ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research

Vol. 17, Issue, 01, pp.31458-31469, January, 2025
DOI: https://doi.org/10.24941/ijcr.48358.01.2025

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

CUSTOMER SEGMENTATION USING CLUSTERING

***Dr. S. Raja and Karthick Raja, J.**

Master of Computer Applications, Center for Open and Digital Education, Hindustan Institute of Technology and Science, Chennai, India

ARTICLE INFO

Article History:
Received 20th October, 2024
Received in revised form
17th November, 2024
Accepted 24th December, 2024
Published online 31st January, 2025

Key Words:
Customer Segmentation, K-means Clustering, Demographic Attributes, Annual income, Spending Behavior, Unsupervised learning, Data Visualization, Customer Relationship Management.

****Corresponding author:*** **Dr. S. Raja**

ABSTRACT

Customer segmentation is a pivotal strategy for businesses to tailor their marketing efforts and optimize customer satisfaction. This project focuses on utilizing K-means clustering, a popular unsupervised learning algorithm, to partition customers into distinct segments based on their demographic attributes, annual income, and spending behaviour. Through comprehensive data exploration and visualization techniques, we analyse the distribution of customer characteristics and identify meaningful clusters. By applying K-means clustering, we aim to uncover hidden patterns and preferences among customers, enabling businesses to develop targeted marketing strategies and enhance customer engagement. The effectiveness of the segmentation process is evaluated through metrics such as silhouette score and within-cluster sum of squares (WCSS). Insights gained from this analysis can empower businesses with valuable insights into customer behavior, facilitating informed decision-making and personalized customer experiences. This project contributes to the field of customer relationship management by providing a data-driven approach to segmentation and highlighting the significance of clustering algorithms in understanding customer dynamics.

Citation: Dr. S. Raja and Karthick Raja, J.. 2024. "A potential target of mosquito species using molecular docking *". International Journal of Current Research*, 17, (01), 31458-31469.

INTRODUCTION

In the modern world, data has become ubiquitous, driving decision-making processes across various domains. From business operations to healthcare and beyond, organizations rely on data to gain insights, make predictions, and optimize processes. With the proliferation of digital devices and online platforms, vast amounts of data are generated every second, creating both opportunities and challenges. Harnessing this data effectively has become crucial for businesses to stay competitive and relevant in today's market.

MACHINE LEARNING: Machine learning, a subset of artificial intelligence, plays a pivotal role in extracting meaningful patterns and knowledge from data. Unlike traditional programming paradigms, where explicit instructions are provided to perform tasks, machine learning algorithms learn from data to improve their performance over time. These algorithms enable computers to identify complex patterns, make predictions, and automate decision-making processes without explicit programming.

APPROACHES IN MACHINE LEARNING: Supervised Learning: In supervised learning, algorithms learn from labelled data, where the input features are associated with corresponding target labels. The goal is to learn a mapping function that can accurately predict the target variable for new instances. Common tasks include classification and regression. Unsupervised Learning: Unsupervised learning involves learning patterns and structures from unlabelled data. Unlike supervised learning, there are no predefined target labels. Instead, the algorithms aim to discover inherent structures or relationships within the data, such as clusters or associations. Semi-Supervised Learning: Semi-supervised learning combines elements of both supervised and unsupervised learning. It leverages a small amount of labelled data along with a large pool of unlabelled data to improve model performance. This approach is beneficial when acquiring labelled data is expensive or time-consuming.

Reinforcement Learning: Reinforcement learning involves training agents to interact with an environment to achieve specific goals. The agent learns through trial and error, receiving feedback in the form of rewards or penalties based on its actions. Over time, the agent learns optimal strategies to maximize cumulative rewards.

CLUSTERING: Clustering is a fundamental technique in unsupervised machine learning, aimed at grouping similar data points together based on their intrinsic characteristics or attributes. It is widely used for data exploration, pattern recognition, and segmentation tasks, helping to uncover hidden structures within datasets.

TYPES OF CLUSTERING: K-means Clustering: K-means clustering is one of the most popular and widely used clustering algorithms. It partitions the dataset into K clusters, where K is predefined by the user. The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids based on the mean of the assigned points. It converges when the centroids no longer change significantly or after a specified number of iterations.

Hierarchical Clustering: Hierarchical clustering builds a tree-like hierarchy of clusters, known as a dendrogram. It does not require the user to specify the number of clusters in advance. The algorithm starts with each data point as a separate cluster and merges the closest clusters iteratively until only one cluster remains. Hierarchical clustering can be agglomerative (bottom-up) or divisive (top-down), offering flexibility in clustering large datasets. Density-based Clustering (e.g., DBSCAN): Density-based clustering identifies clusters based on regions of high data density. It groups together data points that are closely packed, separated by areas of low density. Unlike K-means, density-based clustering does not require specifying the number of clusters beforehand and can handle clusters of arbitrary shapes and sizes. It is particularly useful for datasets with irregular or noisy distributions. Gaussian Mixture Models (GMM): Gaussian Mixture Models assume that the data points are generated from a mixture of several Gaussian distributions. Each cluster is modelled as a Gaussian distribution with its mean and covariance matrix. The algorithm iteratively estimates the parameters of the Gaussian distributions and assigns probabilities of data points belonging to each cluster. GMM is flexible and can capture complex data distributions.

