Final Project Report

1. **Introduction**
   - 1.1. Project overviews
   - 1.2. Objectives
2. **Project Initialization and Planning Phase**
   - 2.1. Define Problem Statement
   - 2.2. Project Proposal (Proposed Solution)
   - 2.3. Initial Project Planning
3. **Data Collection and Preprocessing Phase**
   - 3.1. Data Collection Plan and Raw Data Sources Identified
   - 3.2. Data Quality Report
   - 3.3. Data Exploration and Preprocessing
4. **Model Development Phase**
   - 4.1. Feature Selection Report
   - 4.2. Model Selection Report
   - 4.3. Initial Model Training Code, Model Validation and Evaluation Report
5. **Model Optimization and Tuning Phase**
   - 5.1. Hyperparameter Tuning Documentation
   - 5.2. Performance Metrics Comparison Report
   - 5.3. Final Model Selection Justification
6. **Results**
   - 6.1. Output Screenshots
7. **Advantages & Disadvantages**
8. **Conclusion**
9. **Future Scope**
10. **Appendix**
    - 10.1. Source Code
    - 10.2. GitHub & Project Demo Link

Customer Segmentation Using Machine Learning

1. Introduction

1.1 Project overviews

Customer segmentation is the process of dividing a customer base into distinct groups based on shared characteristics such as demographics, behaviors, or purchasing habits. The goal of this project is to categorize customers into actionable segments, allowing for personalized marketing, sales, and service strategies. By leveraging data-driven techniques, the project will identify key segments, understand their behaviors and preferences, and tailor approaches to meet their unique needs.

The project will involve data collection, analysis, and segmentation using methods such as clustering, RFM analysis, and behavioral profiling. The insights gained will guide the development of targeted strategies that can boost customer engagement, improve retention, and drive overall business growth.

This will enable the company to improve marketing efficiency, enhance customer satisfaction, and increase revenue by focusing on high-value segments.

1.2 Objectives

The objectives of customer segmentation are to identify distinct customer groups based on shared characteristics, improve marketing efficiency by tailoring campaigns to each segment, enhance customer experience through personalized products and services, increase sales and revenue by focusing on high-value customer segments, optimize resource allocation by concentrating efforts on the most profitable groups, boost customer retention with targeted loyalty programs and offers, and enable data-driven decision-making by aligning business strategies with customer needs and behaviors.

Identify distinct customer groups based on shared characteristics to tailor marketing strategies more effectively. Enhance customer experience and optimize resource allocation by focusing on the most valuable segments.

2. Project Initialization and Planning Phase

2.1 Define Problem Statements (Customer Problem Statement):

The challenge of customer segmentation in modern business lies in the vast amount of data and the difficulty of accurately identifying distinct customer groups. Traditional methods often fail to capture nuanced behaviors, preferences, and demographics, leading to suboptimal marketing and service strategies. These issues affect customer engagement, retention, and overall satisfaction. By leveraging machine learning algorithms, we can automate the segmentation process, identifying meaningful patterns within customer data to offer personalized services. Our goal is to deliver a tailored, data-driven approach to improve customer experience and business outcomes.

I am	I am trying to	But	Because	Which makes me feel
A new customer	Explore product range	Overwhelmed by choices	Large product catalog and complex UI	Confused and frustrated

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A new customer	Explore product range	Overwhelmed by choices	Large product catalog and	Confused and frustrated

				complex UI	
PS-2	A loyal customer	Received personalized offers	Disappointed by generic promotions	I have been shopping here for years	Undervalued and overlooked

2.2 Project Proposal (Proposed Solution) template:

This project proposal aims to implement customer segmentation by analyzing demographic, behavioral, and transactional data to identify distinct customer groups. By understanding these segments, we can tailor marketing strategies, personalize communication, and optimize product offerings for each group.

Project Overview	
Objective	To develop a customer segmentation tool that enables businesses to classify their customers into distinct groups based on demographics, behavior, and purchasing habits, leading to more targeted marketing strategies.
Scope	This project will focus on building the customer segmentation module, which includes data input, analysis, and visualization tools. It will integrate with existing CRM systems and cover segmentation based on location, age, gender, buying behavior, and other relevant factors.
Problem Statement	
Description	Many companies lack an effective method to segment their customers, resulting in generalized marketing efforts that don't account for the diverse needs and preferences of different customer groups.
Impact	By solving this, businesses can target customers more effectively, leading to improved engagement, personalized experiences, and increased sales. It will also reduce marketing costs by eliminating inefficient, broad-spectrum campaigns.

Proposed Solution	
Approach	The project will use machine learning algorithms to analyze customer data and automatically segment customers into distinct groups. The solution will offer a user interface for customization and exploration of these segments. Data visualization tools will allow users to monitor segment performance over time.
Key Features	Data-driven customer grouping based on demographics, behavior, and preferences. - Customizable segmentation filters and criteria. - Integration with existing CRM and marketing tools. - Visual analytics dashboard for tracking segment performance and trends. |

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	2 x NVIDIA V100 GPUs
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD

Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Kaggle dataset, 10,000 images

