Customer Segmentation using Data Science

TEAM Member : Sakthibala A

TNM ID : aut22leai03

Customer Segmentation:

It is the process of grouping customers according to how and why they are buy products.

Problem Definition:

The problem is to implement data science techniques to segment customers based on their behavior, preferences, and demographic attributes. The goal is to enable businesses to personalize marketing strategies and enhance customer satisfaction. This project involves data collection, data preprocessing, feature engineering, clustering algorithms, visualization, and interpretation of results.

Main Objectives:

- The main goal for the customer segmentation using data science is to divide the customer base into distinct groups based on similar characteristics.
- This segment will helpful for many Business purpose.

Project Phases:

1. Data Collection
2. Data Preprocessing
3. Feature Engineering
4. Visualization
5. Interpretation of Results

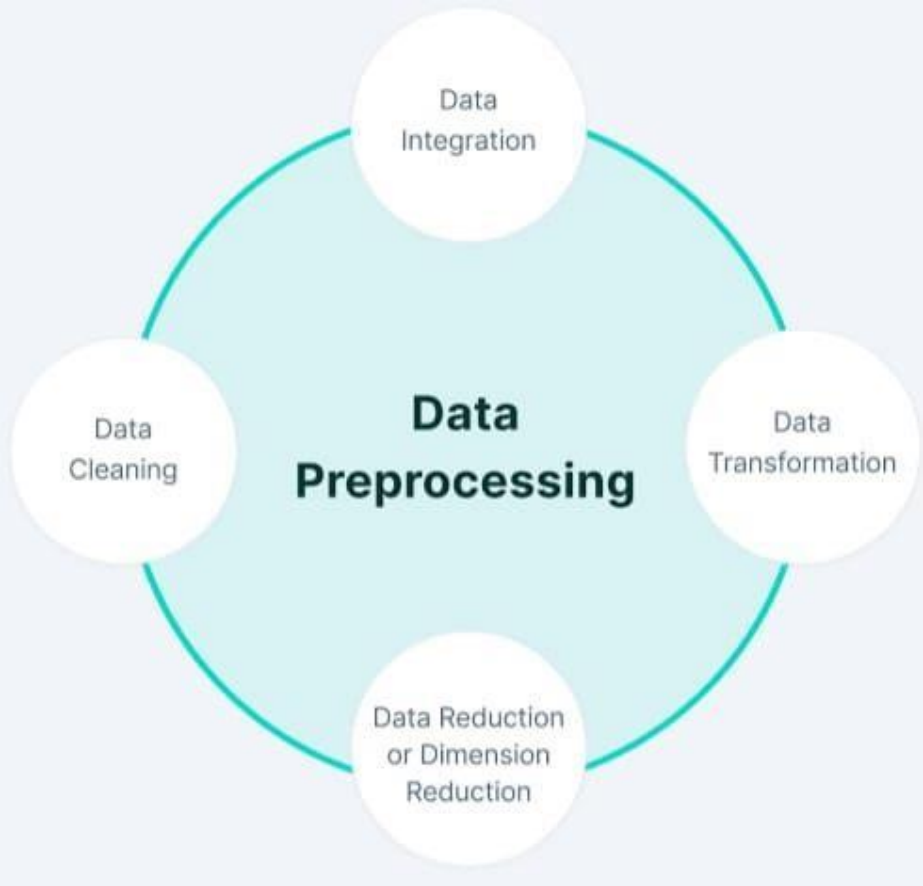
Phase 1: Data Collection

- The main goal for the data collection phase is gathering the necessary data sources for customer segmentation.
- Collecting the customer transaction data are should be include here.
- This phase also collecting the data about the behavior of the customer.
- Huge volumes of data are needed for analysis.
- Example : Considering our given Mall dataset, we have collect the each customer's annual income for predicting the spending time.

Phase 2: Data Preprocessing

- After collecting the data that should be well prepared and clean for analysis.
- Handling the missing values, outliers, and data inconsistencies are should be including here.
- It transform the data through scaling, encoding categorical variables, and feature engineering.
- It integrate the data from different sources.
- Data cleaning, Data integration, Data transformation and Dimension reduction are the important factors in Data Preprocessing.

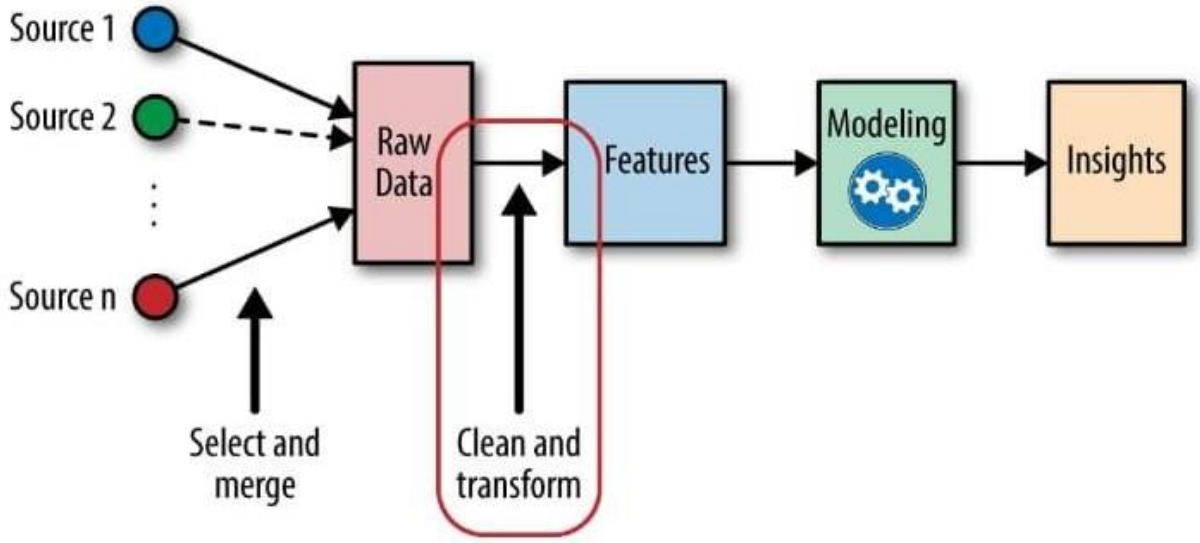
- Example: Considering Mall dataset we have to find out if any outliers or any missing values. The perfect data will output a perfect result.