APPLICATIONS OF CLUSTERING: Clustering, a technique in unsupervised learning, finds applications across various domains for data analysis, pattern recognition, and decision-making. Here are some common applications of clustering along with their definitions:

Market Segmentation

Definition: Market segmentation involves dividing a heterogeneous market into smaller, more homogenous groups based on characteristics such as demographics, behavior, or purchasing patterns.

Application: Clustering helps businesses identify distinct customer segments with similar needs, preferences, and behaviours. This enables targeted marketing strategies, product customization, and better customer engagement.

Image Segmentation

Definition: Image segmentation partitions an image into multiple regions or segments to simplify its representation and facilitate further analysis.

Application: Clustering algorithms like K-means or hierarchical clustering are used to group pixels in images based on colour, texture, or intensity. Image segmentation finds applications in medical imaging, object detection, and computer vision tasks.

Anomaly Detection

Definition: Anomaly detection involves identifying unusual patterns or outliers in data that deviate significantly from normal behavior.

Application: Clustering techniques, particularly density-based methods like DBSCAN, can detect anomalies by considering data points that do not belong to any cluster or fall in low-density regions. This is valuable in fraud detection, network security, and system monitoring.

Document Clustering

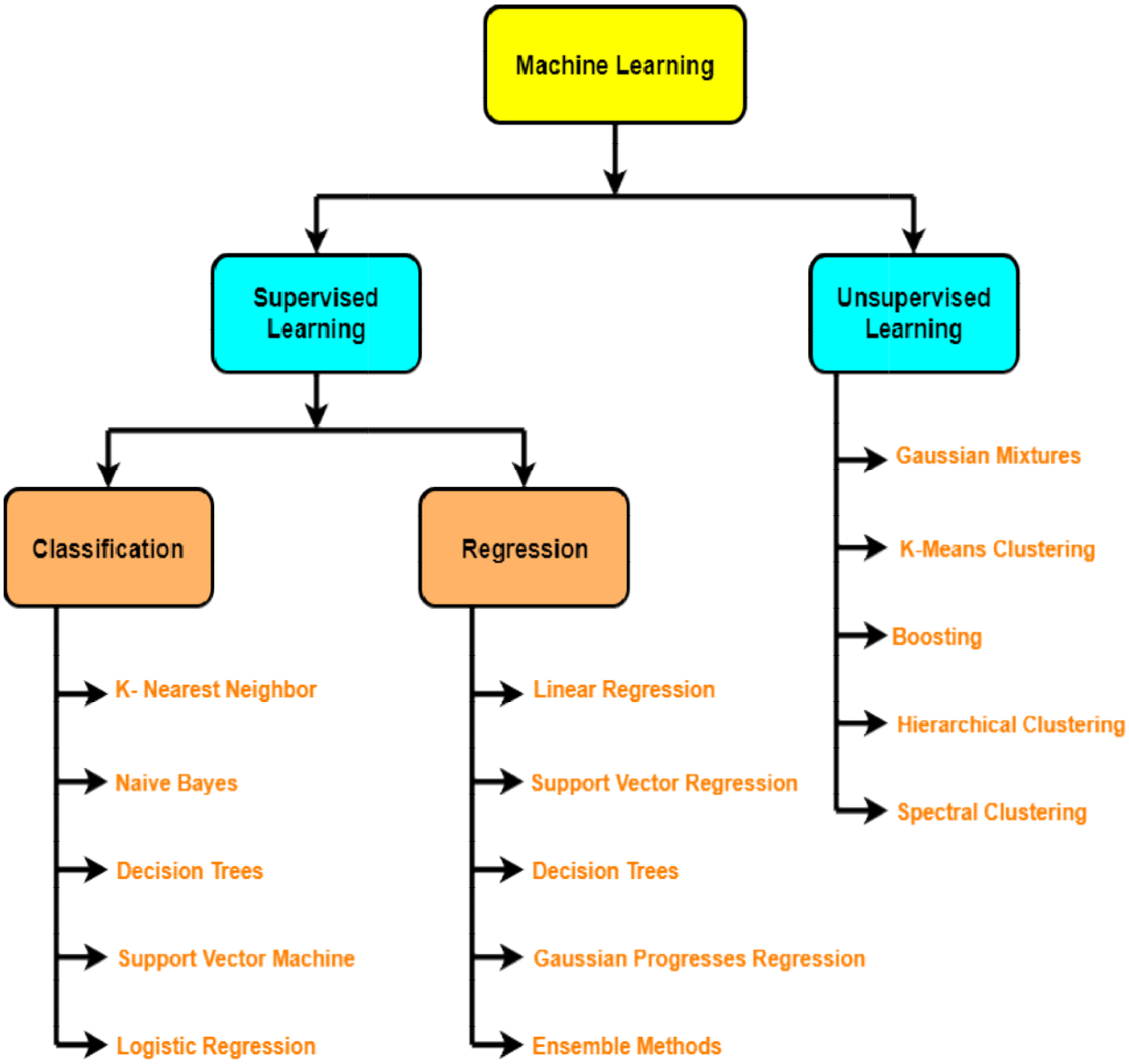
Definition: Document clustering organizes text documents into clusters based on their content similarity, topic, or theme.
Application: Clustering algorithms like Latent Dirichlet Allocation (LDA) or K-means are applied to group documents with similar words, phrases, or topics. Document clustering aids in information retrieval, content recommendation, and text summarization tasks.