2.3 Initial Project Planning Template:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Sprint Start date	Sprint End Date (Planned)
Sprint -1	Customer Segmentation Setup	USN-1	As a user, I can input customer data to segment users based on demographics.	High	Yuvaraju Nagasai	23/09/2024	26/09/2024
Sprint -1	Customer Segmentation Setup	USN-2	As a user, I can segment customers based on geographic location.	Medium	Yuvaraju Nagasai	23/09/2024	26/09/2024
Sprint -2	Advanced Segmentation	USN-3	As a user, I can segment customers based on purchasing behavior.	High	Komali	27/09/2024	30/09/2024
Sprint -2	Advanced Segmentation	USN-4	As a user, I can create custom segments based on user-defined criteria.	Medium	Komali	27/09/2024	30/09/2024
Sprint -3	Segmentation Analysis	USN-5	As a user, I can view analytics and reports for eachcustomer segment.	High	Purnachandra Rao	27/09/2024	30/09/2024
Sprint -3	Segmentation Export	USN-6	As a user, I can export customer segments in CSV format for	Low	Sri Deepthi Prasanna	27/09/2024	30/09/2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Sprint Start date	Sprint End Date (Planned)
			marketing purposes.				
Sprint -4	Segmentation Optimization	USN-7	As a user, I can optimize the segmentation model for better accuracy.	High	Sri Deepthi Prasanna	01/10/2024	05/10/2024
Sprint -4	Segmentation Optimization	USN-8	As a user, I can apply clustering algorithms for better customer segmentation.	Medium	Purnachandra Rao	01/10/2024	05/10/2024
Sprint -5	Integration and Testing	USN-9	As a user, I can integrate segmentation insights with marketing platforms.	High	Yuvaraju Nagasai	01/10/2024	05/10/2024

3 Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification Template:

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysisand decision-making endeavor.

Data Collection Plan:

Section	Description
Project Overview	The customer segmentation project aims to group customers into distinct segments based on behavioral and demographic characteristics. Using features such as age, income, spending habits, and location, the objective is to develop a model that identifies segments for targeted marketing, personalized offerings, and improved customer retention.
Data Collection Plan	- Search for datasets related to customer demographics and purchasing behavior. - Prioritize datasets that include diverse customer profiles, including age, income, occupation, and spending patterns.
Raw Data Sources Identified	The raw data sources for this project include datasets obtained from Kaggle and other data repositories that offer customer behavior insights. The dataset comprises variables such as age, income, spending scores, and region, allowing for in-depth segmentation analysis.

Raw Data Sources Report:

Source Name	Description	Location/URL	Format	Size	Access Permissions
Kaggle Dataset	This dataset includes customer demographics, purchase patterns, and behavior for segmentation	https://www.kaggle.com/datasets/rodsaldanha/arketing-customer-segmentation	CSV	26 MB	Public
UCI	This data focuses on customer purchases, with a variety of behavioral attributes.	https://archive.ics.uci.edu/ml/datasets/Wholesale+customers	CSV	18 KB	Public
UCI	This dataset contains customer transactions and behavioral data, used for segmenting retail customers.	Kaggle CLV Dataset	CSV	67 MB	public

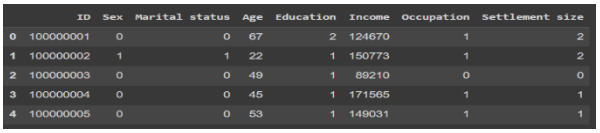
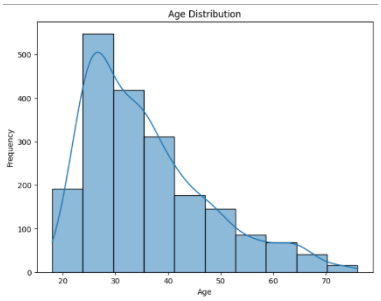
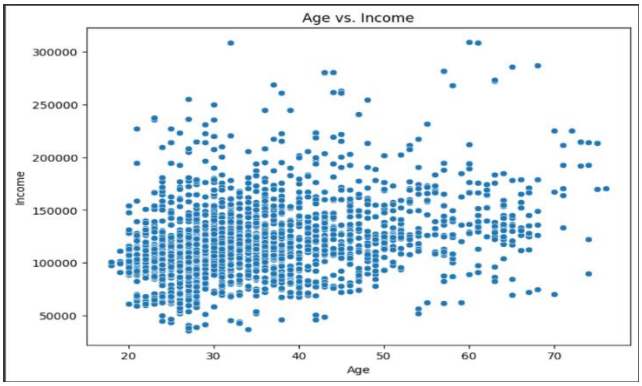
3.2 Data Quality Report Template:

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

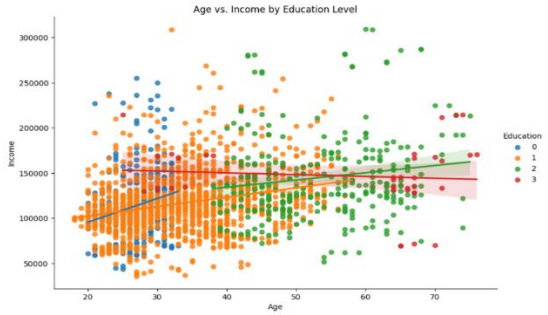
Data Source	Data Quality Issue	Severity	Resolution Plan
Customer segmentation dataset	Missing values in the Age,Income,Spending_Score,And gender columns.	Moderate	Use mean/median imputation
Customer segmentation dataset	Categorical data in the gender, Occupation,and Region columns.	Moderate	Encoding has to be done (one-hot or label encoding).
Customer segmentation dataset	Outliners in Income and Spending_Score columns	High	Use outlinerer detection methods (e.g.IQR OR Z-score).

