Mall Customer Segmentation Data

Bonafide record of work done by

Mithilesh E N (21Z229)
N. Arun Eshwer (21Z232)
Rakkulpravesh M (21Z242)
Venkatprasadh M (21Z268)
Y. Jyaswanth Sai (21Z270)

19Z610 – MACHINE LEARNING LABORATORY

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

BRANCH: COMPUTER SCIENCE AND ENGINEERING

APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

1. PROBLEM STATEMENT

As the owner of a mall, our goal is to effectively identify and understand the target customers to inform strategic marketing initiatives. By analyzing demographic data, psychographic factors, and consumer behavior patterns, we aim to pinpoint the characteristics and preferences of individuals most likely to frequent the mall. Armed with this knowledge, we can provide invaluable insights to the marketing team, enabling them to tailor campaigns and promotions that resonate with the target audience. Through data-driven decision-making and a focus on customer satisfaction, we seek to enhance the overall shopping experience, drive foot traffic, increase sales, and foster long-term customer loyalty, positioning your mall for sustained success in a competitive retail landscape.

2. DATASET DESCRIPTION

We work on a CSV that contains details like, customer id, age, gender, income, marital status, number of children, residence, distance to mall, frequency to mall visits, preferred shopping time, favorite stores, average spending, mode of transportation, purpose of visit, membership status. With all of this vital information we can group customers into a variety of segments. With the help of these segments we can analyze and work on our strategies for marketing appropriately.

3. METHODOLOGY/ MODELS USED

3.1. GAN (Generative Adversarial Network) Image Generation

GANs are ML algorithms for generating new data, like images. They use two neural networks, a generator and a discriminator, trained together. The generator creates fake data from noise, while the discriminator tells real from fake. Over time, the generator improves at making realistic data, while the discriminator gets better at spotting fakes. With the help of this we can generate promotional contents that the user might be interested in and boost the sales.

3.2. NLTK Sentiment Analysis

NLTK is a Python library for natural language processing. It includes tools for sentiment analysis, which involves analyzing the emotional tone of text. NLTK preprocesses text, like tokenization and normalization, then uses machine learning to classify sentiment as positive, negative, or neutral. It offers pre-trained models like Vader, popular for analyzing social media text. By utilizing the above we get to know the customers intention towards certain products, stores and how they would feel about bringing change to the mall in general and react accordingly.

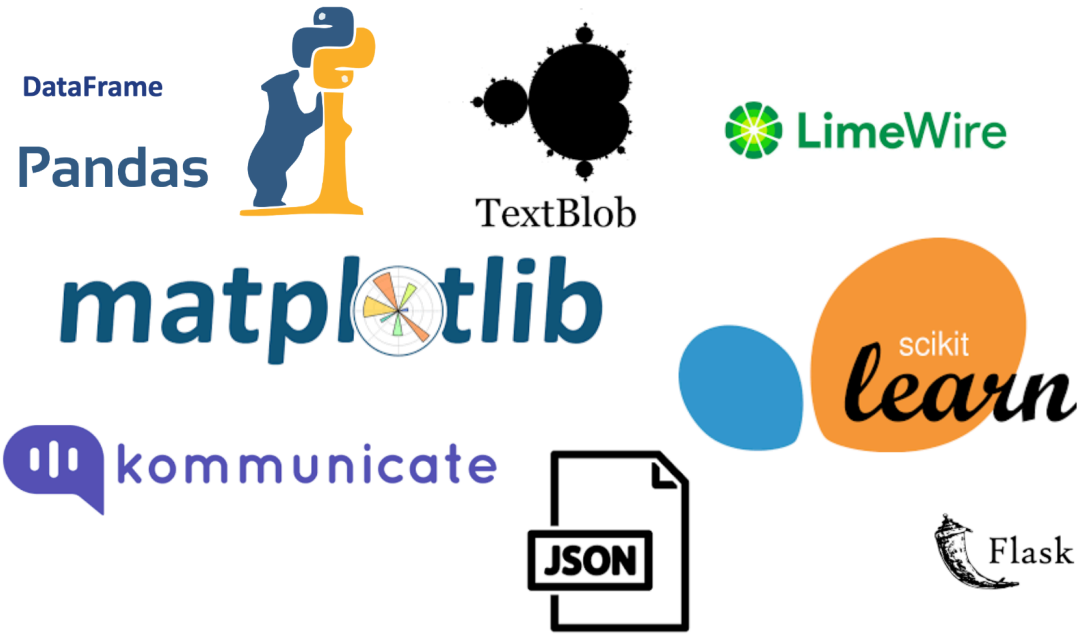
3.3. Customer Classification: k-means clustering

This is the quintessential part of the whole system, k-means clustering is an unsupervised machine learning algorithm used for partitioning a dataset into k clusters based on similarities in the feature space. With the help of this we can group the customers into clusters/ groups based on their age, income, preference to the shopping. By doing so, we understand the customer base as a whole and thereby commit making changes as per their need and increasing the margin.