Customer Relationship Management (CRM)

Definition: CRM involves managing interactions with existing and potential customers to improve business relationships and drive sales growth.

Application: Clustering helps in segmenting customers based on their purchasing behavior, preferences, and demographics. This enables personalized marketing campaigns, targeted promotions, and effective customer retention strategies. These applications demonstrate the versatility and utility of clustering in solving real-world problems across diverse domains, providing valuable insights from complex datasets.

MACHINE LEARNING WORKFLOW



METHODOLOGY

OBJECTIVE OF THE PROJECT: Customer segmentation is the practice of dividing a company's customers into groups that reflect similarities among customers in each group. The main objective of segmenting customers is to decide how to relate to customers in each segment to maximize the value of each customer to the business The emergence of many competitors and entrepreneurs has caused a lot of tension among competing businesses to find new buyers and keep the old ones. As a result of the predecessor, the need for exceptional customer service becomes appropriate regardless of the size of the business. Furthermore, the ability of any business to understand the needs of each of its customers will provide greater customer support in providing targeted customer services and developing customized customer service plans. This understanding is possible through structured customer service.

SOFTWARE AND HARDWARE REQUIREMENTS

Software requirements

- Python
- Anaconda
- Jupiter Notebooks
- Hardware requirements:

- Processor: Intel Core i5
- RAM: 4GB
- OS: Windows

LIBRARIES USED:

Pandas: Used for data manipulation and analysis, including loading the dataset, cleaning, and pre-processing.

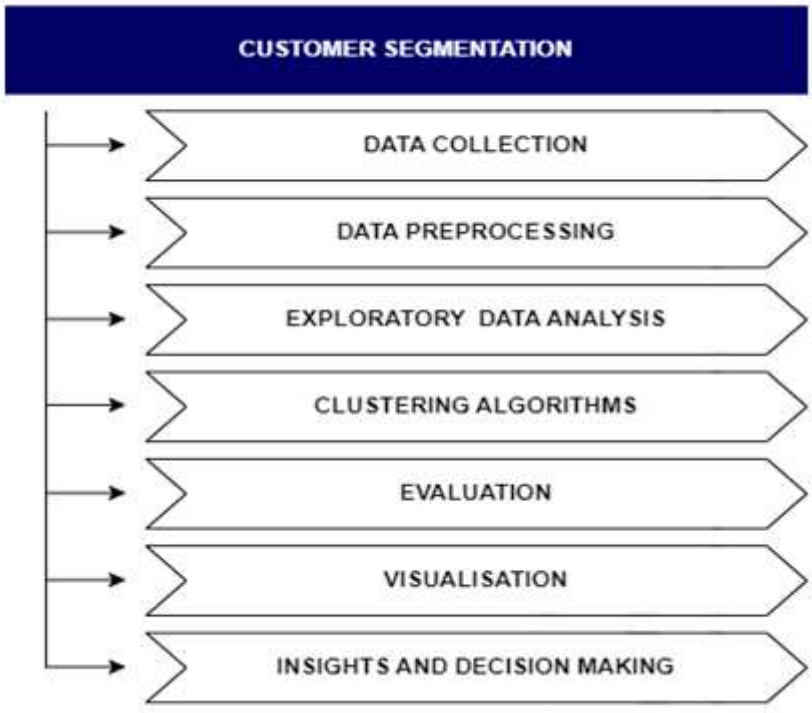
- **Num Py:** Provides support for mathematical operations on arrays and matrices, used for numerical computations.
- **Matplotlib:** Enables visualization of data through various plots like histograms, box plots, bar plots, and scatter plots.
- **Seaborn:** Provides high-level interface for drawing attractive statistical graphics, enhances the visualization aesthetics.
- **Scikit-learn (sklearn):** Implements machine learning algorithms for clustering, including K-means clustering.
- **SciPy:** Provides hierarchical clustering algorithms and distance metrics for clustering analysis.