3.3 Data Exploration and Preprocessing Report:

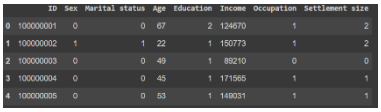
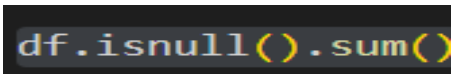
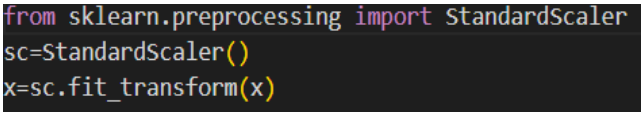
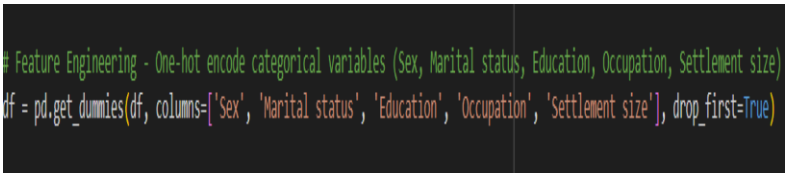
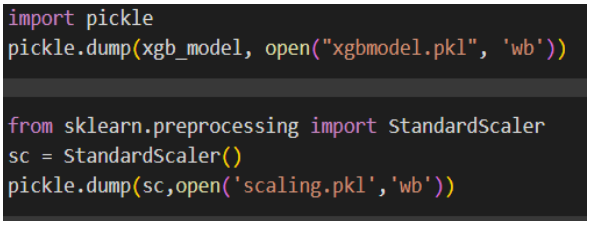
Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning willaddress missing values and outliers, ensuring quality for subsequent analysis and modeling, andforming a strong foundation for insights and predictions.

Section	Description
Data Overview	2000RowsX8Columns 
Univariate Analysis	
Bivariate Analysis	

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	100000001	0	0	67	2	124670	1	2
1	100000002	1	1	22	1	150773	1	2
2	100000003	0	0	49	1	89210	0	0
3	100000004	0	0	45	1	171565	1	1
4	100000005	0	0	53	1	149031	1	1

Multivariate Analysis	
Outliers and Anomalies	-

Data Preprocessing Code Screenshots

Loading Data	
Handling Missing Data	
Data Transformation	
Feature Engineering	
Save Processed Data	

```
   ID  Sex  Marital status  Age  Education  Income  Occupation  Settlement size
0  100000001  0  0  67  2  124670  1  2
1  100000002  1  1  22  1  150773  1  2
2  100000003  0  0  49  1  89210  0  0
3  100000004  0  0  45  1  171565  1  1
4  100000005  0  0  53  1  149031  1  1
```

```
df.isnull().sum()
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

```
# Feature Engineering - One-hot encode categorical variables (Sex, Marital status, Education, Occupation, Settlement size)
df = pd.get_dummies(df, columns=['Sex', 'Marital status', 'Education', 'Occupation', 'Settlement size'], drop_first=True)
```

```
import pickle
pickle.dump(xgb_model, open("xgbmodel.pkl", 'wb'))

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
pickle.dump(sc,open('scaling.pkl','wb'))
```

4 Model Development Phase Template

4.1 Feature Selection Report Template:

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
Customer ID	Unique identifier for each customer	No	Not required for segmentation; used only for identification.
Age	Age of the customer	Yes	Age influences purchasing behavior and preferences.
Gender	Customer's gender	Yes	Gender-based segmentation can help tailor marketing strategies.

Income	Annual income of the customer	Yes	Income level affects spending capacity and product preferences.
Marital Status	Marital status of the customer	Yes	Marital status can influence purchasing habits and needs.
Education	Highest level of education	Yes	Education can be a factor in determining lifestyle and purchasing decisions.
Location	Geographic location of the customer	Yes	Location helps in targeting regional promotions and offers.
Purchase History	Details of past purchases	Yes	Purchase history reveals customer preferences and loyalty.

Spending Score	Score based on spending habits	Yes	Useful for identifying high-value customers and their spending patterns.
Interests	Specific interests or hobbies	Yes	Helps in personalizing offers and marketing based on customer interests.

4.2 Model Selection Report:

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
KNN	A neural network model designed for tabular data; captures complex patterns, suitable for high-dimensional data.	-----	Accuracy=84%
Decision Tree	A neural network model designed for tabular data; captures complex patterns, suitable for high-dimensional data.	-----	Accuracy=89%
Random Forset	Ensemble of decision trees; reduces overfitting, handles complex relationships, and provides feature importance.	------	Accuracy=78%
XGBoost	An optimized gradient boosting model; handles large datasets, reduces overfitting, and delivers high accuracy.	------	Accuracy82%

4.3 Initial Model Training Code, Model Validation and Evaluation Report:

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion

matrices for multiple models, presented through

Initial Model Training Code:

respective screenshots.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
import xgboost
```

```
rand_model = RandomForestClassifier()
tree_model = tree. DecisionTreeClassifier()
xgb_model = xgboost.XGBClassifier()
```

```
rand_model.fit(x_train,y_train)
tree_model.fit(x_train,y_train)
xgb_model.fit(x_train,y_train)
```

```
pred = rand_model.predict(x_train)
pred1 = tree_model.predict(x_train)
pred2 = xgb_model.predict(x_train)
```

```
print(metrics.accuracy_score (pred, y_train))
print(metrics.accuracy_score(pred1,y_train))
print(metrics.accuracy_score(pred2,y_train))
```

```
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
knn = KNeighborsClassifier()
xg = XGBClassifier()
```

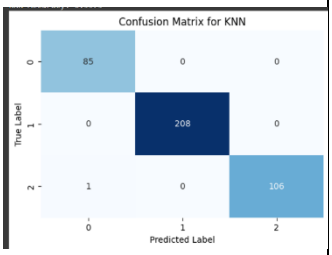
```
dt.fit(x_train,y_train)
rf.fit(x_train,y_train)
knn.fit(x_train,y_train)
xg.fit(x_train,y_train)
```