3.4. Chatbot - Text Understanding Model (Mistral 7B)

This model helps us understand the data briefly and quickly. By doing this data analysis is much faster. We have used a Text Understanding model - Mistral 7B that can take up to 7 Billion parameters. So we can save a lot of data for analysis. For these quick analyses they don't have to be an expert at the analysis. The language used is fairly simple and anyone can understand and use them, thus, by giving a whole picture of the data in a quick span of time this helps in the data analysis even further.

4. TOOLS USED



4. 1. Pandas

Pandas is a strong Python toolkit for data manipulation and analysis. It includes data structures and functions for cleaning, converting, and analyzing structured data, making it an indispensable tool for preparing and examining data sets prior to training machine learning models.

4.2. matplotlib.pyplot
Matplotlib.pyplot is a Python toolkit that allows you to create static, interactive, and publication-quality visualizations. The pyplot package offers a MATLAB-like interface for creating plots and visualizations.

4.3 Textblob
For sentiment analysis of the product reviews posted by user the textblob NLTK is used,which is an efficient text understanding tool

4.4. seaborn
Seaborn is a Python data visualization package built on matplotlib. It provides a high-level interface for constructing visually appealing and useful statistical graphs. Seaborn makes it easier to create sophisticated visualizations by including built-in themes, color palettes, and algorithms for showing statistical correlations.

4.5. sklearn.cluster.KMeans
KMeans is an unsupervised machine learning technique given by scikit-learn that divides data into K clusters based on feature similarity. It is frequently used to group comparable data points in a collection without prior knowledge of the ground truth labels.

4.6. sklearn.preprocessing.StandardScaler
sklearn.preprocessing.StandardScaler is a preprocessing class offered by scikit-learn that standardizes features by eliminating the mean and scaling to unit variance. It is frequently used to preprocess numerical features before training machine learning models, particularly when the features have varied scales or units.

4.7. Limewire GAN
For textual input to image generation the limewire API is used ,which is GAN model that supports natural language input to image conversion and vice versa

4.8. Kommunicate.io
An open source chat model api trained specifically for our dataset and also can be fed with any input data and can be queried for better understanding and analysis of the insights and also serves as a general language chatbot

4.9. Flask
A python based lightweight REST api framework used in our project for the deployment of the web app.

5. CHALLENGES FACED

- The dataset used must contain many columns, else we can't do much analysis.
- Development of the model: k-means and binary classification to fit our requirements.
- Integrating the User Interface and model with the rest of the backend.
- Keeping the User Interface fairly simple as technical persons won't operate them.

6. CONTRIBUTION OF TEAM MEMBERS

Roll No	Name	Contribution
21Z229	Mithilesh E N	Implementation-UI,Flask based backend,integration of frontend and backend,image generation model using limewire GAN,sentiment analysis using textblob,document chat using Kommunicate.io and kompose LLM
21Z232	N. Arun Eshwer	Data Preprocessing, K means clustering of customers, Report Making
21Z242	Rakul Pravesh M	Report Making
21Z268	Venkatprasadh Mari	Charts and Plots in UI, K means clustering of customers
21Z270	Y. Jyaswanth Sai	Dataset Preparation