Phase 3: Feature Engineering

- The main objective of the feature engineering stage is to create the relevant features that capture customer behavior and preferences.

- It also generate new features based on customer interactions and demographics.
- It reduce the dimensionality in case of the necessary situation.



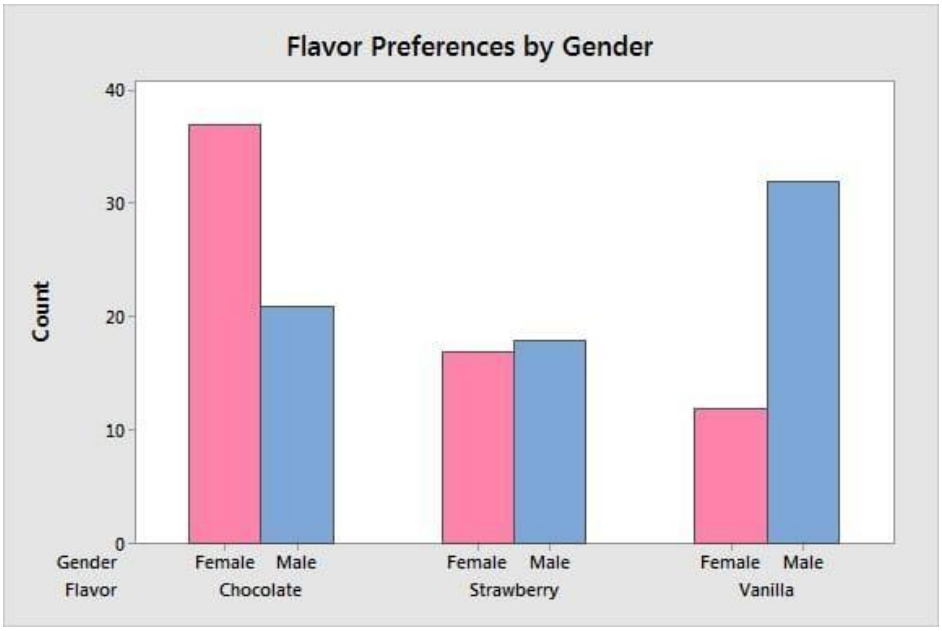
Phase 4: Visualization

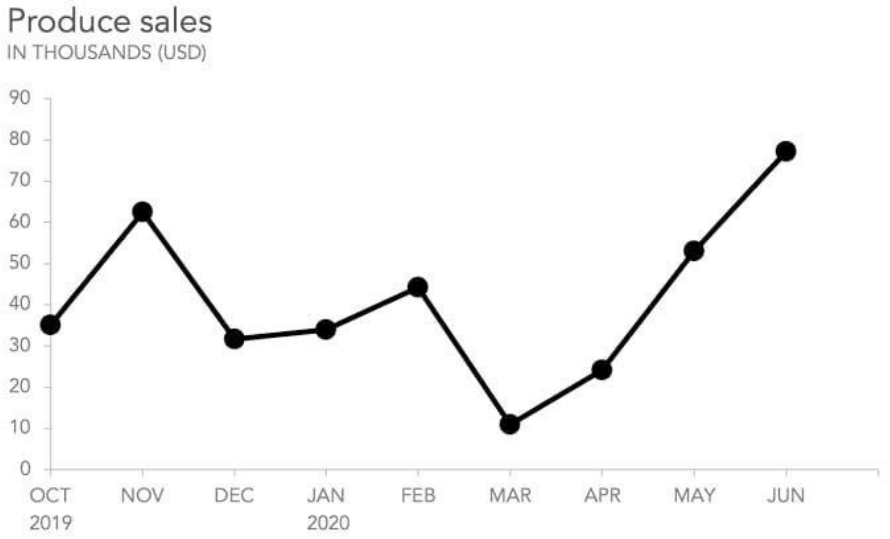
- Visualization is one of the key concept in data science which can be used for give the pictorial or virtual representation about the data.
- Several plots are used for visualize the customer segments.
- Plots example:

1. Bar Chart
2. Scatter Plot
3. Pie Plot
4. Line Plot
5. Histogram

- In python, Matplotlib library used for the visualization.
- Using these charts we can clearly virtualize our Mall Dataset especially Bar chart is used in popularly for virtualize the dataset.
- Syntax,

" Import matplotlib.pyplot as plt "





Phase 5: Interpretation of Results

- The goal of this phase is to interpret customer segments and derive actionable insights.
- It identifies the distinguishing characteristics for each segment.
- Profile each customer segment regarding behavior, preferences, and demographics.
- It also formulates the personalized marketing strategies for each segment.

According to given Dataset:

Mall Customers Dataset

Considering the mall Customers dataset we can analysis the customer's spending time.