PROGRAMMING LANGUAGES

Python: Python is the best programing language fitted to Machine Learning. In step with studies and surveys, Python is the fifth most significant language yet because the preferred language for machine learning and information science. It's owing to the subsequent strengths that Python has – Easy to be told and perceive- The syntax of Python is simpler; thence it's comparatively straightforward, even for beginners conjointly, to be told and perceive the language. Multi-purpose language − Python could be a multi-purpose programing language as a result of it supports structured programming, object-oriented programming yet as practical programming. Support of open supply community − As being open supply programing language, Python is supported by a giant developer community. Because of this, the bugs square measure is simply mounted by the Python community. This characteristic makes Python strong and adaptive.

DOMAIN: Machine learning could be a subfield of computer science (AI). The goal of machine learning typically is to know the structure information of knowledge of information and match that data into models which will be understood and used by folks. Although machine learning could be a field inside technology, it differs from ancient process approaches. In ancient computing, algorithms are sets of expressly programmed directions employed by computers to calculate or downside solve. Machine learning algorithms instead give computers to coach on knowledge inputs and use applied math analysis to output values that fall inside a particular vary. Thanks to this, machine learning facilitates computers in building models from sample knowledge to modify decision-making processes supported knowledge inputs.

SYSTEM ARCHITECTURE



- **Data Collection:**
  - Initial dataset containing customer information such as age, gender, annual income, and spending score.
- **Data Pre-processing:**
  - Cleaning the dataset by handling missing values, outliers, and irrelevant columns.

- Encoding categorical variables like gender if necessary.
- Scaling numerical features to ensure all features contribute equally to the clustering process.

- **Exploratory Data Analysis (EDA):**
  - Visualization of the dataset to gain insights into customer demographics and spending behavior.
  - Histograms, box plots, and correlation matrices help understand the distribution and relationships between features.
- **Clustering Algorithms:**
  - K-means Clustering:
- The main algorithm used for customer segmentation.
- Iteratively assigns data points to clusters based on their proximity to cluster centroids.
- Number of clusters (k) determined using techniques like the Elbow Method or Silhouette Score.
- Evaluation:
  - Silhouette Score Method:
- Evaluates the quality of clustering by measuring the cohesion within clusters and separation between clusters.
- Helps in determining the optimal number of clusters.
  - Within-Cluster Sum of Squares (WCSS):
- Measures the compactness of clusters by summing the squared distances between data points and their respective centroids.
- Helps identify the "elbow point" to determine the optimal number of clusters.

Visualization

- Visualization of clustered data points in 2D or 3D space.
- Scatter plots or 3D plots showing clusters based on customer attributes like age, annual income, and spending score.
- Insights and Decision Making:
- Interpretation of clustered customer segments to derive actionable insights.
- Decision-making support for marketing strategies, product offerings, and customer engagement initiatives.

ALGORITHMS USED

- K-Means Clustering: It is a partitioning algorithm that divides the dataset into K distinct non-overlapping clusters, where each data point belongs to only one cluster. The algorithm iteratively assigns each data point to the nearest centroid and then recalculates the centroids based on the mean of the data points assigned to each cluster. This process continues until the centroids no longer change significantly or a specified number of iterations is reached.

Step-by-step approach

- Choose the number of clusters (k) based on domain knowledge or using techniques like the Elbow Method or Silhouette Score.
- Initialize k centroids randomly in the feature space.
- Assign each data point to the nearest centroid, forming k clusters.
- Update the centroids by calculating the mean of all data points assigned to each cluster.
- Repeat the assignment and centroid update steps until convergence, i.e., when the centroids no longer change significantly, or a specified number of iterations is reached.

Silhouette Score: This is not an algorithm, but a metric used to evaluate the quality of clustering. It measures how similar a data point is to its own cluster compared to other clusters. A higher silhouette score indicates that the data point is well matched to its own cluster and poorly matched to neighbouring clusters.

Step-by-step approach

- For each value of k (number of clusters), perform K-means clustering.
- Calculate the silhouette score for each data point, which measures the cohesion within clusters and separation between clusters.
- Average the silhouette scores across all data points to obtain the overall silhouette score for the clustering.
- Select the value of k that maximizes the silhouette score, indicating the optimal number of clusters.

Within-Cluster Sum of Squares (WCSS): WCSS measures the compactness of clusters by calculating the sum of squared distances between data points and their respective cluster centroids. It helps determine the optimal number of clusters by identifying the point where the rate of decrease in WCSS slows down significantly.

