**CRICKET SCORE PREDICTOR**

Machine Learning Mini Project (LP-III)

Submitted by

ARANAV MAHALPURE (Roll No: 63)

VANSH KANAKIA (Roll No: 32)

FINAL YEAR COMPUTER ENGINEERING



Department of Computer Engineering

International Institute of Information Technology

Hinjawadi, Pune – 411057

SEMESTER I (AY 2024-25)

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO.** |
| 1. ABSTARCT | 1 |
| 1. INTRODCUTION | 2 |
| * 1. SCOPE | 4 |
| * 1. REQUIRMENT ANALYSIS | 5 |
| * 1. SOFTWARE AND HARDWARE DETAILS | 7 |
| * 1. LIBRARIES / PACKAGES USED | 7 |
| 1. DATA SET DESCRIPTION AND SOURCE CODE | 8 |
| 1. OUTPUT | 14 |
| 1. CONCLUSION | 15 |

**CHAPTER 1. ABSTARCT**

This project focuses on leveraging machine learning techniques for customer segmentation, enabling businesses to classify their customers into distinct groups based on behavioral and demographic data. By analyzing features such as customer spending habits, income, age, family size, and tenure, we apply unsupervised learning methods including Principal Component Analysis (PCA) and Agglomerative Clustering to group similar customers together. This segmentation helps businesses understand customer preferences, allowing for more targeted marketing strategies and improved customer relationship management. The project also features a Flask-based web application that provides users with an intuitive interface to visualize customer clusters and predict the segment for new customers, enabling data-driven decision-making.

*Department of Computer Engineering 1*

*1*

**CHAPTER 2. INTRODCUTION**

**Problem Statement:**

In today's competitive business landscape, understanding customer behavior is crucial for personalized marketing and effective customer relationship management. However, analyzing large datasets with diverse customer information such as income, age, spending habits, and family size can be challenging. The goal of this project is to develop a machine learning-based customer segmentation system that groups customers into distinct categories based on their characteristics. By leveraging clustering algorithms, we aim to help businesses identify and target specific customer segments, optimize marketing strategies, and improve customer retention. The system will provide valuable insights into customer preferences and allow businesses to make data-driven decisions.

**Introduction:**

1. Introduction: In today's highly competitive market, understanding customer behavior is crucial for effective marketing and personalized customer engagement.

2.Vast Data: Businesses accumulate extensive customer data, including demographics, purchasing habits, and behavioral patterns. However, turning this data into actionable insights is a challenge.

3. Complexity: With numerous variables like income, age, family size, and spending habits, deriving meaningful insights from customer data can be complex due to its scale and diversity.

4. Objective: Our project aims to harness the power of machine learning techniques to classify customers into distinct segments based on their behavior and demographics.

5. Web Integration: Using Flask, we provide an interactive frontend interface, allowing users to explore customer segments and predict customer categories based on input data.

6. Challenges Addressed: We address key challenges, including identifying patterns in customer behavior, spending patterns, and demographic factors amidst complex datasets.

*Department of Computer Engineering 2*

*1*

7. Behavioral Analysis: Understanding customer behavior, such as spending patterns across different product categories and family size, is essential and requires advanced analysis.

8. Prediction: Our project goes beyond descriptive analysis, using machine learning to predict customer segmentation for new customers based on their attributes.

9. Machine Learning Integration: By applying algorithms like Principal Component Analysis (PCA) and Agglomerative Clustering, we extract meaningful segments from raw data.

10. Decision Support: The project empowers businesses with actionable insights into customer preferences, enabling more effective marketing strategies and resource allocation.

11. Enhanced Understanding: We aim to deepen the understanding of different customer segments, providing a clearer view of the diverse behaviors and spending patterns.

12. Data-Driven Decisions: With this project, businesses can make data-driven decisions to improve customer targeting and optimize marketing campaigns.

13. Impact: The project holds the potential to transform how businesses segment and understand their customer base, leading to more personalized and effective marketing strategies.

14. Contribution: By turning raw customer data into valuable insights, we contribute to the advancement of data-driven marketing and customer relationship management.