Given data,

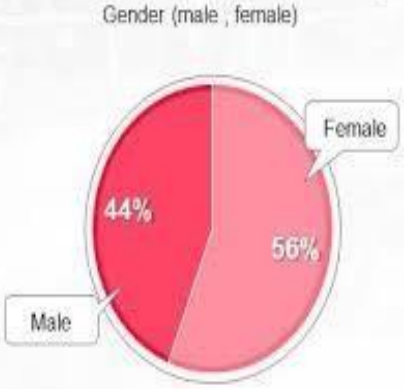
1. Customer_Id
2. Age
3. Annual Income
4. Spending Score

- If the customer's annual income will increases then they are spending more time comparing to before.
- Considering our dataset, female having a majority for spending more time in a mall.

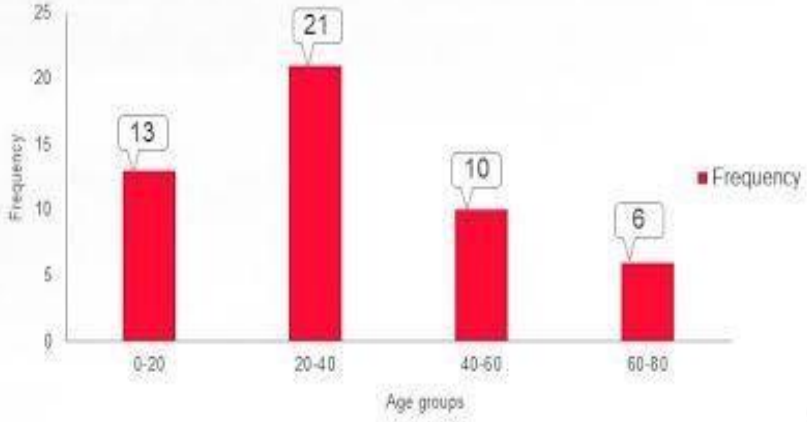
Customer segmentation for shopping mall customers

This slide represents customer segmentation for shopping mall customers. It include information regarding segmentation on the basis of gender and age group.

Customer segmentation on the basis of gender



Customer segmentation on the basis of age groups

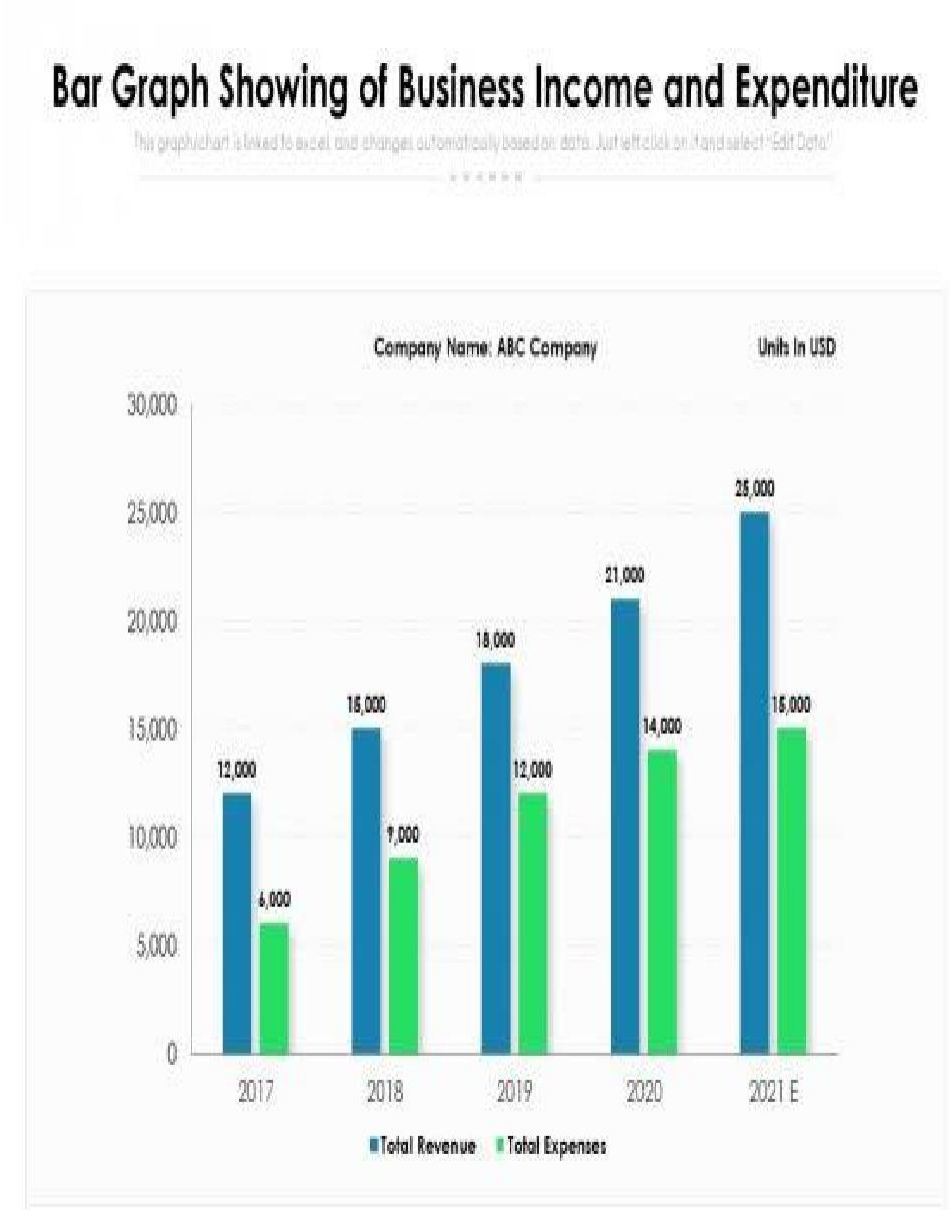


Key insight

Female customers are more for shopping center by **12%**

Age group for **20-40** years have more frequency visits at shopping center

Reason group with high and regular income



Dataset:

Dataset link :(https://www.kaggle.com/datasets/akram24/mall-customers)

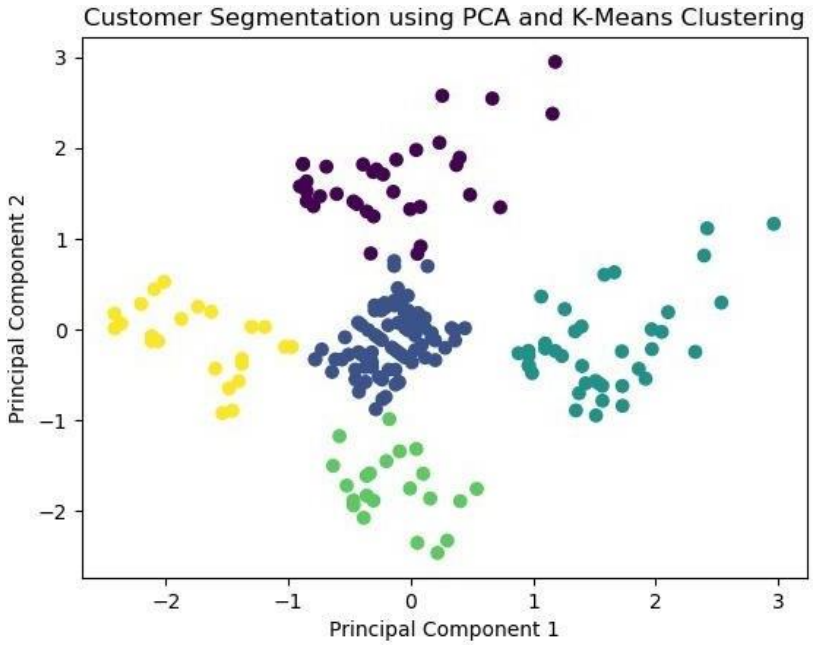
CustomerI	Genre	Age	Annual Inc	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35
18	Male	20	21	66
19	Male	52	23	29
20	Female	35	23	98
21	Male	35	24	35
22	Male	25	24	73
23	Female	46	25	5
24	Male	31	25	73
25	Female	54	28	14
26	Male	29	28	82
27	Female	45	28	32
28	Male	35	28	61

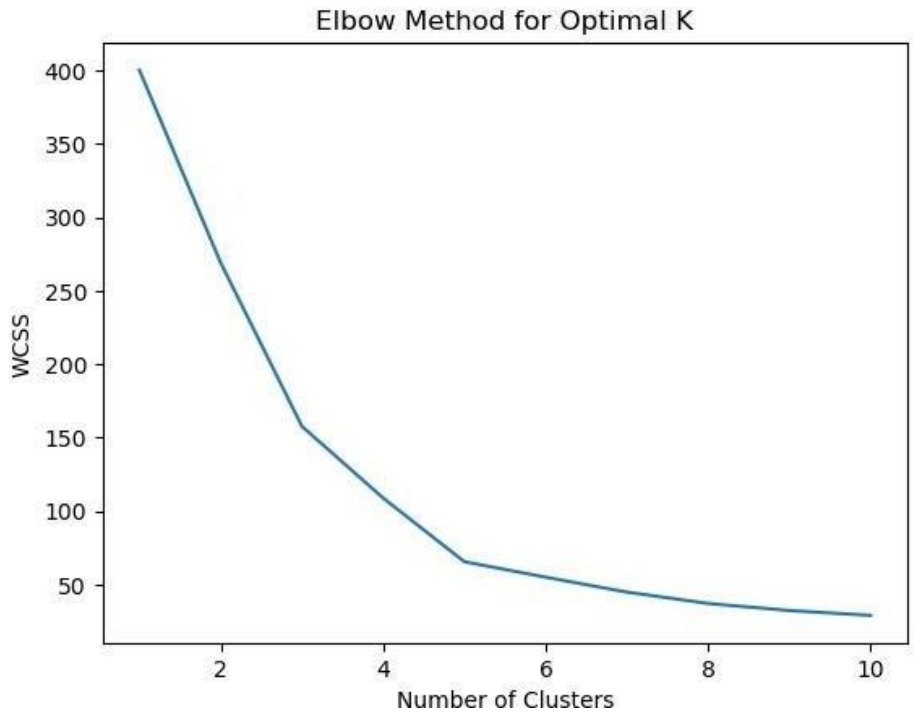
Principal Component Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import
StandardScaler from sklearn.decomposition
import PCA
from sklearn.cluster import
KMeans # Load the mall
customer dataset
data = pd.read_csv("C:\\Users\\Student\\Downloads\\archive
(1)\\Mall_Customers.csv") # Select relevant features for segmentation
```

```
X = data[['Annual Income (k$)', 'Spending Score
(1-100)']] # Standardize the data
scaler = StandardScaler()
X_scaled =
scaler.fit_transform(X)
# Perform PCA to reduce
dimensionality pca =
PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Determine the optimal number of clusters using the
"elbow method" wcss = []
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_pca)
   wcss.append(kmeans.inertia_
   )
# Plot the Within-Cluster-Sum-of-Squares (WCSS) to find the elbow
point plt.plot(range(1, 11), wcss)
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal
K') plt.show()
```