ANNEXURE I: Code

Flask backend:

```
from flask import Flask, render_template, request, redirect, url_for,flash
from pymongo import MongoClient
from bson.objectid import ObjectId
from flask import Flask, request, jsonify
from textblob import TextBlob
import requests
import pandas as pd
from sklearn.cluster import KMeans

app = Flask(__name__)
client = MongoClient('mongodb://localhost:27017/')
db = client['Chart']
complaints_collection = db['complaints']

customers_df = pd.read_csv('customer.csv')
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(customers_df[['Spending_Score']])
cluster_preferences = {
    0: ['Fashion Boutique', 'Beauty Products Store', 'Coffee Shop'],
    1: ['Electronics Store', 'Gaming Store', 'Tech Gadgets Outlet'],
    2: ['Bookstore', 'Art Supplies Shop', 'Music Store'],
    3: ['Fitness Center', 'Sportswear Store', 'Healthy Food Cafe'],
    4: ['Fine Dining Restaurant', 'Luxury Fashion Store', 'Wine Shop']
}

def recommend_stores(predicted_spending_score):
    customer_cluster = kmeans.predict([[predicted_spending_score]])[0]
    return cluster_preferences[customer_cluster]

@app.route('/predict', methods=['GET','POST'])
def predict():
  if request.method == 'POST':
    # Get customer details from the request
    customer_data = request.json
    age = customer_data['age']
    income = customer_data['income']
    spending_score = customer_data['spending_score']

    # Predict potential spending (for demonstration, just using spending score)
    potential_spending=spending_score
```

```
    # Recommend stores/products based on potential spending
    recommended_stores = recommend_stores(spending_score)

    # Prepare response
    response = {
        'potential_spending': potential_spending,
        'recommended_stores': recommended_stores
    }

    return jsonify(response)
  else:
    return render_template('predict.html')


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

@app.route('/process_cheque', methods=['GET', 'POST'])
def process_cheque():
    if request.method == 'POST':
        cheque_image = request.files['cheque_image']
        if cheque_image:
            cheque_image.save('uploads/cheque_image.jpg')
            flash('Cheque image uploaded successfully')
            return redirect(url_for('home'))
        else:
            flash('Please upload a cheque image')
    return render_template('process_cheque.html')


@app.route('/raise_complaint', methods=['GET', 'POST'])
def raise_complaint():
    if request.method == 'POST':
        issuetype = request.form['issuetype']
        complaint = request.form['complaint']
        name = request.form['name']
        email = request.form['email']

        complaints_collection.insert_one({
            'issuetype': issuetype,
            'complaint': complaint,
            'name': name,
            'email': email,
            'reply': ''
        })
```

```
        # flash('Complaint raised successfully')
        return redirect(url_for('complaint_status'))
    return render_template('raise_complaint.html')

@app.route('/sentiment', methods=['GET','POST'])
def sentiment():
    # Get the review from the request
 if request.method == 'POST':
    review = request.form.get('product-review')

    # Perform sentiment analysis
    blob = TextBlob(review)
    sentiment_score = blob.sentiment.polarity

    # Classify the exhibited emotion
    if sentiment_score > 0:
        emotion = 'positive'
    elif sentiment_score < 0:
        emotion = 'negative'
    else:
        emotion = 'neutral'

    # Return the result
    return jsonify({
        'review': review,
        'sentiment': sentiment_score,
        'emotion': emotion
    })
 else:
    return render_template('sentiment.html')

@app.route('/generate', methods=['GET','POST'])
def generate():
 if request.method == 'POST':

    url = "https://api.limewire.com/api/image/generation"

    # Extract the prompt from the JSON data sent in the request
    prompt = request.json.get('prompt', '')

    # Set up payload and headers
    payload = {
        "prompt": prompt,
        "aspect_ratio": "1:1"
    }
    headers = {
```

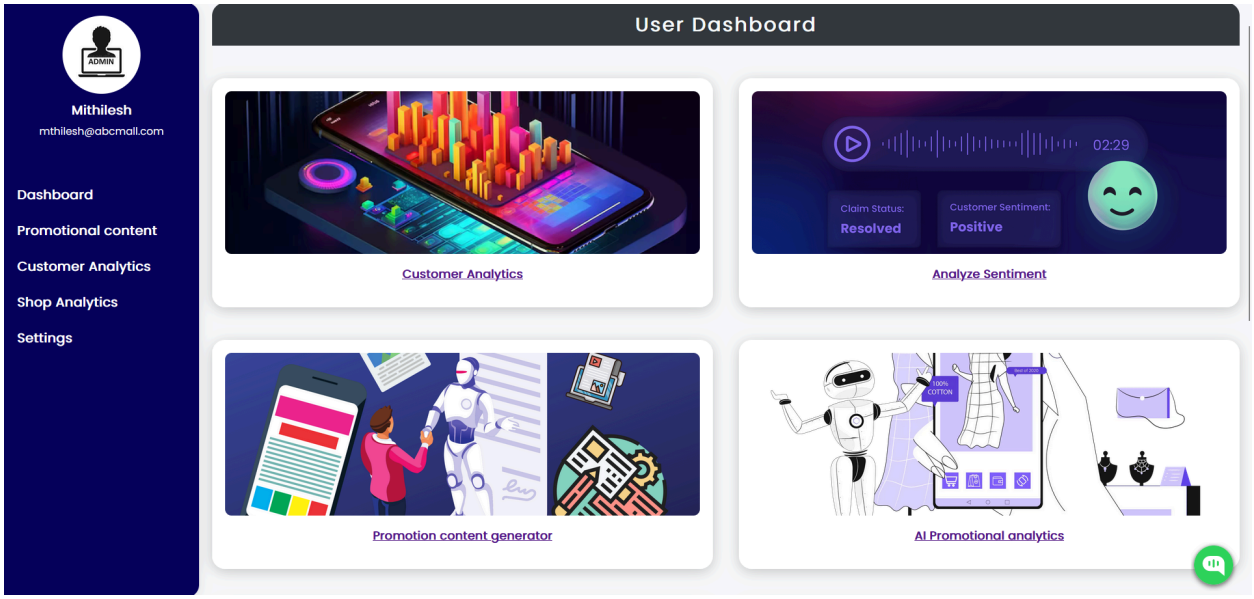
```
        "Content-Type": "application/json",
        "X-Api-Version": "v1",
        "Accept": "application/json",
        "Authorization": "Bearer
lmwr_sk_29rZYc5LdA_yKwChwUETjnklI63VEcyEEBfbjUZE8eGhPpwk"
    }

    # Send a POST request to the Limewire API
    response = requests.post(url, json=payload, headers=headers)

    # Check if the request was successful
    if response.status_code == 200:
        data = response.json()
        return jsonify(data), 200
    else:
        return jsonify({"error": "Failed to generate image"}), response.status_code
 else:
    return render_template('generate.html')

@app.route('/complaint_status')
def complaint_status():
    complaints = complaints_collection.find()
    return render_template('complaint_status.html', complaints=complaints)

@app.route('/reply_complaint/<complaint_id>', methods=['GET', 'POST'])
def reply_complaint(complaint_id):
    if request.method == 'POST':
        reply = request.form['reply']
        complaints_collection.update_one({'_id': ObjectId(complaint_id)}, {'$set': {'reply':
reply}})
        #flash('Complaint replied successfully')
        return redirect(url_for('complaint_status'))
    complaint = complaints_collection.find_one({'_id': ObjectId(complaint_id)})
    return render_template('reply_complaint.html', complaint=complaint)

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


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

@app.route('/shops')
def shops():
```

