**PROJECT TITLE - CUSTOMER SEGMENTATION**

**TEAM MEMBER**

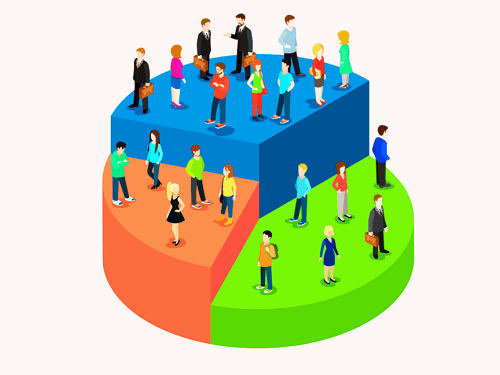
**DHARUNRAJ N**

**210621104012**

**APPLIED DATA SCIENCE (Group -1)**

**Phase-5**

**Project Documentation & Submission**



# CUSTOMER SEGMENTATION

# 1. INTRODUCTION

# 

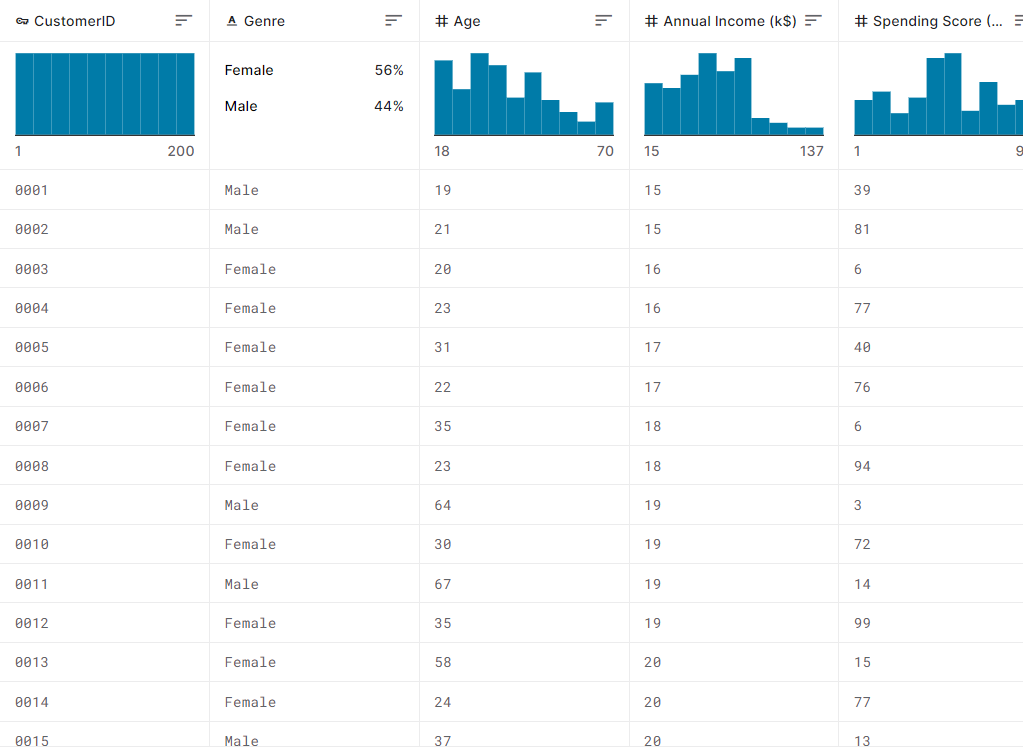
## The primary objective of this project is to employ sophisticated data science techniques to categorize customers based on their behaviour, preferences, as well as demographic information such as age and location. This segmentation strategy holds significant importance as it enables businesses to tailor specific marketing strategies for each customer group. This, in turn, can lead to increased customer satisfaction and higher likelihood of repeat business. The project involves key stages including data acquisition, data refinement, generating valuable insights from the data, employing diverse methods for customer segmentation, visualizing the data, and interpreting the significance of the customer groups.

