Abstract

The prevalence of online shopping made the use of recommendation system more prevalent. In response to this, we developed a comprehensive recommendation system based on the purchased data as part of a mini project. This tool integrates multiple modules, including backend, frontend website and a database to systematically recommend users with the best suitable products.

Our tool provides a user-friendly interface, allowing users to easily navigate through the page and insert URL of the product they want. Through the integration of these modules, our tool offers a robust solution for organizations seeking to enhance their database for recommending users based on their likes and dislikes.

Contents

List of Figures

1. INTRODUCTION

1.1 PROBLEM STATEMENT

Design and implement a recommendation system leveraging purchased data to enhance user experience and drive sales. The recommendation system aims to analyse past purchase behaviour and provide personalized recommendations to users, increasing customer engagement and satisfaction.

1.2 PROJECT SUMMARY INTRODUCTION

1.2.1 PURPOSE

A recommendation system based on purchased data serves a crucial purpose in enhancing user experience and driving business growth in various industries, particularly in e-commerce and retail. By analysing past purchase behaviours, preferences, and patterns of users, such a system can accurately suggest products or services that are likely to be of interest to individual customers. This personalized approach not only improves customer satisfaction by presenting relevant offerings but also increases the likelihood of repeat purchases and customer loyalty. Furthermore, a recommendation system based on purchased data enables businesses to optimize their marketing strategies by targeting specific segments with tailored promotions or product recommendations, thereby maximizing revenue and profitability.

Moreover, beyond its direct impact on sales and customer engagement, a recommendation system powered by purchased data can also provide valuable insights for product development and inventory management. By identifying trends and predicting demand based on historical purchasing data, businesses can make informed decisions about which products to stock, adjust pricing strategies, and even innovate new offerings that align with customer preferences. This data-driven approach enhances operational efficiency and agility, enabling businesses to stay ahead of competitors in a dynamic marketplace. Overall, a recommendation system leveraging purchased data serves as a powerful tool for driving revenue growth, improving customer satisfaction, and fostering innovation across various sectors.

1.2.2 SCOPE

A recommendation system leveraging purchased data holds immense potential across various domains, particularly e-commerce, media streaming, and personalized services. By analysing historical purchasing behaviour, such a system can accurately predict and suggest products or content tailored to individual preferences, enhancing user engagement, satisfaction, and ultimately driving revenue. By incorporating advanced machine learning algorithms, collaborative filtering techniques, and user profiling, the system can continually adapt and refine recommendations, ensuring relevance and effectiveness. Additionally, it can enable targeted marketing campaigns, foster customer loyalty, and optimize inventory management, thereby unlocking significant value for businesses while enriching the user experience.

1.2.3 OVERVIEW

A recommendation system leveraging purchased data employs algorithms to analyse consumer purchasing behaviour and preferences, aiming to offer personalized suggestions to users. Through mining historical transaction data, the system identifies patterns, correlations, and user preferences to generate tailored recommendations. Techniques such as collaborative filtering, content-based filtering, and hybrid approaches are often utilized to enhance recommendation accuracy. By continuously learning from user interactions and refining its models, the system optimizes recommendations over time, fostering increased user engagement, customer satisfaction, and potentially higher sales conversion rates for businesses. Additionally, incorporating factors like item popularity, seasonality, and user demographics can further enhance the relevance and effectiveness of recommendations, contributing to a more dynamic and responsive recommendation ecosystem.

2. LITERATURE REVIEW

2.1 OVERVIEW OF EXISTING RECOMMENDATION SYSTEM

An existing recommendation system based on purchased data utilizes algorithms to analyse past purchasing behaviour, preferences, and interactions of users to generate personalized recommendations. It employs collaborative filtering techniques, content-based filtering, or a hybrid approach to suggest products or services that are likely to be of interest to individual users. The system continuously learns and updates its recommendations based on new data, providing users with tailored suggestions, enhancing their shopping experience, and potentially increasing sales and customer satisfaction for the platform or service provider.

2.2 COMMON RECOMMENDATION SYSTEM AND THEIR IMPLICATIONS

A common recommendation system based on purchased data employs collaborative filtering techniques, where user-item interactions are analysed to predict preferences and suggest relevant items. By leveraging purchase history, this system can identify patterns and similarities among users' behaviours, recommending products that similar users have bought. Implications of such systems include increased sales through personalized recommendations, enhanced user experience by offering relevant suggestions, and improved customer satisfaction and loyalty due to tailored shopping experiences. However, there are potential challenges such as privacy concerns regarding the use of personal data and the risk of creating filter bubbles where users are only exposed to a limited range of items, potentially limiting their exploration of new products. Therefore, it's crucial for businesses to balance personalization with transparency and user control to ensure ethical and effective recommendation systems.

2.3 REVIEW OF RELEVANT TECHNOLOGIES AND METHODOLOGIES

Recommendation systems based on purchased data rely on a variety of methodologies and technologies to provide personalized suggestions to users. One common approach is collaborative filtering, which analyses past purchase behaviour to identify patterns and recommend items that similar users have bought. This method often employs techniques such as user-item matrix factorization or nearest neighbour algorithms to generate recommendations. Another widely used technique is content-based filtering, which recommends items based on their attributes and features, leveraging natural language processing or image recognition to understand item characteristics and match them to user preferences. Additionally, hybrid recommendation systems combine collaborative and content-based approaches to enhance recommendation accuracy and coverage. Furthermore, advancements in deep learning have enabled the development of more sophisticated recommendation systems capable of capturing intricate user-item interactions and generating highly personalized recommendations. Overall, the landscape of recommendation system methodologies for purchased data is diverse and constantly evolving, with ongoing research and innovation driving improvements in recommendation accuracy and user satisfaction.