15. Overall Aim: In summary, our project seeks to unlock the potential hidden within customer data, allowing businesses to make informed decisions and strategically engage their customer base.

*Department of Computer Engineering 3*

*1*

**2.1 SCOPE**

1. Goals: Develop a machine learning-based solution for customer segmentation, aimed at providing actionable insights into customer behavior, improving marketing strategies, and enhancing customer targeting.
2. Deliverables: A Flask-based frontend interface for interactive data visualization and customer classification, along with trained machine learning models for segmenting customers based on their demographics and spending habits.
3. Tasks: Data collection, cleaning, and preprocessing; feature engineering; exploratory data analysis (EDA); frontend development using Flask; implementation of clustering algorithms (PCA, Agglomerative Clustering); model evaluation; deployment; documentation; testing and validation.
4. Resources: Customer data sources, machine learning libraries, development tools, computing resources, and personnel for data collection, analysis, and web development.
5. Boundaries: Analysis is focused on customer behavior data, limited to features like age, income, family size, and spending habits; adherence to ethical guidelines for data privacy, fairness, and transparency.
6. Key Stakeholders: Marketing teams, business analysts, data scientists, and decision-makers who aim to understand customer behavior for optimized marketing strategies.
7. Processes: Data preprocessing, feature engineering, model development, frontend design, testing, deployment, and documentation.
8. Assumptions: Availability of reliable customer data, access to necessary machine learning libraries and computational resources, and expertise in clustering algorithms and frontend development.
9. Constraints: Time constraints, budget limitations, and technical constraints regarding data availability and system requirements for handling large customer datasets.

*Department of Computer Engineering 4*

*1*

**2.2 REQUIRMENT ANALYSIS:**

### **Functional Requirements**

**Data Collection and Preprocessing**:

Gather Data: Collect customer behavior data from reliable sources such as company databases, CRM systems, or third-party data providers.

Clean and Preprocess: Handle missing values, encode categorical variables (e.g., education, marital status), and scale numerical features (e.g., income, age) for proper model training.

**Feature Engineering**:

Create New Features: Generate meaningful features like customer tenure, total spending, family size, and parenthood status to capture customer characteristics.

Identify Key Metrics: Focus on metrics that define customer segments such as total spend, income, age, and household composition to influence marketing strategies.

**Exploratory Data Analysis (EDA):**

Data Visualization: Use visualization libraries (e.g., Matplotlib, Seaborn) to understand the distribution of features like income, age, and spending behavior.

Key Insights: Highlight patterns in spending, demographics, and family size that can help in identifying different customer segments.

**Machine Learning Model Development**:

Apply Clustering Algorithms: Use PCA for dimensionality reduction and Agglomerative Clustering to segment customers based on the transformed data.

Model Optimization: Fine-tune the model using clustering metrics to ensure appropriate segmentation and accuracy in group identification.

*Department of Computer Engineering 5*

*1*

**Frontend Interface Development**:

Develop Interactive Interface: Implement a Flask-based web interface that allows users to explore customer data, filter features, and visualize customer segments.

Model Integration: Integrate machine learning models into the frontend for real-time customer segmentation and insights.

**Prediction for New Customers:**

Prediction Models: Implement classification models that predict the segment of new customers based on their demographic and spending features.

Actionable Insights: Provide businesses with insights on how new customer data fits into existing segments, enabling targeted marketing.

### **Non-Functional Requirements:**

**Performance**:

The system should efficiently handle large datasets of customer data without significant delays in processing or segment prediction.

**Scalability**:

The platform must be scalable to accommodate future growth in customer data and handle more features or advanced models as needed.

**Usability**:

The user interface should be intuitive and easy to navigate, allowing users to segment customers with minimal effort and visualize clear, informative results.

**Security**:

Ensure the system handles sensitive customer data securely, with privacy measures in place to protect personal information.