## 2. DATASET

The dataset has been sourced from Kaggle, a prominent platform for collaborative data science. You can access the dataset through the following link:

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

The data is formatted in CSV (Comma Separated Values), an efficient format for storing and processing data. With the help of the pandas library, we can easily handle and manipulate this data for analysis.



**The dataset comprises the following attributes:**

1. CustomerID

2. Gender

3. Age

4. Annual Income

5. Spending Score

However, CustomerID is deemed redundant for our analysis and will not be utilized. We will focus on exploring and analyzing the remaining columns (Gender, Age, Annual Income, and Spending Score) through Exploratory Data Analysis (EDA). When it comes to clustering, our primary features of interest will be Annual Income and Spending Score.

**3. LIBRARIES USED**

**Pandas**

Pandas is an essential library for data manipulation and analysis in Python, offering a plethora of functionalities. It excels in tasks like loading and reading data from diverse sources, including CSV files, Excel spreadsheets, and SQL databases. It proves invaluable in data cleaning and pre-processing, enabling tasks such as handling missing values, imputing data, removing duplicates, and transforming variables.

**Matplotlib and Seaborn**

Matplotlib and Seaborn are two indispensable libraries for data visualization in Python. It simplifies the process of generating complex plots like distribution plots, pair plots, and correlation matrices with concise and intuitive syntax. Seaborn also comes with built-in themes and colour palettes that enhance the visual appeal of plots. While Matplotlib is versatile and customizable, Seaborn excels in producing visually appealing and informative statistical graphics, making them a powerful duo for data visualization tasks in Python.

**SkLearn**

With sklearn, you can efficiently preprocess and prepare datasets for modeling, employing techniques like feature scaling, encoding categorical variables, and handling missing data. The library offers an extensive collection of machine learning algorithms, including popular ones like decision trees, support vector machines, and random forests, as well as tools for model evaluation and selection.

**KMeans**

K-means is a popular unsupervised machine learning algorithm used for clustering similar data points into groups or clusters. It works by partitioning the data into 'k' number of clusters, where 'k' is a user-defined parameter. The algorithm iteratively assigns data points to clusters based on their proximity to the cluster centroids, which are the mean coordinates of the data points in each cluster. K-means aims to minimize the variance within clusters, effectively grouping data points that are close together in feature space. It's widely used in tasks like customer segmentation, image compression, and anomaly detection.

**Code**

**Importing Libraries**

import pandas as pd

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

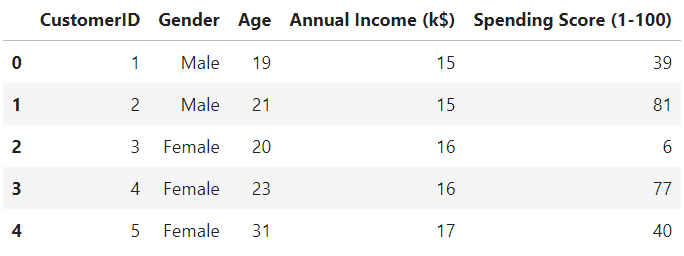
from sklearn.metrics import silhouette\_score

**Loading the data**

data = pd.read\_csv("Mall\_Customers.csv")

data.head()

**Output**

****

**4. EXPLORATORY DATA ANALYSIS (EDA)**

EDA stands for Exploratory Data Analysis. It is a crucial initial step in data analysis where analysts or data scientists explore and summarize the main characteristics of a dataset. This process involves generating descriptive statistics, visualizing the data through graphs and plots, and identifying patterns or trends. EDA helps in gaining a deeper understanding of the dataset, identifying outliers or missing values, and formulating hypotheses for further analysis. It serves as a foundation for making informed decisions about data cleaning, feature engineering, and selecting appropriate modelling techniques in subsequent stages of a data science project.

**Age**

We'll employ a categorical plot to illustrate the distribution of genders and their respective counts.