3. METHODOLGY

3.1 DESCRIPTION OF THE TOOL'S ARCHITECTURE & DESIGN

3.1.1 ARCHITECTURE OVERVIEW

A recommendation system architecture based on purchased data typically involves several key components working together to analyse user behaviour, preferences, and historical purchase data to provide personalized recommendations. Here's a high-level overview of such an architecture:

- **Data Ingestion Layer:** This layer is responsible for collecting and ingesting data from various sources such as e-commerce platforms, customer relationship management (CRM) systems, and other relevant databases.
- **Data Storage Layer:** The ingested data is stored in a scalable and efficient storage system such as a data warehouse or a data lake.
- **Data Processing Layer:** This layer performs data processing tasks such as data cleaning, feature engineering, and data aggregation.
- **Machine Learning Models:** Recommendation algorithms are at the heart of the recommendation system. These models analyse user behaviour and historical purchase data to generate personalized recommendations. Common approaches include collaborative filtering, content-based filtering, and hybrid models that combine multiple techniques.
- **Real-time Recommendation Engine:** For real-time recommendations, a recommendation engine continuously processes user interactions and updates recommendations in near realtime.
- **Scalability and Performance:** As the system needs to handle large volumes of data and serve recommendations to a growing user base, scalability and performance are critical considerations.
- **Monitoring and Analytics:** Monitoring tools and analytics dashboards are essential for tracking the performance of the recommendation system, including metrics such as clickthrough rates, conversion rates, and user engagement. Monitoring helps detect anomalies, identify performance bottlenecks, and guide ongoing optimization efforts.

3.1.2 DESIGN PRINCIPLES

- **Personalization:** Tailor recommendations based on individual preferences, past purchase behaviour, and demographic information. By understanding each user's unique tastes and interests, the system can suggest relevant products or services that are more likely to resonate with them.
- **Transparency:** Ensure transparency in the recommendation process by providing clear explanations of why certain items are being suggested. Users should understand the factors influencing recommendations, such as purchase history, ratings, and item attributes. This helps build trust and enables users to better evaluate the suggestions provided.
- **Dynamic Adaptation:** Implement algorithms that continuously adapt and evolve based on user interactions and feedback. As preferences change over time, the recommendation system should dynamically adjust to reflect these shifts, ensuring that recommendations

remain relevant and engaging. This may involve employing machine learning techniques to analyse and respond to user behaviour in real-time.

3.1.3 DATA FLOW

A recommendation system based on purchase data typically involves several key steps in its data flow. Initially, it begins with the collection of purchase data from various sources such as transaction records, online orders, or point-of-sale systems. This data is then pre-processed to extract relevant features such as item IDs, purchase quantities, and timestamps. Next, the data undergoes analysis and modelling to understand patterns and relationships, often utilizing techniques like collaborative filtering, content-based filtering, or hybrid methods. These models generate personalized recommendations by predicting items a user might be interested in based on their past purchases or similar users' behaviour. Finally, the recommendations are delivered to users through various channels such as websites, mobile apps, or email, completing the data flow loop of the recommendation system. Throughout this process, ongoing monitoring and evaluation ensure the system remains accurate and effective in providing valuable recommendations to users.

3.2 MODULE'S FUNCTIONALITY

A recommendation system based on purchased data typically involves several key modules to effectively analyse user behaviour and preferences. Here's a breakdown of the functionality each module might provide:

Data Collection Module:

- This module gathers relevant data related to user purchases, including product details, purchase history, timestamps, and user identifiers.
- Data may be collected from various sources such as e-commerce platforms, point-ofsale systems, customer databases, etc.
- Ensure data integrity, accuracy, and compliance with privacy regulations.

Data Preprocessing Module:

- Clean and preprocess the collected data to handle missing values, outliers, and inconsistencies.
- Transform the data into a suitable format for analysis, such as user-item matrices or transactional datasets.

Feature Engineering Module:

- Extract meaningful features from the pre-processed data that capture user behaviour and item characteristics.
- Features might include product attributes (e.g., category, brand, price), user demographics, temporal factors, etc.
- Dimensionality reduction techniques like PCA or feature selection methods may be applied to reduce computational complexity.

Modelling Module:

- Utilize machine learning algorithms or statistical models to analyse user behaviour and generate recommendations.
- Common approaches include collaborative filtering, content-based filtering, and hybrid methods combining both.
- Train models using techniques like matrix factorization, deep learning, or similarity based algorithms.
- Evaluate model performance using metrics such as precision, recall, and accuracy.

Recommendation Generation Module:

- Generate personalized recommendations for users based on their historical purchase data and preferences.
- Use trained models to predict items that users are likely to purchase or find relevant.
- Rank recommendations based on predicted preferences or relevance scores.

Post-processing Module:

- Apply any necessary post-processing steps to refine recommendations or improve user experience.
- Filter out items that users have already purchased or shown disinterest in.
- Incorporate business rules or constraints, such as promotional offers, inventory constraints, or diversity requirements.

Deployment Module:

- Deploy the recommendation system in a production environment, integrated with the target platform (e.g., e-commerce website, mobile app).
- Monitor system performance, user engagement, and feedback to continuously improve recommendation quality.
- Scale the system to handle increasing volumes of data and user traffic.

Feedback and Iteration Module:

- Collect user feedback and interaction data to iteratively improve the recommendation models.
- Incorporate user feedback into the modelling process through techniques like collaborative filtering with feedback or reinforcement learning.
- Continuously update and refine the recommendation system to adapt to changing user preferences and market dynamics.

4. SYSTEM OVERVIEW

4.1 USER INTERFACE

The user interface for the recommendation system based on purchased data offers a streamlined experience focused on personalized suggestions. Upon accessing the platform, users are greeted with a clean and intuitive layout displaying recommended products or services tailored to their purchasing history. Clear navigation tools enable easy exploration of categories and filtering options for refining recommendations. Interactive elements allow users to provide feedback on suggested items, further enhancing the system's ability to deliver relevant suggestions. Additionally, a user-friendly dashboard provides insights into past purchases and preferences, empowering users to make informed decisions. Overall, the interface prioritizes simplicity, personalization, and usability to optimize the user experience.

4.2 MODULES

A recommendation system for leveraging purchase data typically comprises several key modules. Firstly, data ingestion and preprocessing modules are crucial for gathering and cleaning transactional data. Next, feature engineering modules transform raw data into meaningful features, such as customer preferences or product similarities. Following this, the core recommendation engine employs algorithms like collaborative filtering or content-based filtering to generate personalized recommendations based on user behaviour and item attributes. Additionally, evaluation modules assess the performance of recommendation algorithms, while deployment modules ensure seamless integration into platforms for real time recommendations. Finally, feedback loops continuously update and refine the system based on user interactions, ensuring ongoing relevance and effectiveness.