```
# Based on the plot, choose the optimal number of

clusters optimal_k = 5

# Perform K-Means clustering with the chosen number of clusters

kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300,
n_init=10, random_state=0)

cluster_labels = kmeans.fit_predict(X_pca)

# Add the cluster labels to the original dataset

data['Cluster'] = cluster_labels

# Visualize the clusters

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=data['Cluster'], cmap='viridis')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')
plt.title('Customer Segmentation using PCA and K-Means

Clustering') plt.show()
```





Data Loading

```
import pandas as pd
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
data.head(5)
```

	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

Data Preprocessing :

Data preprocessing is a important step in data analysis and machine learning. It involves cleaning, integrating, and transforming raw data into a format suitable for analysis or model training. Data preprocessing is essential because real-world data is often incomplete and inconsistent. By preparing the data properly, we can improve the quality and reliability of our analysis or machine learning models.

Techniques :

1) Data Cleaning
2) Data Transformation
3) Data Reduction

Data Cleaning :

- Data cleaning is also known as data cleansing or data scrubbing. It is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset. It is a crucial step in data preprocessing, necessary to ensure that the data used for analysis or machine learning is accurate, reliable, and free from noise.
- **Handling Missing Data:** Identifying and dealing with missing values in the dataset. This can involve filling in missing values, removing rows or columns with too many missing values
- **Removing Duplicates:** Identifying and removing duplicate records or observations from the dataset.

Correcting “Gender” Column name

```
import pandas as pd
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
df=df.rename(columns={'Genre':'Gender'})
df.head()
```

Checking the null values

```
import pandas as pd
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
df.isnull().sum()
```

Output :

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

```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Data Transformation :

Data transformation is the context of data analysis, refers to the process of converting data from one format, structure, or

representation into another. This transformation is performed to make the data suitable for analysis, reporting, visualization, or modeling.

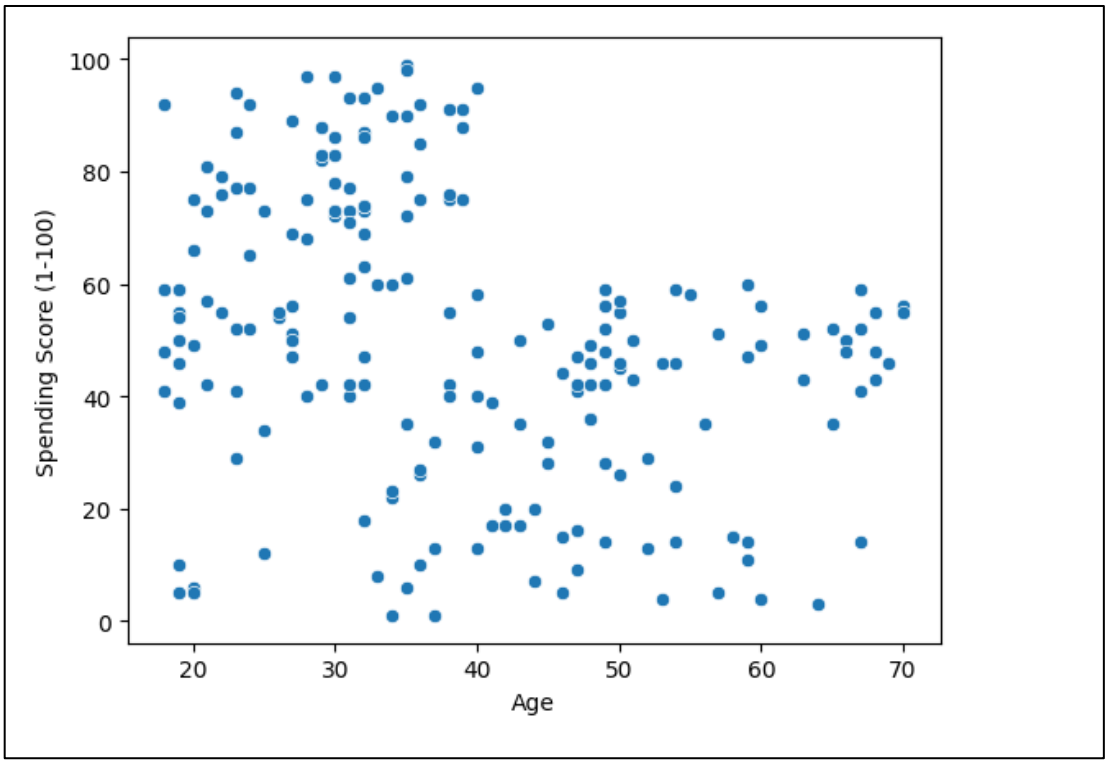
Describing the dataset

```
import pandas as pd
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
df.describe()
```