**2.3 SOFTWARE AND HARDWARE DETAILS:**

*Department of Computer Engineering 6*

*1*

* **Software Requirement:**

|  |  |
| --- | --- |
| Name | Details |
| Operating System | **Windows / Linux** |
| Installations | **Python 3.1** |
| Technology | **Jupyter notebook** |

# Hardware Requirement:

|  |  |
| --- | --- |
| **Name** | **Details** |
| **Processor** | 64-bit Processor |
| **RAM** | 4 GB |
| **Hard Disk** | 16 GB |

**2.4 LIBRARIES / PACKAGES USED:**

* Libraries Used:

· Pandas

· Flask

· Scikit-learn

· NumPy

· Matplotlib

· Seaborn

· StandardScaler

· LabelEncoder

· Agglomerative Clustering

*Department of Computer Engineering 7*

*1*

**CHAPTER 3. DATA SET DESCRIPTION & SOURCE CODE**

The dataset contains 63,888 entries and 9 columns. Here's an overview of the columns:

1. Unnamed: 0: Index column (integer values, likely not significant for analysis).
2. match\_id: Integer identifier for each match.
3. batting\_team: Name of the team batting.
4. bowling\_team: Name of the team bowling.
5. ball: Fractional value representing the ball number in the over (e.g., 0.1 for the first ball of the over).
6. runs: Number of runs scored on the ball.
7. Player\_dismissed: Details if a player was dismissed (0 if no dismissal).
8. city: Name of the city where the match was played (some missing values).
9. venue: Venue name where the match was held.

The dataset appears to represent ball-by-ball data from T20 cricket matches.

Dataset Link: [Crick World Cup 2024 ICC Men's T20 (kaggle.com)](https://www.kaggle.com/datasets/muhammadroshaanriaz/icc-mens-t20-worldcup)

Source Code:

from flask import Flask, render\_template, request, redirect, url\_for

import numpy as np

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.cluster import AgglomerativeClustering

import warnings

import sys

if not sys.warnoptions:

    warnings.simplefilter("ignore")

np.random.seed(42)

data = pd.read\_csv("static/marketing\_campaign.csv", sep="\t")

*Department of Computer Engineering 8*

*1*

app = Flask(\_\_name\_\_)

*Department of Computer Engineering 9*

*1*

data = data.dropna()

data["Dt\_Customer"] = pd.to\_datetime(data["Dt\_Customer"],format='%d-%m-%Y')

dates = []

for i in data["Dt\_Customer"]:

    i = i.date()

    dates.append(i)

days = []

d1 = max(dates) #taking it to be the newest customer

for i in dates:

    delta = d1 - i

    days.append(delta)

data["Customer\_For"] = days

data["Customer\_For"] = pd.to\_numeric(data["Customer\_For"], errors="coerce")

#Feature Engineering

#Age of customer today

data["Age"] = 2021-data["Year\_Birth"]

#Total spendings on various items

data["Spent"] = data["MntWines"]+ data["MntFruits"]+ data["MntMeatProducts"]+ data["MntFishProducts"]+ data["MntSweetProducts"]+ data["MntGoldProds"]

#Deriving living situation by marital status"Alone"

data["Living\_With"]=data["Marital\_Status"].replace({"Married":"Partner", "Together":"Partner", "Absurd":"Alone", "Widow":"Alone", "YOLO":"Alone", "Divorced":"Alone", "Single":"Alone",})

#Feature indicating total children living in the household

data["Children"]=data["Kidhome"]+data["Teenhome"]

#Feature for total members in the householde

data["Family\_Size"] = data["Living\_With"].replace({"Alone": 1, "Partner":2})+ data["Children"]

#Feature pertaining parenthood

data["Is\_Parent"] = np.where(data.Children> 0, 1, 0)

#Segmenting education levels in three groups

data["Education"]=data["Education"].replace({"Basic":"Undergraduate","2n Cycle":"Undergraduate", "Graduation":"Graduate", "Master":"Postgraduate", "PhD":"Postgraduate"})

#For clarity

data=data.rename(columns={"MntWines": "Wines","MntFruits":"Fruits","MntMeatProducts":"Meat","MntFishProducts":"Fish","MntSweetProducts":"Sweets","MntGoldProds":"Gold"})

