UNDERSTANDING AUDIENCE: A MACHINE LEARNING APPROACH TO CUSTOMER SEGMENTATION

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

MOGILI VIDHYA	**20UK1A6607**
PAVAN KOLIPAKA	**20UK1A6613**
DARA ADITHYA	**20UK1A6646**

Under the guidance of

Mr.'s D. Gowthami

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) –

506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "UNDERSTANDING AUDIENCE: A MACHINE LEARNING APPROACH TO CUSTOMER SEGMENTATION" is being submitted by MOGILI VIDHYA (20UK1A6607), PAVAN KOLIPAKA (20UK1A6613), DARA ADITYA (20UK1A6646) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering toJawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024.

Project Guide
Mr.'s. D. Gowthami
(Assistant Professor)

HOD
Dr. K. Sharmila
(Professor)

External

ABSTRACT

In this study, we propose a groundbreaking method for customer segmentation by integrating machine learning techniques into the analysis of diverse datasets. Recognizing the limitations of conventional segmentation strategies in capturing the complexities of modern markets, our research focuses on leveraging advanced analytics to reveal hidden patterns and trends. Real-world case studies will be used to showcase the superior efficacy of our machine learning approach compared to traditional segmentation methods, with implications for industries seeking more precise and adaptable customer targeting strategies.

The anticipated outcome of this research is the development of a versatile framework that enables businesses to elevate their customer understanding, refine marketing strategies, and foster personalized interactions. As the market landscape continues to evolve, our proposed approach offers a timely solution for organizations striving to meet the rapidly changing expectations of their diverse customer base. By embracing machine learning in audience segmentation, businesses can position themselves at the forefront of innovation, ensuring that their strategies remain agile and responsive in an era where individualization and tailored experiences are paramount to success.

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA**, Head of the Department of CSD, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **D. Gowthami,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout thesis.

MOGILI VIDHYA **(20UK1A6607)**
PAVAN KOLIPAKA **(20UK1A6613)**
DARA ADITYA **(20UK1A6646)**

TABLE OF CONTENTS

1.INTRODUCTION

In the contemporary landscape of business and marketing, understanding customer behavior is pivotal for success. This paper introduces a novel approach to customer segmentation, integrating machine learning techniques to uncover nuanced insights within large datasets. Traditional methods often fall short in capturing the complexities of modern markets, prompting the need for innovative strategies. By leveraging advanced analytics and machine learning algorithms, this research aims to provide businesses with a more precise understanding of their audience, enabling tailored strategies and personalized interactions.

1.1. OVERVIEW

Customer segmentation is a crucial aspect of marketing and business strategy. It involves dividing a customer base into smaller groups or segments, each sharing common characteristics, behavior or needs. The purpose of customer segmentation is to enable companies to tailor their marketing efforts, product offerings, and customer experiences to better meet the specific requirements and preferences of each segment.

In recent years, the field of customer segmentation has seen significant advancements due to the integration of machine learning techniques. Machine learning, a subset of artificial intelligence, empowers businesses to analyze vast amounts of data and extract valuable insights from it. When applied to customer segmentation, machine learning can help identify hidden patterns, relationships, and trends that might be challenging to discern using traditional methods.

1.2. PURPOSE

The project, "Understanding Audience: A Machine Learning Approach to Customer Segmentation," is a game-changer in the world of business strategy. Its primary goal is to harness the formidable power of machine learning to break down a company's customer base into finely tuned segments, each with its own unique characteristics and behaviors. This visionary approach is poised to turbocharge marketing efforts, ensuring laser-focused campaigns that maximize conversions and minimize costs.

It also paves the way for customer experiences that resonate with each segment's distinct preferences and needs, cultivating unwavering customer loyalty. Furthermore, this project positions organizations for a resounding competitive edge, enabling them to pivot rapidly in response to market dynamics and emerging trends.

With resource allocation at its most efficient, data-driven decision-making at its core, and a commitment to ongoing refinement, this project ultimately seeks to deliver substantial revenue growth, enhanced customer retention, and the ability to measure the tangible impact of personalized strategies on each segment, guaranteeing a position of strength and adaptability in today's fast-paced business landscape.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

The prevailing problem that necessitates the project, "Understanding Audience: A Machine Learning Approach to Customer Segmentation," is the persistence of generic and non-targeted marketing and business strategies. Many organizations still rely on outdated, blanket approaches that do not consider the diversity and specific needs of their customer base. This leads to suboptimal resource allocation, with marketing budgets often spread thinly and ineffectively. Customers are bombarded with irrelevant marketing messages, causing disengagement and dissatisfaction. Additionally, the absence of data-driven segmentation makes companies sluggish in adapting to rapidly changing market trends, leaving them vulnerable to more agile competitors. This project seeks to rectify these challenges by leveraging machine learning to create precision-driven strategies that optimize marketing endeavor's, elevate customer contentment, and fortify businesses' competitive edge in an ever-shifting business ecosystem.