	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

Scatter Plot

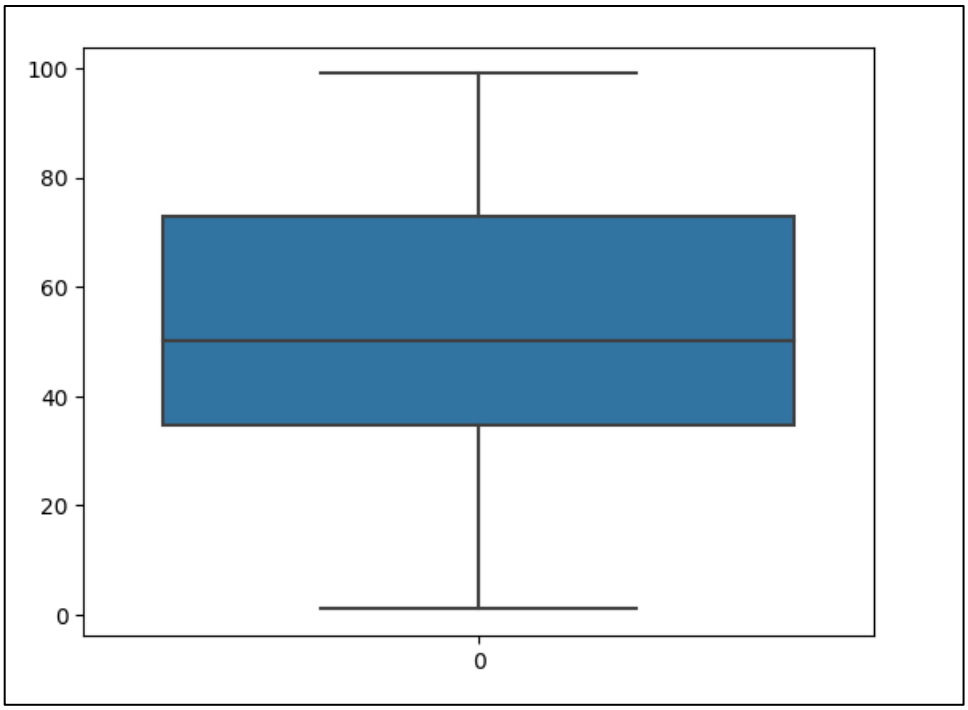
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
sns.scatterplot(x='Age',y='Spending Score (1-100)',data=df)
```



Box Plot

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
sns.boxplot(df['Spending Score (1-100)'])
df['Spending Score (1-100)'])
```

	CustomerID	Age	Genre
0	1	19	Male
1	2	21	Male
2	3	20	Female
3	4	23	Female
4	5	31	Female



Data Reduction :

Data reduction is a process used in data analysis and data mining to decrease the volume but produce the same or similar analytical results.It is also called as “Dimensionality Reduction”.

Reducing the Column

```
import pandas as pd
mall_data = pd.read_csv("C:\\Users\\Student\\Downloads\\archive (1)\\Mall_Customers.csv")
selected_features = ["CustomerID", "Age", "Genre"]
reduced_mall_data = mall_data[selected_features]
reduced_mall_data.head
```

Exploratory Data Analysis:

Exploratory Data Analysis (EDA) for customer segmentation in a mall dataset involves first importing and cleaning the data. Then, it requires summarizing key variables and employing data visualization techniques to understand customer characteristics. EDA helps identify patterns, correlations, and outliers, which are essential for informed decision-making. Subsequently, relevant variables are chosen for segmentation, and clustering techniques are applied to create customer segments. These segments are then interpreted to understand the distinct characteristics of each group. EDA facilitates insights into shopping behavior and assists in tailoring marketing strategies and store layouts to better meet the diverse needs of mall customers, enhancing business outcomes.

Program :

```
# import required packages
import numpy as
np import pandas
as pdimport os
import matplotlib.pyplot
as pltimport seaborn as
sns
sns.set(context="notebook", palette="Spectral", style = 'darkgrid'
```

```
,font_scale = 1.5,color_codes=True)
from sklearn.linear_model import
LinearRegressionfrom
sklearn.preprocessing import
MinMaxScaler from tensorflow import
keras
from keras.models import
Sequential from keras.layers
import LSTM, Dense
from sklearn.metrics import mean_squared_error

# import dataset
df=pd.read_csv("C:/Users/ASUS/Downloads/archive/Mall_Custo
mers.csv") df.head()
df.tail()
```

In [86]:
```
df=pd.read_csv("C:/Users/ASUS/Downloads/archive/Mall_Customers.csv")
df.head()
```

Out[86]:

	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

In [62]:
```
df.tail()
```

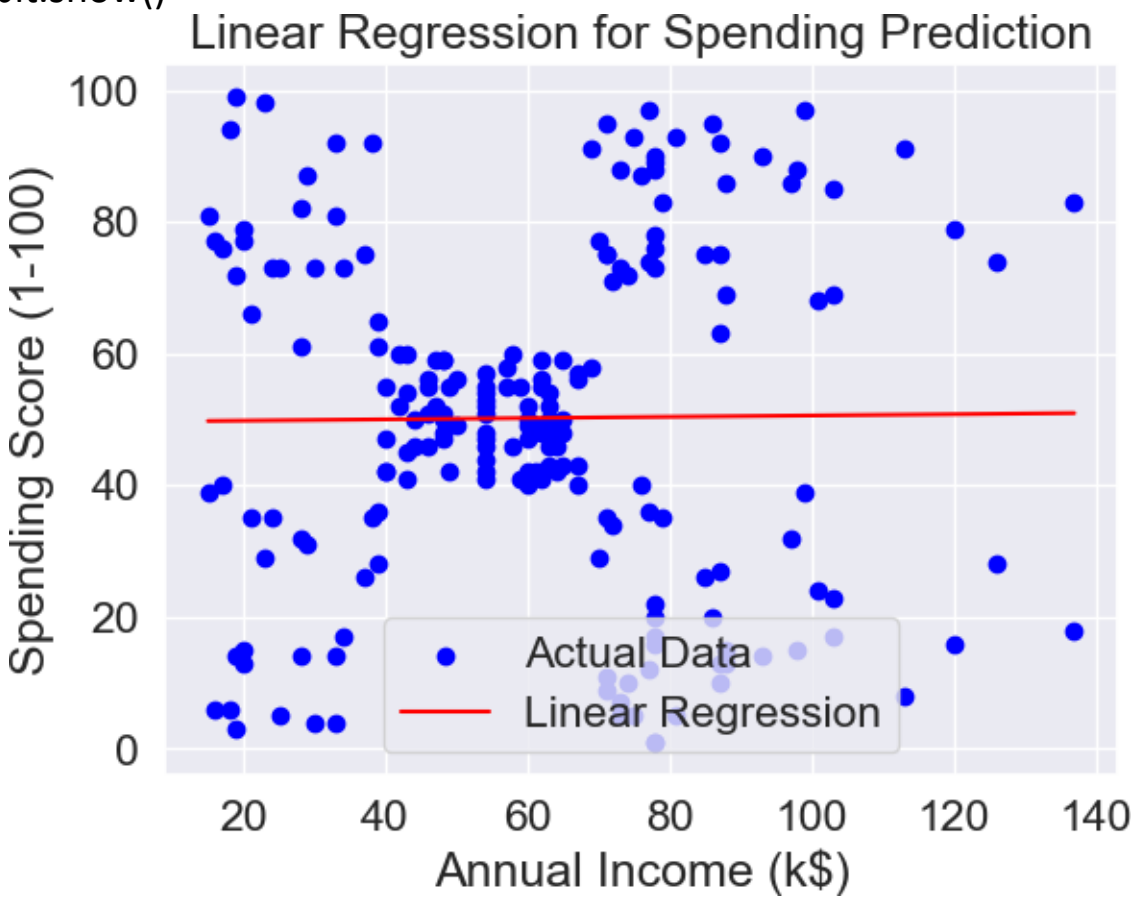
Out[62]:

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

Perfoming Linear Regression:

```
X = df[['Annual Income
(k$)']] y = df['Spending
Score (1-100)']model =
LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
plt.scatter(X, y, color='blue', label='Actual
Data') plt.plot(X, y_pred, color='red',
label='Linear Regression')plt.title('Linear
Regression for Spending Prediction')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-
100)')plt.legend()
```

plt.show()



Performing Moving Average

feature = 'Spending Score (1-

100)'window_size = 3

df['Moving_Average'] =

df[feature].rolling(window=window_size).mean()

plt.figure(figsize=(12, 6))

plt.plot(df['CustomerID'], df[feature], label='Actual Income(k$) ' + feature)

plt.plot(df['CustomerID'], df['Moving_Average'],
label='Moving Average (' +str(window_size) + ' months)')

plt.xlabel('CutomerID

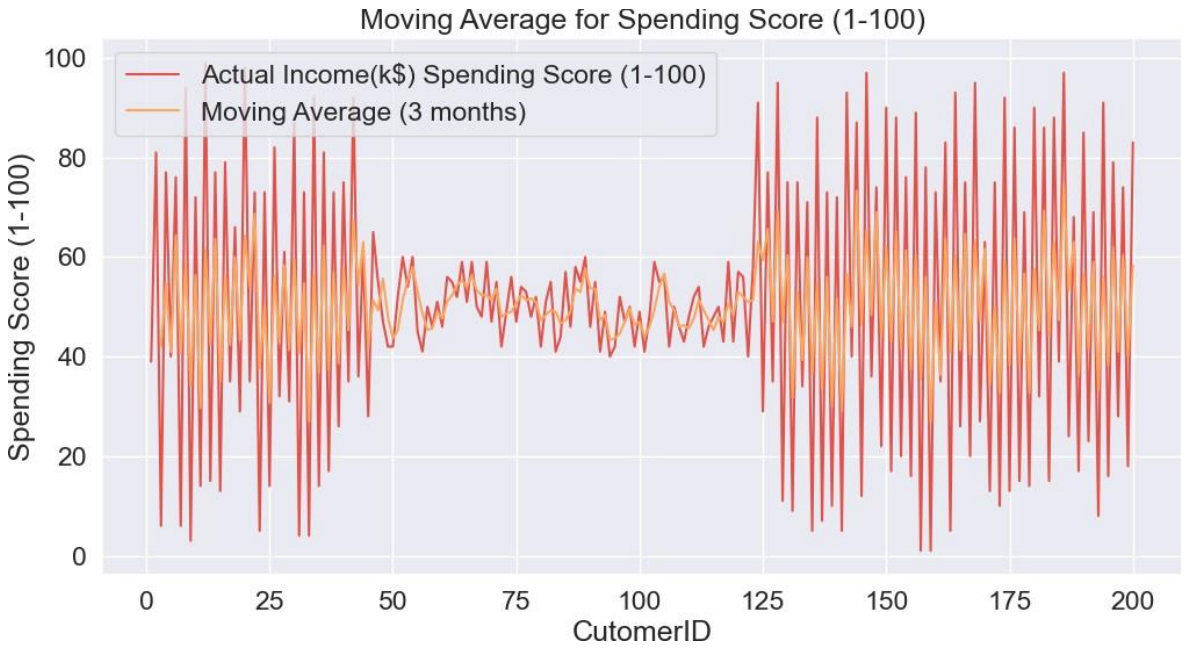
')plt.ylabel(feature)

plt.title('Moving Average for ' +

feature)plt.legend()

plt.grid(True)

plt.show()



Performing LSTM

feature = 'Spending Score (1-

100)'data = df[[feature]]

scaler = MinMaxScaler()

data_scaled =

scaler.fit_transform(data)

train_size = int(len(data) * 0.8)