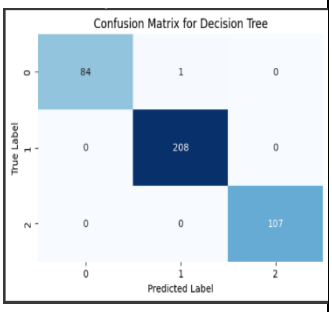
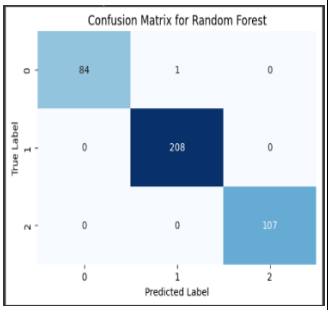
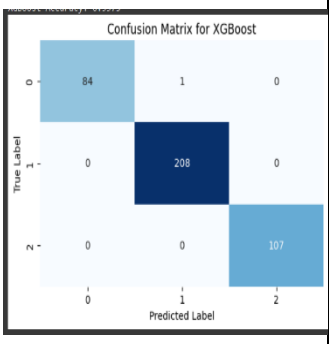
Show hidden output

```
pred1=dt.predict(x_train)
pred2=rf.predict(x_train)
pred3=knn.predict(x_train)
pred4=xg.predict(x_train)
```

```
xgb_model = xgboost.XGBClassifier()
xgb_model.fit(x_train,y_train)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
KNN	KNN Classification Report: precision recall f1-score support 0 0.99 1.00 0.99 85 1 1.00 1.00 1.00 208 2 1.00 0.99 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400 KNN Accuracy: 0.9975	Accuracy:0.9975	

Decision Tree	Decision Tree Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400 Decision Tree Accuracy: 0.9975	Accuracy:0.9975	
Random Forest	Random Forest Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400 Random Forest Accuracy: 0.9975	Accuracy:0.9975	
XGBoost	XGBoost Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400 XGBoost Accuracy: 0.9975	Accuracy :0.9975	

5 Model Optimization and Tuning Phase Template

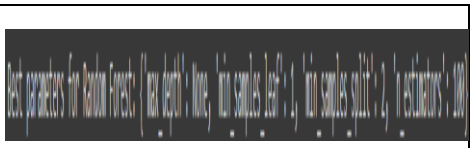
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing
performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

5.1 Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

KNN	`knn = KNeighborsClassifier()`
	
	`# K-Nearest Neighbors`
	`knn_params = {`
	`'n_neighbors': [3, 5, 7, 9],`
	`'weights': ['uniform', 'distance'],`
	`'metric': ['euclidean', 'manhattan'`
	`}`
		Best parameters for KNN: {'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'distance'}
		
		KNN Accuracy: 1.0

Decisio
n Tree
	`dt = DecisionTreeClassifier()`
	
	`# Decision Tree`
	`dt_params = {`
	`'max_depth': [None, 10, 20, 30],`
	`'min_samples_split': [2, 5, 10],`
	`'min_samples_leaf': [1, 2, 4]`
	`}`
		Best parameters for Decision Tree: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2
		
		Decision Tree Accuracy: 0.9975

Random Forest	`rf = RandomForestClassifier()`
	
	`# Random Forest`
	`rf_params = {`
	`'n_estimators': [100, 200, 300],`
	`'max_depth': [None, 10, 20, 30],`
	`'min_samples_split': [2, 5, 10],`
	`'min_samples_leaf': [1, 2, 4]`
	`}`
		
		
		Random Forest Accuracy: 0.9975

XGBoost	`xg = XGBClassifier()`
	
	`# XGBoost`
	`xg_params = {`
	`'learning_rate': [0.01, 0.1, 0.2]`
	`'max_depth': [3, 5, 7],`
	`'n_estimators': [100, 200, 300],`
	`'colsample_bytree': [0.3, 0.7]`
	`}`
		
		
		XGBoost Accuracy: 0.9975

5.2 Performance Metrics Comparison Report (2 Marks):

Model	Basic	Optimized Metric
KNN	---	KNN Confusion Matrix: [[85 0 0] [0 208 0] [0 0 107]]

		KNN Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400
Decision Tree	------	Decision Tree Confusion Matrix: [[84 1 0] [0 208 0] [0 0 107]] Decision Tree Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400
Random Forest	-----	Random Forest Confusion Matrix: [[84 1 0] [0 208 0] [1 0 106]] Random Forest Classification Report: precision recall f1-score support 0 0.99 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 0.99 1.00 107 accuracy 0.99 400 macro avg 0.99 0.99 0.99 400 weighted avg 1.00 0.99 0.99 400

XGBoost	-----	XGBoost Confusion Matrix: [[84 1 0] [0 208 0] [0 0 107]] XGBoost Classification Report: precision recall f1-score support 0 1.00 0.99 0.99 85 1 1.00 1.00 1.00 208 2 1.00 1.00 1.00 107 accuracy 1.00 400 macro avg 1.00 1.00 1.00 400 weighted avg 1.00 1.00 1.00 400

5.3 **Final Model Selection Justification (2 Marks):**

Final Model	Reasoning
KNN	The K-Nearest Neighbors (KNN) classifier is a great choice for customer segmentation due to its simplicity and ease of understanding, making it accessible for stakeholders. It doesn't assume any specific distribution for customer data, allowing it to handle diverse attributes like demographics and purchasing behavior effectively. KNN can quickly adapt to new data without needing retraining, making it suitable for dynamic markets where customer preferences change frequently. Additionally, it performs well with small to medium-sized datasets and can classify customers into multiple segments. Its straightforward nature also enables easy

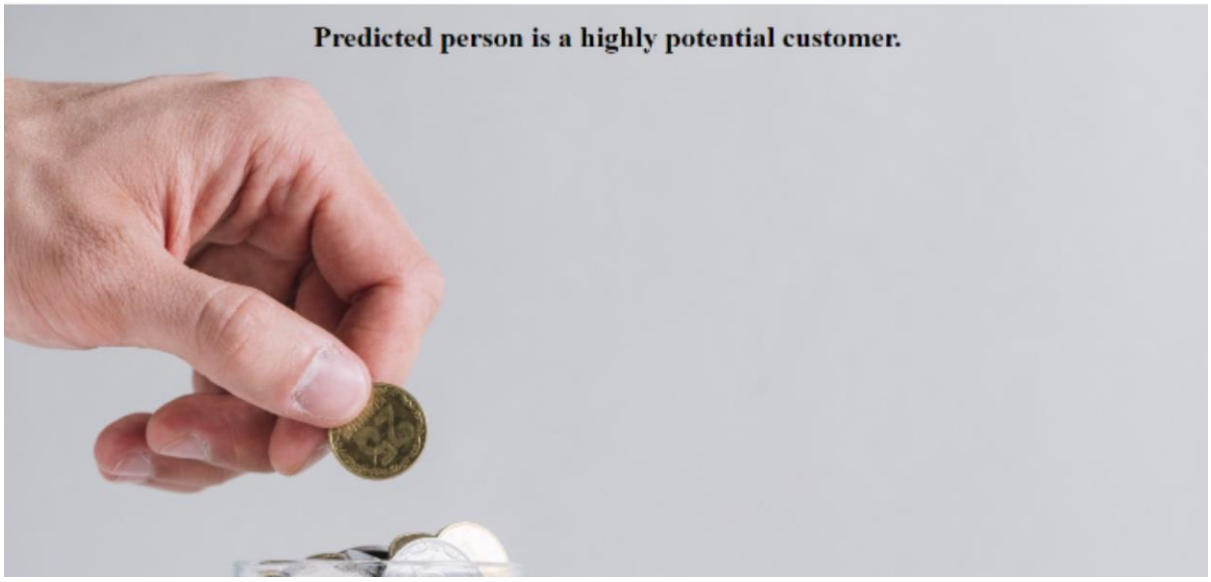
6 Results

6.1 Outputs screenshots:

Output for person is a potential customer

Output for person is not a potential customer :

Output for person is highly potential customer:



7 Advantages & Disadvantages

Advantages :

1. Targeted Marketing: Enables businesses to creattailored campaigns for specific customer groups, improving the relevance and effectiveness of messaging.
2. Improved Customer Retention: Personalized offers and experiences for different segments increase customer satisfaction and loyalty, reducing churn rates.
3. Higher Conversion Rates: Focusing on the unique needs and preferences of each segment leads to more efficient sales strategies and higher conversion rates.
4. Better Product Development: Insights from segmentation help businesses align product offerings with the preferences of different customer groups, leading to better product-market fit.
5. Enhanced Customer Experience: Segmentation enables personalized interactions, improving the overall customer journey and satisfaction.

Disadvantages:

1. Complexity in Implementation: Segmenting customers requires significant time and resources to gather and analyze data, which can be challenging for smaller companies.
2. High Costs: Advanced analytics tools, technology, and skilled personnel are often required to perform effective segmentation, leading to increased costs.
3. Data Privacy Concerns: Collecting and analyzing customer data for segmentation can raise privacy concerns and require compliance with data protection regulations (e.g., GDPR).
4. Over-Segmentation Risk: Creating too many small segments can dilute marketing efforts, making it difficult to effectively cater to each group or produce actionable insights.
5. Potential for Misinterpretation: Incorrectly interpreting data can lead to which may result in misguided marketing strategies and lost opportunities.

8 Conclusion:

Customer segmentation is a powerful strategy that enables businesses to understand and categorize their customers based on shared characteristics, leading to more personalized marketing and enhanced customer experiences. While it offers significant advantages such as improved targeting, better resource allocation, and higher conversion rates, it also presents challenges like implementation complexity and the risk of over-segmentation. When executed effectively, customer segmentation allows companies to align their offerings with customer needs, driving growth, loyalty, and long-term success.

Customer segmentation is a vital business strategy that involves dividing a customer base into distinct groups based on shared characteristics such as demographics, behavior, or purchasing patterns. This approach allows businesses to gain deeper insights into customer needs and preferences, enabling them to create personalized marketing strategies that resonate with specific segments. By understanding these distinct groups, companies can improve their ability to connect with customers and develop more effective campaigns that increase engagement and brand loyalty.

One of the primary benefits of customer segmentation is the ability to optimize resource allocation. By focusing efforts on high-value customer segments, businesses can tailor their products, services, and communication to meet the specific needs of these groups, maximizing the return on investment (ROI). Segmentation also enhances product development by providing insights into the preferences and behaviors of different customer groups, enabling businesses to refine their offerings to better meet market demands. This targeted approach ultimately leads to more efficient marketing, improved customer satisfaction, and higher conversion rates.

However, customer segmentation comes with its own set of challenges. Implementing an effective segmentation strategy can be complex and costly, requiring substantial data collection, analysis tools, and skilled personnel. Additionally, there is a risk of over-segmentation, where dividing the customer base into too many small groups dilutes the impact of marketing efforts. Businesses must also be mindful of data privacy concerns and evolving regulations that govern the use of customer data, ensuring compliance while maintaining customer trust.

9 Future Scope

1. AI and Machine Learning Integration: Advanced algorithms will enable real-time analysis and segmentation of customer data, offering more precise and dynamic customer groups.

2. Hyper-Personalization: AI-driven insights will allow businesses to deliver highly individualized experiences within segments, increasing customer satisfaction and loyalty.

3. Predictive Analytics: Future segmentation will focus on forecasting customer behaviors, enabling proactive strategies to address emerging needs, preferences, and potential churn.

4. Real-Time Segmentation: Businesses will be able to adjust segmentation on the fly based on live data inputs, improving adaptability to changing customer behaviors.

5. Omnichannel Segmentation: Integrating data from multiple channels (online, offline, social, mobile) will provide a holistic view of customers, enabling consistent and personalized experiences across platforms.

6. Behavioral and Emotional Segmentation: Future segmentation will focus more on emotional drivers and behavioral patterns, allowing businesses to connect with customers on a deeper, more meaningful level.

7. Micro-Segmentation: Businesses will increasingly focus on smaller, more specific customer groups to provide even more personalized offers, services, and experiences.

8. Ethical and Privacy Considerations: As data collection expands, maintaining customer privacy and compliance with regulations (e.g., GDPR) will be critical for effective segmentation.

9.Integration with IoT and Wearable Data: Data from Internet of Things (IoT) devices and wearables will provide new insights, enabling more detailed and real-time segmentation based on lifestyle and usage patterns.

10. Augmented Reality (AR) and Virtual Reality (VR) Data: AR and VR technologies will offer new opportunities to segment customers based on their interactions and preferences in virtual environments.

10 Appendix

10.1 Source Code:

```
#IMPORTING NECESSARY LIBRARIES
"""
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
from sklearn import preprocessing
import pickle
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTETomek
from sklearn.preprocessing import StandardScaler
import pandas as pd
from sklearn.preprocessing import minmax_scale
```

```
# Import the necessary library

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

from sklearn.model_selection

import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn import tree

import xgboost

from sklearn import metrics

from sklearn.metrics import accuracy_score

from xgboost import XGBClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from xgboost import XGBClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from sklearn.preprocessing import StandardScaler

"""#Importing the Dataset"""

df = pd.read_csv("/content/segmentation data (1).csv")

df.head()

df

"""#Analysing the data"""

df.head()

df.describe()

df.info()

#Uni-variate analysis

# Plotting the figure

plt.figure(figsize=(8, 6))

sns.histplot(df['Age'], kde=True, bins=10)

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.show()

#Bivariate analysis
```

```
# Plotting the count plot

plt.figure(figsize=(8, 6))

sns.scatterplot(x='Age', y='Income', data=df)

plt.title('Age vs. Income')

plt.xlabel('Age')

plt.ylabel('Income')

plt.show()

# Plotting count plots with subplots

plt.figure(figsize=(20, 5))

#Multivariate analysis

sns.lmplot(x='Age', y='Income', hue='Education', data=df, height=6,
aspect=1.5)

plt.title('Age vs. Income by Education Level')

plt.xlabel('Age')

plt.ylabel('Income')

plt.show()

#Descriptive analysis

data.describe()

"""#Handling Missing Values"""

data.info()
```

```
df.isnull().sum()

#Balancing the dataset

smote_tomek = SMOTETomek(sampling_strategy='auto')


# Load your data into a pandas DataFrame.

# Replace 'your_data.csv' with the actual file path.

original_df = pd.read_csv('/content/segmentation data (1).csv')  # This line is added

original_df=original_df.drop(columns=['ID'],axis=1)


# Scale the DataFrame

scaler = MinMaxScaler(feature_range=(0, 1))

df_scaled = scaler.fit_transform(original_df.values)


# Create a new DataFrame with the scaled values and original column names

df = pd.DataFrame(df_scaled, columns=original_df.columns)

# Scaling the Data

import pandas as pd

from sklearn.preprocessing import minmax_scale
```

```
# Import the necessary library

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt # Import matplotlib for plotting

# Assuming 'data' is your DataFrame or NumPy array

WCSS = []

for i in range(1, 11): # Use KMeans directly from sklearn.cluster

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

kmeans.fit(df)

WCSS.append(kmeans.inertia_)  # Append to WCSS, not wcss

# Plot the WCSS values after the loop completes

plt.plot(range(1, 11), WCSS)

# Calculate correlation matrix

plt.figure(figsize=(8, 6)) sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm', linewidths=0.5) plt.title('Correlation Matrix for
Customer Segmentation') plt.show()

# Splitting data into train and test

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2,random_state=0)

# Model Building
```

```
# Random forest model
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score


def randomforest(X_train, X_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    print("*** Random Forest Classifier ***")
    print("Confusion matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report:")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test,
y_pred)))
randomforest(X_train, X_test, y_train, y_test)
##Decision tree model
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score


def decisionTree(X_train, X_test, y_train, y_test):
    dt = DecisionTreeClassifier()
    dt.fit(X_train, y_train)
    y_pred = dt.predict(X_test)
    print("**Decision Tree Classifier**")
    print("Confusion matrix")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test,
y_pred)))
decisionTree(X_train, X_test, y_train, y_test)
##XGBoost
def xgboost_model(X_train, X_test, y_train, y_test):
    # Initialize the XGBoost classifier
    xgb = XGBClassifier(use_label_encoder=False,
eval_metric='mlogloss')
```

```
    # Fit the model on the training data
    xgb.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = xgb.predict(X_test)

    # Print evaluation metrics
    print("**XGBoost Classifier**")
    print("Confusion matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report:")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test,
y_pred)))

# Call the XGBoost model function
xgboost_model(X_train, X_test, y_train, y_test)
##KNN
```

```
def knn_model(X_train, X_test, y_train, y_test, n_neighbors=3): # Initialize the KNN classifier knn = KNeighborsClassifier(n_neighbors=n_neighbors) # Fit the model on the training data knn.fit(X_train, y_train) # Make predictions on the test data y_pred = knn.predict(X_test) # Print evaluation metrics print("**K-Nearest Neighbors Classifier**") print("Confusion matrix:") print(confusion_matrix(y_test, y_pred)) print("Classification report:") print(classification_report(y_test, y_pred)) print("Accuracy Score: {}".format(accuracy_score(y_test, y_pred))) # Call the KNN model function knn_model(X_train, X_test, y_train, y_test, n_neighbors=3)