2.2PROPOSED SOLLUTION

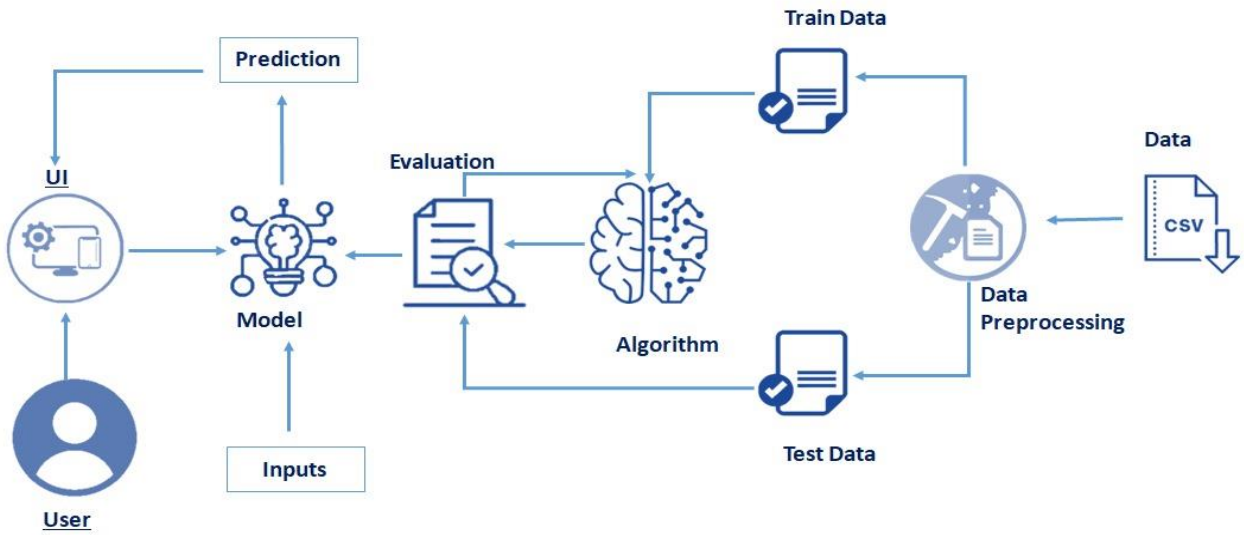
- **Identified Problem:** The current challenge lies in the inadequacy of existing marketing and business strategies, often characterized by non-targeted approaches that fail to address the diverse needs and preferences of customers.
- **Proposed Solution:** The suggested remedy involves the implementation of a robust, data-driven customer segmentation strategy. This approach aims to collect a comprehensive dataset, encompassing demographics, purchase history and online interactions, which will be integrated into a unified database.
- **Data Harmonization:** To ensure coherence and consistency, diverse sets of customer data will be harmonized into a unified database, laying the foundation for meaningful analysis.
- **Machine Learning Algorithms:** Advanced machine learning algorithms, specifically clustering methods and classification models, will be employed to analyze the integrated dataset. These algorithms will unveil hidden patterns and group customers into distinct segments based on shared characteristics and behaviors.
- **Preprocessing and Feature Engineering:** To enhance data accuracy and relevance, preprocessing techniques and feature engineering will be applied. This step ensures that the data is refined and

optimized for effective analysis.

- **Segment Profiling:** The resulting segments will undergo meticulous profiling to gain a profound understanding of each group's unique preferences and needs. This in-depth analysis enables businesses to tailor their strategies to specific customer segments.

3 THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **Requirements Analysis:** Began by comprehensively understanding and documenting the functional and non-functional requirements of the software. This involves engaging stakeholders to gather information about the desired features, performance expectations, and user experience.
- **System Design:** Develop a high-level system architecture that outlines the overall structure of the software. This involves defining modules, components, and their interactions. Consideration should be given to scalability, flexibility, and future expansion.
- **Database Design:** Design the database schema based on the data requirements of the software. This includes defining tables, relationships, and constraints. Consider data normalization to minimize redundancy and ensure data integrity.

- **User Interface (UI) Design:** Create mockups and prototypes to design the user interface. Consider usability, accessibility, and user experience principles. Interaction flows and design elements should align with the intended user journey and the overall aesthetic of the application.
- **Algorithm Design:** Define and design algorithms for critical functions within the software. This includes algorithms for data processing, computation, and decision-making. Optimize algorithms for efficiency and scalability.
- **Security Design:** Implement security measures to protect the software from potential threats. This involves designing secure authentication and authorization mechanisms, encrypting sensitive data, and ensuring secure communication channels.
- **Data Flow Design:** Map out the flow of data within the software. Identify how data is input, processed, stored, and output throughout the system. This includes considering data validation, error handling, and logging mechanisms.
- **Error Handling and Recovery Design:** Plan for error scenarios and define strategies for error handling and recovery. This includes designing user-friendly error messages, logging error details for debugging, and implementing mechanisms for system recovery after a failure.
- **Testing Strategy:** Develop a comprehensive testing strategy that includes unit testing, integration testing, and system testing. Define test cases and scenarios to ensure that the software meets the specified requirements and functions as intended.

4. EXPERIMENTAL INVESTIGATIONS

Experimental investigations involve systematically conducting experiments to gather data, test hypotheses, and draw meaningful conclusions. Here are key points related to conducting experimental investigations:

1. **Research Question Formulation:** Clearly define the research question or hypothesis that the experiment aims to address. The question should be specific, measurable, and aligned with the goals of the investigation.

2. **Experimental Design:** Develop a robust experimental design outlining the methods, procedures, and variables involved. Consider factors such as control groups, independent and dependent variables, and randomization to ensure the validity of the experiment.