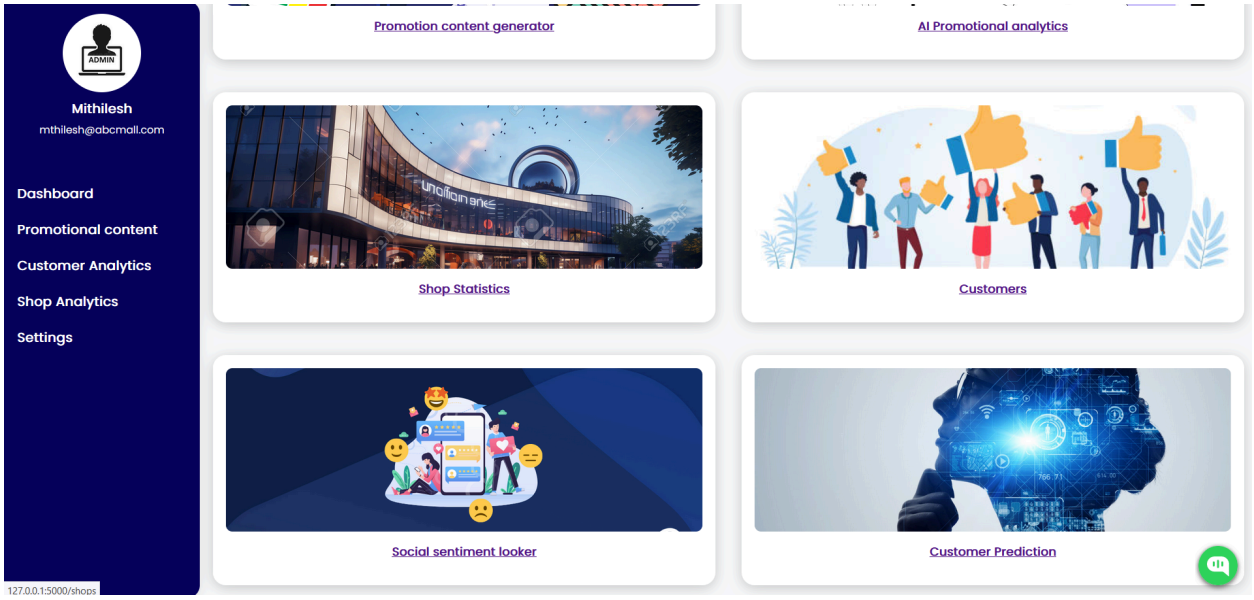
```
    return render_template('shops.html')
@app.route('/chat_ui')
def chat_ui():
    return render_template('chat_ui.html')

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

The complete implementation code is available at github
https://github.com/MithileshEN/cutomer_segmentation

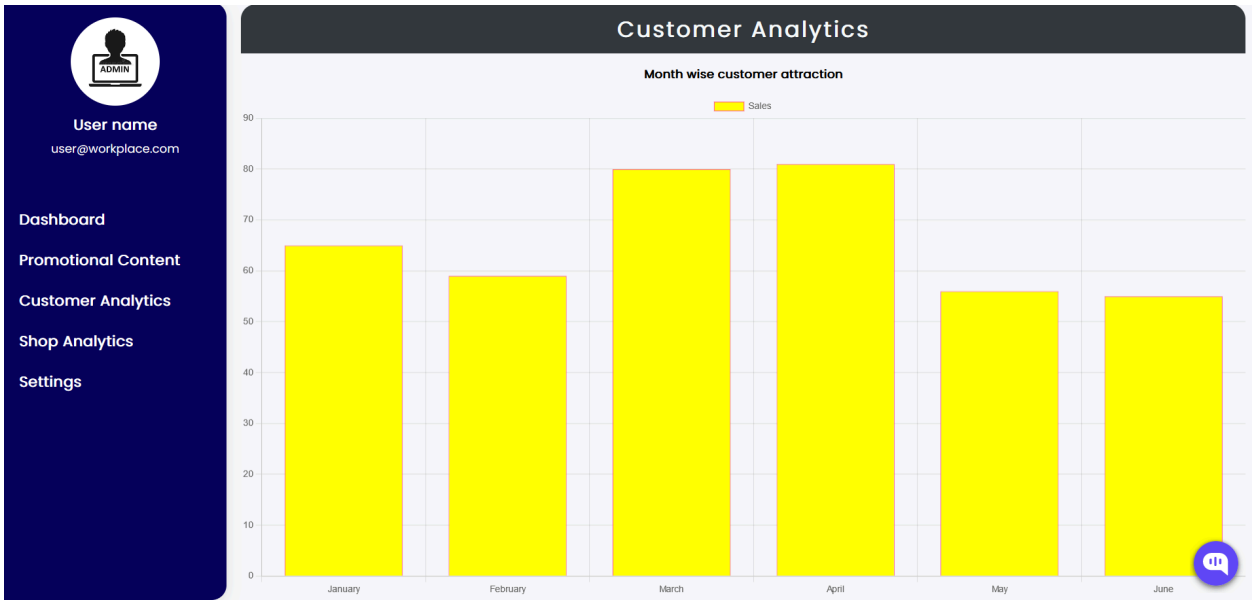
ANNEXURE II: Snapshots of the Output

Dashboard:





Customer analytics



Sentiment analysis