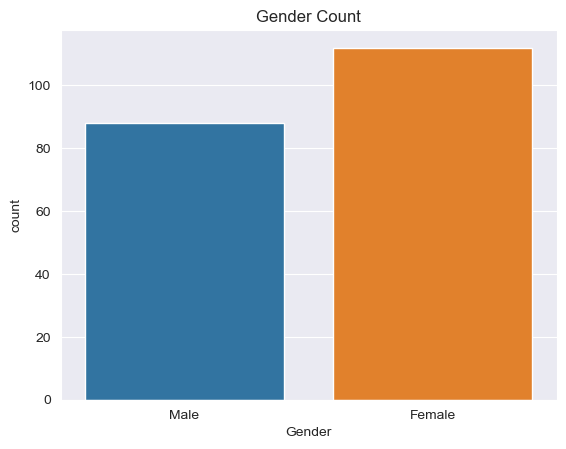
**Code**

sns.countplot(data=data, x="Gender")

plt.title("Gender Count")

plt.show()

**Output**

****

It appears that there is a higher number of customers identified as "Female" compared to those identified as "Male".

**Age with Gender**

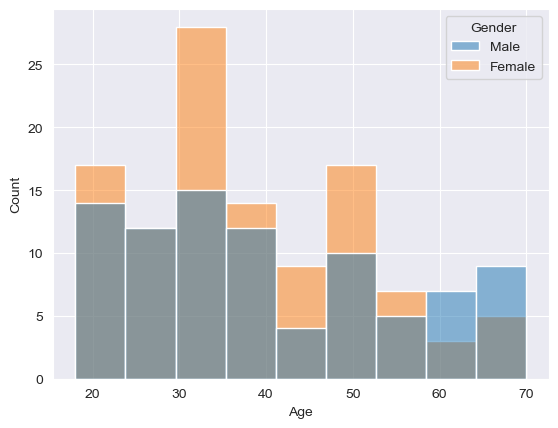
We will utilize a histogram plot to examine the age distribution within specific intervals, separately for each gender.

**Code**

sns.histplot(data=data, x="Age", hue="Gender")

plt.show()

**Output**

****

A notable concentration of "Female" customers falls within the age range of "30-35".

**Spending Score with Gender**

**Code**

sns.histplot(data=data, x="Spending Score (1-100)", hue="Gender")

plt.show()

**Output**

A graph of a graph with numbers and a bar chart

Description automatically generated with medium confidence

The majority of customers identified as "Female" exhibit a spending score between 70 and 100.

**Age and Income**

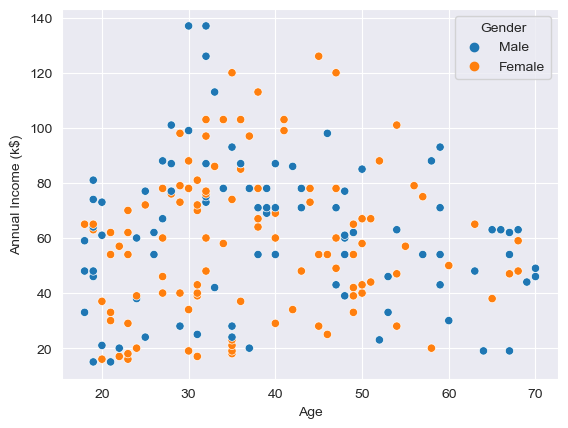
We'll employ a scatter plot to visualize the relationship between Age and Annual Income.

**Code**

sns.scatterplot(data=data,x="Age", y="Annual Income (k$)", hue="Gender")

plt.show()

**Output**

****

Individuals under the age of 30 and over the age of 60 tend to have relatively lower "Annual Income".

**Age and Spending Score**

**Code**

sns.scatterplot(data=data, x="Age", y="Spending Score (1-100)", hue="Gender")

plt.show()

**Output**

**A graph with blue and orange dots

Description automatically generated**

Individuals aged above 40 tend to have lower spending habits.