#Dropping some of the redundant features

to\_drop = ["Marital\_Status", "Dt\_Customer", "Z\_CostContact", "Z\_Revenue", "Year\_Birth", "ID"]

data = data.drop(to\_drop, axis=1)

data = data[(data["Age"]<90)]

data = data[(data["Income"]<600000)]

corrmat= data.corr(numeric\_only=True)

s = (data.dtypes == 'object')

object\_cols = list(s[s].index)

LE=LabelEncoder()

for i in object\_cols:

    data[i]=data[[i]].apply(LE.fit\_transform)

ds = data.copy()

# creating a subset of dataframe by dropping the features on deals accepted and promotions

cols\_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']

ds = ds.drop(cols\_del, axis=1)

#Scaling

scaler = StandardScaler()

scaler.fit(ds)

scaled\_ds = pd.DataFrame(scaler.transform(ds),columns= ds.columns )

pca = PCA(n\_components=3)

pca.fit(scaled\_ds)

PCA\_ds = pd.DataFrame(pca.transform(scaled\_ds), columns=(["col1","col2", "col3"]))

from sklearn.decomposition import PCA

# Set number of components to retain 90% of variance

pca = PCA(n\_components=0.90)  # Retains enough components to explain 90% of the variance

pca.fit(scaled\_ds)

PCA\_ds = pd.DataFrame(pca.transform(scaled\_ds))

AC = AgglomerativeClustering(n\_clusters=4)

# fit model and predict clusters

yhat\_AC = AC.fit\_predict(PCA\_ds)

PCA\_ds["Clusters"] = yhat\_AC

#Adding the Clusters feature to the orignal dataframe.

data["Clusters"]= yhat\_AC

data["Total\_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]+ data["AcceptedCmp3"]+ data["AcceptedCmp4"]+ data["AcceptedCmp5"]

Personal = [ "Kidhome","Teenhome","Customer\_For", "Age", "Children", "Family\_Size", "Is\_Parent", "Education","Living\_With"]

# Cluster properties on the unscaled data

cluster\_unscaled\_centers = data.groupby("Clusters").mean()

# Cluster properties on the unscaled data

cluster\_unscaled\_centers = data.groupby("Clusters").mean()

# Function to name clusters based on data insights

def name\_clusters(cluster\_centers\_unscaled):

*Department of Computer Engineering 10*

*1*

    cluster\_names = []

    cnt=0

    for i, row in cluster\_centers\_unscaled.iterrows():

        cluster\_names.append([])

        if row['Spent'] > 600 and row['Income'] > 5000:

            cluster\_names[cnt].append("High-Spender, Wealthy")

        if row['Spent'] < 500 and row['Income'] < 30000:

            cluster\_names[cnt].append(" Low-Spender, Low-Income")

        if row['Is\_Parent'] > 0.5 and row['Family\_Size'] > 3:

            cluster\_names[cnt].append(" Family\_Oriented, Moderate Spender")

        if row['Is\_Parent']>0.9 :

            cluster\_names[cnt].append(" Has Many Chidren")

        cnt+=1

    return cluster\_names

# Name clusters based on their representative points (unscaled data)

cluster\_names = name\_clusters(cluster\_unscaled\_centers)

# Define all required features

all\_features = ['Education', 'Income', 'Kidhome', 'Teenhome', 'Recency', 'Wines', 'Fruits', 'Meat',

                'Fish', 'Sweets', 'Gold', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',

                'NumStorePurchases', 'NumWebVisitsMonth', 'Customer\_For', 'Age', 'Spent', 'Living\_With',

                'Children', 'Family\_Size', 'Is\_Parent']

from sklearn.neighbors import NearestCentroid

def classify\_new\_customer(new\_data, clustering\_model, scaler, scaled\_ds):

    # Fit the NearestCentroid classifier on the scaled data and clusters

    nc = NearestCentroid()

    nc.fit(scaled\_ds, clustering\_model.labels\_)

    # Scale the new data with the same scaler used previously

    new\_data\_scaled = scaler.transform(new\_data)

