K-MEANS CLUSTERING-BASED CUSTOMER SEGMENTATION: A PYTHON IMPLEMENTATION WITH REAL-WORLD APPLICATIONS

A DESIGN PROJECT REPORT

Submitted by

RAHUL R (811721104081)

SAMSON JEBASEELAN C (811721104086)

SURAJ M (811721104108)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER,2024

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"K-MEANS CLUSTERING-BASED CUSTOMER SEGMENTATION: A PYTHON IMPLEMENTATION WITH REAL-WORLD APPLICATIONS"** is the bonafide work of **RAHUL R (811721104081)**, **SAMSON JEBASEELAN C (811721104086), SURAJ M (811721104108)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. A. Delphin Carolina Rani M.E., Ph.D.,
HEAD OF THE DEPARTMENT

PROFESSOR AND HEAD
Department of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Dr. A. Delphin Carolina Rani M.E., Ph.D.,
SUPERVISOR

PROFESSOR AND HEAD
Department of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"K-MEANS CLUSTERING-BASED CUSTOMER SEGMENTATION: A PYTHON IMPLEMENTATION WITH REAL-WORLD APPLICATIONS"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

RAHUL R

SAMSON JEBASEELAN C

SURAJ M

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution, "**K. Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

We would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

We thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Head of the Department of **COMPUTER SCIENCE AND ENGINEERING,** for providing her encouragement in pursuing this project.

We wish to convey our profound and heartfelt gratitude to our esteemed project guide **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance andpatience, which motivated us to carry out this project.

We render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

This project focuses on customer segmentation using the K-Means clustering algorithm implemented in Python. By leveraging Python's data analysis libraries like Pandas, NumPy, and visualization tools such as Matplotlib and Seaborn, the study analyzes customer data to identify distinct groups based on behavioral and demographic attributes. The K-Means algorithm, an unsupervised machine learning technique, is employed to minimize intra-cluster variance while maximizing inter-cluster differences, enabling meaningful segmentation. The project highlights the use of techniques like the elbow method and silhouette score to determine the optimal number of clusters. The insights gained from this segmentation can assist businesses in enhancing customer relationship management, enabling personalized marketing strategies, and improving overall customer satisfaction and retention.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

EDA – Exploratory Data Analysis

CLV – Customer Lifetime Value

RFM – Recency, Frequency, and Monetary

PCA – Principal Component Analysis

CLR – Click Through Rates

GDPR – General Data Protection Regulation

CCPA – California Consumer Privacy Act

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Customer segmentation is the process of dividing a customer base into distinct groups based on shared characteristics, behaviors, or preferences, enabling businesses to tailor marketing strategies and improve decision-making. Python offers powerful libraries like Pandas, NumPy, Scikit-learn, Matplotlib, and Seaborn, making it an ideal tool for implementing customer segmentation. The process begins with collecting and preprocessing data, addressing missing values, outliers, and normalization to ensure quality. Exploratory Data Analysis (EDA) helps identify patterns and trends, while feature engineering creates metrics like Recency, Frequency, and Monetary (RFM) scores or Customer Lifetime Value (CLV). Clustering algorithms, such as K-Means, DBSCAN, or Hierarchical Clustering, group customers based on similarities, with dimensionality reduction techniques like Principal Component Analysis (PCA) aiding in handling complex data. After clustering, segments are profiled and visualized to reveal actionable insights, such as identifying high-value customers or price-sensitive groups.

1.2 OBJECTIVE

This Python-based project focuses on creating a data-driven solution to identify these segments, helping businesses optimize their marketing strategies, personalize customer interactions, and enhance overall decision-making. The process begins with data collection, where customer data, such as demographic details, purchase history, and behavioral patterns, are gathered from sources like CSV files, databases, or APIs. Afterward, data cleaning and preprocessing are conducted to handle missing values, duplicates, and outliers, ensuring the dataset is consistent and reliable. Exploratory Data Analysis (EDA) is then performed to uncover key patterns and trends.

1.3 SCOPE

segmentation is vital in industries like retail, e-commerce, banking, healthcare, and telecommunications, where personalized marketing, loyalty programs, and targeted recommendations significantly impact customer retention and revenue. By grouping customers based on behaviors, preferences, and demographics, companies can optimize resource allocation, improve campaign effectiveness, and predict trends. Moreover, Python's scalability and adaptability make it ideal for projects of varying complexity, ensuring it meets the needs of small businesses and large enterprises alike. With growing data availability and advancements in AI, the scope of customer segmentation continues to expand, empowering businesses to stay competitive in dynamic markets.

1.4 FEATURES

It includes data preprocessing to clean, normalize, and structure data for analysis and exploratory data analysis (EDA) to uncover patterns and trends in customer behavior. Python enables feature engineering to create meaningful metrics like Recency, Frequency, and Monetary (RFM) scores or Customer Lifetime Value (CLV). Using machine learning algorithms such as K-Means, DBSCAN, or Hierarchical Clustering, it segments customers into distinct groups based on shared characteristics. Advanced techniques like Principal Component Analysis (PCA) can reduce dimensionality for better clustering accuracy. Python's visualization tools like Matplotlib and Seaborn allow clear representation of customer segments, while integration with reporting tools helps generate actionable insights. These features empower businesses to personalize marketing, enhance customer experiences, and make data-driven decisions effectively.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: EVLOUTION OF SEGMENTATION PROCESS

AUTHOR: ATIS VERDENHOFS

YEAR:2019

The evolution of the segmentation process has transitioned from manual, intuition-based grouping to data-driven, automated methodologies, reflecting advancements in technology and analytics. Initially, businesses relied on basic demographic segmentation, categorizing customers by age, income, or location based on surveys or simple records. As markets grew more complex, behavioral and psychographic segmentation emerged, focusing on purchase habits, preferences, and lifestyles. The advent of digital data collection and computational power revolutionized segmentation by enabling the use of statistical techniques like cluster analysis. Today, the process has advanced further with machine learning and artificial intelligence, allowing for dynamic and predictive segmentation based on vast datasets. Tools like Python and sophisticated algorithms now enable businesses to analyze real-time data, uncover hidden patterns, and create highly personalized segments.

2.2 TITLE: USER ENGAGEMENT DYNAMICS

AUTHOR: NADINE KÖHLE

YEAR:2020

User engagement dynamics refer to the patterns and behaviors exhibited by users when interacting with a product, service, or platform, providing insights into their level of interest, satisfaction, and loyalty. These dynamics encompass various metrics such as time spent on a platform, frequency of interactions, click-through rates, content consumption patterns, and user retention rates. High engagement often indicates a strong alignment between the user's needs and the value proposition of the product or service. Key drivers of user engagement include personalized experiences, intuitive interfaces, timely feedback mechanisms, and consistent value delivery.

2.3 TITLE: EMERGENCE OF STATISTICAL TECHNIQUES

AUTHOR: ANTÓNIO PEDRO COSTA

YEAR:2022

The **emergence of statistical techniques** in customer segmentation marked a significant shift from manual and intuition-based methods to data-driven approaches that offered greater precision and scalability. This phase, beginning in the mid-20th century, was fueled by advancements in mathematics and computing, enabling businesses to analyze large datasets systematically. Techniques like **cluster analysis**, introduced to group customers based on similarities in behavior or demographics, became foundational. Regression analysis allowed for predicting customer responses to variables like pricing or promotions, while discriminant analysis helped in differentiating customer groups effectively.

2.4 TITLE: ETHICAL AND RESPONSIBLE SEGMENTATION

AUTHOR: MICHAEL PORTER

YEAR:2023

Ethical and responsible segmentation focuses on ensuring that customer segmentation practices respect individual privacy, promote fairness, and align with legal and societal standards. As businesses increasingly rely on vast amounts of customer data for segmentation, concerns about data misuse, discrimination, and lack of transparency have come to the forefront. Ethical segmentation emphasizes collecting and using data responsibly, ensuring customers' informed consent, and adhering to privacy regulations such as the **General Data Protection Regulation (GDPR)** and **California Consumer Privacy Act (CCPA)**.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 OVERVIEW

Customer segmentation is a strategic process that divides a company's customer base into distinct groups based on shared characteristics, behaviors, or needs. This segmentation enables businesses to understand their customers better and tailor their products, services, and marketing efforts accordingly. Traditional methods of segmentation often relied on demographic factors such as age, gender, income, or location. However, as markets have become more complex, modern segmentation incorporates behavioral, psychographic, and transactional data, allowing for more precise targeting and personalization.