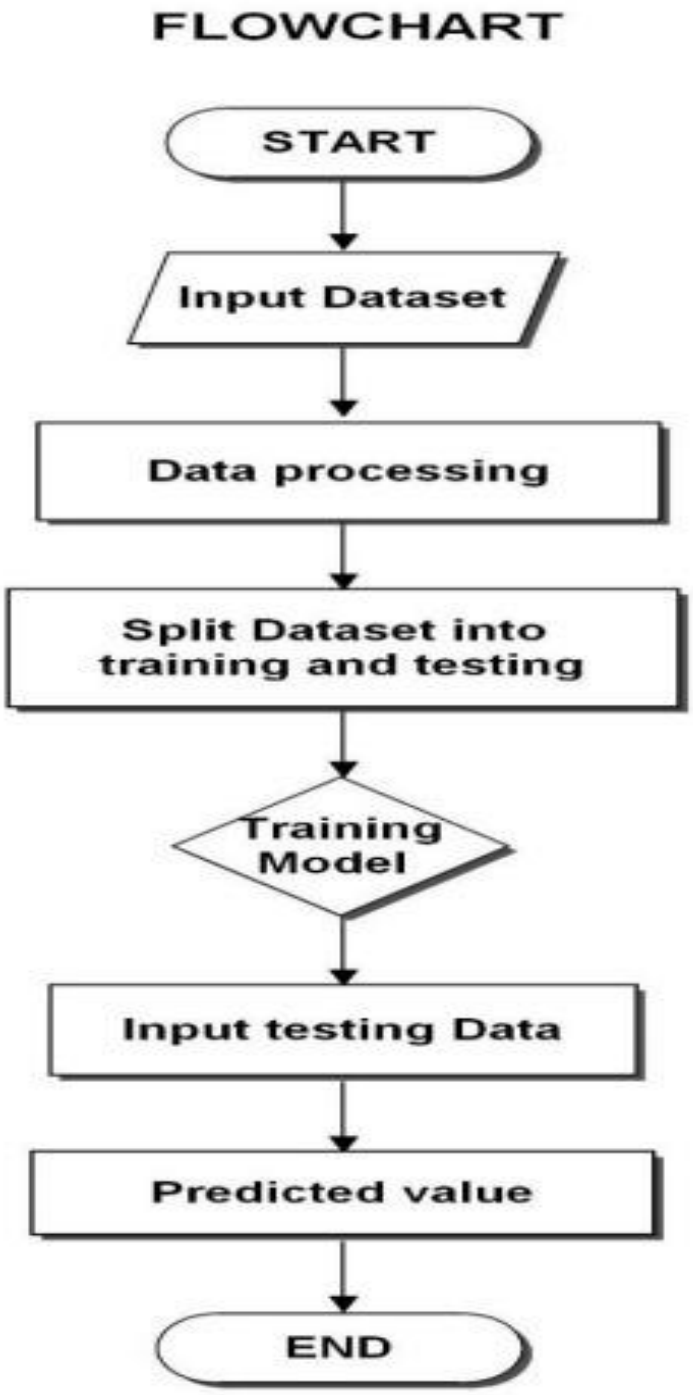
3. **Variables Definition:** Identify and clearly define the independent variables (factors manipulated by the researcher) and dependent variables (outcomes measured as a result of changes in the independent variables). Control variables should also be considered to minimize confounding factors.

4. **Sampling Strategy:** Define the target population and implement a suitable sampling strategy. Ensure that the chosen sample size is representative and provides statistically significant results.