Step-by-step approach

- For each value of k, perform K-means clustering.
- Calculate the sum of squared distances between each data point and its assigned centroid for all clusters.
- Sum these squared distances to obtain the within-cluster sum of squares (WCSS) for the clustering.
- Plot the WCSS against the number of clusters and identify the "elbow point" where the rate of decrease in WCSS slows down significantly, indicating the optimal number of clusters.

DATASET

- **Customer ID:** Unique identifier for each customer. Data type: String or object.
- Gender: Gender of the customer (Male or Female). Data type: String or object.
- Age: Age of the customer. Data type: Integer or numeric.
- Annual Income (k$): Annual income of the customer in thousands of dollars. Data type: Integer or numeric.
- Spending Score (1-100): Spending score assigned to the customer based on their purchasing behavior and other factors. Data type: Integer or numeric.

IMPLEMENTATION

MODULES

Dataset Collection

- **Objective:**
  - Obtain the dataset required for customer segmentation analysis.
- **Activities:**
  - Load the dataset from a specified source (e.g., CSV file, database).
  - Perform initial exploration to understand the structure, features, and data types.
  - Pre-process the dataset if necessary, including handling missing values, encoding categorical variables, and scaling numerical features.
- **Tools:**
  - Pandas for data loading and manipulation.
  - NumPy for numerical operations.
- **Output:**
  - Cleaned and pre-processed dataset ready for model training.

Training the Model

- **Objective:**
  - Implement and train the K-means clustering model on the dataset.
- **Activities:**
  - Choose the appropriate number of clusters (K) using evaluation metrics like Silhouette Score and WCSS.
  - Train the K-means clustering algorithm on the dataset.
  - Assign cluster labels to each data point based on the trained model.
- **Tools:**
  - scikit-learn (sklearn) for implementing K-means clustering.
  - SciPy for hierarchical clustering (optional).
- **Output:**
  - Trained clustering model with assigned cluster labels.

Deployment

- **Objective:**
  - Deploy the trained model for customer segmentation analysis.
- **Activities:**
  - Visualize the clusters using various plots to understand customer segments.
  - Provide insights and interpretations based on the clustering results.
  - Save the trained model and necessary pre-processing steps for future use.
- **Tools:**

- Matplotlib and Seaborn for data visualization.
- Joblib for saving/loading the trained model.

- **Output:**
  - Visualizations and insights derived from customer segmentation analysis.
  - Saved model and pre-processing steps for deployment in production environment.

CALCULATIONS

Distance Calculation: The Euclidean distance formula is commonly used to calculate the distance between two points in a multi-dimensional space. For each data point and centroid, the distance is calculated to determine which centroid a data point is closest to Euclidean Distance between two points P1(x1,y1,z1,...) and P2(x2,y2,z2,...):

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \dots}$$

Centroid recalculation: After assigning data points to clusters, the centroids of the clusters are recalculated by taking the mean of the data points belonging to each cluster. The formula for the centroid of a cluster is: Centroid C(xc,yc,zc,...):

$$\left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n}, \frac{\sum z_i}{n}, \dots\right)$$

where xi, yi, zi are the coordinates of data points in the cluster and n is the number of data points in the cluster.

WCSS (Within-Cluster Sum of Squares): This metric is used to evaluate the clustering quality by measuring the compactness of the clusters. It is calculated as the sum of squares of the distances between each data point and its assigned centroid within a cluster. The formula for WCSS is:

$$WCSS = \sum_{i=1}^{k} \sum_{j=1}^{n} (x_{ij} - \mu_i)^2$$

where k is the number of clusters, n is the number of data points in cluster i, xij is the jth data point in cluster i, and μi is the centroid of cluster i.

Silhouette score: The silhouette score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters. The silhouette score s(i) for each sample i is calculated using the following formula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- Is the average distance from the sample i to other points in the same cluster.
- Is the smallest average distance from the sample i to points in a different cluster, minimized over clusters.

The silhouette score for the entire dataset is the average of the silhouette score for each sample. Higher silhouette scores indicate better clustering. This score can be calculated using the silhouette_score function provided by scikit-learn' s metrics module.

CONCLUSION

The project successfully demonstrated the application of K-means clustering for customer segmentation based on age, annual income, and spending score. Through comprehensive data exploration and visualization, we gained valuable insights into the distribution and characteristics of customers in the dataset. By employing internal evaluation metrics such as the Within-Cluster Sum of Squares (WCSS) and silhouette score, we determined the optimal number of clusters and assessed the quality of the clustering results.