In recent years, advancements in technology and analytics have revolutionized customer segmentation. The integration of big data and machine learning allows companies to analyze vast datasets, uncover hidden patterns, and create dynamic segments. Tools like Python and libraries such as Scikit-learn, Pandas, and Matplotlib facilitate data preprocessing, clustering, and visualization. Clustering algorithms like K-Means, DBSCAN, or Hierarchical Clustering are commonly employed to group customers, while techniques like Principal Component Analysis (PCA) help reduce dimensionality in complex datasets. These innovations enable businesses to move from static, rule-based segmentation to adaptive, data-driven approaches that evolve with customer behavior.

The benefits of customer segmentation extend across industries, from retail and e-commerce to finance and healthcare. Personalized marketing campaigns can target specific groups effectively, while loyalty programs can be designed for high-value customers. Predictive segmentation, which leverages historical data to forecast customer needs and preferences, helps businesses stay ahead of market trends.

3.2 REQUIREMENT SPECIFICATION

ID	NAME	FUNCTIONAL	NON-FUNCTIONAL	DESCRIPTION	PRIORITY	ACTOR
1	User registration	Yes		Functionality for user to create account	High	User
2	Login	Yes		Functional for user to get access	High	User
3	Logout	Yes		Functionality for user to delete session	High	User
4	Data Collection and Integration	Yes		Collection of data	High	Admin
5	Data Preprocessing	Yes		Cleaning of missing, inconsistent	High	User
6	Clustering and Segmentation	Yes		Support for clustering algorithms such as K-Means	Medium	User
7	Security		Yes	User privacy	High	Admin
8	Robustness		Yes	Dealing with errors	High	Admin
9	Performance		Yes	Application performance must be better	High	Admin
10	Usability		Yes	Easy for newbies	High	Admin
11	Reliability		Yes	Trusted by users	High	Admin

Table 3.1 Requirement Specification

3.3 USECASE DIAGRAMS

A use case diagram for customer segmentation illustrates the interactions between users and the system, highlighting core functionalities. The primary actors include the Business Analyst, Marketing Manager, Data Scientist, and Customer Database. Key use cases involve Importing Customer Data from various sources, Preprocessing Data to clean and normalize it, and Feature Engineering to create metrics like Recency, Frequency, and Monetary (RFM) values. The Perform Segmentation use case applies clustering algorithms like K-Means to group customers, while Analyze Segments profiles these groups for actionable insights. Additional functionalities include Visualizing Segments through charts and heatmaps, Generating Reports to summarize findings, and Exporting Segment Data for integration with other systems.

3.3.1 USECASE SPECIFICATION

Use Case ID	Use Case Description	Actors	Trigger	Preconditions	Postconditions
UC-01	User Login	User	User provides login details	User accesses login page	User authenticated and directed to the dashboard
UC-02	Import Customer Data	Admin	Load data from various sources such as CSV files, databases	Data Sources Identified	Data Integrity Verified
UC-03	Business Analyst	Admin	Analyzes segmentation insights	Data must be preprocessed	Analyst must have access to relevant and accurate customer data
UC-04	Generate Reports	Admin	recommendations for decision- making	Share segmented data with external systems	Provide summarized insights

Table 3.2 Usecase Specification

3.3.2 AUTHENTICATION

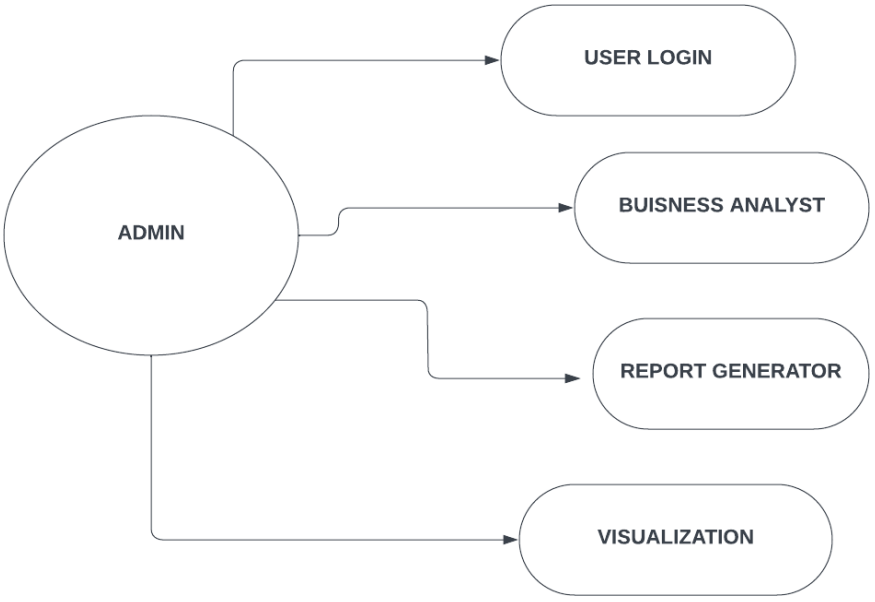


Fig 3.1 Authentication

3.3.3 SYSTEM DESIGN ARCHITECTURE

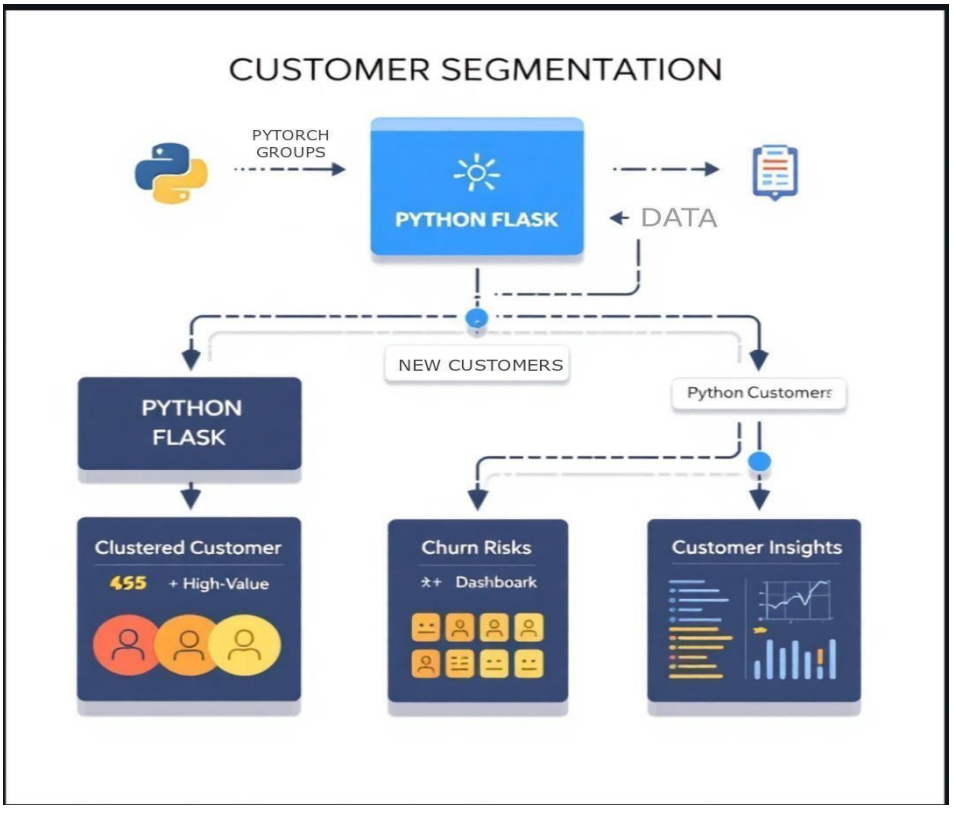


Fig 3.2 System design architecture

CHAPTER 4

BACKEND DEVELOPMENT

4.1 DATA PROCESSING

Data processing in customer segmentation involves collecting and preparing customer data from various sources such as transaction histories, social media activity, and demographic information. The first step is data cleaning, where irrelevant, missing, or erroneous data is removed or corrected. This is followed by data transformation, which might include normalizing or standardizing data for consistency. Next, feature engineering is performed, where new variables or attributes are created to highlight important patterns or behaviors. Afterward, clustering algorithms like K-means or DBSCAN are applied to group customers into distinct segments based on similarities in their characteristics. These segments can represent categories such as high-value customers, potential churn risks, or frequent buyers. The processed data is then ready for analysis, allowing businesses to tailor marketing strategies, improve customer service, or develop personalized offerings.