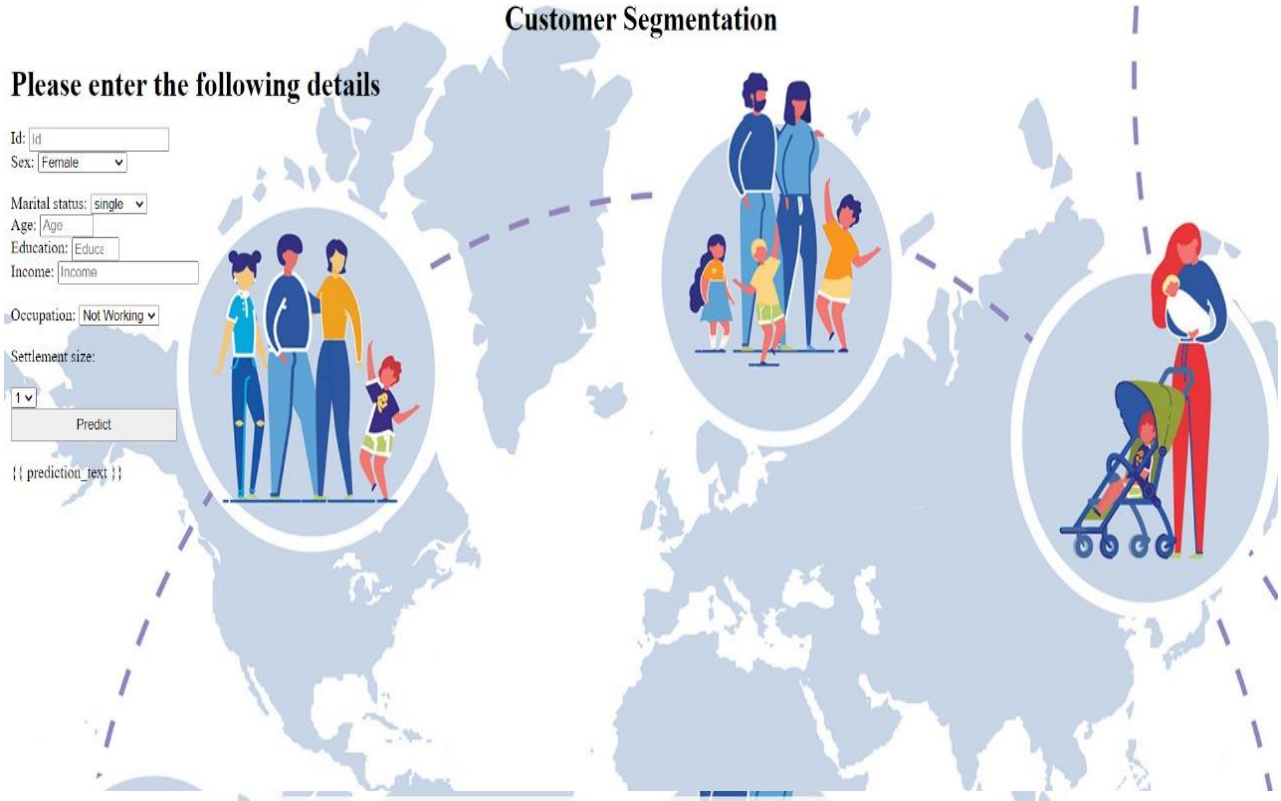
5. **Ethical Considerations:** Adhere to ethical guidelines and obtain necessary approvals, especially when human subjects or sensitive data are involved. Prioritize participant confidentiality, informed consent, and the well-being of individuals involved in the experiment.

6. **Data Collection:** Implement a systematic approach to collect relevant data. This may involve using instruments, surveys, observations, or other data collection methods. Ensure that data collection methods align with the experimental design.

5.FLOWCHART


FLOWCHART
START
Input Dataset
Data processing
Split Dataset into training and testing
Training Model
Input testing Data
Predicted value
END

6 RESULTS


Customer Segmentation
Please enter the following details
Id: Id
Sex: Female
Marital status: single
Age: Age
Education: Educa
Income: Income
Occupation: Not Working
Settlement size:
1
Predict
{{ prediction_text }}


Customer Segmentation
Please enter the following details
Sex: Female
Marital status: Single
Age: Age
Education: Education
Income: Income
Occupation: Not Working
Settlement size: 1
Predict
Not a potential customer

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Precision:** Machine learning enables precise customer segmentation by identifying subtle patterns and relationships in large datasets, resulting in more accurate targeting.
2. **Personalization:** Businesses can deliver highly personalized marketing messages and product recommendations, leading to improved customer satisfaction and engagement.
3. **Efficiency:** Machine learning streamlines the process of segmentation, reducing the time and effort required to create and maintain segments.
4. **Dynamic Adaptation:** Machine learning models can adapt to changing customer behavior and market dynamics, ensuring that the segments remain relevant over time.
5. **Improved ROI:** Targeted marketing efforts reduce wasted resources, leading to higher return on investment (ROI) for marketing campaigns.

DISADVANTAGES:

1. **Data Quality:** Machine learning models heavily rely on the quality of input data. Inaccurate or biased data can lead to flawed segmentation results.
2. **Complexity:** Implementing machine learning can be complex, requiring specialized expertise and resources that not all businesses may have.
3. **Initial Cost:** Developing and implementing machine learning models can be expensive, especially for smaller businesses with limited budgets.
4. **Interpretability:** Some machine learning models, such as deep learning neural networks, can be challenging to interpret, making it difficult to understand why specific segmentation decisions are made.
5. **Privacy Concerns:** Handling sensitive customer data for segmentation may raise privacy and security concerns, necessitating robust data protection measures.
6. **Overfitting:** Machine learning models are at risk of overfitting the training data, resulting in segments that do not generalize well to new customer data.

8.APPLICATIONS

1. **Marketing Personalization:** Machine learning enables businesses to create highly personalized marketing campaigns. By tailoring messages and offers to specific customer segments, companies can increase customer engagement and conversion rates.
2. **Customer Retention:** By understanding customer behavior and preferences, businesses can implement strategies to improve customer retention rates. Customized loyalty programs and retention campaigns can be developed to cater to the needs of each segment.
3. **Email Marketing:** Email marketing campaigns can be fine-tuned to target specific segments with relevant content, increasing email open rates and click-through rates.
4. **Customer Support:** Segmentation can guide customer support teams to provide more targeted and effective assistance to different customer groups.
5. **Fraud Detection:** In the financial industry, customer segmentation can help identify unusual or fraudulent transactions by recognizing patterns that deviate from typical customer behavior.

9.CONCLUSION

- In conclusion, the project, "Understanding Audience: A Machine Learning Approach to Customer Segmentation," represents a transformative shift in how businesses understand and engage with their customer base. The implementation of machine learning for customer segmentation offers a myriad of benefits, ranging from precision marketing to adaptive strategies. The advantages are clear: personalized campaigns, enhanced customer experiences, efficient resource allocation, and a competitive edge. By leveraging machine learning algorithms, businesses can derive actionable insights from vast amounts of customer data, unlocking the potential for highly targeted marketing efforts that improve ROI. This data-driven approach equips organizations with the agility to adapt to changing market dynamics, providing a distinct competitive advantage in a rapidly evolving landscape.

