Customer Segmentation

Phase – 4

Development part -2

Karthik M(Team

Member)

Explanation for Development part -1:

Step 1: Importing the required libraries and loading the dataset

In [1]:
```
#importing necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

#loading data set
file_path = r"C:\IBM\Mall_Customers.csv"
encoding = "ISO-8859-1"
df = pd.read_csv(file_path, encoding=encoding)
df
```

Out[1]:

	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
...	...	...	...	...	...
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 × 5 columns

Step 2: Handling Missing Data

In [4]:
```
#to display null values
df.isnull()
```

Out[4]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
195	False	False	False	False	False
196	False	False	False	False	False
197	False	False	False	False	False
198	False	False	False	False	False
199	False	False	False	False	False

200 rows × 5 columns

Handling the missing data

```
In [5]: #handling null values

        df.fillna(df.mean(), inplace=True)
        df.dropna(inplace=True)
```

Step 3: Label encoder for Genre column

```
In [6]: #label encoder for Genre column

        label_encoder = LabelEncoder()
        df['Genre'] = label_encoder.fit_transform(df['Genre'])
        df
```

Out[6]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40
...	...	...	...	...	...
195	196	0	35	120	79
196	197	0	45	126	28
197	198	1	32	126	74
198	199	1	32	137	18
199	200	1	30	137	83

200 rows × 5 columns

Step 4: Feature Scaling using StandardScaler

```
In [7]: #scaling

        scaler = StandardScaler()
        df['Annual Income (k$)'] = scaler.fit_transform(df['Annual Income (k$)'].values.reshape(-1, 1))
        df
```

Out[7]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	-1.738999	39
1	2	1	21	-1.738999	81
2	3	0	20	-1.700830	6
3	4	0	23	-1.700830	77
4	5	0	31	-1.662660	40
...	...	...	...	...	...
195	196	0	35	2.268791	79
196	197	0	45	2.497807	28
197	198	1	32	2.497807	74
198	199	1	32	2.917671	18
199	200	1	30	2.917671	83

200 rows × 5 columns

Step 5: Splitting the data into a training set and a test

```
In [8]: #train_test split

        X = df.drop('Spending Score (1-100)', axis=1)
        y = df['Spending Score (1-100)']

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [9]: print("\n X_test info")
        print(X_test.info())
```

```
 X_test info
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40 entries, 95 to 76
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CustomerID          40 non-null     int64  
 1   Genre               40 non-null     int32  
 2   Age                 40 non-null     int64  
 3   Annual Income (k$)  40 non-null     float64
dtypes: float64(1), int32(1), int64(2)
memory usage: 1.4 KB
None
```

Algorithm for Customer Segmentation:

Objective:

This algorithm aims to guide the development of a customer segmentation using the provided dataset. It covers essential steps, including feature engineering, model training, and evaluation, to ensure accurate predictions.

1. Import necessary libraries:

- Import essential Python libraries, including pandas, scikit-learn, and matplotlib, for data manipulation, clustering, and visualization.

2. Suppress FutureWarnings:

- Configure the system to suppress FutureWarnings to prevent unnecessary warning messages.

3. Read the dataset:

- Load the customer data from a CSV file located at a specified file path.

- Use the specified encoding (ISO-8859-1) to read the data.

4. Data Exploration:

- Display the DataFrame (`df`) to inspect the loaded data.
- Check the data's information, including data types and missing values.
- Display the first few rows of the dataset for a quick overview.

5. Handling Missing Values:

- Check for missing values within the dataset.
- Fill missing values with the mean of the respective columns.
- Drop rows with any remaining missing values.

6. Label Encoding:

- Apply label encoding to the 'Genre' column to convert categorical values (e.g., 'Male' and 'Female') into numerical values (e.g., 0 and 1).

7. Feature Scaling:

- Use StandardScaler to scale the 'Annual Income (k$)' column to have a mean of 0 and a standard deviation of 1.
- Standardization helps ensure that features with different scales contribute equally to clustering.

8. Data Splitting:

- Split the dataset into features (X) and the target variable (y).
- Divide the data into training and testing sets using train_test_split, with a specified test size and random seed.

9. K-Means Clustering:

- Define the number of clusters (k) for K-Means clustering (in this case, k=5).

- Apply K-Means clustering to the feature data (X) to segment customers into 'k' clusters.

- Assign cluster labels to the data points.