4.2 MODEL HANDLING

Model handling for customer segmentation involves selecting, training, and fine-tuning machine learning models to accurately group customers based on their behavior and characteristics. The first step is to choose an appropriate algorithm for segmentation, such as K-means clustering, hierarchical clustering, or DBSCAN, depending on the data type and the desired output. Once a model is selected, it's trained using the prepared customer data, where the model learns patterns and clusters based on predefined features. Model evaluation is then carried out using metrics like silhouette score or inertia to assess the quality and coherence of the segments. If necessary, the model is fine-tuned by adjusting hyperparameters, such as the number of clusters.

4.3 FILE MANAGEMENT

File management in customer segmentation involves organizing and storing the data, models, and outputs related to the segmentation process in a structured and efficient way. The first step is to manage **input data files**, which may include raw customer information in formats like CSV, Excel, or JSON. These files are cleaned and pre-processed, and the cleaned data is stored in a standardized format (e.g., CSV or Parquet) for easy access during model training. During the segmentation process, intermediate files such as **feature matrices**, **transformed data sets**, and **model parameters** are generated. These files are typically stored in a designated directory or cloud storage service to ensure they are accessible and organized. Proper file naming conventions, such as version control and clear folder structures, are essential to keep track of different iterations of data, models, and outputs.

4.4 FLASK

Flask can be used in customer segmentation to deploy machine learning models as web applications or APIs for real-time predictions and data visualization. After training a customer segmentation model, such as K-means clustering, Flask can load the model and expose endpoints where new customer data can be submitted, processed, and categorized into segments. It can also serve as the backend for interactive dashboards, displaying visualizations like cluster charts and segment statistics, allowing users to explore and analyze customer groups.

CHAPTER 5

FRONTEND DEVELOPMENT

5.1 RESPONSIVE DESIGN

Responsive design in customer segmentation refers to the development of web applications or dashboards that adapt seamlessly to different screen sizes and devices, ensuring a consistent user experience across desktops, tablets, and smartphones. In the context of customer segmentation, a responsive design enables users to access insights and interact with segmentation data from various devices, whether they're at their desk or on the go. For example, a marketing team can view customer segments, analyze segment characteristics, and visualize data patterns on a mobile device during a meeting, or on a desktop for detailed analysis. This adaptability is achieved using flexible layouts, scalable images, and media queries in CSS, allowing the interface to adjust dynamically to different screen resolutions. By employing responsive design in customer segmentation applications, businesses can improve accessibility and ensure that users can engage with the segmentation results anytime and anywhere, enhancing decision-making and real-time responsiveness.

5.2 CLIENTSIDE SCRIPTING

In the context of customer segmentation, client-side scripting can be used for tasks like dynamically displaying customer groups, visualizing segments through interactive charts (e.g., using libraries like D3.js or Chart.js), and filtering or sorting segment data in real time without needing to reload the page. For example, when a user selects a segment from a dropdown, JavaScript can update the displayed customer data or visualizations instantly, improving the user experience. Client-side scripting can also handle tasks like form validation when inputting new customer data for segmentation or providing immediate feedback to users about segmentation results. This reduces server load by offloading some tasks to the client, making the application more responsive and reducing delays in user interaction. By incorporating client-side scripting, customer segmentation applications can be more interactive, faster, and provide a more engaging user experience.

5.3 JAVA SCRIPT

JavaScript in customer segmentation can play a crucial role in building interactive and dynamic web applications that allow users to explore and visualize segmented customer data. It is commonly used for client-side tasks like manipulating customer data in real-time, creating interactive charts, and enhancing the user experience on segmentation dashboards. For instance, JavaScript libraries like Chart.js or D3.js can be used to create dynamic, interactive visualizations of customer segments, allowing users to explore different segments based on attributes like age, spending behavior, or churn risk. JavaScript can also facilitate the filtering and sorting of customer data directly in the browser without needing to reload the page, which improves performance and user interactivity. Additionally, JavaScript can be used to handle user inputs, such as selecting specific customer segments, updating filters, or applying segmentation models to new data. By integrating JavaScript into customer segmentation applications, businesses can provide real-time insights, interactive data visualizations, and a more responsive experience for users engaging with customer segmentations.

5.4 HTML & CSS

HTML and CSS form the foundational building blocks of the Photogram website, working in tandem to structure and style its content. HTML, the markup language, defines the website's structure, organizing elements like headers, paragraphs, images, and forms into a coherent layout. It establishes the backbone of the site, outlining the relationships between different sections and content components. CSS, on the other hand, takes charge of the website's presentation and visual appeal. Through CSS, the design elements specified in HTML are styled, including colors, fonts, spacing, and layout attributes. CSS enables the application of consistent design them across the website, ensuring a cohesive visual identity. In the context of Photogram, HTML structures the various sections of the platform, such as user profiles, image galleries, and engagement features, while CSS is responsible for styling these elements to create an engaging and aesthetically pleasing user interface. Together, HTML and CSS form an integral part of Photogram's development, working harmoniously to deliver a seamless and visually appealing web experience to its users

CHAPTER 6

TESTING AND DEPLOYMENT

6.1 DEPLOYMENT PROCESS

The deployment process of customer segmentation begins with preparing the model for deployment. After training the segmentation model (such as K-means, DBSCAN, or hierarchical clustering), the first step is to ensure that the model is properly saved in a format suitable for deployment (e.g., pkl or .h5 for machine learning models). The necessary dependencies and environment for running the model (including Python libraries like scikit-learn or TensorFlow) should be bundled together in a virtual environment or containerized using Docker. This ensures that the model can be run on any machine with the same environment, making it portable and reducing issues related to software versioning.

The next step involves setting up an appropriate server environment to host the model. This often involves using cloud platforms like AWS, Google Cloud, or Azure, or deploying on local servers. A common approach is to use Flask or Fast API for creating a web API, where the customer data can be sent via HTTP requests (like POST) and the model will return the predicted segment. This allows real-time predictions, where new customer data is processed and assigned to a specific segment as it enters the system. Additionally, a database or storage system is integrated to store and manage customer data, segment labels, and any associated metadata.

Finally, monitoring and maintenance are crucial parts of the deployment process. Once the segmentation model is live, it is important to monitor its performance to ensure it continues to deliver accurate results. This can involve tracking key metrics, such as prediction accuracy and response times, and setting up alerts for any performance issues. Regular updates and retraining might be necessary as customer behavior evolves or new data becomes available. Continuous integration/continuous deployment (CI/CD) pipelines can be set up to automate the testing and updating of the model, ensuring that the deployed system is always using the most current version of the segmentation model.

CHAPTER 7

FUTURE ENHANCEMENT

7.1 FUTURE ENHANCEMENT

The future enhancement of customer segmentation using the K-Means algorithm can incorporate a coin-based reward system to drive personalized advertising. By assigning coins to customers based on their interactions, such as purchases or referrals, the K-Means algorithm can segment customers not only by traditional factors like demographics and behavior but also by their coin accumulation. This segmentation enables targeted advertising where high-coin customers are shown premium products or exclusive deals, while low-coin customers receive incentives to engage more, such as discounts or additional coins for specific actions. By leveraging this enhanced segmentation, businesses can create dynamic marketing campaigns tailored to each customer group, improving customer engagement, loyalty, and conversion rates.

Another potential future enhancement for customer segmentation using the K-Means algorithm is integrating real-time data streams and advanced AI models for dynamic clustering. By continuously monitoring customer interactions through online and offline channels, businesses can update clusters in real-time, adapting to changing customer preferences and market trends. Additionally, integrating sentiment analysis from customer reviews and social media posts can provide deeper insights into customer attitudes, enabling businesses to fine-tune their offerings. By combining K-Means with predictive analytics, businesses can also anticipate future purchasing patterns and proactively design personalized recommendations. This advanced approach can empower organizations to stay competitive, enhance customer satisfaction, and drive sustained business growth.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

Customer segmentation plays a vital role in modern businesses, allowing them to tailor products, services, and marketing efforts to different groups based on their behaviors, preferences, and characteristics. By analyzing customer data, businesses can identify distinct segments that share common traits, such as high-value customers, frequent buyers, or those at risk of churn. This segmentation enables businesses to allocate resources more effectively, create personalized marketing campaigns, and deliver improved customer experiences, ultimately increasing customer loyalty and revenue.