    # Predict the cluster for the new customer

    cluster = nc.predict(new\_data\_scaled)

    return cluster

@app.route('/')

def index():

*Department of Computer Engineering 11*

*1*

    return render\_template('index.html')

@app.route('/search', methods=['GET'])

def search():

    # Redirect to result route with all query parameters from the form

    return redirect(url\_for('result', \*\*request.args))

@app.route('/result')

def result():

    # List of all expected features

    all\_features = [

        'Education', 'Income', 'Kidhome', 'Teenhome', 'Recency', 'Wines', 'Fruits',

        'Meat', 'Fish', 'Sweets', 'Gold', 'NumDealsPurchases', 'NumWebPurchases',

        'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Customer\_For',

        'Age', 'Spent', 'Living\_With', 'Children', 'Family\_Size', 'Is\_Parent'

    ]

    # Retrieve and convert query parameters dynamically

    customer\_data = {feature: int(request.args.get(feature, 0)) for feature in all\_features}

    customer\_segmentation = {

    "Group 1": {

        "description": "Definitely a parent, max 4 family members, at least 2. Many are single parents with teenagers at home, relatively older.",

        "characteristics": {

            "parent\_status": "definitely a parent",

            "max\_family\_members": 4,

            "min\_family\_members": 2,

            "common\_age": "older",

            "notable\_traits": ["single parents", "teenagers at home"]

        }

    },

    "Group 2": {

        "description": "Definitely not a parent, max 2 family members, mostly couples, high income group, spans all ages.",

        "characteristics": {

            "parent\_status": "definitely not a parent",

            "max\_family\_members": 2,

            "common\_age": "all ages",

            "income": "high",

            "notable\_traits": ["mostly couples"]

        }

    },

    "Group 3": {

        "description": "Majority are parents, max 3 family members, mostly one kid, typically not teenagers, relatively younger.",

        "characteristics": {

*Department of Computer Engineering 12*

*1*

            "parent\_status": "mostly parents",

            "max\_family\_members": 3,

            "common\_age": "younger",

            "notable\_traits": ["mostly one kid", "not teenagers"]

        }

    },

    "Group 4": {

        "description": "Definitely a parent, max 5 family members, at least 2, many have teenagers at home, relatively older, lower income group.",

        "characteristics": {

            "parent\_status": "definitely a parent",

            "max\_family\_members": 5,

            "min\_family\_members": 2,

            "common\_age": "older",

            "income": "low",

            "notable\_traits": ["many have teenagers"]

        }

    }

}

    # Convert the dictionary into a Pandas DataFrame

    new\_customer = pd.DataFrame([customer\_data])

    # Call the classification function to classify the new customer

    new\_customer\_cluster = classify\_new\_customer(new\_customer, AC, scaler, scaled\_ds)

     # Get the cluster number

    cluster\_number = new\_customer\_cluster[0]

    # Access the description and characteristics from the segmentation dictionary

    group\_name = f"Group {cluster\_number}"  # Adjust for 0-indexed to 1-indexed

    group\_info = customer\_segmentation[group\_name]

    # Render the result page with the predicted cluster and its details

    return render\_template('result.html',

                           cluster\_number=cluster\_number + 1,

                           description=group\_info['description'],

                           characteristics=group\_info['characteristics'])

@app.route('/visual')

def visualization\_page():

    return render\_template('visual.html')