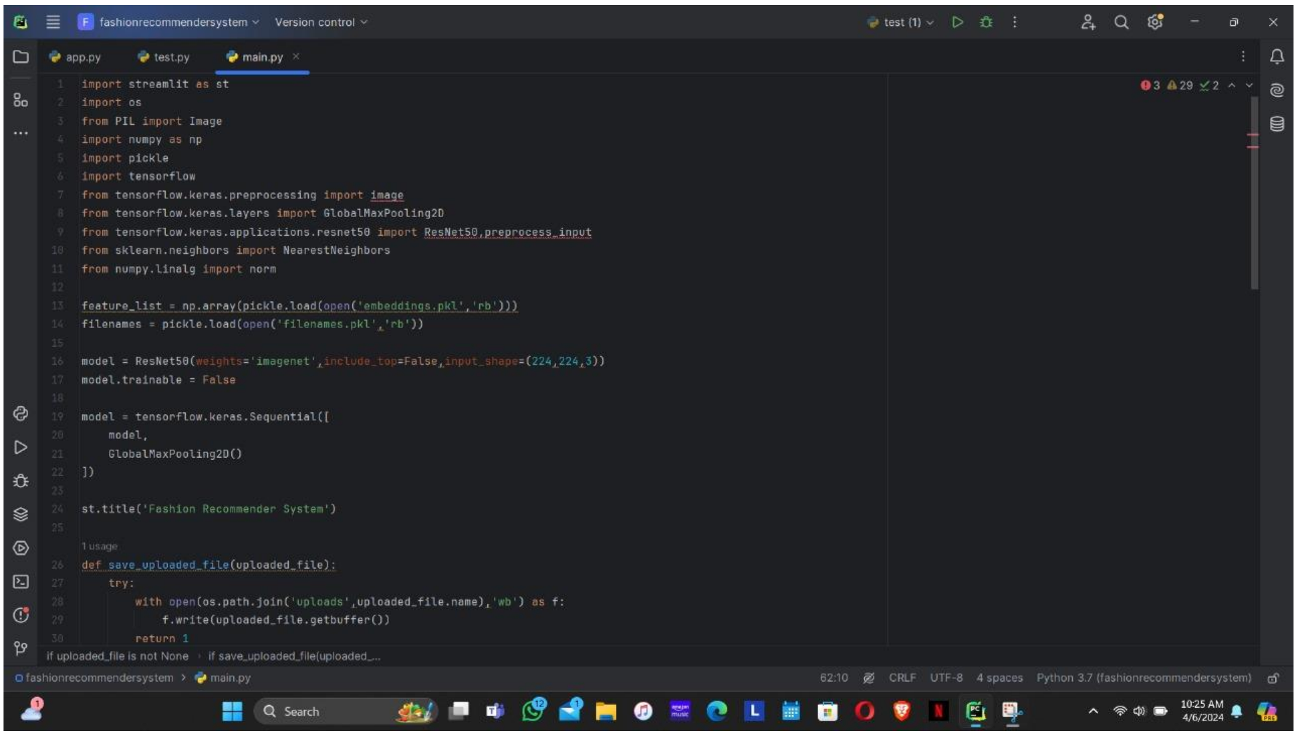
4.3 EXTERNAL TOOLS AND LIBRARIRES

For a recommendation system based on purchased data, several external tools and libraries can be utilized. Apache Mahout offers scalable machine learning algorithms for collaborative filtering, content-based filtering, and clustering. TensorFlow and PyTorch are powerful deep learning frameworks enabling the implementation of advanced recommendation models such as neural collaborative filtering or deep learning-based content recommenders. Additionally, scikit-learn provides a wide range of machine learning algorithms suitable for recommendation tasks, including support vector machines and decision trees. Furthermore, for data preprocessing and visualization, libraries such as Pandas, NumPy, and Matplotlib can be employed to clean, transform, and visualize the purchased data before feeding it into the recommendation models.

4.4 REPORTING AND ANALYSIS

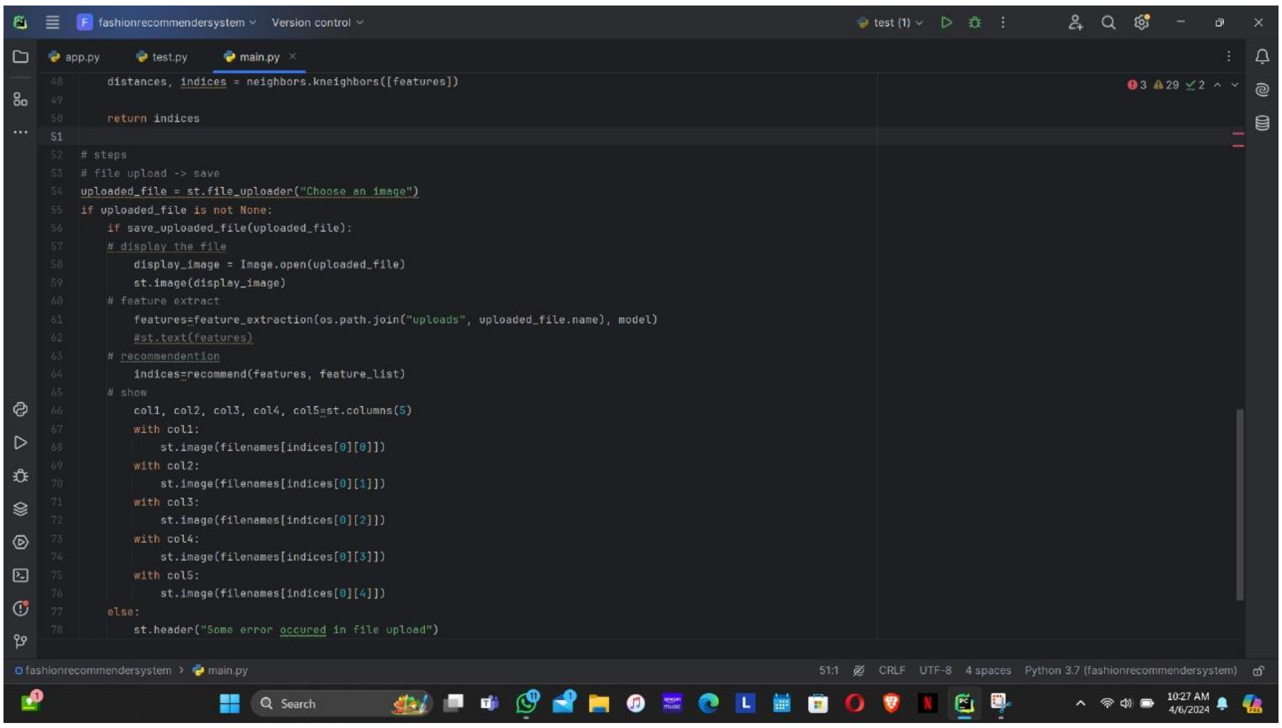
The reporting and analysis for the recommendation system based on purchased data involves a comprehensive approach aimed at extracting actionable insights. Through meticulous examination of purchasing patterns, user behaviour, and product interactions, the system identifies trends and correlations to enhance personalized recommendations. Utilizing advanced analytics techniques such as collaborative filtering and machine learning algorithms, it continually refines its suggestions to optimize user experience and drive increased sales and customer satisfaction. Regular reporting provides stakeholders with clear visibility into system performance and effectiveness, facilitating informed decision-making and strategic adjustments to further enhance the recommendation engine's efficacy.