- However, it's crucial to recognize the challenges that come with this approach, including data quality issues, complex implementation, and the need for ongoing maintenance. Overcoming these hurdles demands a commitment to data integrity, expertise in machine learning, and an investment in resources.

10.FUTURE SCOPE

Future Scope of the AQI Prediction and Management System:

1. **Evolution of Machine Learning and Data Analytics**: Anticipate the continuous evolution of machine learning algorithms and data analytics, providing businesses with more sophisticated tools for customer segmentation.
2. **Real-Time Customer Segmentation:** Expect the development of real-time customer segmentation, enabling businesses to dynamically adapt their strategies based on immediate changes in customer behavior and preferences.
3. **Hyper-Personalization:** Envision a future where hyper-personalization becomes the norm, utilizing advanced algorithms to tailor products, services, and marketing messages at an individual level.
4. **Predictive Analytics:** Explore the potential for enhanced predictive analytics, allowing businesses to anticipate customer needs and behaviors, thereby optimizing decision-making processes.

11.BIBILOGRAPHY

- Dr. R. Gardener “The Essential R Reference” (2014),
- Concepts of customer segmentation http://www.business-science.io
- https://labs.openviewpartners.com/customer-segmentation/
- Source data related to our analysis has been collected from

  https://github.com/mdancho84/orderSimulatoR/tree/master/data
- https://www.kaggle.com/
- https://www.r-project.org/
- https://www.rstudio.com/

11.APPENDIX
(SOURCE CODE)

App.py

```
import numpy as npimport

pickle import pandas import

os

import joblib

from flask import Flask, request, jsonify, render_template



app = Flask(__name__)

model = pickle.load(open(r'F:/notebook/customer/Flask/xgbmodel1.pkl', 'rb')) #scale =

pickle.load(open(r'C:/Users/SmartbridgePC/Desktop/AIML/Guided

projects/rainfall_prediction/IBM flask push/Rainfall IBM deploy/scale.pkl','rb'))


@app.route('/')# route to display the home pagedef home():

    return render_template('index.html') #rendering the home page


@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UIdef

predict():

    # reading the inputs given by the user input_feature=[float(x) for x in

    request.form.values() ]features_values=[np.array(input_feature)]

    names = [['Id','Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation','Settlementsize']]

    data = pandas.DataFrame(features_values,columns=names)#data =
```

```
    scale.fit_transform(features_values




# predictions using the loaded model file
prediction=model.predict(data) print(prediction)


    if (prediction == 0):
      return render_template("index.html",prediction_text ="Not a potential customer")elif
    (prediction == 1):
      return render_template("index.html",prediction_text = "Potential customer")else:
      return render_template("index.html",prediction_text = "Highly potential customer")# showing
     the prediction results in a UI
 if__name__=="__main__":


    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```

Index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Customer Segmentation</title>
</head>

<body background="https://www.imf.org/external/pubs/ft/fandd/2020/03/images/032020/picture-1600.jpg" text="black">
    <div class="login">

        <center>
            <h1>Customer Segmentation</h1>
        </center>
        <!-- Main Input For Receiving Query to our ML -->
        <form action="{{ url_for('predict')}}" method="post">
            <h1>Please enter the following details</h1>
            </style>
            </head>
            <label>Id:</label>
```

```
        <input type="number" name="Id" placeholder="Id" required="required" /><br>
        <label for="Sex">Sex:</label>
        <select id="Sex" name="Sex">
           <option value=0>Female</option>
           <option value=1>Malek/option>
        </select>   <br>
        <br><label for="Marital status">Marital status:</label> <select id="Marital status" name="Marital
status">
           <br>
           <br>
           <option value=0>single</option>
           <option value=1>Married</option>
        </select>   <br>
        <label>Age:</label>
        <input type="number" min="20" max="80" name="Age" placeholder="Age" required="required"
/><br>
        <label>Education:</label>

        <input type="number" min="0" max="3" name="Education" placeholder="Education"
required="required" /><br>

        <label>Income:</label>
        <input type="number" min="5000" name="Income" placeholder="Income" required="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=1>Business</option>
        </select>   <br>

<br>
        <label for="Settlement size">Settlement size:</label>
        <br><br>
        <select id="Settlement size" name="Settlement size">
            <option value=1>1</option>
            <option value=0>0</option>
            <option value=2>2</option>
        </select>   <br>
        <button type="submit" class="btn btn-primary btn-block btn-large"
            style="height:30px;width:200px">Predict</button>
        <br>
        <br>

    </form>
    {{ prediction_text }}
    <br> <br>
    <img src="{{url_3}}" alt="Submit Form" height="180" width="233"
```

```
        onerror="this.style.display='none'" /> <img src="{{url_1}}" alt="Submit
Form"
        height="180" width="233" onerror="this.style.display='none'" /> <img
src="{{url_4}}"
        alt="Submit Form" height="180" width="233" onerror="this.style.display='none'" />



<br>
      <br>
      <img src="{{url_2}}" alt="Submit Form" height="150" width="711"
        onerror="this.style.display='none'" />
   </div>
</body>
```