The visualization of clusters in a 3D scatter plot provided a clear depiction of distinct customer segments, allowing businesses to discern patterns and preferences among different groups of customers. These insights can inform targeted marketing strategies, personalized product recommendations, and enhanced customer experiences. Overall, customer segmentation using K-means clustering offers businesses a powerful tool for understanding their customer base, identifying valuable segments, and tailoring marketing efforts to meet specific needs and preferences. By leveraging the findings from this project, businesses can optimize resource allocation, improve customer engagement, and ultimately drive business growth and profitability in today's competitive market landscape.

APPENDICES

SOURCE CODE

#Importing libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

#Load dataset

```
df = pd.read_csv("/content/Mall_customer_list.csv")
# Data exploration
print(df.head())
print(df.info())
print(df.describe())
```

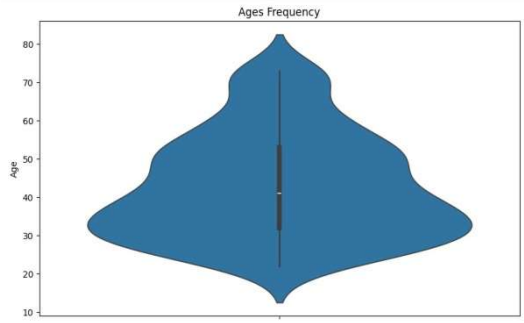
#Data Visualization

```
df.drop(["CustomerID"], axis = 1, inplace=True)
plt.figure(figsize=(10,6))
plt.title("Ages Frequency")
sns.axes_style("dark")
sns.violinplot(y=df["Age"])
plt.show()

plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.boxplot(y=df["Spending Score (1-100)"], color="red")
plt.subplot(1,2,2)
sns.boxplot(y=df["Annual Income (k$)"])
plt.show()

genders = df['Genders'].value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10, 4))
sns.barplot(x=genders.index, y=genders.values)
plt.xlabel('Gender')  # Set the x-axis label
plt.ylabel('Count')   # Set the y-axis label
plt.title('Distribution of Gender')
plt.show()

age18_25 = df.Age[(df.Age<= 25) & (df.Age>= 18)]
age26_35 = df.Age[(df.Age<= 35) & (df.Age>= 26)]
age36_45 = df.Age[(df.Age<= 45) & (df.Age>= 36)]
age46_55 = df.Age[(df.Age<= 55) & (df.Age>= 46)]
age55above = df.Age[df.Age>= 56]

x = ["18-25","26-35","36-45","46-55","55+"]
y = [len(age18_25.values),len(age26_35.values),len(age36_45.values),len(age46_55.values),
len(age55above.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=x, y=y, palette="rocket")
plt.title("Number of Customer and Ages")
plt.xlabel("Age")
plt.ylabel("Number of Customer")
plt.show()

ss1_20 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 1) & (df["Spending Score (1-100)"] <= 20)]
ss21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 21) & (df["Spending Score (1-100)"] <= 40)]
ss41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 41) & (df["Spending Score (1-100)"] <= 60)]
ss61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 61) & (df["Spending Score (1-100)"] <= 80)]
ss81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 81) & (df["Spending Score (1-100)"] <= 100)]
```