**Annual Income and Spending Score**

**Code**

sns.scatterplot(data=data, x="Annual Income (k$)", y="Spending Score (1-100)", hue="Gender")

plt.show()

**Output**

**A graph with blue and orange dots

Description automatically generated**

There is evidently a discernible pattern that allows for segmentation of the provided customers based on their Annual Income and Spending Score. We will delve deeper into this pattern using the KMeans algorithm in the subsequent steps.

**5. PREPROCESSING**

Data preprocessing is a crucial step in data science that involves preparing and cleaning raw data for analysis. It encompasses tasks like handling missing values, removing duplicates, and transforming variables to ensure data quality. Additionally, data preprocessing may involve scaling features, encoding categorical variables, and handling outliers to make the data suitable for machine learning algorithms.

**Removing Useless Column**

We'll eliminate the "CustomerID" column from our dataset since its values are non-informative and unnecessary for our analysis.

**Code**

data.drop(["CustomerID"], axis=1, inplace=True)

**Missing Values**

Having missing values in a dataset can lead to biased analyses and inaccurate model predictions. It can also reduce the effectiveness of machine learning algorithms, as they may struggle to handle incomplete information.

**Code**

data.isna().sum()

**Output**

A close-up of a score

Description automatically generated

**Feature Scaling**

Feature scaling is a critical preprocessing step for machine learning algorithms that rely on distance calculations between data points. Without proper scaling, features with larger value ranges can overpower the distance computations, potentially leading to biased results. This is particularly significant for algorithms like K-Means clustering, which employs metrics like Euclidean distance. Thus, ensuring feature scaling is essential to achieve accurate and meaningful results when applying K-Means and similar algorithms in data analysis and modelling tasks.

**Code**

scaler **=** StandardScaler()

data\_new **=** data**.**iloc[:, [2, 3]]

scaler**.**fit(data\_new)

data\_scaled **=** scaler**.**transform(data\_new)

Now our data is scaled and ready for model creation.

**6. MODEL BUILDING**

K-means clustering is employed in scenarios where data lacks predefined categories or labels. This algorithm excels at automatically categorizing unlabelled data into a specified number of clusters, determined by the value of 'k'. This process relies on identifying similarities among data points to create distinct groupings.

**Creating an Optimal Model**

We will determine the optimal number of clusters by employing metrics such as WCSS (Within-Cluster-Sum-of-Squares) and Silhouette score.

**Elbow Method (WCSS)**

It quantifies the distance between each data point and its respective centroid, then computes the squared difference between them. The objective is to minimize the cumulative sum of these squared differences.

**Code**

wcss = []

for i in range(1, 10):

kmeans = KMeans(n\_clusters=i)

kmeans.fit(data\_scaled)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 10), wcss)

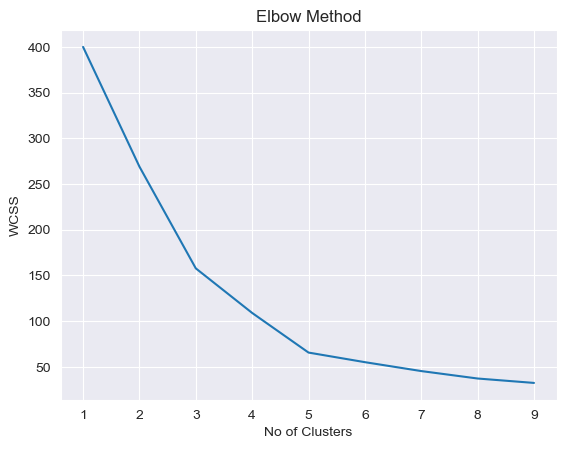
plt.title("Elbow Method")

plt.xlabel("No of Clusters")

plt.ylabel("WCSS")

plt.show()

**Output**

****

Based on the WCSS metric, we identify 3 and 5 clusters as potential optimal choices.