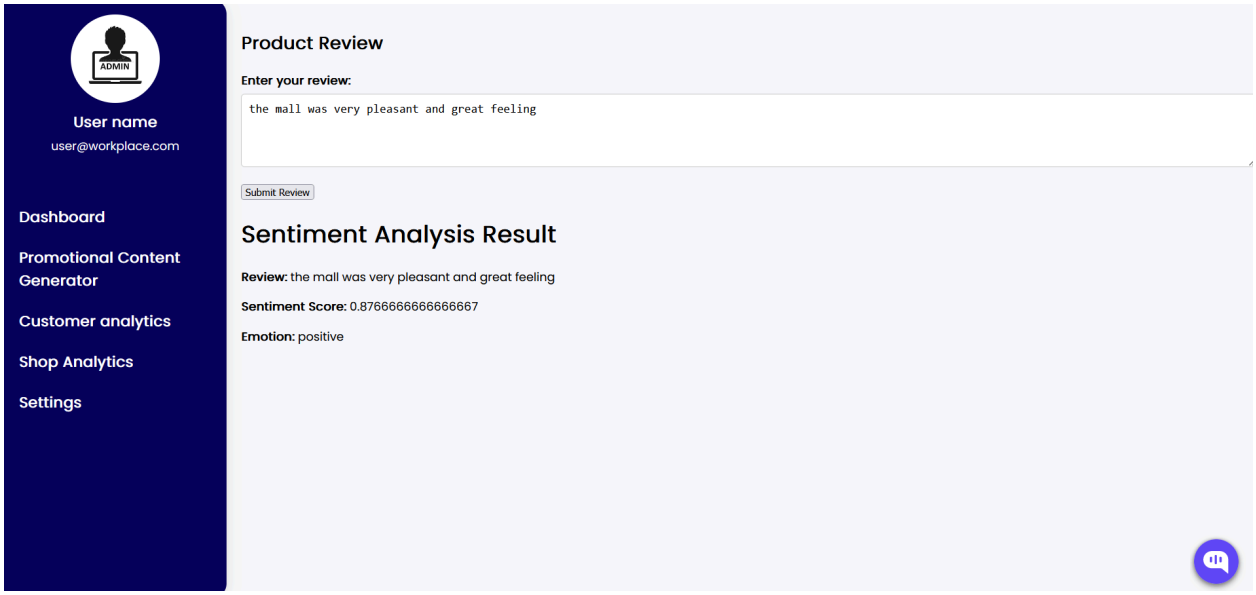
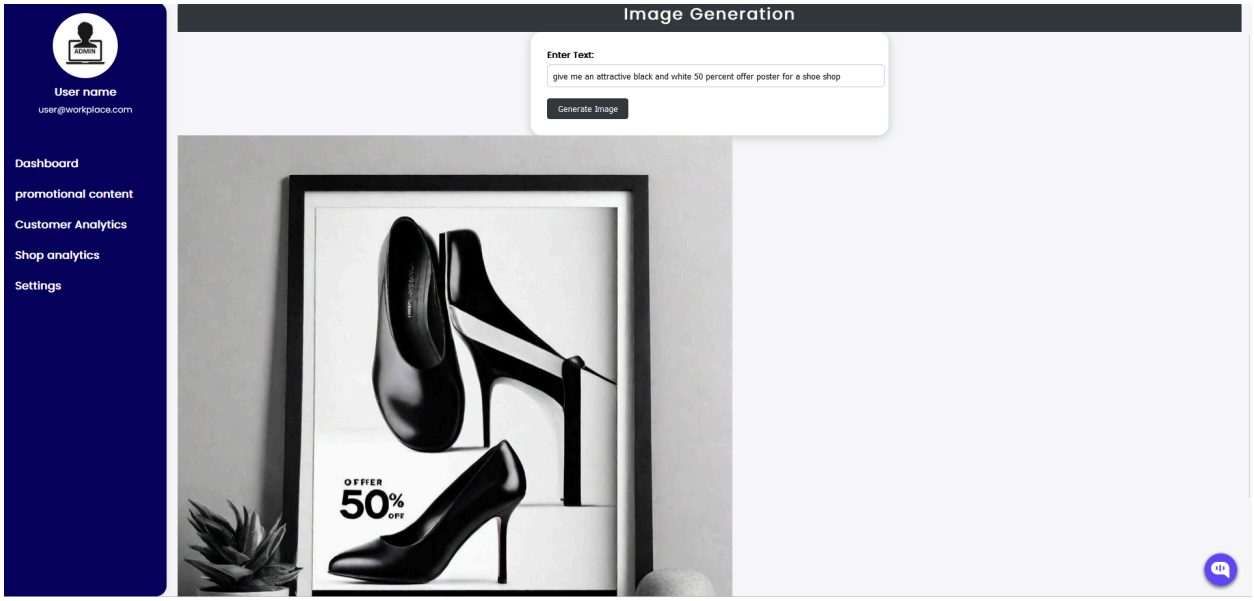
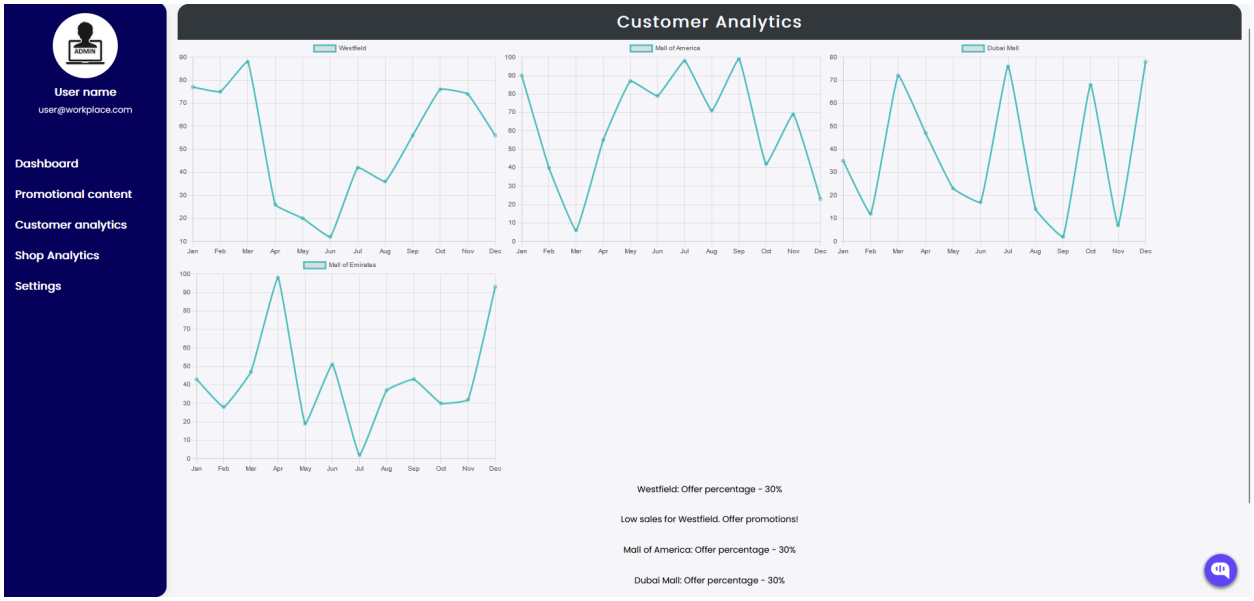


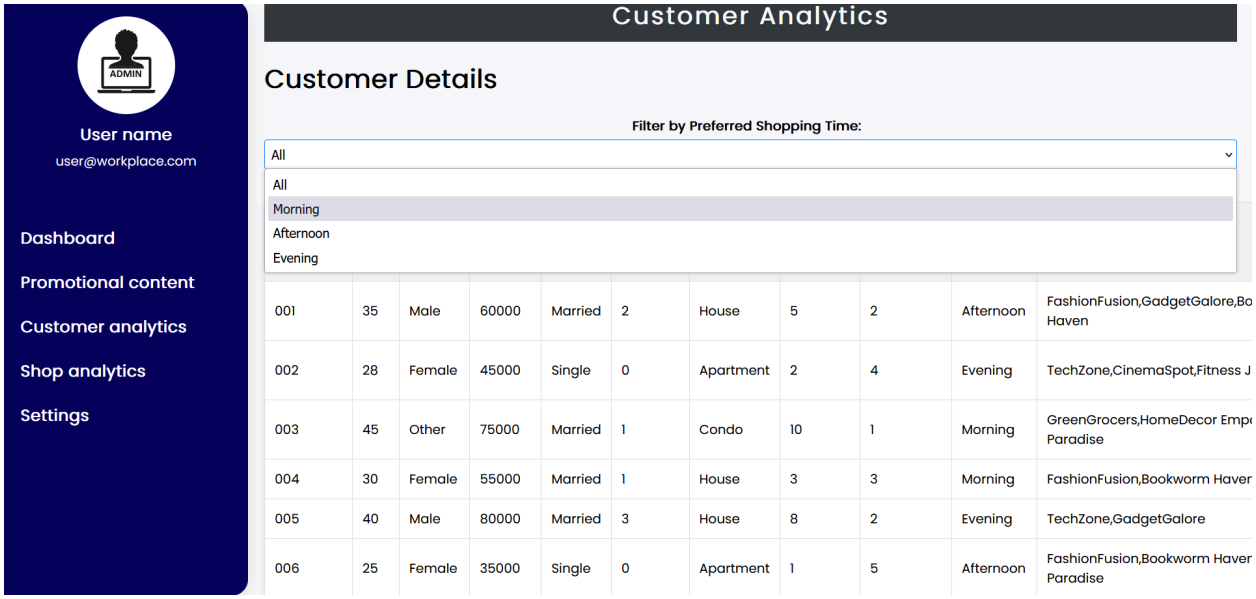
Image generation



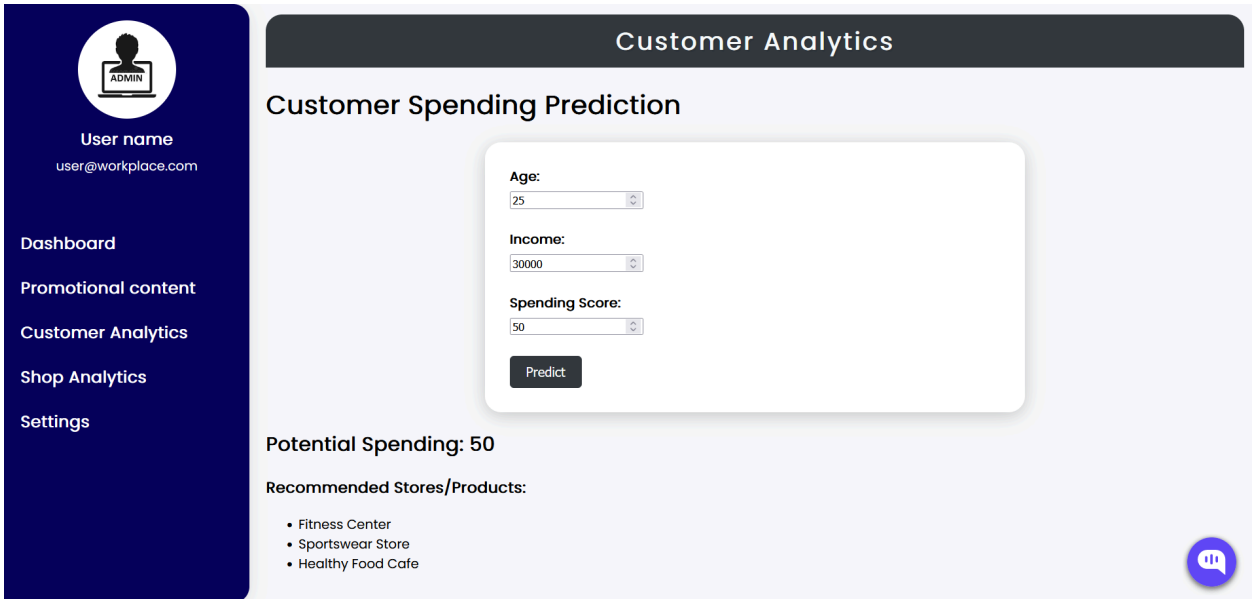
Shops and customer analytics



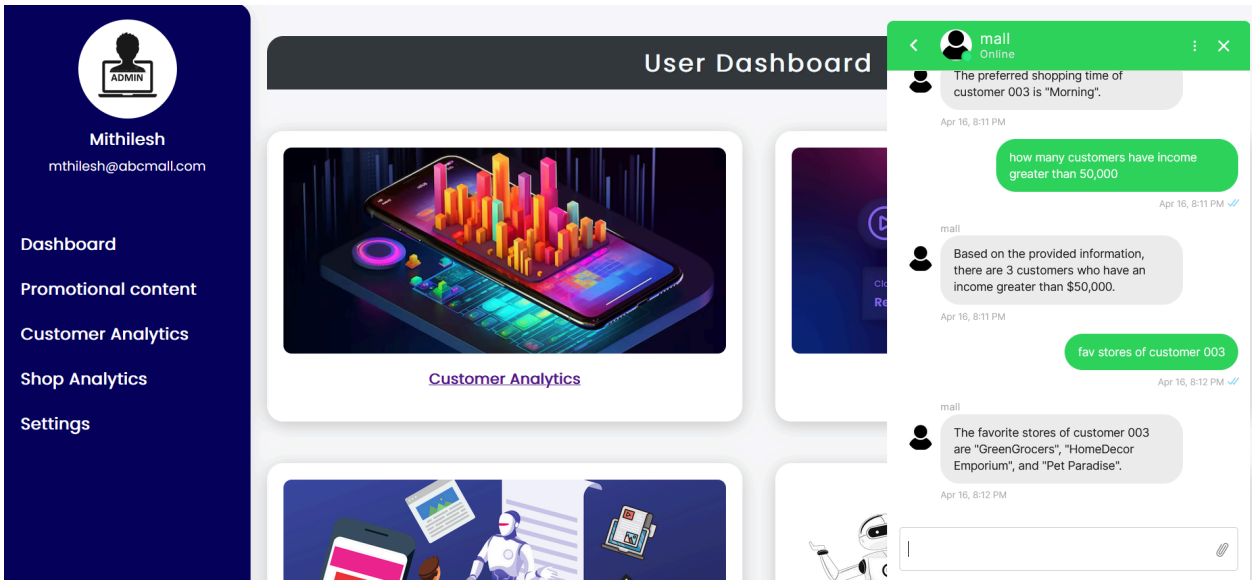
Customer details



Prediction using k means and classification algorithm



Chat bot


ADMIN
Mithilesh
mthilesh@abcmall.com
Dashboard
Promotional content
Customer Analytics