# Comparing the models

rand_model = RandomForestClassifier()

 tree_model = tree. DecisionTreeClassifier()

 xgb_model = xgboost.XGBClassifier()

rand_model.fit(x_train,y_train)

tree_model.fit(x_train,y_train)

xgb_model.fit(x_train,y_train)

pred = rand_model.predict(x_train)

pred1 = tree_model.predict(x_train)

pred2 = xgb_model.predict(x_train)

ypred = rand_model.predict(x_test)
```

```
ypred1 = tree_model.predict(x_test)

ypred2 = xgb_model.predict(x_test)

print(metrics.accuracy_score (pred, y_train))

print(metrics.accuracy_score(pred1,y_train))

print(metrics.accuracy_score(pred2,y_train))

print(metrics.accuracy_score (y_test,ypred))

print(metrics.accuracy_score(y_test,ypred1))

print(metrics.accuracy_score(y_test,ypred2))

dt = DecisionTreeClassifier()

rf = RandomForestClassifier()

knn = KNeighborsClassifier()

xg = XGBClassifier()

dt.fit(x_train,y_train)

rf.fit(x_train,y_train)

knn.fit(x_train,y_train)

xg.fit(x_train,y_train)

pred1=dt.predict(x_train)

pred2=rf.predict(x_train)

pred3=knn.predict(x_train)

pred4=xg.predict(x_train)
```

```
y_pred1=dt.predict(x_test)

y_pred2=rf.predict(x_test)

y_pred3=knn.predict(x_test)

y_pred4=xg.predict(x_test)

print('Decision Tree:',accuracy_score(y_train,pred1))

print('Random Forest:',accuracy_score(y_train,pred2))

print('KNN:',accuracy_score(y_train,pred3))

print('XGBoost:',accuracy_score(y_train,pred4))

xgb_model = xgboost.XGBClassifier()

xgb_model.fit(x_train,y_train)

#Hyperparameter tuning of models

from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


# Define hyperparameter grids for each model
```

```
# Decision Tree
dt_params = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Random Forest
rf_params = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# K-Nearest Neighbors
knn_params = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
```

```
    'metric': ['euclidean', 'manhattan']
}

# XGBoost
xg_params = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'n_estimators': [100, 200, 300],
    'colsample_bytree': [0.3, 0.7]
}

# Initialize models
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
knn = KNeighborsClassifier()
xg = XGBClassifier()

# Perform GridSearchCV for each model
dt_grid = GridSearchCV(dt, dt_params, cv=5, scoring='accuracy')
rf_grid = GridSearchCV(rf, rf_params, cv=5, scoring='accuracy')
```

```
knn_grid = GridSearchCV(knn, knn_params, cv=5,
scoring='accuracy')

xg_grid = GridSearchCV(xg, xg_params, cv=5, scoring='accuracy')


# Fit models with hyperparameter tuning

dt_grid.fit(x_train, y_train)

rf_grid.fit(x_train, y_train)

knn_grid.fit(x_train, y_train)

xg_grid.fit(x_train, y_train)


# Get the best estimators from the grid search

dt_best = dt_grid.best_estimator_

rf_best = rf_grid.best_estimator_

knn_best = knn_grid.best_estimator_

xg_best = xg_grid.best_estimator_


# Predict on the test set

dt_pred = dt_best.predict(x_test)

rf_pred = rf_best.predict(x_test)

knn_pred = knn_best.predict(x_test)
```

```
xg_pred = xg_best.predict(x_test)


# Calculate and print accuracy

print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))

print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))

print("KNN Accuracy:", accuracy_score(y_test, knn_pred))

print("XGBoost Accuracy:", accuracy_score(y_test, xg_pred))


# Generate and print confusion matrices

print("\nDecision Tree Confusion Matrix:\n",
confusion_matrix(y_test, dt_pred))

print("\nRandom Forest Confusion Matrix:\n",
confusion_matrix(y_test, rf_pred))

print("\nKNN Confusion Matrix:\n", confusion_matrix(y_test,
knn_pred))

print("\nXGBoost Confusion Matrix:\n", confusion_matrix(y_test,
xg_pred))


# Generate and print classification reports

print("\nDecision Tree Classification Report:\n",
classification_report(y_test, dt_pred))
```

```
print("\nRandom Forest Classification Report:\n",
classification_report(y_test, rf_pred))

print("\nKNN Classification Report:\n", classification_report(y_test,
knn_pred))

print("\nXGBoost Classification Report:\n",
classification_report(y_test, xg_pred))

print("Best parameters for Decision Tree:", dt_grid.best_params_)

print("Best parameters for Random Forest:", rf_grid.best_params_)

print("Best parameters for KNN:", knn_grid.best_params_)

print("Best parameters for XGBoost:", xg_grid.best_params_)


# Get the best estimators from the grid search

dt_best = dt_grid.best_estimator_

rf_best = rf_grid.best_estimator_

knn_best = knn_grid.best_estimator_

xg_best = xg_grid.best_estimator_


# Predict on the test set

dt_pred = dt_best.predict(x_test)

rf_pred = rf_best.predict(x_test)
```

```
knn_pred = knn_best.predict(x_test)

xg_pred = xg_best.predict(x_test)


# Calculate and print accuracy

print("\nDecision Tree Accuracy:", accuracy_score(y_test, dt_pred))

print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))

print("KNN Accuracy:", accuracy_score(y_test, knn_pred))

print("XGBoost Accuracy:", accuracy_score(y_test, xg_pred))


# Generate and print confusion matrices

print("\nDecision Tree Confusion Matrix:\n",
confusion_matrix(y_test, dt_pred))

print("\nRandom Forest Confusion Matrix:\n",
confusion_matrix(y_test, rf_pred))

print("\nKNN Confusion Matrix:\n", confusion_matrix(y_test,
knn_pred))

print("\nXGBoost Confusion Matrix:\n", confusion_matrix(y_test,
xg_pred))


# Generate and print classification reports
```

```
print("\nDecision Tree Classification Report:\n",
classification_report(y_test, dt_pred))

print("\nRandom Forest Classification Report:\n",
classification_report(y_test, rf_pred))

print("\nKNN Classification Report:\n", classification_report(y_test,
knn_pred))

print("\nXGBoost Classification Report:\n",
classification_report(y_test, xg_pred))