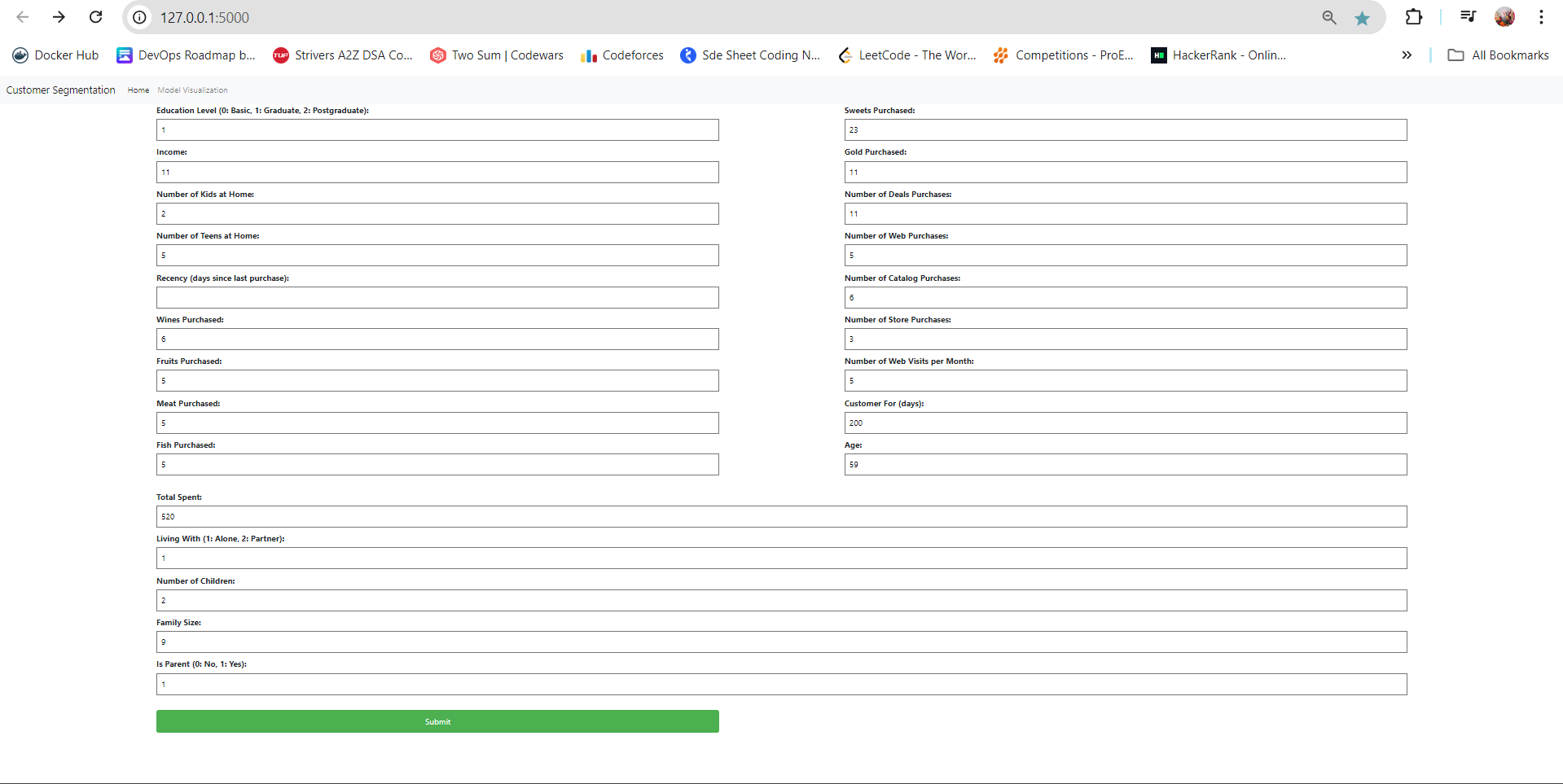
if \_\_name\_\_ == '\_\_main\_\_':