(CODE SNIPPETS)

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
import numpy as np
import pandas as pd
import sklearn
import scipy
import seaborn as sns
from matplotlib import pyplot as plt
```

```
import numpy as np
import pandas as pd
```

```
data=pd.read_excel("/content/drive/MyDrive/segmentation data.xlsx")
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
...	...	...	...	...	...	...	...	...
1995	100001996	1	0	47	1	123525	0	0
1996	100001997	1	1	27	1	117744	1	0
1997	100001998	0	0	31	0	86400	0	0
1998	100001999	1	1	24	1	97968	0	0
1999	100002000	0	0	25	0	68416	0	0

2000 rows × 8 columns

```
pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
data.head()
```

	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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               2000 non-null   int64
 1   Sex              2000 non-null   int64
 2   Marital status   2000 non-null   int64
 3   Age              2000 non-null   int64
 4   Education        2000 non-null   int64
 5   Income           2000 non-null   int64
 6   Occupation       2000 non-null   int64
 7   Settlement size  2000 non-null   int64
dtypes: int64(8)
memory usage: 125.1 KB
```

```
data.shape
```

```
(2000, 8)
```

```
data.describe()
```

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
count	2.000000e+03	2000.000000	2000.000000	2000.000000	2000.00000	2000.000000	2000.000000	2000.000000
mean	1.000010e+08	0.457000	0.496500	35.909000	1.03800	120954.419000	0.810500	0.739000
std	5.774946e+02	0.498272	0.500113	11.719402	0.59978	38108.824679	0.638587	0.812533
min	1.000000e+08	0.000000	0.000000	18.000000	0.00000	35832.000000	0.000000	0.000000
50%	1.000010e+08	0.000000	0.000000	33.000000	1.00000	115548.500000	1.000000	1.000000
75%	1.000015e+08	1.000000	1.000000	42.000000	1.00000	138072.250000	1.000000	1.000000
max	1.000020e+08	1.000000	1.000000	76.000000	3.00000	309364.000000	2.000000	2.000000

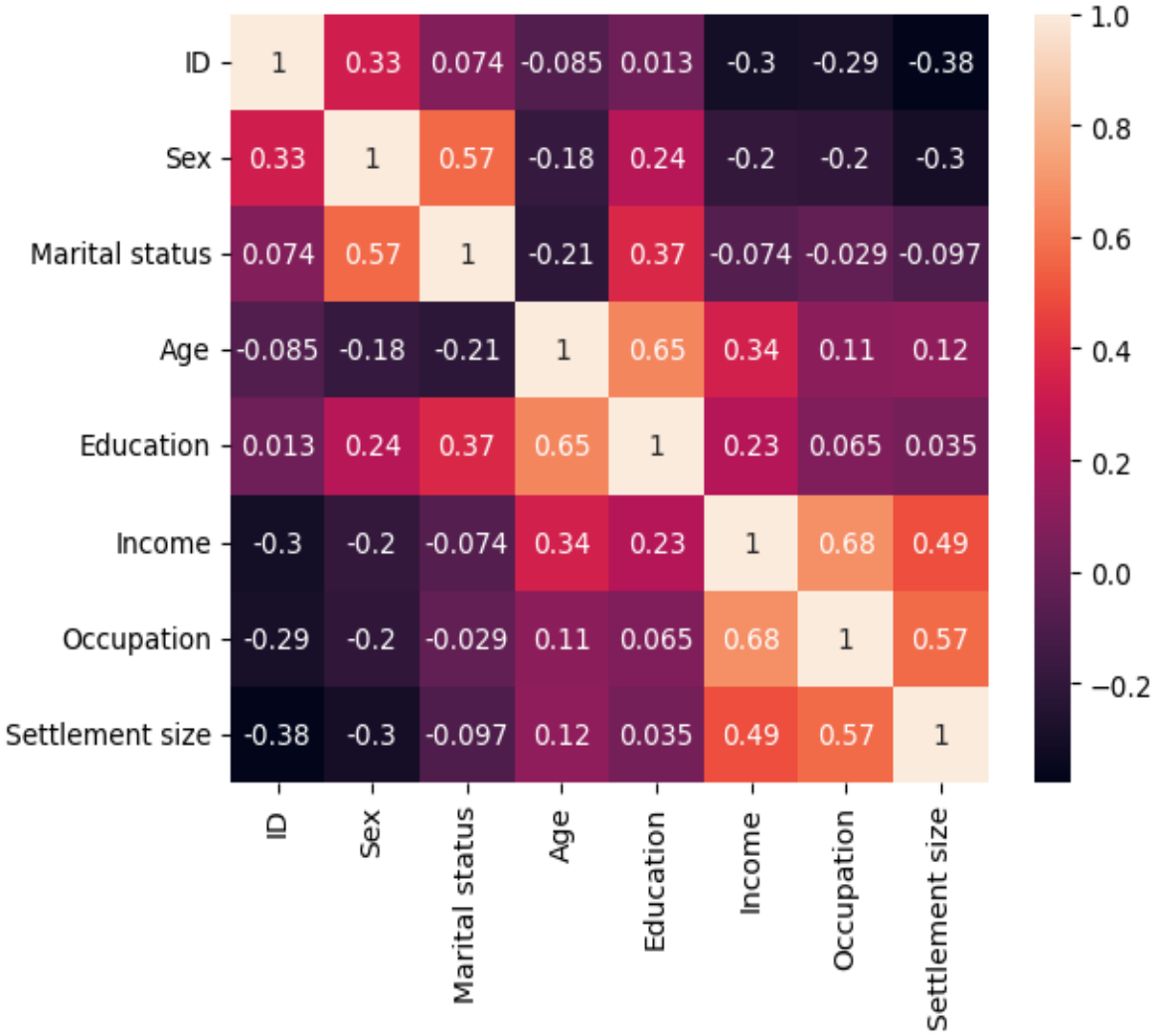
```
cor = data.corr()
cor
```

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
ID	1.000000	0.328262	0.074403	-0.085246	0.012543	-0.303217	-0.291958	-0.378445
Sex	0.328262	1.000000	0.566511	-0.182885	0.244838	-0.195146	-0.202491	-0.300803
Marital status	0.074403	0.566511	1.000000	-0.213178	0.374017	-0.073528	-0.029490	-0.097041
Age	-0.085246	-0.182885	-0.213178	1.000000	0.654605	0.340610	0.108388	0.119751
Education	0.012543	0.244838	0.374017	0.654605	1.000000	0.233459	0.064524	0.034732
Income	-0.303217	-0.195146	-0.073528	0.340610	0.233459	1.000000	0.680357	0.490881
Occupation	-0.291958	-0.202491	-0.029490	0.108388	0.064524	0.680357	1.000000	0.571795
Settlement size	-0.378445	-0.300803	-0.097041	0.119751	0.034732	0.490881	0.571795	1.000000

