PHASE 5: Project Documentation & Submission

Problem Statement: Implement data science techniques to segment customers based on their behavior, preferences, and demographic attributes, enabling businesses to personalize marketing strategies and enhance customer satisfaction.

Problem Explanation: The problem at hand is to leverage data science techniques to effectively segment customers based on various aspects such as their behavior, preferences, and demographic attributes. The ultimate goal of this project is to empower businesses with the ability to tailor their marketing strategies in a personalized manner, ultimately leading to improved customer satisfaction and potentially increased sales.

The process involved in achieving this objective can be broken down into several key steps :

- ➔ Data Collection
- ➔ Data Preprocessing
- ➔ Feature Engineering
- ➔ Clustering Algorithms
- ➔ Visualization
- ➔ Interpretation

Data Collection: The first step is to gather customer data from various sources. This data encompasses a wide range of attributes, including but not limited to purchase history, demographic information (age, gender, location), and customer interaction behavior (e.g., website visits, social media

engagement, email response rates). Collecting diverse data points is crucial as it provides a holistic view of each customer.

Data Preprocessing: Once the data is collected, it needs to be cleaned and preprocessed. This involves tasks such as handling missing values, dealing with outliers, and ensuring data consistency. Additionally, categorical features like gender or product categories may need to be converted into numerical representations (one-hot encoding, label encoding) that machine learning algorithms can work with.

Feature Engineering: Creating new features is a vital step in understanding customer behavior. Features such as total spending, frequency of purchases, and average order value can be engineered from the raw data. These features help in capturing meaningful patterns and characteristics of each customer.
Clustering Algorithms: The heart of the project lies in applying clustering algorithms to segment the customer base. Common algorithms like K-Means, DBSCAN, or hierarchical clustering can be employed for this purpose. These algorithms group customers with similar attributes and behavior into distinct clusters or segments.

Visualization: Visualization plays a crucial role in presenting the results in an understandable and actionable manner. Techniques like scatter plots, bar charts, and heatmaps can be used to visualize the identified customer segments. This step helps stakeholders grasp the differences and similarities between segments intuitively.

Interpretation: Finally, the project requires in-depth analysis and interpretation of the identified customer segments. This involves understanding the characteristics, needs, and preferences of each segment. By doing so, businesses can derive actionable insights that can inform marketing strategies, product offerings, and customer engagement tactics.

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

This dataset is used in retail and marketing analytics to understand customer behavior and
preferences. It includes the following types of information:

- Customer ID
- Gender
- Age
- Annual Income
- Spending Score

A "customer ID" (Customer Identification) is a unique identifier assigned to each customer in a database or system. It is used to distinguish one customer from another and track their activities, purchases, interactions, and other relevant information.

Gender is one of the key factors in segmenting customers into distinct groups. For example, stores may tailor their product offerings and marketing strategies differently for male and
female customers.

Age is a fundamental factor for segmenting customers into groups. Different age groups
may have distinct preferences, shopping behaviors, and income levels. For example, retailers
often distinguish between teenagers, young adults, middle-aged individuals, and seniors.

The annual income of mall customers is a crucial demographic variable that helps businesses and mall operators understand the spending capacity and shopping preferences of their customer base.

Spending score is a metric used to assess and quantify a customer's purchasing behavior within a mall.

3.2 Loading the dataset
3.3 Preprocessing Dataset
3.4 Performing different analysis

Executed in Python Notebook (.ipynb) file in git repository.

Git repository link:
https://github.com/adrieljoshua/CustomerSegmentation-AppliedDataScience.git

Performing K-Means Clustering on given data

K-Means Clustering:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

Cost Function: Inertia is a metric used to evaluate the quality of a clustering algorithm, particularly the K-means algorithm. It measures the sum of squared distances between each data point and its assigned centroid. In other words, it measures how far the data points are from their assigned cluster centers.

The K-means algorithm tries to minimize the inertia by iteratively updating the cluster centers until the inertia cannot be reduced any further. A lower inertia value indicates that the clusters are more compact and well-separated, while a higher inertia value indicates that the clusters are more spread out and overlapping.