Shop Analytics
Settings
User Dashboard
Customer Analytics
mall
Online
The preferred shopping time of customer 003 is "Morning".
Apr 16, 8:11 PM
how many customers have income greater than 50,000
Apr 16, 8:11 PM
mall
Based on the provided information, there are 3 customers who have an income greater than $50,000.
Apr 16, 8:11 PM
fav stores of customer 003
Apr 16, 8:12 PM
mall
The favorite stores of customer 003 are "GreenGrocers", "HomeDecor Emporium", and "Pet Paradise".
Apr 16, 8:12 PM

References

[1] https://www.anaconda.com/blog/how-to-build-ai-chatbots-with-mistral-and-llama2

[2] https://algoscale.com/blog/how-to-use-gan-to-generate-images/

[3] https://kaushiklade27.medium.com/image-generation-using-generative-adversarial-networks-gans-cd82afd71597

[4] https://medium.com/@ugursavci/step-by-step-customer-segmentation-using-k-means-and-pca-in-python-5733822295b6

[5] https://www.youtube.com/watch?v=SrY0sTJchHE

[6] https://medium.com/@facujallia/stock-classification-using-k-means-clustering-8441f75363de

[7] https://www.youtube.com/watch?v=CMtpzNJbbWQ

[8] https://www.freecodecamp.org/news/binary-classification-made-simple-with-tensorflow/

[9] https://www.analyticsvidhya.com/blog/2021/08/a-beginners-guide-to-machine-learning-binary-classification-of-legendary-pokemon-using-multiple-ml-algorithms/

[10] https://www.behance.net/search/projects/homepage%20ui%20design