5. RESULTS



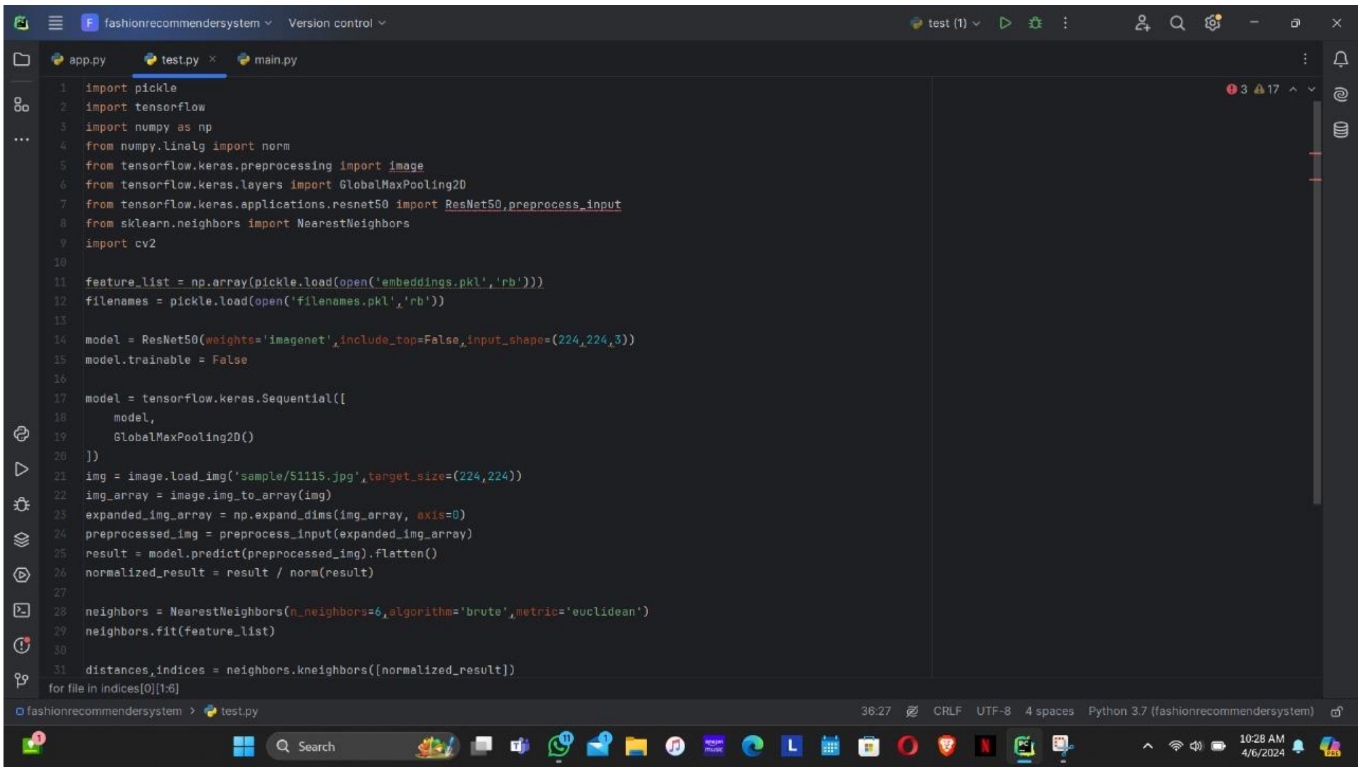
```python
import streamlit as st
import os
from PIL import Image
import numpy as np
import pickle
import tensorflow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
from sklearn.neighbors import NearestNeighbors
from numpy.linalg import norm

feature_list = np.array(pickle.load(open('embeddings.pkl','rb')))
filenames = pickle.load(open('filenames.pkl','rb'))

model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

st.title('Fashion Recommender System')

def save_uploaded_file(uploaded_file):
    try:
        with open(os.path.join('uploads',uploaded_file.name),'wb') as f:
            f.write(uploaded_file.getbuffer())
        return 1
```

Figure 1: MAIN FOR WEB APP CODE



```python
    distances, indices = neighbors.kneighbors([features])

    return indices

# steps
# file upload -> save
uploaded_file = st.file_uploader("Choose an image")
if uploaded_file is not None:
    if save_uploaded_file(uploaded_file):
    # display the file
        display_image = Image.open(uploaded_file)
        st.image(display_image)
    # feature extract
        features=feature_extraction(os.path.join("uploads", uploaded_file.name), model)
        #st.text(features)
    # recommendention
        indices=recommend(features, feature_list)
    # show
        col1, col2, col3, col4, col5=st.columns(5)
        with col1:
            st.image(filenames[indices[0][0]])
        with col2:
            st.image(filenames[indices[0][1]])
        with col3:
            st.image(filenames[indices[0][2]])
        with col4:
            st.image(filenames[indices[0][3]])
        with col5:
            st.image(filenames[indices[0][4]])
    else:
        st.header("Some error occured in file upload")
```