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be different from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means re-assign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

In [ ]:
```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.interpolate import griddata
class PerformKMeans:

    def __init__(self, X) -> None:
        self.X = X

    def get_cost_func(self, fit_cols, no_k = 10):

        # record inertia values in a list
        cost_function_values = []
        for k in range(1,no_k+1):
            # get k_means algorithm
            km = KMeans(n_clusters=k, init='random', max_iter=100, n_init=1,
                    algorithm = 'lloyd', verbose=False, random_state=9)

            # fit
            dataset = self.X.loc[:,fit_cols]
            km.fit_predict(dataset)

            # get inertia
            inertia = km.inertia_
            cost_function_values.append(inertia)

        return cost_function_values
```

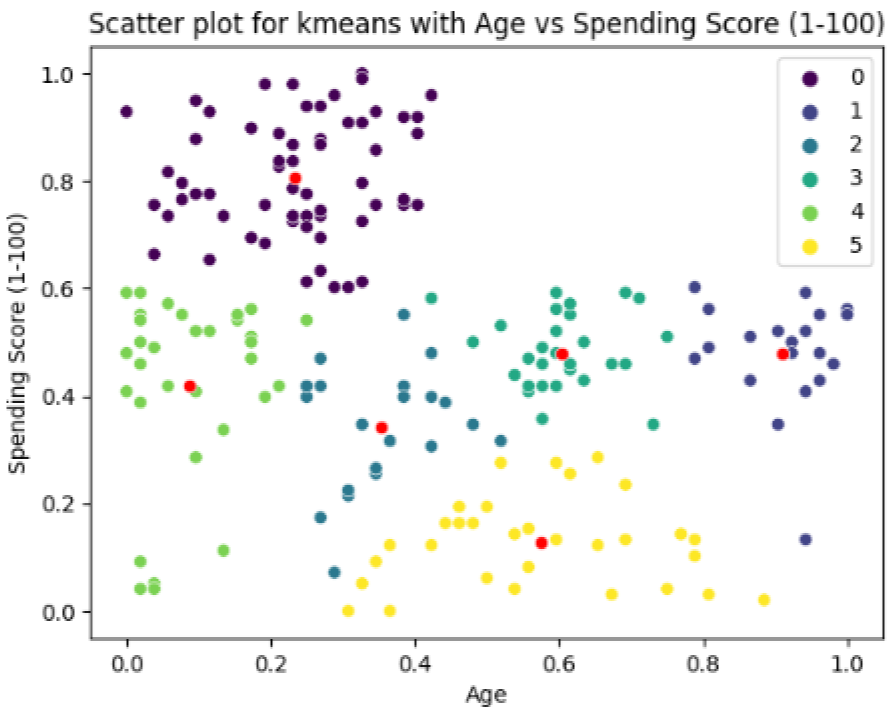
```python
    def plot_clusters(self, fit_cols, k=5):

        if len(fit_cols) != 2:
            raise Exception("clusters can be plotted only for 2 features using this fucntion")

        # get k_means algorithm
        km = KMeans(n_clusters=k, init='random', max_iter=100, n_init=1,
                    algorithm = 'lloyd', verbose=False, random_state=9)

        # fit
        dataset = self.X.loc[:,fit_cols]
        labels = km.fit_predict(dataset)
        centers = km.cluster_centers_

        # plot
        import matplotlib.pyplot as plt
        plt.title("Scatter plot for kmeans with {} vs {}".format(fit_cols[0],fit_cols[1]))
        sns.scatterplot(x= self.X.loc[:,fit_cols[0]], y= self.X.loc[:,fit_cols[1]],
               hue = labels, palette= "viridis")
        sns.scatterplot(x=centers[:,0], y=centers[:,1], color = 'red')
        plt.show()

        return dataset, labels
```