**Silhouette Score**

The silhouette score is a metric that assesses both the cohesion of a data point with its assigned cluster and its separation from other clusters. It provides a score between -1 and 1, where 1 signifies well-separated and distinctly defined clusters. A high silhouette score indicates a good match between the data point and its assigned cluster, while also indicating a notable distinction from neighboring clusters. Conversely, negative values indicate that a data point may have been erroneously assigned to its cluster.

**Code**

silhouette\_avg = []

for i in range(2, 10):

temp\_kmeans = KMeans(n\_clusters=i)

temp\_kmeans.fit(data\_scaled)

cluster\_label = temp\_kmeans.labels\_

silhouette\_avg.append(silhouette\_score(data\_scaled, cluster\_label))

plt.plot(range(2, 10), silhouette\_avg)

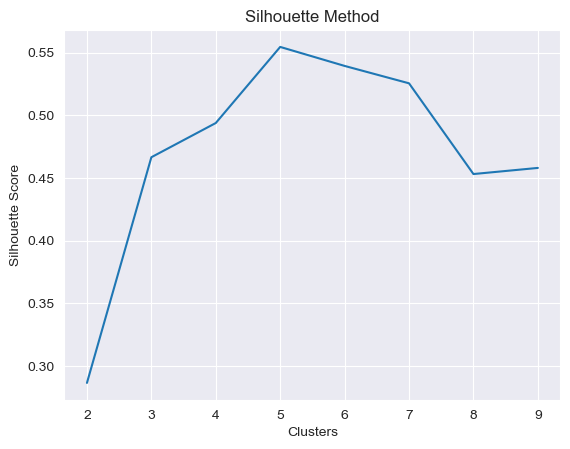
plt.xlabel("Clusters")

plt.ylabel("Silhouette Score")

plt.title("Silhouette Method")

plt**.**show()

**Output**

****

Based on the silhouette score, we ascertain that 5 is the optimal number of clusters for our analysis.

**Creating the model**

We have instantiated a KMeans model with 5 clusters, a decision informed by both the WCSS and silhouette score evaluations.

**Code**

kmeans **=** KMeans(5)

clusters **=** kmeans**.**fit\_predict(data\_new)

data\_new["Clusters"] **=** clusters

data\_new**.**head()

**7. MODEL VISUALIZATION**

Data visualization is an essential aspect of clustering analysis, as it helps you understand and interpret the results of your clustering algorithms. Visualizing clustering results can provide insights into the structure and patterns within your data.

**Code**

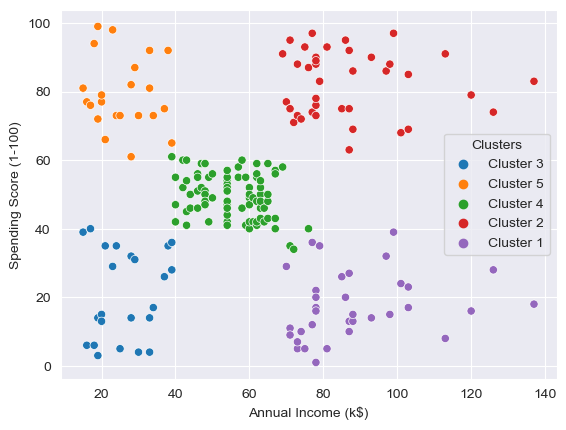
clusters = kmeans.cluster\_centers\_

data\_new["Clusters"] = data\_new["Clusters"].apply(lambda x: f"Cluster {x + 1}")

sns.scatterplot(data=data\_new, x="Annual Income (k$)", y="Spending Score (1-100)", hue="Clusters")

plt.show()

**Output**

****

**8. CONCLUSION**

By applying the KMeans algorithm, we have successfully segmented the customers into five distinct groups based on their spending behavior and income levels:

1. Low Income & High Spender

2. Mid Income & Mid Spender

3. Low Income & Low Spender

4. High Income & Low Spender

5. High Income & High Spender

This segmentation provides valuable insights for tailoring personalized advertisements and services to cater to the specific preferences and needs of each group. It allows for more targeted and effective marketing strategies, ultimately enhancing customer satisfaction and retention.