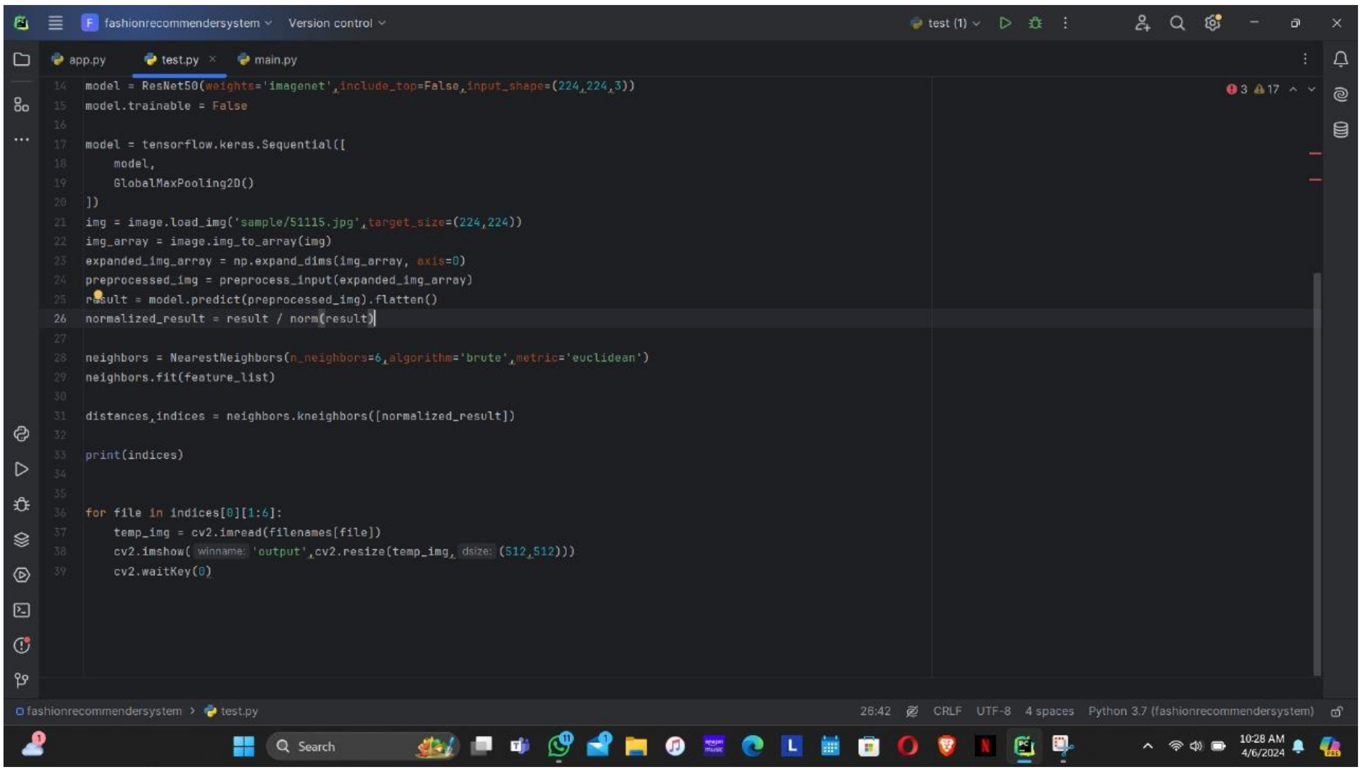
Figure 2: MAIN FOR WEB APP CODE



fashionrecommendersystem
Version control
test (1)
app.py
test.py
main.py
for file in indices[0][1:6]
fashionrecommendersystem › test.py
36:27 CRLF UTF-8 4 spaces Python 3.7 (fashionrecommendersystem)

```python
import pickle
import tensorflow
import numpy as np
from numpy.linalg import norm
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
from sklearn.neighbors import NearestNeighbors
import cv2

feature_list = np.array(pickle.load(open('embeddings.pkl','rb')))
filenames = pickle.load(open('filenames.pkl','rb'))

model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])
img = image.load_img('sample/51115.jpg',target_size=(224,224))
img_array = image.img_to_array(img)
expanded_img_array = np.expand_dims(img_array, axis=0)
preprocessed_img = preprocess_input(expanded_img_array)
result = model.predict(preprocessed_img).flatten()
normalized_result = result / norm(result)

neighbors = NearestNeighbors(n_neighbors=6,algorithm='brute',metric='euclidean')
neighbors.fit(feature_list)

distances,indices = neighbors.kneighbors([normalized_result])
```