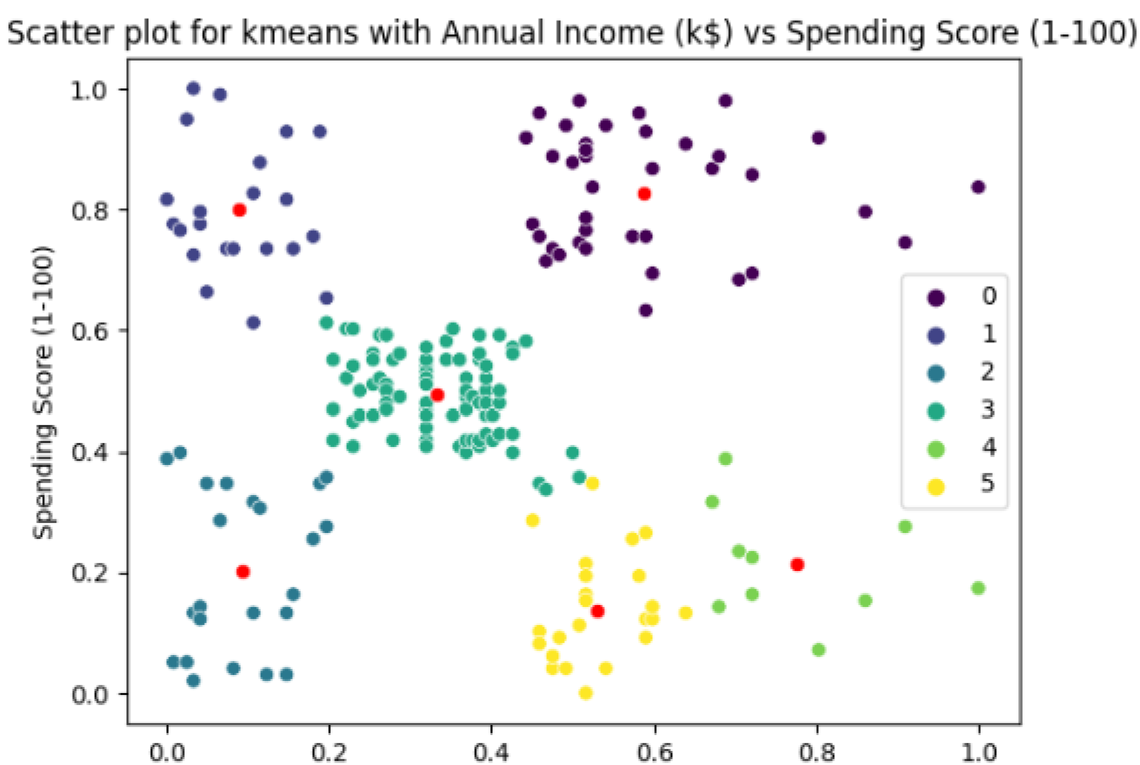
[ ]:

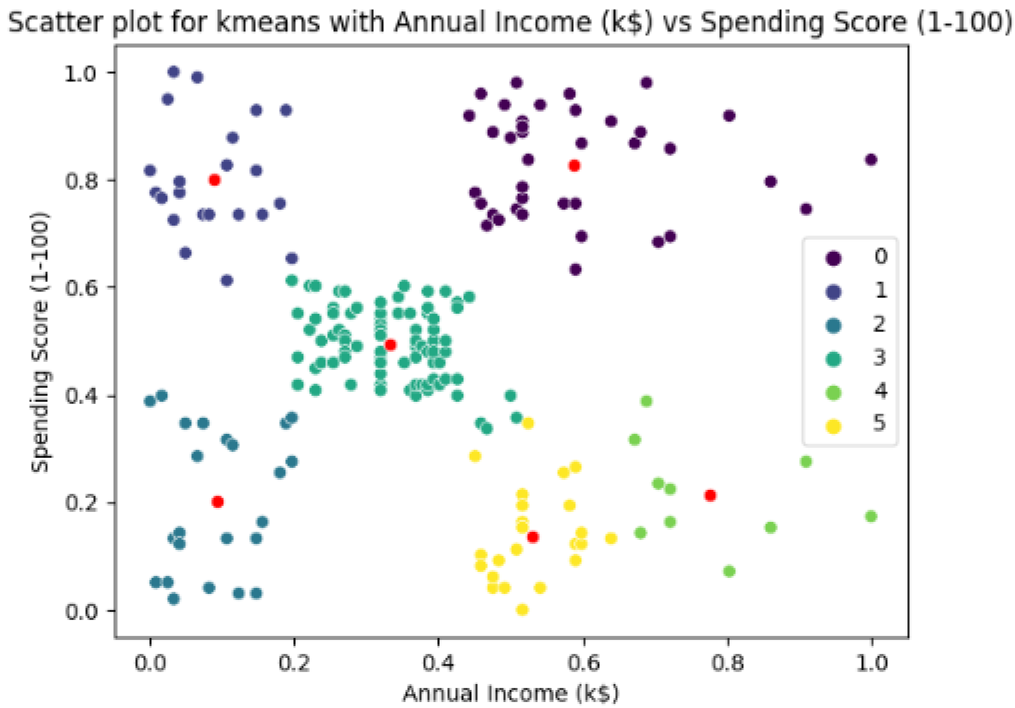
```python
pkm = PerformKMeans(X)
feature_set1 = X.columns.drop(["Age","Gender"])
feature_set2 = X.columns.drop(["Gender", "Spending Score (1-100)"])
feature_set3 = X.columns.drop(["Gender", "Annual Income (k$)"])
feature_set4 = X.columns.drop(["Gender"])
```