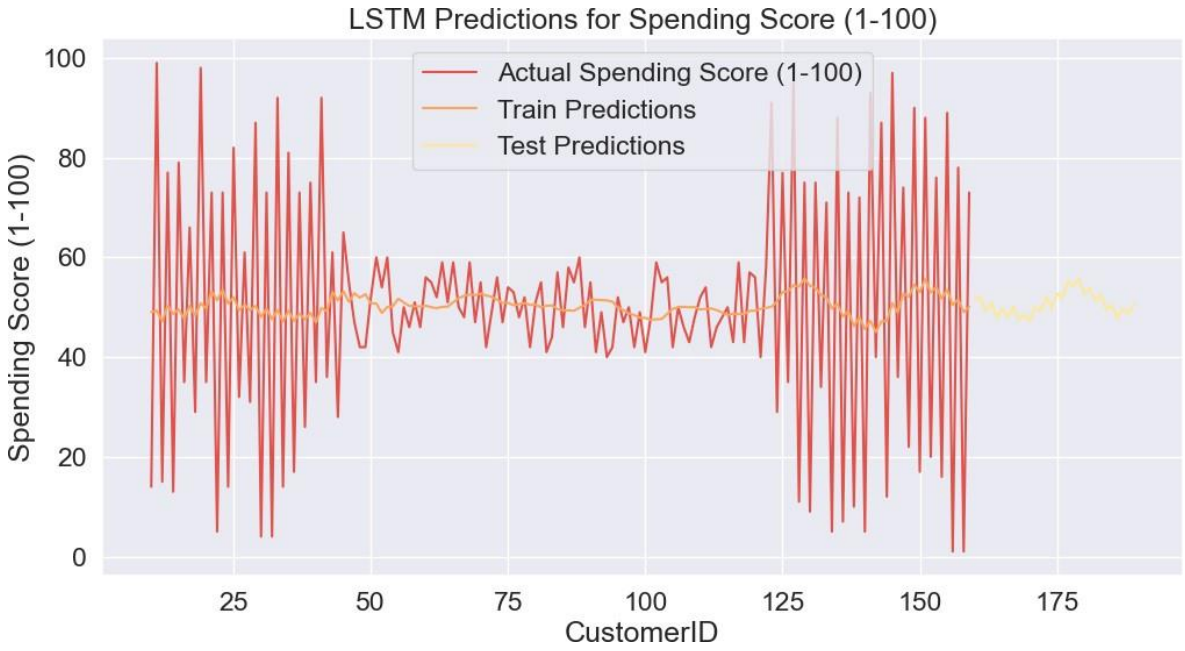
train_data, test_data = data_scaled[:train_size],

```
data_scaled[train_size:]def create_sequences(data,
sequence_length):
  X, y = [], []
  for i in range(len(data) -
    sequence_length):
    X.append(data[i:i+sequence_leng
    th])
    y.append(data[i+sequence_length
    ])
  return np.array(X), np.array(y)

sequence_length = 10  # Adjust this as needed
X_train, y_train = create_sequences(train_data,
sequence_length)X_test, y_test =
create_sequences(test_data, sequence_length)
model = Sequential()
model.add(LSTM(50,
input_shape=(sequence_length, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error',
optimizer='adam')
model.fit(X_train, y_train, epochs=15, batch_size=32, verbose=1)
```

```
Epoch 1/15
5/5 [==============================] - 2s 4ms/step - loss: 0.3066
Epoch 2/15
5/5 [==============================] - 0s 4ms/step - loss: 0.1686
Epoch 3/15
5/5 [==============================] - 0s 6ms/step - loss: 0.0828
Epoch 4/15
5/5 [==============================] - 0s 6ms/step - loss: 0.0557
Epoch 5/15
5/5 [==============================] - 0s 5ms/step - loss: 0.0671
Epoch 6/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0627
Epoch 7/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0549
Epoch 8/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0551
Epoch 9/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0567
Epoch 10/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0556
Epoch 11/15
5/5 [==============================] - 0s 5ms/step - loss: 0.0543
Epoch 12/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0541
Epoch 13/15
5/5 [==============================] - 0s 5ms/step - loss: 0.0540
Epoch 14/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0536
Epoch 15/15
5/5 [==============================] - 0s 4ms/step - loss: 0.0534
```

```
train_predictions =

model.predict(X_train)

test_predictions =

model.predict(X_test)

train_predictions =

scaler.inverse_transform(train_predictions)

test_predictions =

scaler.inverse_transform(test_predictions)
```

```
plt.figure(figsize=(12, 6))
plt.plot(df.index[sequence_length:sequence_length +
len(train_predictions)],
df[feature][sequence_length:sequence_length +
len(train_predictions)], label='Actual ' +feature)

plt.plot(df.index[sequence_length:sequence_length +
len(train_predictions)],train_predictions, label='Train
Predictions')

test_index = df.index[sequence_length +
len(train_predictions):sequence_length +len(train_predictions) +
len(test_predictions)]

plt.plot(test_index, test_predictions, label='Test Predictions')


plt.xlabel('CustomerI

D')plt.ylabel(feature)

plt.title('LSTM Predictions for ' +

feature)plt.legend()

plt.grid(True)

plt.show()
```



conclusion :

Customer segmentation using data science on a mall dataset helps businesses understand their customers better by grouping them into distinct categories based on their preferences, behaviors, and demographics. This enables targeted marketing strategies, personalized product recommendations, and improved customer experiences, ultimately leading to increased customer satisfaction and higher profitability for the business.