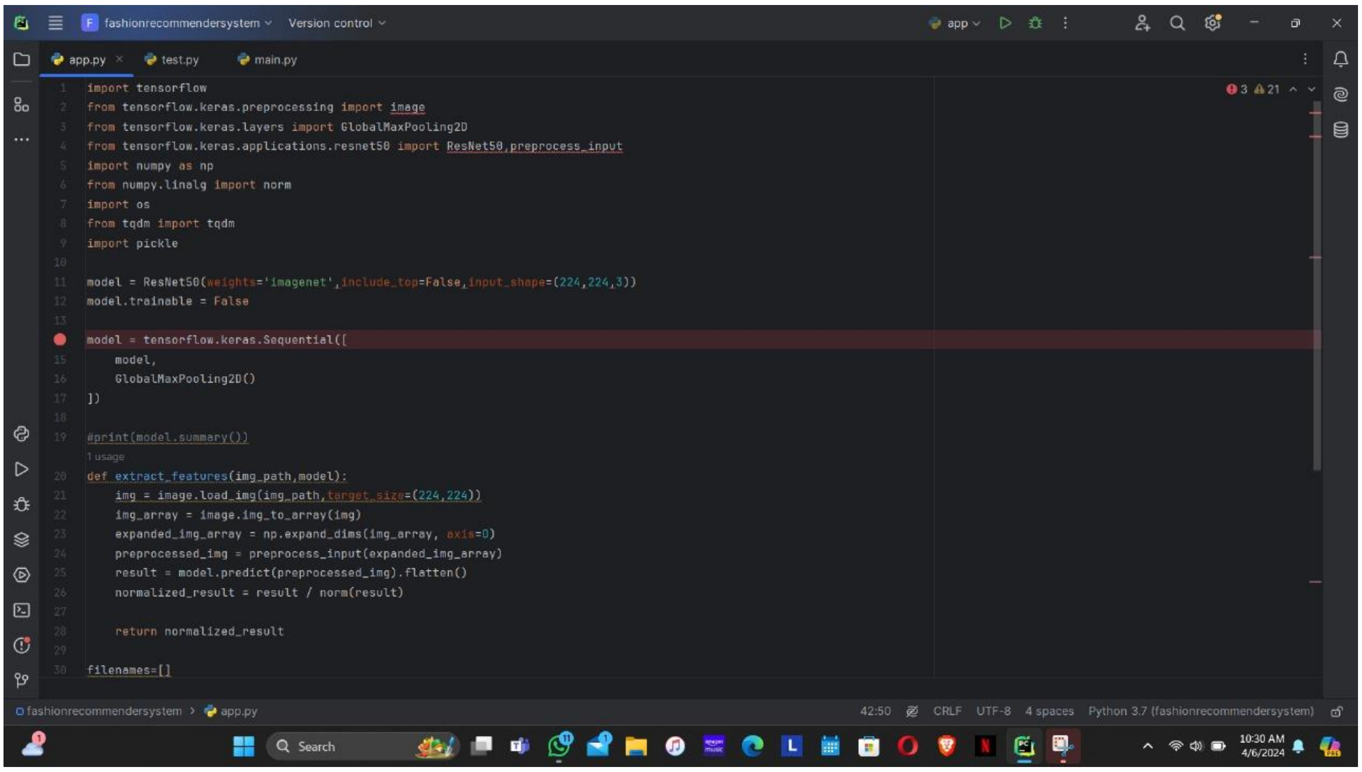
Figure 3: TESTING THE MODEL



fashionrecommendersystem
Version control
test (1)
app.py
test.py
main.py
fashionrecommendersystem › test.py
26:42 CRLF UTF-8 4 spaces Python 3.7 (fashionrecommendersystem)

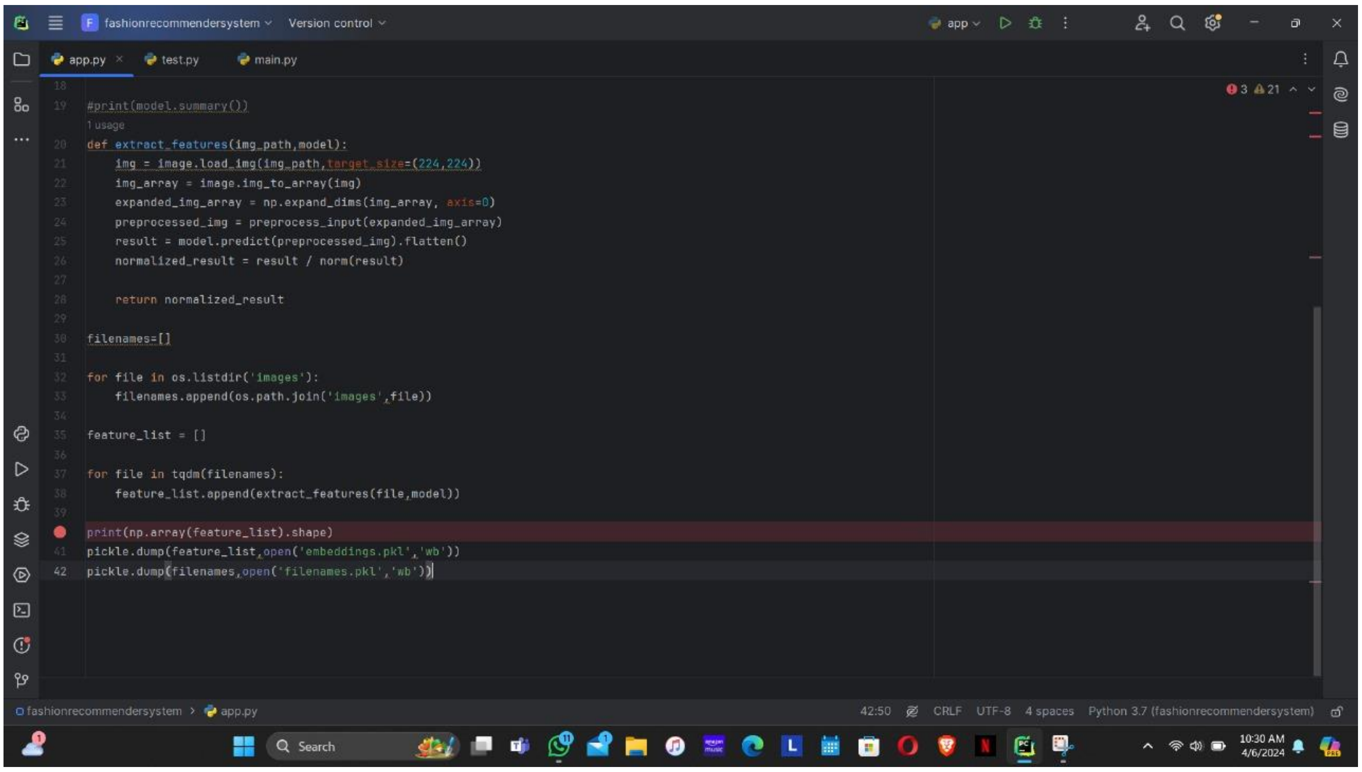
```python
model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])
img = image.load_img('sample/51115.jpg',target_size=(224,224))
img_array = image.img_to_array(img)
expanded_img_array = np.expand_dims(img_array, axis=0)
preprocessed_img = preprocess_input(expanded_img_array)
result = model.predict(preprocessed_img).flatten()
normalized_result = result / norm(result)

neighbors = NearestNeighbors(n_neighbors=6,algorithm='brute',metric='euclidean')
neighbors.fit(feature_list)

distances,indices = neighbors.kneighbors([normalized_result])

print(indices)


for file in indices[0][1:6]:
    temp_img = cv2.imread(filenames[file])
    cv2.imshow( winname: 'output',cv2.resize(temp_img, dsize: (512,512)))
    cv2.waitKey(0)
```