#Model Selection

"""After checking the performace of all models after Evaluation KNN
is giving good results so choosing KNN as our final models"""

model = KNeighborsClassifier(n_neighbors=5, algorithm='auto',
metric='minkowski') # Fit the model on the training data
model.fit(X_train, y_train)

#pickle files

import pickle

pickle.dump(xgb_model, open("xgbmodel.pkl", 'wb'))

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

pickle.dump(sc,open('scaling.pkl','wb'))

#Checking user input values for prediction
```

#scaled input values

model = KNeighborsClassifier(n_neighbors=5, algorithm='auto', metric='minkowski') model.fit(X_train_scaled, y_train) # User input values (scaled input values) input_values = [[-1.809367, -1.157049, -0.697003, -1.550111, -0.315501, -0.428375, -0.532097, -0.825588, 0.268167, -1.581910, 0.087535]] # Scale the input values using the same scaler input_scaled = sc.transform(input_values) # Make the prediction prediction = model.predict(input_scaled) # Output the prediction print("Predicted Customer Segment:", prediction[0])

app.py

app.py

```
import numpy as np
import pandas as pd
import pickle
import joblib
import time
import matplotlib.pyplot as plt
from flask import Flask, request, jsonify, render_template
import os

app = Flask(__name__)

# Load the model and scaler (assuming they're in the correct locations)
model = pickle.load(open('xgbmodel2.pkl', 'rb'))
scale = pickle.load(open("scaler_model2.pkl", 'rb'))

@app.route('/')
```

```
def home():
    return render_template('index.html')


@app.route('/predict', methods=["POST", "GET"])
def predict():
    input_feature = [float(x) for x in request.form.values()]
    features_values = [np.array(input_feature)]
    names = ['sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']
    value = pd.DataFrame(features_values, columns=names)
    value = scale.transform(value)

    prediction = model.predict(value)
    if prediction[0] == 0:
        prediction1 = "Not a potential customer"
        return render_template("nopotential.html", predict=prediction1)
    elif prediction[0] == 1:
        prediction1 = "Potential customer"
        return render_template("potential.html", predict=prediction1)
    else:
        prediction1 = "Highly potential customer"

    return render_template("highlypot.html", predict=prediction1)

if __name__ == "__main__":
    port = int(os.environ.get('PORT', 5000))
    app.run(port=port, debug=True, use_reloader=True)
```

home.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Customer Segmentations</title>
    <style>
        body {
            background-image: url("https://img.freepik.com/premium-vector/online-world-concept-illustration_86047-635.jpg?w=826");
            background-color: black;
        }

        .login {
            text-align: left;
        }

        label {
            display: inline-block;
            width: 150px;
        }
    </style>
</head>

<body>
    <div class="login">
        <center>
            <h1>Customer Segmentation</h1>
```

```
        <p>Main Input For Main Input Receiving Query</p>
        <form action="{{ url_for('predict') }}" method="post">
            <h1>Please enter the following details</h1>
            <label for="Sex">Sex:</label>
            <select id="Sex" name="Sex">
                <option value="0">Female</option>
                <option value="1">Male</option>
            </select>
            <br><br>
            <label for="Marital status">Marital status:</label>
            <select id="Marital status" name="Marital status">
                <option value="0">Single</option>
                <option value="1">Married</option>
            </select>
            <br><br>
            <label for="Age">Age:</label>
            <input type="number" id="Age" name="Age" min="20" max="80"
placeholder="Age required" required />
            <br><br>
            <label for="Education">Education:</label>
            <select id="Education" name="Education">
                <option value="0">High School</option>
                <option value="1">Bachelor's Degree</option>
                <option value="2">Master's Degree</option>
                <option value="3">Ph.D.</option>
            </select>
            <br><br>
            <label for="Income">Income:</label>
            <input type="number" id="Income" name="Income" min="0"
placeholder="Income" required />
            <br><br>
```

```
            <label for="Occupation">Occupation:</label>
            <select id="Occupation" name="Occupation">
               <option value="0">Not Working</option>
               <option value="1">Working</option>
               <option value="2">Business</option>
            </select>
            <br><br>
            <label for="Settlement size">Settlement size:</label>
            <select id="Settlement size" name="Settlement size">
               <option value="0">0</option>
               <option value="1">1</option>
               <option value="2">2</option>
            </select>

            <br> <br><br>
            <button>predict</button>
         </form>
         <br>
         <br>
         <br>

         <div>The Customer is :{{predict}}</div>
      </center>

   </div>
</body>

</html>
```

Ouput1.html

```
<!DOCTYPE html>
<html>
<head>
   <style>
      body {
         margin: 0;
         padding: 0;
         height: 80vh;
         width: 100vw;
         background: url('/static/image.png') no-repeat center center fixed;
         background-size: contain;
      }
   </style>
</head>
<body>
</body>
</html>
```

Ouput2.html

```
<!DOCTYPE html>
<html>
<head>
   <style>
      body {
         margin: 0;
         padding: 0;
         height: 80vh;
```

```
            width: 100vw;
            background: url('/static/not potential.png') no-repeat center center fixed;
            background-size: contain;
        }
    </style>
</head>
<body>
</body>
</html>
```

Ouput3.html

```
<!DOCTYPE html>
<html>
<head>
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 80vh;
            width: 100vw;
            background: url('/static/potential.png') no-repeat center center fixed;
            background-size: contain;
        }
    </style>
</head>
<body>
</body>
</html>
```

10.2 GitHub & Project Demo Link:

https://github.com/yuvaraju4541/Customer-Segmentation-using-Machine-Learning

10.3 **Demonstration video link :**

https://drive.google.com/file/d/16o96skMd8UROSC8A2fZExy5HogLlKuwh/view?usp=drivesdk