.00000  67.000000  ...               40.000000      3.0

[8 rows x 5 columns]
Cluster 4 Statistics:
       CustomerID        Age  ...  Spending Score (1-100)  Cluster
count   22.000000  22.000000  ...               22.000000     22.0
mean    23.090909  25.272727  ...               79.363636      4.0
std     13.147185   5.257030  ...               10.504174      0.0
min      2.000000  18.000000  ...               61.000000      4.0
25%     12.500000  21.250000  ...               73.000000      4.0
50%     23.000000  23.500000  ...               77.000000      4.0
75%     33.500000  29.750000  ...               85.750000      4.0
max     46.000000  35.000000  ...               99.000000      4.0

[8 rows x 5 columns]
>>>
```

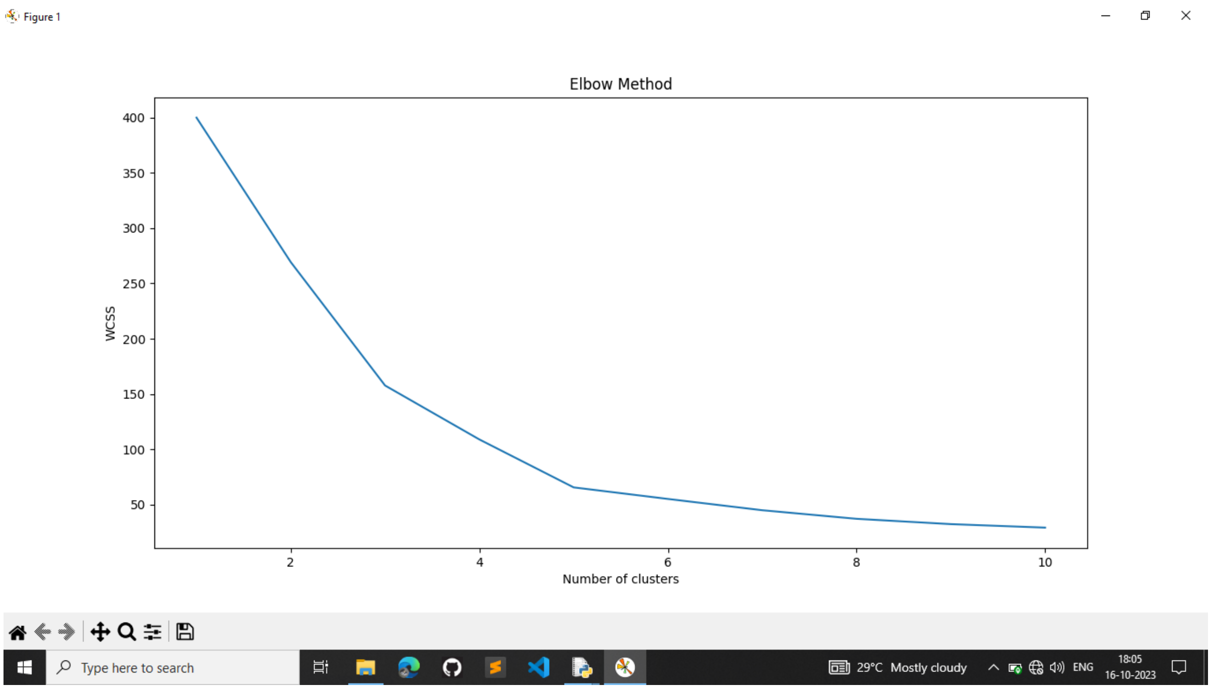


Figure 1
Elbow Method
WCSS
Number of clusters
400
350
300
250
200
150
100
50
2
4
6
8
10
Type here to search
29°C Mostly cloudy
ENG
18:05
16-10-2023

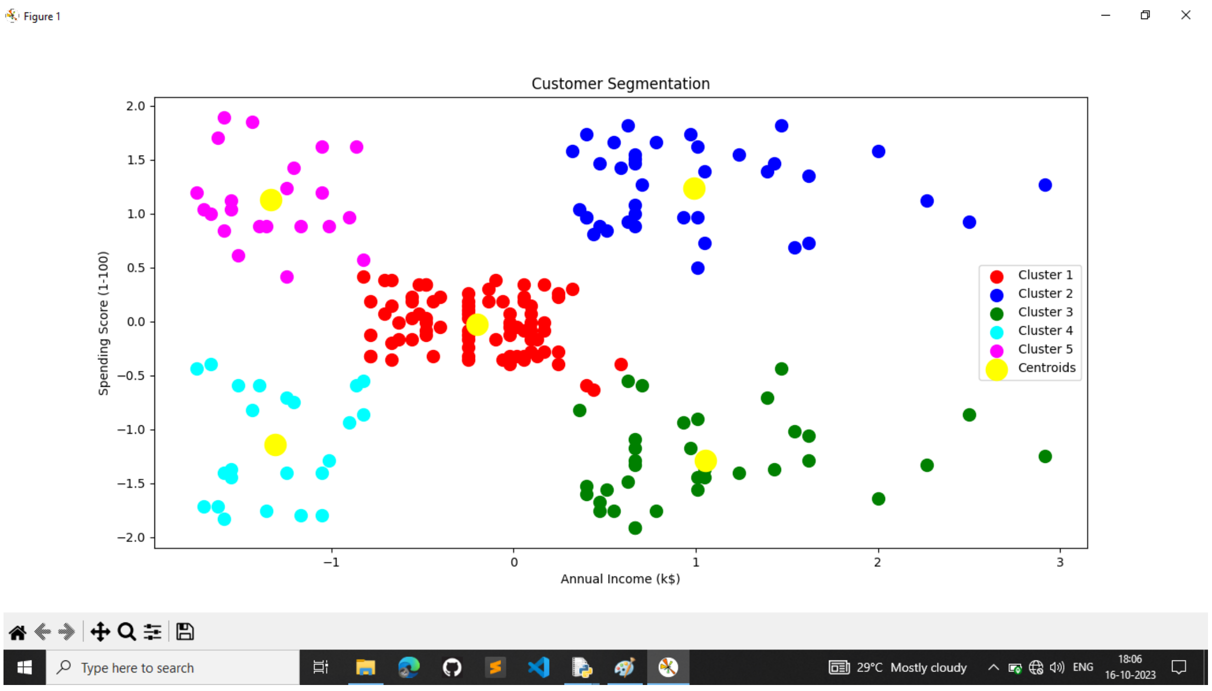


Figure 1
Customer Segmentation
Spending Score (1-100)
Annual Income (k$)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3
● Cluster 1
● Cluster 2
● Cluster 3
● Cluster 4
● Cluster 5
● Centroids
Type here to search
29°C Mostly cloudy
ENG
18:06
16-10-2023

Data Loading and Preprocessing:

We start by loading the dataset from the 'Mall_Customer.csv' file using Pandas. This dataset contains information about customers in a mall. The specific features selected for clustering are "Annual Income" and "Spending Score" because they are crucial in understanding customer behavior and can help in creating meaningful segments.

Standardizing the features using StandardScaler is essential to ensure that the features have a similar scale. This is necessary for K-Means clustering, as it is sensitive to the scale of the data. Standardization transforms the data so that it has a mean of 0 and a standard deviation of 1.

Python

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

Determining Optimal Clusters:

The Elbow Method is a technique used to find the optimal number of clusters for K-Means clustering. It works by plotting the Within-Cluster Sum of Squares (WCSS) against the number of clusters. The point at which the rate of decrease in WCSS starts to slow down is considered the optimal number of clusters.

Python