Implementing customer segmentation involves several key steps, including data collection, preprocessing, model training, and evaluation. Different techniques, such as clustering algorithms (e.g., K-means, DBSCAN), are applied to group customers based on relevant features. It is crucial to properly handle the data to ensure the segmentation model's effectiveness, as well as to continually monitor and fine-tune the model for better accuracy. The process also includes deploying segmentation models into production environments, ensuring they are accessible for real-time predictions and analysis, which can drive dynamic decision-making.

In conclusion, customer segmentation is a powerful tool for businesses to understand their customer base in-depth and make data-driven decisions. Wit the integration of advanced machine learning models, cloud technologies, and interactive user interfaces, businesses can gain actionable insights into customer behavior, leading to improved targeting, better customer relationships, and a competitive edge in the market. As data continues to grow, the importance of effective customer segmentation will only increase, making it an essential strategy for business growth and innovation.

APPENDICES I

A.1 SAMPLE CODING FOR CUSTOMER SEGMENTATION: FILE CONFIGURATION

MACHINE LEARNING CODE:

Importing the Dependencies:

```
import numpy as np  # type: ignore
import pandas as pd # type: ignore
import matplotlib.pyplot as plt # type: ignore
import seaborn as sns # type: ignore
from sklearn.cluster import KMeans # type: ignore
from sklearn.preprocessing import LabelEncoder # type: ignore
```

Data Collection & Analysis:

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

# Missing Value Handling
customer_data.fillna(customer_data.mean(), inplace=True)
print("Missing values after handling:")
print(customer_data.isnull().sum())
X = customer_data[['Age', 'Average Amount Spent', 'Spending Score']].values
print("Any NaN values in X:", np.isnan(X).any())

 # first 5 rows in the dataframe
customer_data.tail()

 # finding the number of rows and columns
customer_data.shape

 # getting some informations about the dataset
Customer data.info()

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

Choosing the number of clusters:

```
customer_data.fillna(customer_data.mean(),  inplace=True)
```

WCSS -> Within Clusters Sum of Squares:

```
X = customer_data[['Age', 'Average Amount Spent', 'Spending Score']].values
```

 # 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()
```

Optimum Number of Clusters = 5

Training the k-Means Clustering Model

```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
Y = kmeans.fit_predict(X)
```

5 Clusters - 0, 1, 2, 3, 4

Visualizing all the Clusters:

 # plotting all the clusters and their Centroids

```
plt.figure(figsize=(8,8))
plt.scatter(X[Y == 0, 0], X[Y == 0, 1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y == 1, 0], X[Y == 1, 1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y == 2, 0], X[Y == 2, 1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y == 3, 0], X[Y == 3, 1], s=50, c='violet', label='Cluster 4')
plt.scatter(X[Y == 4, 0], X[Y == 4, 1], s=50, c='blue', label='Cluster 5')
```

 # Plot the centroids
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='cyan', label='Centroids')

plt.title('Customer Groups')
plt.xlabel('Age')
plt.ylabel('Spending Score')
plt.legend()
plt.show()

FLASK CODE (app.py):

```
import pandas as pd # type: ignore
from flask import Flask, render_template, request, redirect, url_for # type: ignore
import os

app = Flask(_name_)

# Path to the CSV file
csv_file_path = 'customers.csv'

# Create CSV file with headers if it doesn't exist
if not os.path.isfile(csv_file_path):
    df = pd.DataFrame(columns=["age", "purchase_frequency", "average_spent",
"spending_score"])
    df.to_csv(csv_file_path, index=False)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    # Get form data
    age = int(request.form['age'])
    purchase_frequency = int(request.form['purchase_frequency'])
    average_spent = float(request.form['average_spent'])

    # Calculate spending score (example calculation, modify as needed)
    spending_score = purchase_frequency * average_spent

    # Create a new DataFrame with the new data
    new_data = pd.DataFrame([[age, purchase_frequency, average_spent, spending_score]],
                            columns=["age", "purchase_frequency", "average_spent", "spending_score"])

    # Append to the existing CSV file
    try:
        new_data.to_csv(csv_file_path, mode='a', header=False, index=False)
    except Exception as e:
        print("Error while writing to CSV:", e) # Debug print to check for issues

    return redirect(url_for('index'))

if _name_ == '_main_':
    app.run(debug=True)
```

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Segmentation Form</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background: linear-gradient(135deg, #f5f7fa, #c3cfe2);
            color: #333;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }
        .container {
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
            width: 90%;
            max-width: 500px;
            text-align: center;
            transition: transform 0.3s, box-shadow 0.3s;
        }
        .container:hover {
            transform: translateY(-5px);
            box-shadow: 0 12px 24px rgba(0, 0, 0, 0.3);
        }
        h1 {
            color: #007bff;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-weight: bold;
```

```
}
input[type="number"],
select {
    width: 100%;
    padding: 12px;
    margin-bottom: 15px;
    border: 1px solid #007bff;
    border-radius: 5px;
    transition: border-color 0.3s;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
input[type="number"]:focus,
select:focus {
    border-color: #0056b3;
    outline: none;
}
input[type="submit"] {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 15px;
    cursor: pointer;
    border-radius: 5px;
    transition: background-color 0.3s ease, transform 0.2s;
    width: 100%;
    font-size: 16px;
}
input[type="submit"]:hover {
    background-color: #0056b3;
    transform: translateY(-2px);
}
.footer {
    margin-top: 20px;
    font-size: 14px;
    color: #666;
}
.error {
    color: red;
    margin-top: 10px;
}
.input-container {
    position: relative;
    margin-bottom: 20px;
}
```

```
        .input-container i {
            position: absolute;
            top: 12px;
            left: 10px;
            color: #007bff;
        }
        .success-message {
            background-color: #dff0d8;
            color: #3c763d;
            padding: 10px;
            border-radius: 5px;
            margin-top: 15px;
            display: none; /* Hidden by default */
        }
    </style>
    <script>
        function validateForm() {
            const age = document.getElementById("age").value;
            if (age > 100) {
                alert("Age must be less than or equal to 100.");
                return false;
            }
            return true;
        }
    </script>
</head>
<body>

<div class="container">
    <h1><i class="fas fa-users"></i> Customer Segmentation Form</h1>
    <form action="/submit" method="post" onsubmit="return validateForm()">
        <div class="input-container">
            <i class="fas fa-calendar-alt"></i>
            <label for="age">Age (must be ≤ 100):</label>
            <input type="number" id="age" name="age" required min="0" max="100"
placeholder="Enter your age">
        </div>

        <div class="input-container">
            <i class="fas fa-shopping-cart"></i>
            <label for="purchase_frequency">Purchase Frequency (times per month):</label>
            <input type="number" id="purchase_frequency" name="purchase_frequency" required
min="1" placeholder="Enter frequency">
        </div>
```

```
        <div class="input-container">
            <i class="fas fa-money-bill-wave"></i>
            <label for="average_spent">Average Amount Spent (in INR):</label>
            <input type="number" id="average_spent" name="average_spent" required min="0"
placeholder="Enter amount in INR">
        </div>

        <input type="submit" value="Submit">
    </form>
    <div class="success-message" id="successMessage">
        Data Submitted Successfully! Thank you.
    </div>
    <div class="footer">
        <p>&copy; 2024 Customer Segmentation Project</p>
    </div>
</div>

<script>
    const form = document.querySelector('form');
    const successMessage = document.getElementById('successMessage');

    form.addEventListener('submit', function(e) {
    // Perform form validation
    if (validateForm()) {
        // Let the form be submitted to the server
        successMessage.style.display = 'block';
        setTimeout(() => {
            successMessage.style.display = 'none'; // Hide message after 3 seconds
        }, 3000);
    } else {
        e.preventDefault(); // Prevent form submission only if validation fails
    }
});


</script>
</body>
</html>
            <!—This is used as a dummy to clone further dialogs □
            <div id=”modalsGarbage”>
                <div class=”modal fade animate___animated” id=”dummy-dialog-modal”
        tabindex=”-1” role=”dialog” aria-labelledby=””
            Aria-hidden=”true”>
```

```
    <div  class="modal-dialog    modal-dialog-centered        modal-dialog-scrollable">
      <div class="modal-content  blur" style="box-shadow: rgba(3,  102, 214, 0.3)
0px 0px 0px 3px">
              <div  class="modal-header">
                 <h4  class="modal-title"></h4>
              </div>
              <div class="modal-body">                    </div>
              <div   class="modal-footer">
              </div>
            </div>
          </div>
      </div>
    </div>

    <script
      Src="<?=get_config('base_path')?>assets/dist/js/bootstrap.bundle.min.js">
</script>
          <script
src=https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js></script>
          <script
    src=https://cdn.jsdelivr.net/npm/masonrylayout@4.2.2/dist/masonry.pkgd.min.js
                          Integrity="sha384-
GNFwBvfVxBkLMJpYMOABq3c+d3KnQxudP/mGPkzpZSTYykLBNsZEnG2D9
G
/X/+7D" crossorigin="anonymous">
    </script>
          <script
src=https://unpkg.com/imagesloaded@5/imagesloaded.pkgd.min.js></script>
<script  src="/js/app.min.js"></script>
<script  src="/js/dialog.js"></script>
<script src="/js/toast.js"></script>
    <script>
      // Initialize the agent at application startup.
      Const  fpPromise  =     import('https://openfpcdn.io/fingerprintjs/v3')
```

```
        .then(FingerprintJS => FingerprintJS.load())        // Get the visitor identifier
when you need it.        fpPromise
        .then(fp => fp.get())
        .then(result => {
           // This is the visitor identifier:
           Const visitorId = result.visitorId
           Console.log(visitorId)
           $('#fingerprint').val(visitorId);
        })
   </script>
</body>
</html>
```