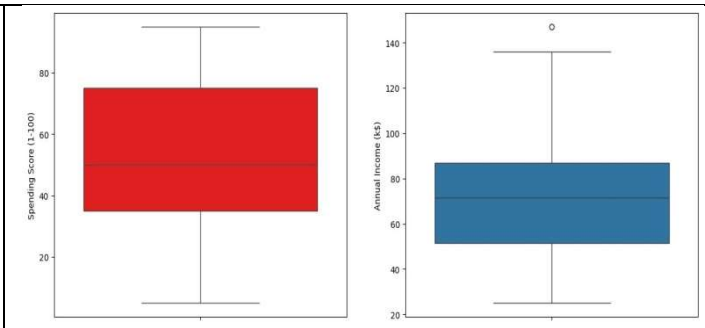
```
ssx = ["1-20", "21-40", "41-60", "61-80", "81-100"]
ssy = [len(ss1_20.values), len(ss21_40.values), len(ss41_60.values), len(ss61_80.values), len(ss81_100.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=ssx, y=ssy, palette="nipy_spectral_r")
plt.title("Spending Scores")
plt.xlabel("Score")
plt.ylabel("Number of Customer Having the Score")
plt.show()

ai0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) & (df["Annual Income (k$)"] <= 30)]
ai31_60 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 31) & (df["Annual Income (k$)"] <= 60)]
ai61_90 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 61) & (df["Annual Income (k$)"] <= 90)]
ai91_120 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 91) & (df["Annual Income (k$)"] <= 120)]
ai121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 121) & (df["Annual Income (k$)"] <= 150)]

aix = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$ 90,001 - 120,000", "$ 120,001 - 150,000"]
aiy = [len(ai0_30.values), len(ai31_60.values), len(ai61_90.values), len(ai91_120.values), len(ai121_150.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=aix, y=aiy, hue=aix, palette="Set2", legend=False)
plt.title("Annual Incomes")
plt.xlabel("Income")
plt.ylabel("Number of Customers")
plt.show()

# Plotting inertia and silhouette scores to choose k
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np

# Assuming you want to exclude the 'Gender' column from clustering
numeric_df = df.select_dtypes(include=[np.number])

silhouette_scores = []


for k in range(2, 11):  # Silhouette score requires at least 2 clusters
kmeans = KMeans(n_clusters=k, init="k-means++", n_init=10)
cluster_labels = kmeans.fit_predict(numeric_df)
silhouette_avg = silhouette_score(numeric_df, cluster_labels)
silhouette_scores.append(silhouette_avg)

plt.figure(figsize=(12, 6))
plt.grid()
plt.plot(range(2, 11), silhouette_scores, linewidth=2, color="blue", marker="o")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score Method for Optimal K")
plt.xticks(np.arange(2, 11, 1))
plt.show()

# Plotting the Within-Cluster Sum of Squares (WCSS) for Different Values of K
from sklearn.cluster import KMeans
# Assuming you want to exclude the 'Gender' column from clustering

numeric_df = df.select_dtypes(include=[np.number])

wcss = []

for k in range(1, 11):
kmeans = KMeans(n_clusters=k, init="k-means++", n_init=10)  # Explicitly set n_init
kmeans.fit(numeric_df)
wcss.append(kmeans.inertia_)
```

```
plt.figure(figsize=(12, 6))
plt.grid()
plt.plot(range(1, 11), wcss, linewidth=2, color="red", marker="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1, 11, 1))
plt.ylabel("WCSS")
plt.show()

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Assuming 'Gender' is a non-numeric column, and you want to exclude it

numeric_df = df.select_dtypes(include=[np.number])

# Explicitly set n_init to suppress the FutureWarning
km = KMeans(n_clusters=5, n_init=10)
clusters = km.fit_predict(numeric_df)
df["label"] = clusters