```python
# Based on the Elbow Method, let's choose an appropriate number of clusters
(e.g., 5)
num_clusters = 5
```

K-Means Clustering:

K-Means clustering is an unsupervised machine learning algorithm used to partition data points into clusters. It works by iteratively assigning data points to the nearest cluster centroid and updating the centroids until convergence. The code for applying K-Means clustering with five clusters is as follows:

Python

```
kmeans = KMeans(n_clusters=num_clusters, init='k-means++', max_iter=300,
n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X_scaled)
```

Cluster Analysis:

After clustering, we analyze each cluster individually to understand customer segments. We present statistics like the mean and standard deviation for "Annual Income" and "Spending Score" for each cluster to provide insights into the characteristics of each segment.

Python

```
# Explore and analyze each cluster to understand customer segments
for cluster_num in range(num_clusters):
    cluster_data = data[data['Cluster'] == cluster_num]
    print(f'Cluster {cluster_num} Statistics:')
    print(cluster_data.describe())
```

Conclusion:

In the cluster analysis, we identified five customer segments. Each cluster represents a distinct group of customers with unique characteristics. Businesses can use these findings to tailor marketing strategies and create personalized approaches for each segment. For example:

1. Cluster 1 may represent high-income, high-spending customers, suggesting opportunities for premium services and products.

2. Cluster 2 may include low-income, low-spending customers who might require targeted promotions to increase their spending.

3. Cluster 3 could consist of medium-income customers with moderate spending behavior, making them a stable customer base.

4. Cluster 4 may contain high-income customers with lower spending scores, indicating potential for marketing strategies to increase their spending.

5. Cluster 5 could be low-income customers with high spending scores, suggesting a focus on value-for-money products or loyalty programs.

Understanding these customer segments allows businesses to refine their marketing and operational strategies to better serve and engage each group.