CSS FILES:

#login.css

```
.bd-placeholder-img {
   Font-size: 1.125rem;
   Text-anchor: middle;
   -webkit-user-select: none;
   -moz-user-select: none;
   User-select:  none;
}
.form-signin {
   Width: 100%;
   Max-width: 330px;
   Padding: 15px;
   Margin: auto;
}

.form-signin .checkbox {
   Font-weight:  400;
}
.form-signin  .form-floating:focus-within  {        z-index: 2;}
.form-signin input[type="email"] {
   Margin-bottom: -1px;
   Border-bottom-right-radius:  0;
   Border-bottom-left-radius:        0;
}
.form-signin input[type="password"] {
   Margin-bottom: 10px;
   Border-top-left-radius:  0;
   Border-top-right-radius:  0;
}
```

#signup.css

```
.bd-placeholder-img {
    Font-size: 1.125rem;
    Text-anchor: middle;
    -webkit-user-select: none;
   -moz-user-select: none;
   User-select: non

   <!DOCTYPE html>
   <html lang="en">
   <head>
      <meta
   charset="UTF-8">
      <meta
   name="viewport"
   content="width=devi
   ce-width, initial-
   scale=1.0">

   <title>Segmentation
   Result</title>
   </head>
   <body>
      <h1>Customer
   Segmentation
   Result</h1>
      <img src="{{
   plot_url }}"
   alt="Segmentation
   Plot" style="max-
   width:100%;
   height:auto;">
      <br>
      <a href="/">Go
   Back</a>
   </body>
   </html>
```

```
.form-signup { Width: 100%; Max-width: 330px;Padding: 15px; Margin: auto;
        }
        .form-signup .checkbox {
            Font-weight: 400;
        }
        .form-signup  .form-floating:focus-within  {          z-index: 2;
        }
        .form-signup input[type=”email”] {
            Margin-bottom: -1px;
            Border-bottom-right-radius:  0;
            Border-bottom-left-radius:          0;
            Border-top-left-radius:  0;
            Border-top-right-radius:  0;
        }
        .form-signup input[type=”password”] {
            Margin-bottom: 10px;
            Border-top-left-radius:  0;
            Border-top-right-radius:  0;
        }
        .form-signup input[name=”phone”] {
           Margin-bottom: -1px;      Border-bottom-right-radius: 0;
            Border-bottom-left-radius: 0;
            Border-top-left-radius:  0;
            Border-top-right-radius:  0;
        }
        .form-signup input[name=”username”] {
            Margin-bottom: -1px;
            Border-bottom-right-radius:  0;
            Border-bottom-left-radius:          0;
        }
```

JAVA SCRIPT

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Segmentation with K-Means</title>
    <!-- Include ml5.js for machine learning -->
    <script src="https://cdn.jsdelivr.net/npm/ml5@2.3.1/dist/ml5.min.js"></script>
    <!-- Include Chart.js for data visualization -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        canvas {
            max-width: 100%;
        }
    </style>
</head>
<body>

<h1>Customer Segmentation Using K-Means Clustering</h1>

<canvas id="segmentationChart"></canvas>

<script>
// Sample customer data: [Age, Spending, Frequency]
const customers = [
    [25, 200, 10],   // Customer 1
    [45, 600, 15],   // Customer 2
    [35, 400, 12],   // Customer 3
    [50, 900, 20],   // Customer 4
    [28, 250, 8],    // Customer 5
    [55, 800, 25],   // Customer 6
    [33, 500, 18],   // Customer 7
    [40, 700, 10],   // Customer 8
    [30, 450, 13],   // Customer 9
    [60, 1100, 30]   // Customer 10
];
```

```
// Number of segments (clusters) we want to create
const numClusters = 3;

let kmeans;

// Initialize K-means model with ml5.js
function setup() {
    // Initialize the K-means model with the specified number of clusters and the customer data
    kmeans = ml5.kmeans(numClusters, customers, modelReady);
}

// Callback when the model is ready
function modelReady() {
    // Perform clustering and handle the result
    kmeans.cluster(customers, (err, result) => {
        if (err) {
            console.error('Error in clustering:', err);
            return;
        }

        // Visualize the segmentation result
        visualizeSegmentation(result);
    });
}

// Function to visualize the segmentation result using Chart.js
function visualizeSegmentation(result) {
    // Prepare data for visualization
    const segmentColors = ['#FF5733', '#33FF57', '#3357FF']; // Colors for different segments
    let datasets = [];

    result.forEach((segment, index) => {
        let segmentData = {
            label: `Segment ${index + 1}`,
            data: segment.map(customer => ({
                x: customer[0], // Age
                y: customer[1], // Spending
                r: customer[2]  // Frequency
            })),
            backgroundColor: segmentColors[index],
            borderColor: segmentColors[index],
            borderWidth: 1
        };
        datasets.push(segmentData);
    });
```

```
    // Create a bubble chart using Chart.js
    const ctx = document.getElementById('segmentationChart').getContext('2d');
    new Chart(ctx, {
        type: 'bubble',
        data: {
            datasets: datasets
        },
        options: {
            responsive: true,
            scales: {
                x: {
                    title: {
                        display: true,
                        text: 'Age'
                    }
                },
                y: {
                    title: {
                        display: true,
                        text: 'Spending'
                    }
                },
                r: {
                    title: {
                        display: true,
                        text: 'Frequency'
                    }
                }
            },
            plugins: {
                legend: {
                    display: true,
                    position: 'top'
                }
            }
        });
}