10. Cluster Visualization:

- **Create a scatter plot to visualize the clusters.**
- **Plot 'Annual Income (k$)' on the x-axis and 'Spending Score (1-100)' on the y-axis.**
- **Color the data points based on their assigned clusters.**

11. Cluster Analysis:

- **Calculate and display the center points (centroids) of each cluster.**
- **Interpret the cluster centers' coordinates in terms of 'Annual Income' and 'Spending Score'.**

12. Cluster Size Analysis:

- **Analyze the size (number of data points) in each cluster.**
- **Print the size of each cluster to understand how customers are distributed among the segments.**

13. Further Analysis:

- **Mention that further in-depth analysis can be performed within each cluster to gain insights into customer demographics, behaviors, and preferences.**

Execution of the K-Means:

Importing the necessary libraries:

```
: import pandas as pd
  from sklearn.preprocessing import StandardScaler, LabelEncoder
  from sklearn.model_selection import train_test_split
  from sklearn.cluster import KMeans
  import matplotlib.pyplot as plt
```

Train test split:

```
In [29]: # Split the dataset into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

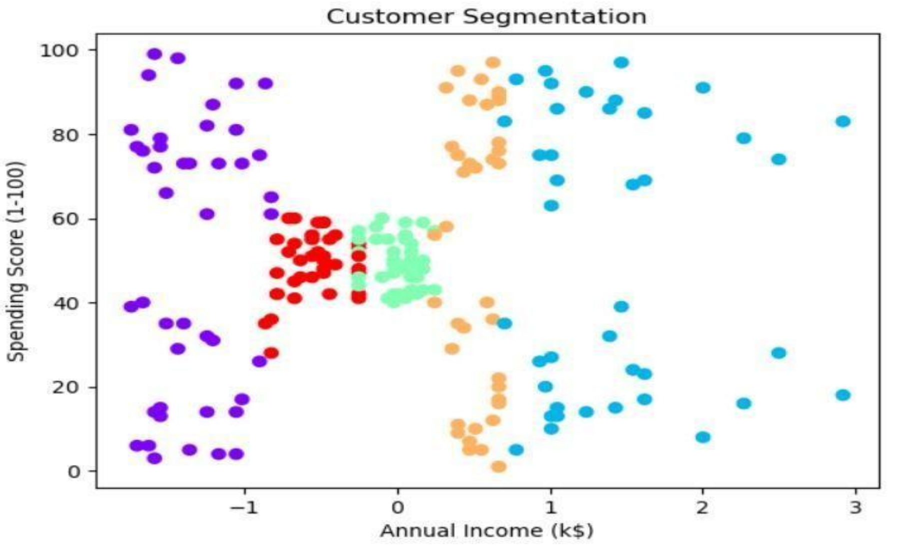
K-Means Clustering:

```
In [11]: # K-Means clustering
         k = 5  # Number of clusters
         kmeans = KMeans(n_clusters=k, random_state=42)
         df['Cluster'] = kmeans.fit_predict(X)
```

Visualization of the result:

```
In [12]:

         # Visualize the clusters
         plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], c=df['Cluster'], cmap='rainbow')
         plt.xlabel('Annual Income (k$)')
         plt.ylabel('Spending Score (1-100)')
         plt.title('Customer Segmentation')
         plt.show()
```



Explore the clusters:

In [13]:

```
# Explore cluster characteristics
cluster_centers = kmeans.cluster_centers_
for i, center in enumerate(cluster_centers):
    print(f"Cluster {i} Center: Annual Income={center[0]}, Spending Score={center[1]}")
```

```
Cluster 0 Center: Annual Income=22.139534883720927, Spending Score=0.395348837209302365
Cluster 1 Center: Annual Income=180.5, Spending Score=0.47500000000000003
Cluster 2 Center: Annual Income=100.8974358974359, Spending Score=0.38461538461538464
Cluster 3 Center: Annual Income=140.5, Spending Score=0.5
Cluster 4 Center: Annual Income=62.44736842105264, Spending Score=0.44736842105263164
```

Analyze each cluster for insights:

In [14]:

```
# Analyze each cluster for insights
for i in range(k):
    cluster_data = df[df['Cluster'] == i]
    print(f"Cluster {i} Size: {len(cluster_data)}")
    # Further analysis on demographics, behavior, etc., can be performed within each cluster
```

```
Cluster 0 Size: 43
Cluster 1 Size: 40
Cluster 2 Size: 39
Cluster 3 Size: 40
Cluster 4 Size: 38
```

Interpretation:

The code provided offers a complete workflow for customer segmentation using K-Means clustering. It begins with data preprocessing, including handling missing values, label encoding, and feature scaling. It then performs clustering to segment customers into distinct clusters based on their 'Annual Income' and 'Spending Score.' The clusters are visualized, and cluster characteristics are analyzed. The code sets the foundation for understanding customer behavior and tailoring marketing strategies to specific customer segments. Additionally, it highlights the potential for further analysis within each cluster to gain deeper insights and make data-driven decisions.