# Plotting the clusters in 3D
fig = plt.figure(figsize=(20, 10))
ax = fig.add_subplot(111, projection='3d')

# Scatter plots for each cluster
for label in range(5):
ax.scatter(df.Age[df.label == label],
df["Annual Income (k$)"][df.label == label],
df["Spending Score (1-100)"][df.label == label],
        label=f'Cluster {label}', s=60)

ax.view_init(30, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

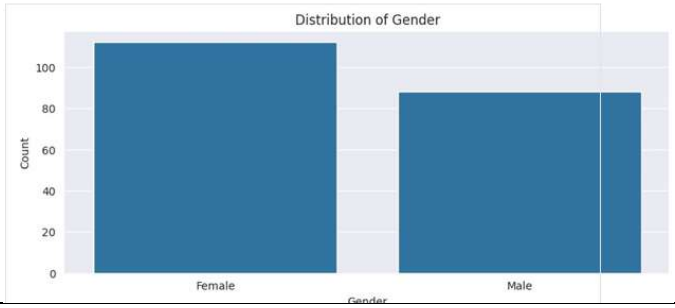
```
   CustomerID Genders  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   25                  25                      40
1           2    Male   27                  25                      80
2           3  Female   26                  26                      10
3           4  Female   29                  26                      75
4           5  Female   35                  27                      45
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genders                 200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   42.700000           70.350000               50.340000
std     57.879185   13.750468           26.150848               25.025703
min      1.000000   22.000000           25.000000                5.000000
25%     50.750000   32.000000           51.500000               35.000000
50%    100.500000   41.000000           71.500000               50.000000
75%    150.250000   53.000000           87.000000               75.000000
max    200.000000   73.000000          147.000000               95.000000
```



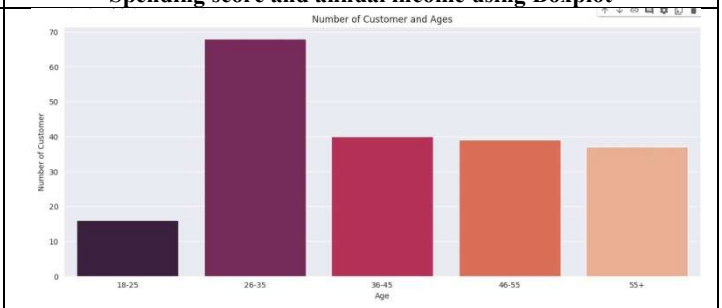
Age frequency using violinplot



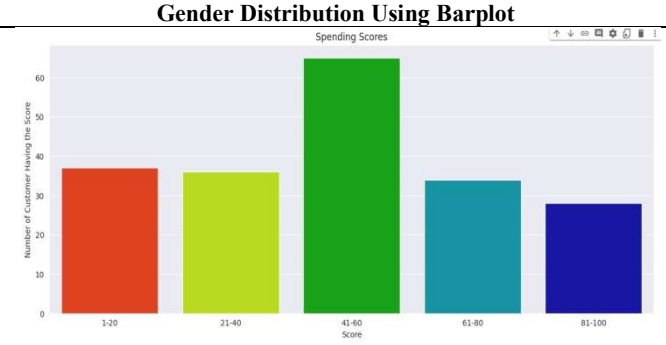
Spending score and annual income using Boxplot



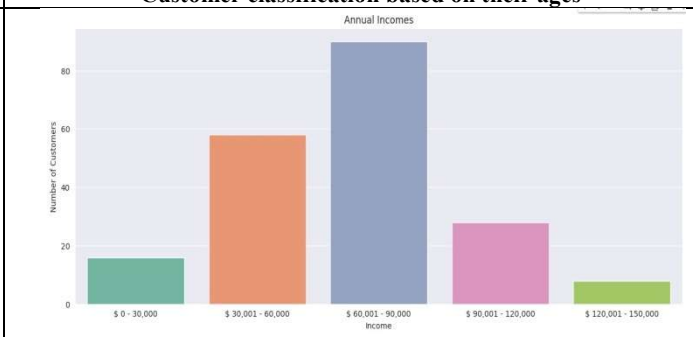
Gender Distribution Using Barplot



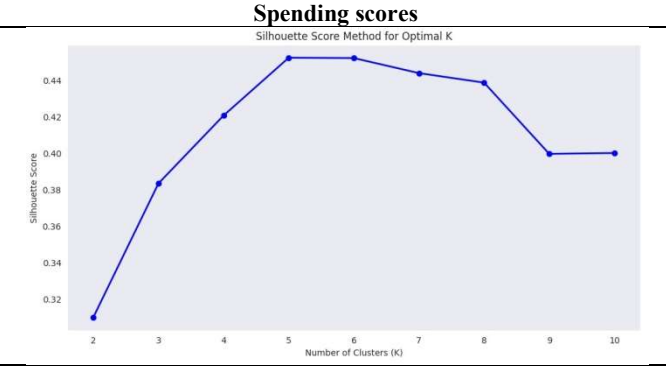
Customer classification based on their ages



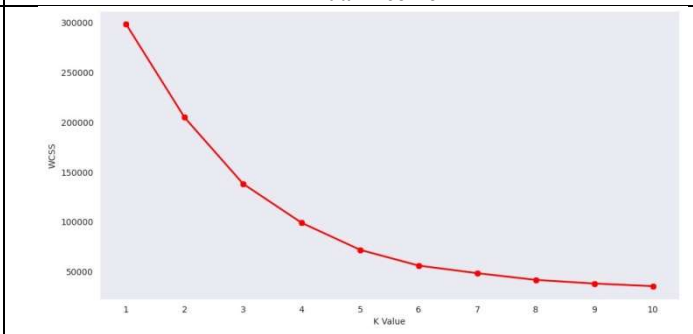
Spending scores



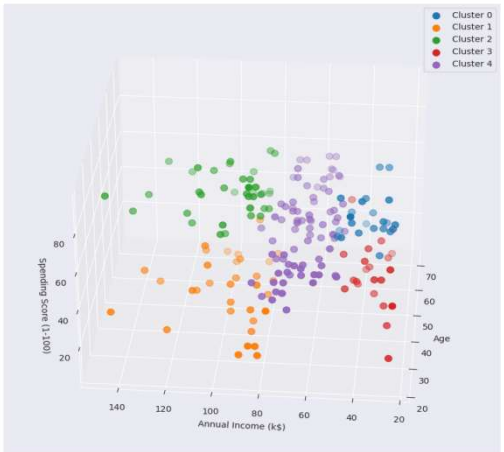
Annual income



Silhouette score



Plotting k value



Plotting clusters in 3d:

LIST OF ABBREVIATIONS

KMeans	K-means clustering algorithm
WCSS	Within-Cluster Sum of Squares.
EDA	Exploratory Data Analysis.
ML	Machine Learning.

REFERENCES

K-means Clustering" by Scikit-learn: Official documentation on K-means clustering from the Scikit-learn library, providing detailed information on implementation and usage.

Understanding K-means Clustering in Machine Learning" by Analytics Vidhya: An article on Analytics Vidhya explaining the concept of K-means clustering, its implementation, and use cases.

K-Means Clustering in Python" by Data Camp: A tutorial on DataCamp covering K-means clustering implementation in Python, along with examples and explanations.

Introduction to K-means Clustering" on Kaggle: A beginner-friendly introduction to K-means clustering on Kaggle, including code examples and practical insights.

K-means Clustering in Python with Example" by KDnuggets: An article on KDnuggets providing a step-by-step guide to implementing K-means clustering in Python, with a practical example.