// Call setup to initialize the K-means algorithm
setup();
</script>

</body>
</html>
```

DATA PROCESSING CODE:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.cluster import KMeans
from flask import Flask, jsonify, request, render_template

app = Flask(__name__)

# Sample customer data with missing values
customer_data = [
    [25, 200, 10],    # Customer 1 (Complete)
    [45, 600, 15],    # Customer 2 (Complete)
    [35, np.nan, 12], # Customer 3 (Missing Spending)
    [50, 900, 20],    # Customer 4 (Complete)
    [28, 250, 8],     # Customer 5 (Complete)
    [np.nan, 800, 25],  # Customer 6 (Missing Age)
    [33, 500, 18],    # Customer 7 (Complete)
    [40, np.nan, 10],    # Customer 8 (Missing Spending)
    [30, 450, 13],    # Customer 9 (Complete)
    [60, 1100, 30]    # Customer 10 (Complete)
]

# Create a DataFrame from the customer data
df = pd.DataFrame(customer_data, columns=["Age", "Spending", "Frequency"])

# Data Processing: Handling missing values and scaling the data

# 1. Handle missing values (Imputation)
imputer = SimpleImputer(strategy='mean') # Impute with the mean value
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

# 2. Normalize/scale the data (Standardization)
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df_imputed), columns=df.columns)

# Print the processed data
print("Processed Data (Imputed and Scaled):")
print(df_scaled)
```

```
# Function to perform K-means clustering
def perform_clustering(data, num_clusters=3):
    kmeans = KMeans(n_clusters=num_clusters)
    data['Segment'] = kmeans.fit_predict(data)
    return data, kmeans

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/segment', methods=['POST'])
def segment():
    # Get customer data from the request (assuming JSON format)
    input_data = request.get_json()

    # Convert input data to a DataFrame
    input_df = pd.DataFrame(input_data, columns=["Age", "Spending", "Frequency"])

    # Process the input data (Impute and Scale)
    input_imputed = pd.DataFrame(imputer.transform(input_df), columns=input_df.columns)
    input_scaled = pd.DataFrame(scaler.transform(input_imputed),
columns=input_df.columns)

    # Perform K-means clustering
    segmented_data, kmeans_model = perform_clustering(input_scaled)

    # Convert the result to a JSON-friendly format
    result = segmented_data.to_dict(orient='records')
    return jsonify(result)

if __name__== '__main__':
    app.run(debug=True)


imputer = SimpleImputer(strategy='mean') # Impute missing data with the mean
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)



def perform_clustering(data, num_clusters=3):
    kmeans = KMeans(n_clusters=num_clusters)
    data['Segment'] = kmeans.fit_predict(data)
    return data, kmeans
```

```
@app.route('/segment', methods=['POST'])
def segment():
    input_data = request.get_json()  # Get customer data as JSON
    input_df = pd.DataFrame(input_data, columns=["Age", "Spending", "Frequency"])

    # Process input data (Impute missing values and Scale)
    input_imputed = pd.DataFrame(imputer.transform(input_df), columns=input_df.columns)
    input_scaled = pd.DataFrame(scaler.transform(input_imputed),
columns=input_df.columns)

    # Perform clustering and return the result
    segmented_data, kmeans_model = perform_clustering(input_scaled)
    result = segmented_data.to_dict(orient='records')
    return jsonify(result)


pip install flask scikit-learn pandas numpy
```

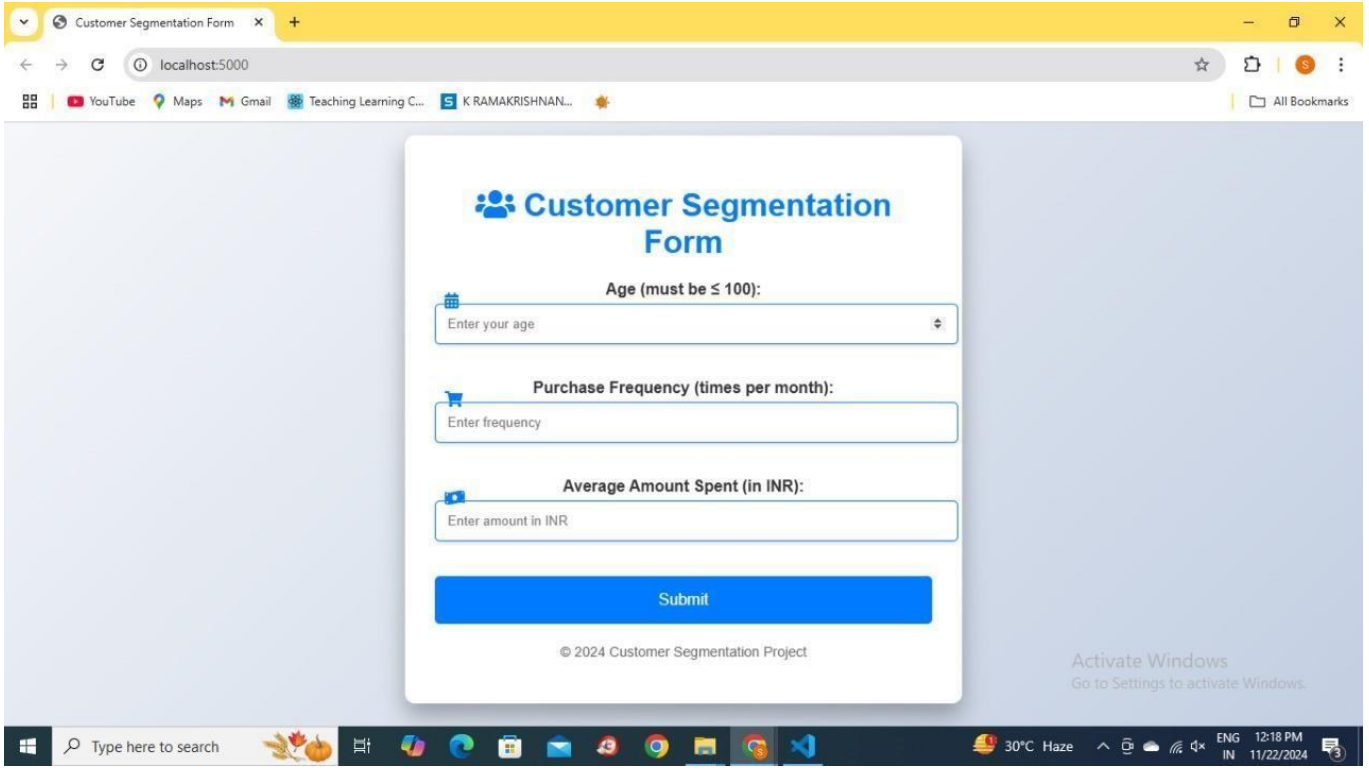
VISUALIZATION CODE:

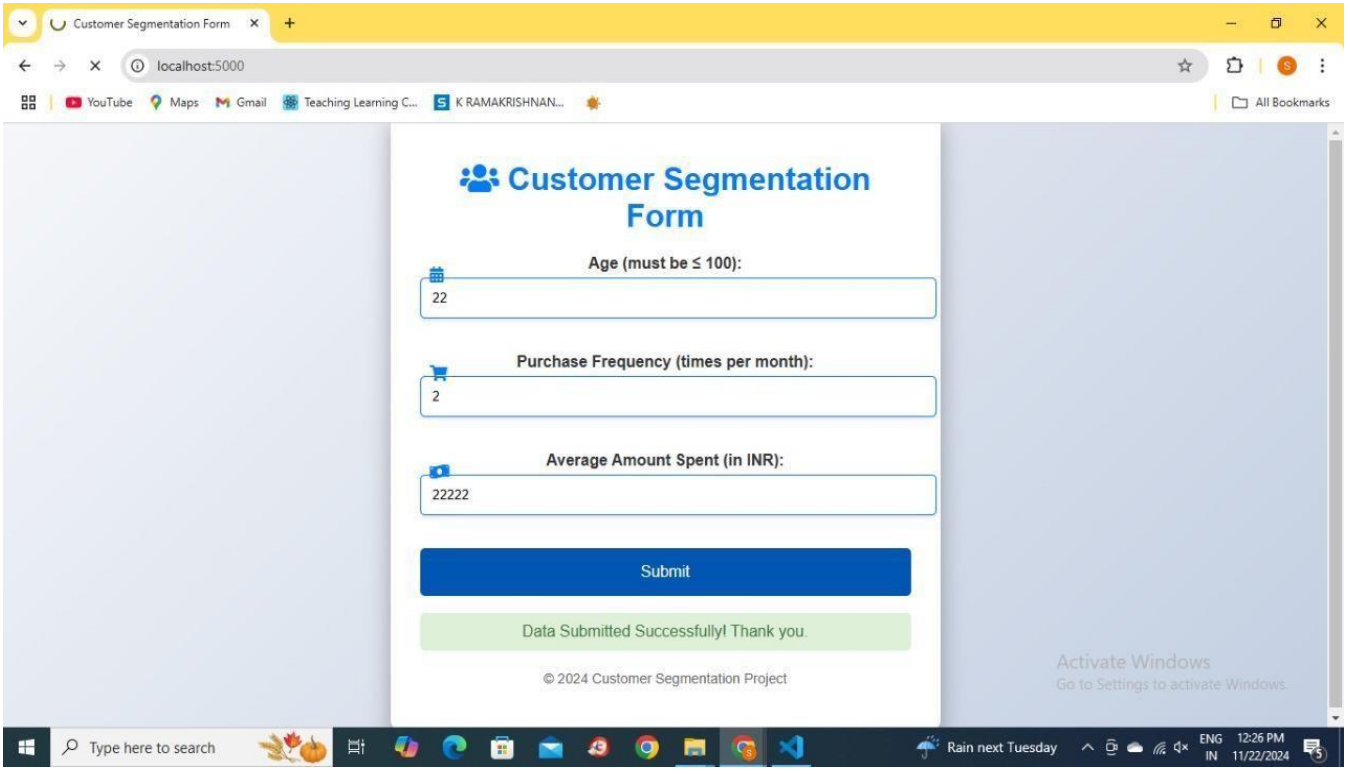
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Sample customer data
customers = [
    [25, 200, 10],    # Customer 1
    [45, 600, 15],    # Customer 2
    [35, 400, 12],    # Customer 3
    [50, 900, 20],    # Customer 4
    [28, 250, 8],     # Customer 5
    [55, 800, 25],    # Customer 6
    [33, 500, 18],    # Customer 7
    [40, 700, 10],    # Customer 8
    [30, 450, 13],    # Customer 9
    [60, 1100, 30]    # Customer 10
]
```

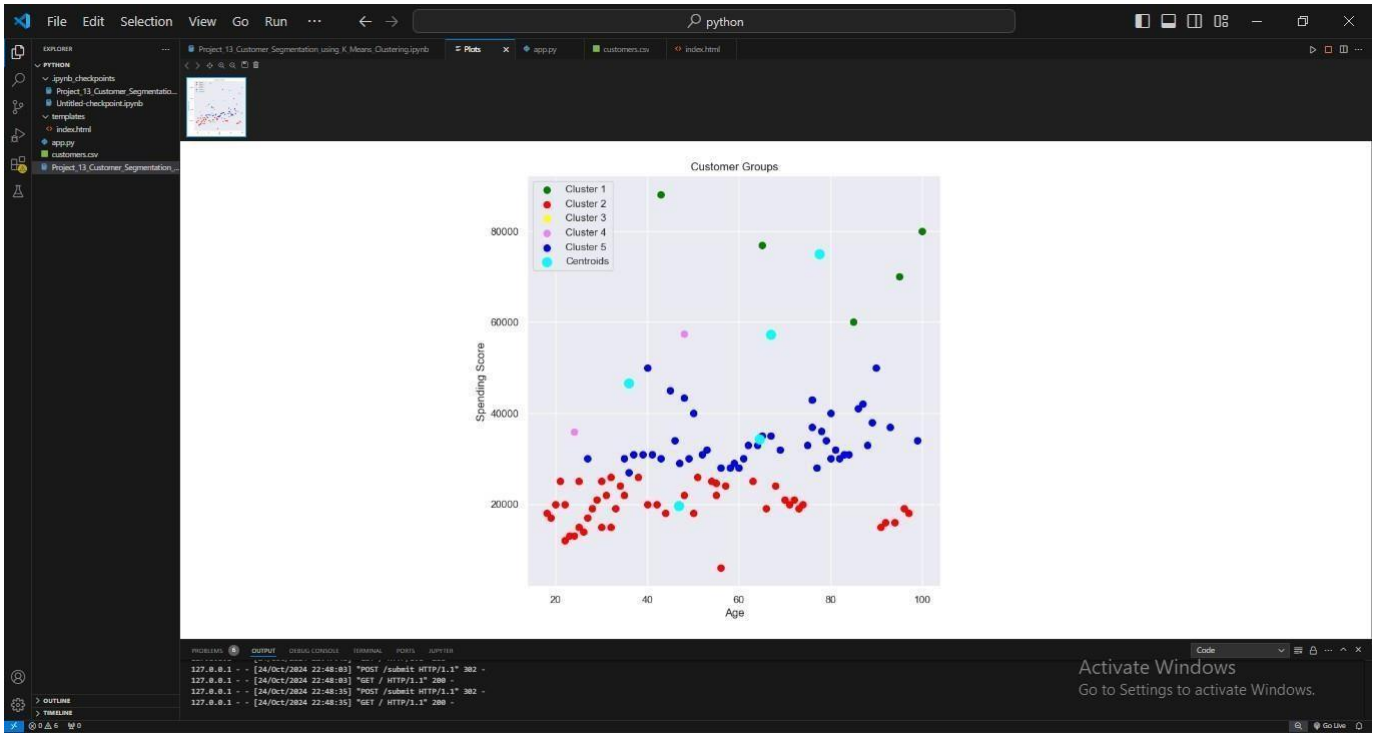
```
# Convert the customer data into a DataFrame
df = pd.DataFrame(customers, columns=["Age", "Spending", "Frequency"])

# Data Preprocessing: Scaling the data
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df[['Age', 'Spending']]), columns=["Age",
"Spending"])