```
import seaborn as sns
```

```
sns.heatmap(cor,annot=True)
```

```
<Axes: >
```


ID 1 0.33 0.074 -0.085 0.013 -0.3 -0.29 -0.38
Sex 0.33 1 0.57 -0.18 0.24 -0.2 -0.2 -0.3
Marital status 0.074 0.57 1 -0.21 0.37 -0.074 -0.029 -0.097
Age -0.085 -0.18 -0.21 1 0.65 0.34 0.11 0.12
Education 0.013 0.24 0.37 0.65 1 0.23 0.065 0.035
Income -0.3 -0.2 -0.074 0.34 0.23 1 0.68 0.49
Occupation -0.29 -0.2 -0.029 0.11 0.065 0.68 1 0.57
Settlement size -0.38 -0.3 -0.097 0.12 0.035 0.49 0.57 1
ID
Sex
Marital status
Age
Education
Income
Occupation
Settlement size
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


+1e8
ID
Sex
Marital status
Age
Education
Income
Occupation
Settlement size

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

```
names = data.columns
```

```
data.isna().sum()
```

```
Sex               0
Marital status    0
Age               0
Education         0
Income            0
Occupation        0
Settlement size   0
dtype: int64
```

```
from sklearn import preprocessing
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
```

```
from sklearn import preprocessing
```

```
scaled_data = preprocessing.minmax_scale(data,feature_range=(0,1))  #scaling the data
```

```
scaled_data
```

```
array([[0.        , 0.        , 0.84482759, ..., 0.32478101, 0.5       ,
        1.        ],
       [1.        , 1.        , 0.06896552, ..., 0.42021043, 0.5       ,
        1.        ],
       [0.        , 0.        , 0.53448276, ..., 0.19514353, 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.22413793, ..., 0.18487051, 0.        ,
        0.        ],
       [1.        , 1.        , 0.10344828, ..., 0.22716172, 0.        ,
        0.        ],
       [0.        , 0.        , 0.12068966, ..., 0.11912317, 0.        ,
        0.        ]])
```

```
import pandas as pd
scaled_data_df = pd.DataFrame(scaled_data, columns=names)
```

```
from sklearn import cluster
```

```
from scipy import spatial
```

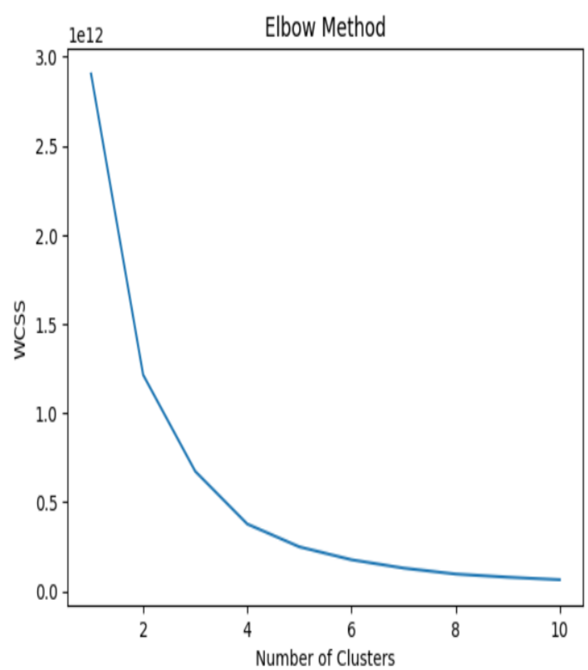
```
from sklearn import cluster
import matplotlib.pyplot as plt

# Assuming 'data' is your dataset
wcss = []
num_samples = len(data)  # Replace 'data' with the actual name of your dataset