Figure 4: TESTING THE MODEL



```python
import tensorflow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
import numpy as np
from numpy.linalg import norm
import os
from tqdm import tqdm
import pickle

model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

#print(model.summary())

def extract_features(img_path,model):
    img = image.load_img(img_path,target_size=(224,224))
    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expanded_img_array)
    result = model.predict(preprocessed_img).flatten()
    normalized_result = result / norm(result)

    return normalized_result

filenames=[]
```

Figure 5: MODEL CREATION



```python
#print(model.summary())

def extract_features(img_path,model):
    img = image.load_img(img_path,target_size=(224,224))
    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expanded_img_array)
    result = model.predict(preprocessed_img).flatten()
    normalized_result = result / norm(result)

    return normalized_result

filenames=[]

for file in os.listdir('images'):
    filenames.append(os.path.join('images',file))

feature_list = []

for file in tqdm(filenames):
    feature_list.append(extract_features(file,model))

print(np.array(feature_list).shape)
pickle.dump(feature_list,open('embeddings.pkl','wb'))
pickle.dump(filenames,open('filenames.pkl','wb'))
```

Figure 6: MODEL CREATION

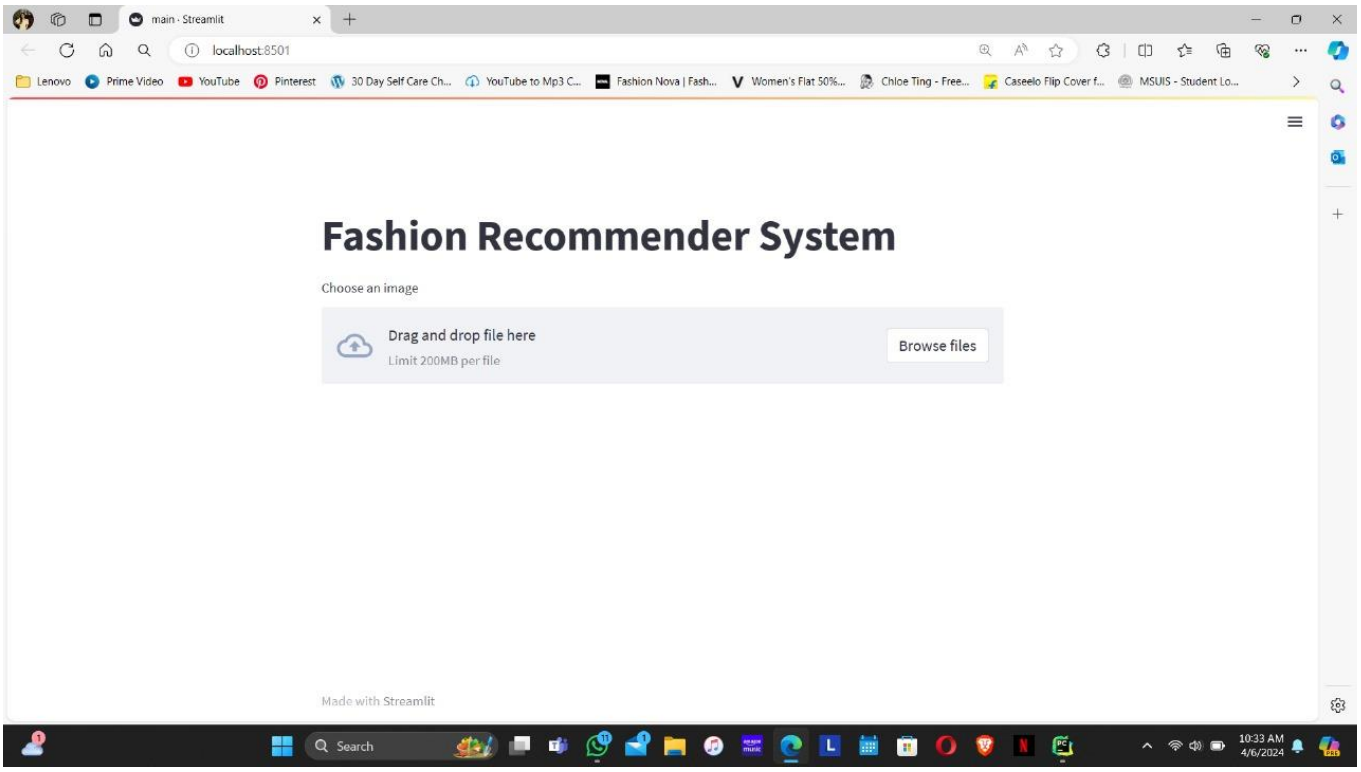


Figure 7: WEB PAGE

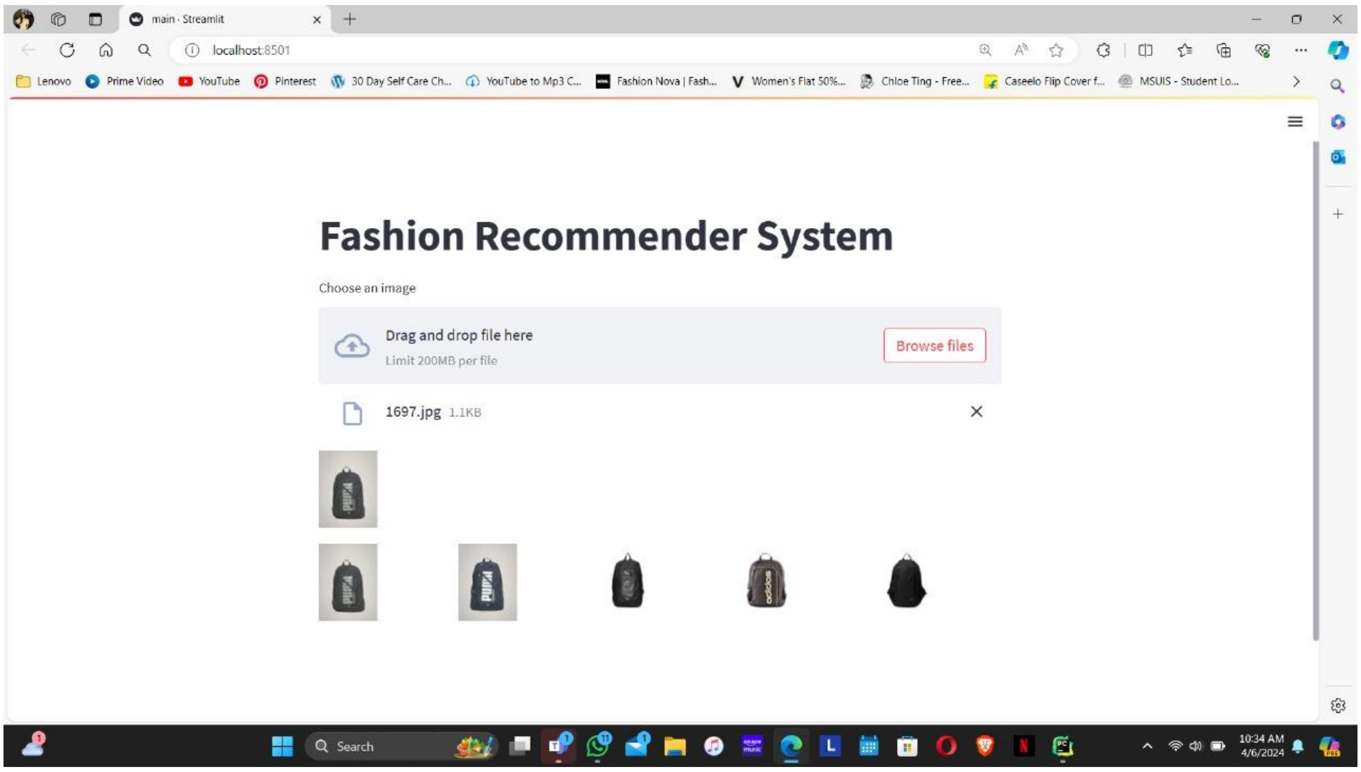
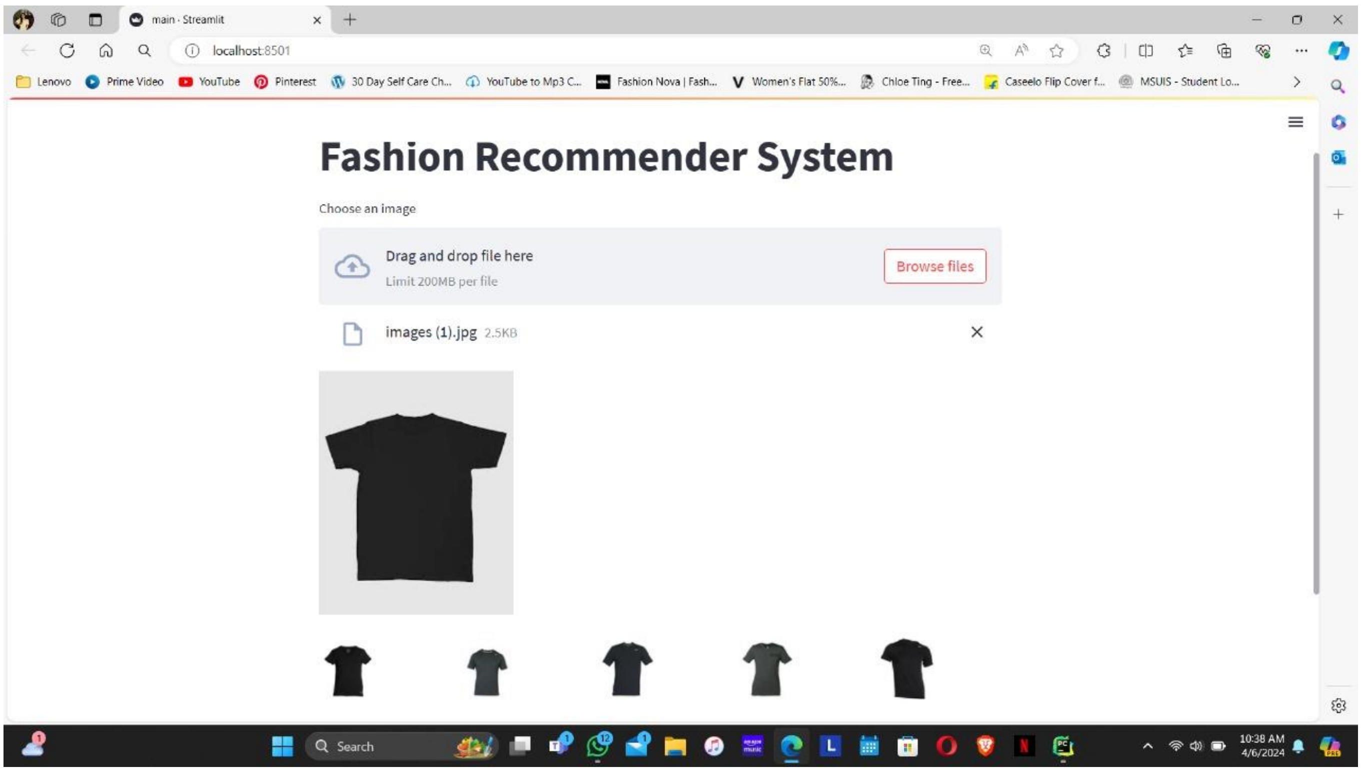


Figure 8: SEARCH RESULT FOR BAG



Fashion Recommender System

Choose an image

Drag and drop file here
Limit 200MB per file

Browse files

images (1).jpg 2.5KB

Figure 9: SEARCH RESULT FOR T-SHIRT

6. CONCLUSION

In conclusion, leveraging purchased data for recommendation systems presents a potent opportunity to enhance user experience and drive business success. By analysing historical purchases, preferences, and behaviours, these systems can generate personalized recommendations that cater to individual tastes and needs, thereby increasing customer satisfaction and loyalty. Furthermore, through continuous refinement and adaptation using machine learning algorithms, such systems can evolve to deliver increasingly accurate and relevant suggestions over time, ultimately fostering stronger customer engagement and maximizing revenue potential for businesses.

7. REFFRENCES

1. https://www.anderson.ucla.edu/faculty/anand.bodapati/Recommendation-Systems-with-Purchase-Data.pdf
2. https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset
3. https://github.com/campusx-official/fashion-recommender-system