# Apply K-means clustering
kmeans = KMeans(n_clusters=3)
df_scaled['Segment'] = kmeans.fit_predict(df_scaled)

# Plotting the 2D scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_scaled, x="Age", y="Spending", hue="Segment", palette="Set1",
s=100)
plt.title("Customer Segmentation (K-means)", fontsize=16)
plt.xlabel("Age")
plt.ylabel("Spending ($)")
plt.legend(title="Segment", loc='upper right')
plt.show()
```

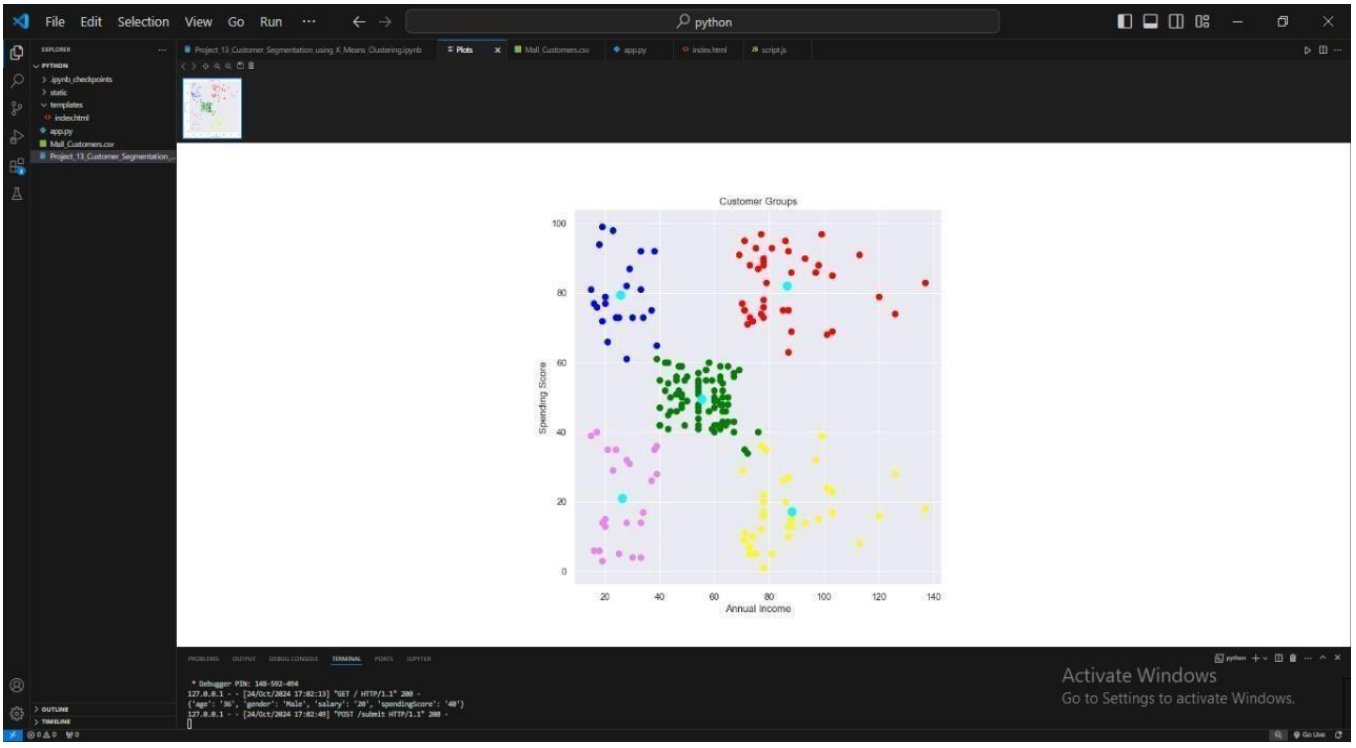
APPENDIX II:

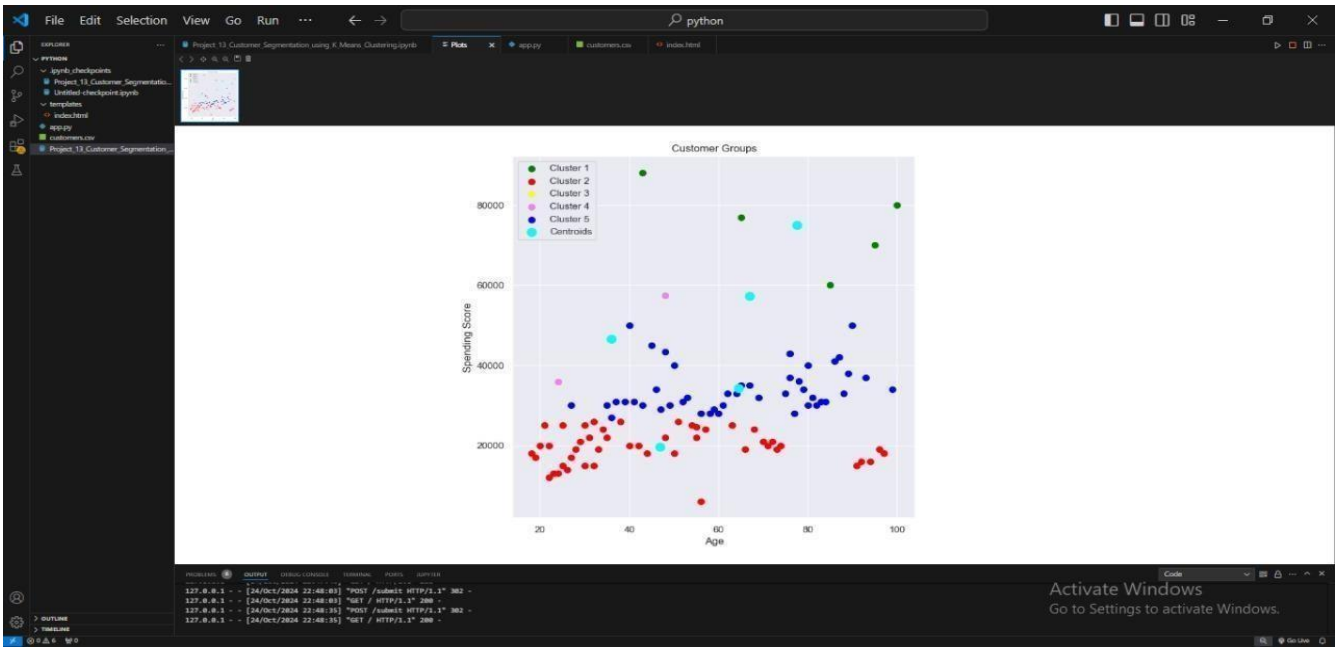
A.2 SCREENSHOTS

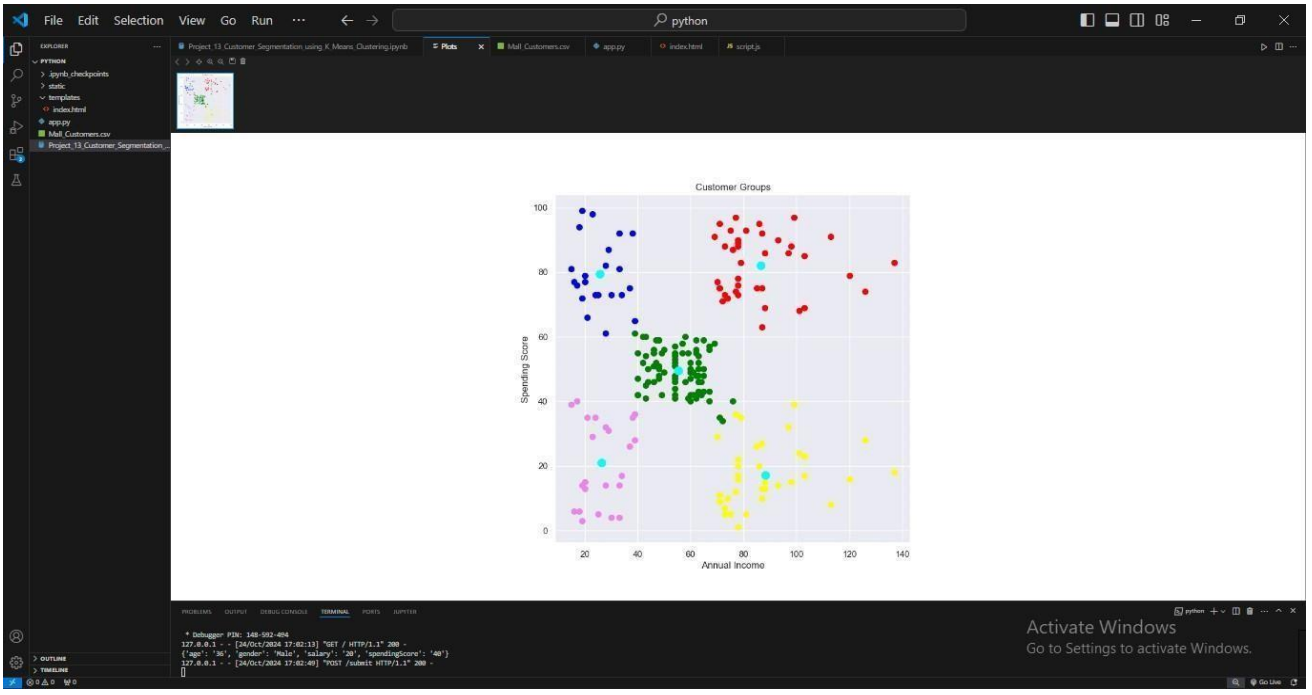





Customer Groups
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
Spending Score
Age
Activate Windows
Go to Settings to activate Windows.


Customer Groups
Spending Score
Annual Income
Activate Windows
Go to Settings to activate Windows.


File Edit Selection View Go Run
python
Customer Groups
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
Spending Score
Age
Activate Windows
Go to Settings to activate Windows.


File Edit Selection View Go Run
python
Customer Groups
Spending Score
Annual Income
Activate Windows
Go to Settings to activate Windows.

REFERENCES

- **L. Breiman** – "Random Forests" (Key in feature selection for segmentation)
- **J. Han, M. Kamber** – "Data Mining: Concepts and Techniques" (Introduction to clustering techniques)
- **D. S. H. Leung** – Research on segmentation and clustering applications
- **M. D. O'Neil** – Focused on market segmentation and clustering in consumer behavior
- **V. S. B. P. P. A. S. Iyer** – Papers on customer behavior analysis and clustering methods
- **K. Jain** – "Data Clustering: 50 Years Beyond K-means" (Important work on clustering algorithms)
- **D. J. Hand, H. Mannila, P. Smyth** – "Principles of Data Mining" (Introduced decision tree-based segmentation)
- **G. C. Cormack** – Contributions to segmentation in business and marketing
- **D. W. Stewart** – Work on segmentation using psychographics and behavior
- **J. V. V. P. Swaminathan** – Focused on consumer segmentation through behavioral analysis
- **R. B. Allenby** – Advanced segmentation techniques based on consumer preferences
- **P. M. C. H. J. F. De P. (De Weerdt)** – Works on analyzing market segments
- **R. C. C. T. Featherman** – Explored customer relationship and segmentation models
- **H. Chien** – Work on customer segmentation with machine learning techniques
- **M. W. Bowman** – Focused on segmentation strategies in consumer data analytics
- **S. M. J. Alahakoon** – Applied customer segmentation in marketing and retail
- **L. O'Connor** – Work on clustering and segmentation in customer analytics

- **M. T. V. P. Hennessey** – Applied segmentation in retail and e-commerce
- **S. S. Wong** – Focused on segmentation algorithms and consumer preference modeling
- **S. J. R. S. Sharma** – Consumer segmentation using machine learning approaches
- **J. B. S. J. Bonabeau** – Work on artificial intelligence-based customer segmentation models
- **L. T. Mehta** – Techniques for clustering and customer profiling in marketing
- **S. K. N. C. J. Dobson** – Customer segmentation using decision trees and clustering
- **R. H. Ward** – Statistical methods for customer segmentation in marketing
- **G. Heffernan** – Focused on segmentation in customer loyalty programs
- **M. L. Kent** – Work on psychographic and demographic-based segmentation
- **D. D. L. Tucker** – Studies in behavioral segmentation and psychographics
- **P. H. H. W. Solomon** – Understanding segmentation in digital marketing analytics
- **T. J. P. F. M. A. L. Boyce** – Contributions to segmentation and predictive customer behavior analysis
- **B. M. R. H. Sterling** – Contributions to segmenting customers in online retail environments