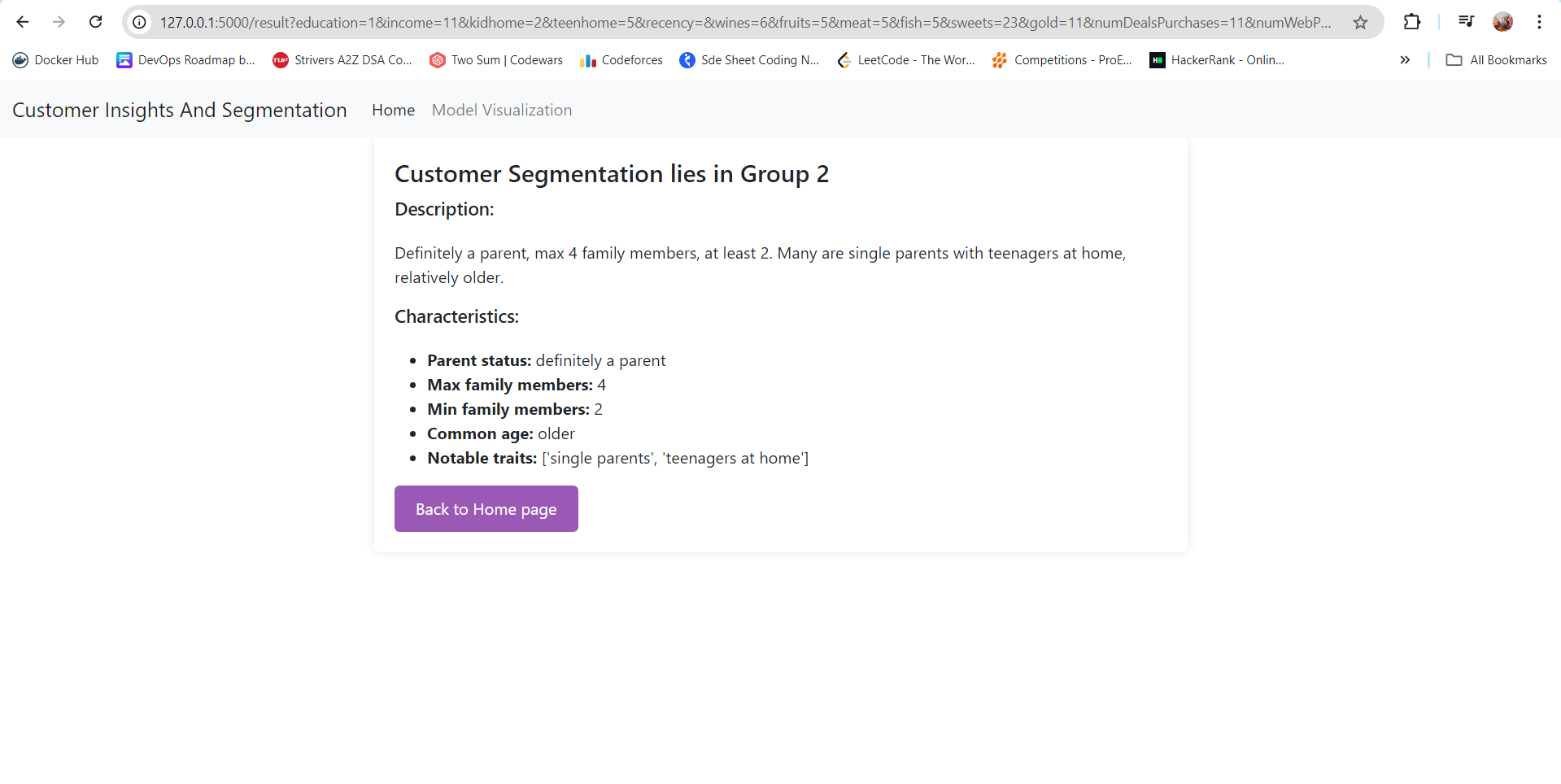
app.run(debug=True)

*Department of Computer Engineering 13*

*1*

**CHAPTER 4. OUTPUT**





*Department of Computer Engineering 14*

**CHAPTER 5. CONCLUSION**

In conclusion, our project aims to transform customer segmentation through advanced machine learning techniques and an intuitive web interface developed using Flask. By analyzing extensive customer datasets, identifying key demographic and behavioral metrics, and deploying clustering algorithms, we empower businesses to make data-driven decisions regarding marketing strategies and customer engagement. Through comprehensive testing and user feedback, we ensure the robustness and usability of our solution. With a focus on ethical data practices and ongoing enhancements, we seek to deepen the understanding of customer behavior and segmentation, paving the way for a more personalized and effective marketing approach in today’s competitive landscape.

*Department of Computer Engineering 15*

*1*