Applied Data Science

Phase 3: Development Part 1

Customer Segmentation using Data Science

Project definition:

- Customer segmentation is the method of distributing a customer base into collections of people based on mutual characteristics so organizations can market to group efficiently and competently individually.
- The purpose of segmenting customers is to determine how to correlate with customers in multiple segments to maximize customer benefits. Perfectly done customer segmentation empowers marketers to interact with every customer in the best efficient approach

Dataset explanation:

The data includes the following features:

1. Customer ID
2. Customer Gender
3. Customer Age
4. Annual Income of the customer (in Thousand Dollars)
5. Spending score of the customer (based on customer behavior and spending nature)

Importing the required libraries:

Let's start the development part of customer segmentation using data science by importing the necessary Python libraries.

Importing the Dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

Importing the dataset:

Data Collection & Analysis

```
# loading the data from csv file to a Pandas DataFrame
customer_data = pd.read_csv('/content/Mall_Customers.csv')
```

Printing first five rows of the dataset:

```
# first 5 rows in the dataframe
customer_data.head()
```

	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

Connected to Python 3 Google Compute Engine backend

Analyzing the dataset:

+ Code + Text

```
# getting some informations about the dataset
customer_data.info()
```

```
<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   Genre                   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
```

```
# checking for missing values
customer_data.isnull().sum()
```

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

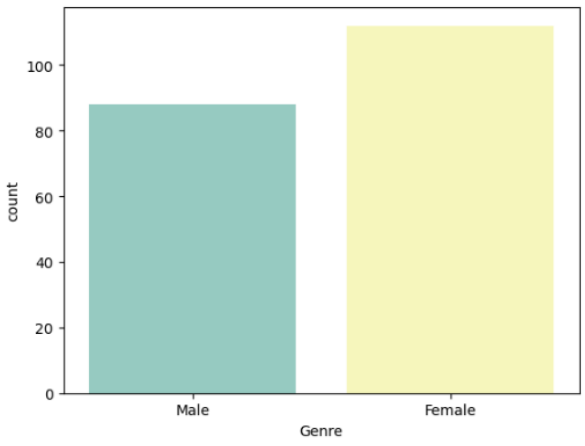
Choosing the Annual Income Column & Spending Score column

```
[ ] X = customer_data.iloc[:,[3,4]].values

print(X)
```

```
[ 76  40]
[ 76  87]
[ 77  12]
[ 77  97]
[ 77  36]
[ 77  74]
[ 78  22]
[ 78  90]
[ 78  17]
[ 78  88]
[ 78  20]
[ 78  76]
[ 78  16]
[ 78  89]
[ 78   1]
[ 78  78]
[ 78   1]
[ 78  73]
[ 79  35]
[ 79  83]
```

```
import seaborn as sns
customer_data = pd.read_csv('/content/Mall_Customers.csv')
sns.countplot(x='Genre', data=customer_data, palette="Set3")
```

```
<Axes: xlabel='Genre', ylabel='count'>
```



✓ 0s completed at 3:04 PM

Encoding the categorical features:

- Encoding categorical data using one- hot encoding (OneHotEncoder) is a process of representing categorical variables in a numerical format that can be used in various machine learning algorithms
- Encoding categorical data using OneHotEncoder is a process in data projects where categorical variables are transformed into a numerical format, specifically binary vectors.
- Each unique category or label within a categorical variable is converted into a separate binary column, and for each observation, the column corresponding to its category is marked with a "1," while all other columns are set to "O."
- This method is used to make categorical data compatible with machine learning algorithms that require numerical input.

- OneHotEncoder ensures that the categorical data is represented in a way that doesn't introduce false ordinal relationships or numerical values, preventing biases in the model's interpretation of the data

Encoding Categorical Features

```
# Generate one-hot dummy columns
customer_data = pd.get_dummies(customer_data).reset_index(drop=True)
customer_data = pd.read_csv('/content/Mall_Customers.csv')
```

Choosing the number of clusters:

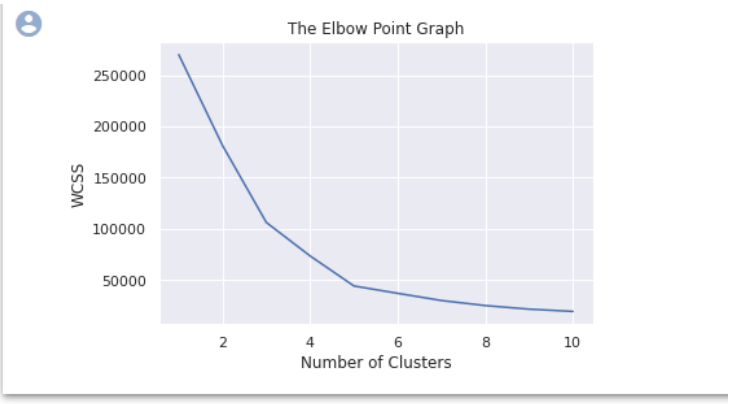
K Means Clustering Using the Elbow Method For each value of K, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of the squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow.

Choosing the number of clusters

WCSS -> Within Clusters Sum of Squares

```
# finding wcss value for different number of clusters

wcss = []

for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(X)

  wcss.append(kmeans.inertia_)
```

```
# plot an elbow graph

sns.set()
plt.plot(range(1,11), wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



Training the k-Means algorithm:

1.Testing set: The testing set is a smaller portion of the data, usually around 20-30% of the dataset. It is kept separate and is not used during the model training phase. Instead, it is used to

evaluate the model's performance by making predictions or performing analyses and comparing them to the actual, known outcomes.

2.Training set: This subset contains a majority of the data, typically around 70-80% of the dataset. It is used to train machine learning models or perform data analysis tasks. The model learns patterns, relationships, and trends within the data from this set.

Optimum Number of Clusters = 5

Training the k-Means Clustering Model

```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)

# return a label for each data point based on their cluster
Y = kmeans.fit_predict(X)

print(Y)
```

```
[3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3
 1 3 1 3 1 3 0 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 2 4 2 0 2 4 2 4 2 0 2 4 2 4 2 4 2 4 2 0 2 4 2 4 2
 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4
 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2]
```

- So, this is how we can do Customer Segmentation using data science.