# Calculate WCSS
for i in range(1, min(11, num_samples + 1)):
    kmeans = cluster.KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)

# Plot the elbow method
plt.plot(range(1, min(11, num_samples + 1)), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_ir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_ir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_ir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_ir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_ir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
  warnings.warn(
```



```python
for i in range(1, min(11, len(data))):
    kmeans = cluster.KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
```

```python
ykmeans=km_model.fit_predict(data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` wi
  warnings.warn(
```

```python
import numpy as np

# Assuming 'data' is your NumPy array
print(data[:5])  # Print the first 5 elements
```

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

```python
import pandas as pd

# Assuming 'data' is your NumPy array
data_df = pd.DataFrame(data)
print(data_df.head())
```

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

```
import pandas as pd

# Assuming 'data' is your NumPy array
data_df = pd.DataFrame(data)
print(data_df.head())
```

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

```
from sklearn import cluster
import sklearn as sk
```

```
clus = cluster.AgglomerativeClustering(n_clusters=3,affinity="euclidean",linkage='complete')
clus
```

```
AgglomerativeClustering(affinity='euclidean', linkage='complete', n_clusters=3)
```

```
clus.fit(data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning: Attribute `affinity` was deprecated in version 1
  warnings.warn(
```

```
AgglomerativeClustering(affinity='euclidean', linkage='complete', n_clusters=3)
```

```
[ ] abc = clus.fit_predict(data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning: Attribute `affinity` was deprecated in version 1.2 and will be removed in 1.4. (
  warnings.warn(
```

```
[ ] hclusdata = pd.DataFrame(data,pd.Series(abc))
```

Creating a labled data with the help of clustering model

```
[ ] hclusdata['clus']= pd.Series(abc)
```

```
hclusdata.head()
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	clus
0	0	0	67	2	124670	1	2	0
0	0	0	67	2	124670	1	2	0
2	0	0	49	1	89210	0	0	2
0	0	0	67	2	124670	1	2	0
0	0	0	67	2	124670	1	2	0

```
[ ] y = hclusdata['clus']
    x = hclusdata.drop(columns=['clus'],axis=1)
```

splitting the test and train data

```
[ ] 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)
```

```
print(x_train.shape)
print(x_test.shape)
```

```
(1600, 7)
(400, 7)
```

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

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

```
from sklearn.model_selection import train_test_split

# Assuming you've already adjusted your 'y' variable to contain only 0 and 1
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

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

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

ɔdel Evaluation

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

```
[ ]  from sklearn import metrics
```

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

```
1.0
1.0
1.0
```

Importing And Initializing The Model

K-means clustering

```
[ ]  from scipy import spatial
```

```
wcss = []
# Determine the maximum number of clusters based on the size of your dataset
max_clusters = min(len(hclusdata), 10)

for i in range(1, max_clusters + 1):
    kmeans = cluster.KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(hclusdata)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'a
  warnings.warn(
```

```
hclusdata.duplicated().sum()
```

```
0
```

```
hclusdata = hclusdata.drop_duplicates()
```

```
import pandas as pd
from sklearn.cluster import KMeans

# Convert specific columns to numeric values
numeric_columns = ['column1', 'column2', 'column3']  # Replace with the actual column names

# Convert the column names to strings and handle any potential discrepancies
hclusdata.columns = hclusdata.columns.astype(str)

# Check if the columns in numeric_columns exist in the DataFrame
missing_columns = set(numeric_columns) - set(hclusdata.columns)
```

```
if missing_columns:
    print(f"The following columns are missing: {missing_columns}")
else:
    i = 3  # Your value of 'i'

    # Your KMeans code
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(hclusdata)
```

```
The following columns are missing: {'column2', 'column1', 'column3'}
```

```
import pandas as pd
from sklearn.cluster import KMeans

# Assuming hclusdata is a DataFrame with mixed types
# Convert any string columns to numeric values
hclusdata_numeric = hclusdata.apply(pd.to_numeric, errors='coerce')

# Fill NaN values with a suitable value, e.g., 0
hclusdata_numeric = hclusdata_numeric.fillna(0)

# Assuming you define i somewhere before using it
i = 3

# Your KMeans code
kmeans = KMeans(n_clusters=i, init='k-means++', random_state=0)
kmeans.fit(hclusdata_numeric)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4
  warnings.warn(
```

KMeans

KMeans(n_clusters=3, random_state=0)

```
km_model = cluster.KMeans(n_clusters=3,init='k-means++',random_state=0)
```

```
ykmeans = km_model.fit_predict(data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4
  warnings.warn(
```

```
data.head()
```

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

```
[ ] data['kclus'] = pd.Series(ykmeans)
```

```
[ ] data.head()
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	kclus
0	0	0	67	2	124670	1	2	1
1	1	1	22	1	150773	1	2	1
2	0	0	49	1	89210	0	0	2
3	0	0	45	1	171565	1	1	0
4	0	0	53	1	149031	1	1	1

Splitting The Dataset Into Dependent And Independent Variable

```
[ ] y = data['kclus']
    x = data.drop(columns=['kclus'],axis=1)
```

```
[ ] import pickle
    pickle.dump(xgb_model,open("xgbmodel1.pkl",'wb'))
```

```
import joblib
joblib.dump(xgb_model,"xgb_model.joblib")
```