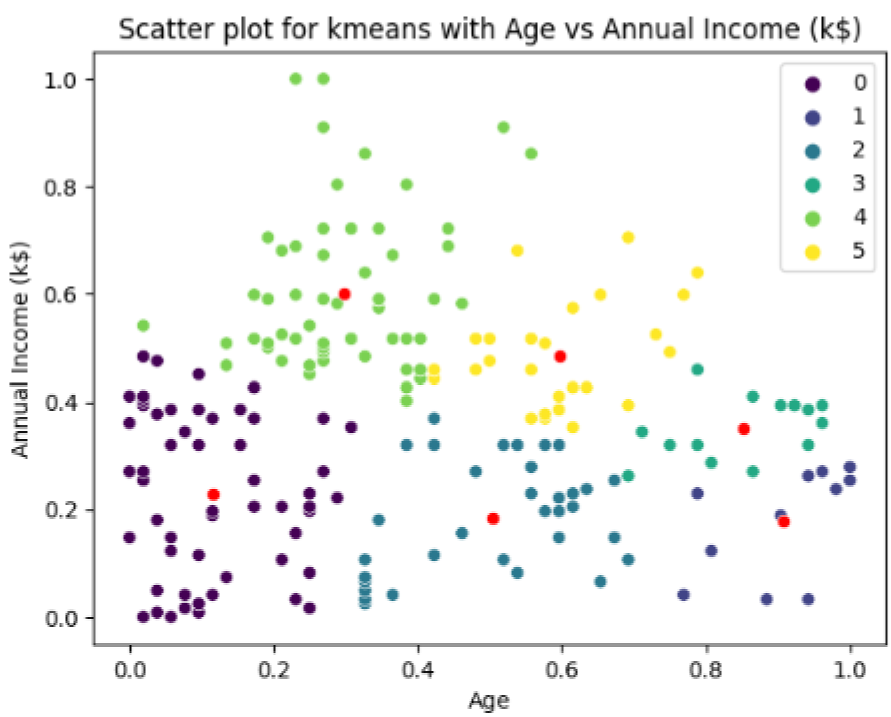
```python
km = KMeans(n_clusters=6, init='random', max_iter=100, n_init=1,
                    algorithm = 'lloyd', verbose=False, random_state=42)

# fit
dataset = X.loc[:,feature_set4]
labels = km.fit_predict(dataset)
centers = km.cluster_centers_
inertia = km.inertia_

# print results
print("inertia: ", inertia)
print("centers: \n", centers)
```

```python
pkm.plot_clusters(feature_set1, k=6)
```


Scatter plot for kmeans with Age vs Spending Score (1-100)
Spending Score (1-100)
Age
0
1
2
3
4
5


Scatter plot for kmeans with Annual Income (k$) vs Spending Score (1-100)
Spending Score (1-100)
0
1
2
3
4
5


Scatter plot for kmeans with Annual Income (k$) vs Spending Score (1-100)
Spending Score (1-100)
Annual Income (k$)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5


Scatter plot for kmeans with Age vs Annual Income (k$)
Annual Income (k$)
Age
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5

Process:

1. We load the dataset, select the relevant columns, and encode the "Genre" column using Label Encoding to convert it to numerical values.

2. We standardize the features using StandardScaler to make them comparable in terms of scale.

3. We specify the number of clusters (k) as 5 in this example, but you can adjust it based on your dataset and problem.

4. We perform K-Means clustering and obtain cluster labels for each data point.

5. We add the cluster labels to the original DataFrame to associate each data point with its cluster.

6. We visualize the clusters by creating a scatter plot of "Annual Income" and "Spending Score," where data points are colored by their cluster labels, and the cluster centroids are marked in red.