E-COMMERCE PRODUCT RECOMMENDATION ENGINE

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

AUM SAH(RA2011003010872)
MOKKAPATI SHIV PRASAD
SAHIL(RA2011003010879)
SHEETAL JATAV(RA2011003010885)

Under the guidance of
Mrs. M. RAJALAKSHMI
Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **“E-COMMERCE PRODUCT RECOMMENDATION ENGINE”** is the bona fide work of **Aum Sah(RA2011003010872), Mokkapati Shiv Prasad Sahil(RA2011003010879), Sheetal Jatav(RA2011003010885)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mrs. M. Rajalakshmi	Dr. M. Pushpalatha
Assistant Professor	**HEAD OF THE DEPARTMENT**
Department of Computing Technologies	Professor & Head
	Department of Computing Technologies

ABSTRACT

The E-commerce industry has gained immense popularity over the years and has revolutionized the way people shop. One of the key challenges for an e-commerce business is to recommend products that are most relevant to its customers. A product recommendation engine is an intelligent system that analyzes the customer’s shopping behavior and provides them with personalized product recommendations.

This project aims to develop an e-commerce product recommendation engine that utilizes collaborative filtering techniques to provide personalized product recommendations to users. The system will collect and analyze user data such as the user's past purchase history, viewed products, and product ratings. The engine will then compare the user’s data with the data of other users to identify similar users and recommend products that were purchased by similar users.

The project will be developed using the Python programming language and will utilize libraries such as NumPy, Pandas, and Scikit-learn. The final system will be evaluated using metrics such as precision, recall, and F1 score to measure its accuracy and effectiveness in providing relevant product recommendations.

The proposed system can be integrated into any e-commerce platform to enhance the user’s shopping experience, increase customer satisfaction, and ultimately increase revenue for the businesses.

TABLE OF CONTENTS

ABSTRACT **iii**

TABLE OF CONTENTS **iv**

1 INTRODUCTION **5**

2 LITERATURE SURVEY **7**

3 SYSTEM ARCHITECTURE AND DESIGN **8**

3.1 Architecture of the recommendation 9

4 METHODOLOGY **10**

4.1 Methodological Steps 11

5 CODING AND TESTING **12**

6 SCREENSHOTS AND RESULTS **15**

7 CONCLUSION AND FUTURE ENHANCEMENT **17**

7.1 Conclusion

7.2 Future Enhancement

REFERENCES **19**

CHAPTER 1

INTRODUCTION

The e-commerce industry has witnessed tremendous growth in recent years and is projected to continue growing at a rapid pace. With the increasing number of online shoppers, the competition in the e-commerce market has become fierce. One of the key challenges for any e-commerce business is to provide a personalized shopping experience to its customers. Personalized product recommendations play a crucial role in enhancing the user’s shopping experience and increasing customer loyalty.

An e-commerce product recommendation engine is an intelligent system that leverages customer data to provide personalized product recommendations. The recommendation engine analyzes the user's purchase history, search queries, viewed products, and other customer data to identify patterns and preferences. Based on the analysis, the system recommends products that are most relevant to the user.

Collaborative filtering is one of the most popular techniques used in e-commerce product recommendation engines. Collaborative filtering is a machine learning technique that analyzes the behavior of similar users and recommends products based on the similarity of user behavior.

This project aims to develop an e-commerce product recommendation engine using collaborative filtering techniques. The system will be developed using the Python programming language and will utilize libraries such as NumPy, Pandas, and Scikit-learn. The final system will be evaluated using metrics such as precision, recall, and F1 score to measure its accuracy and effectiveness in providing relevant product recommendations.

The proposed system has the potential to enhance the user's shopping experience, increase customer satisfaction, and ultimately increase revenue for the e-commerce business.

CHAPTER 2

LITERATURE SURVEY

A literature survey on e-commerce product recommendation engines reveals that several studies have been conducted in this field. Most of the research studies have focused on developing personalized product recommendation engines using machine learning and deep learning techniques. Here are some of the key findings from the literature survey:

Collaborative filtering is one of the most popular techniques used in e-commerce product recommendation engines. Collaborative filtering algorithms analyze the behavior of similar users and recommend products based on the similarity of user behavior.

Content-based filtering is another popular technique used in e-commerce product recommendation engines. Content-based filtering algorithms analyze the characteristics of products and recommend products that are similar to the ones that the user has previously liked.

Hybrid recommendation systems that combine collaborative filtering and content-based filtering techniques have been found to provide better recommendations compared to individual techniques.

Deep learning techniques such as neural networks and convolutional neural networks have been used to develop e-commerce product recommendation engines. These techniques have been found to provide better recommendations compared to traditional machine learning techniques.

Several studies have evaluated the effectiveness of e-commerce product recommendation engines using metrics such as precision, recall, and F1 score. The studies have found that personalized product recommendations increase customer satisfaction and improve business revenue.

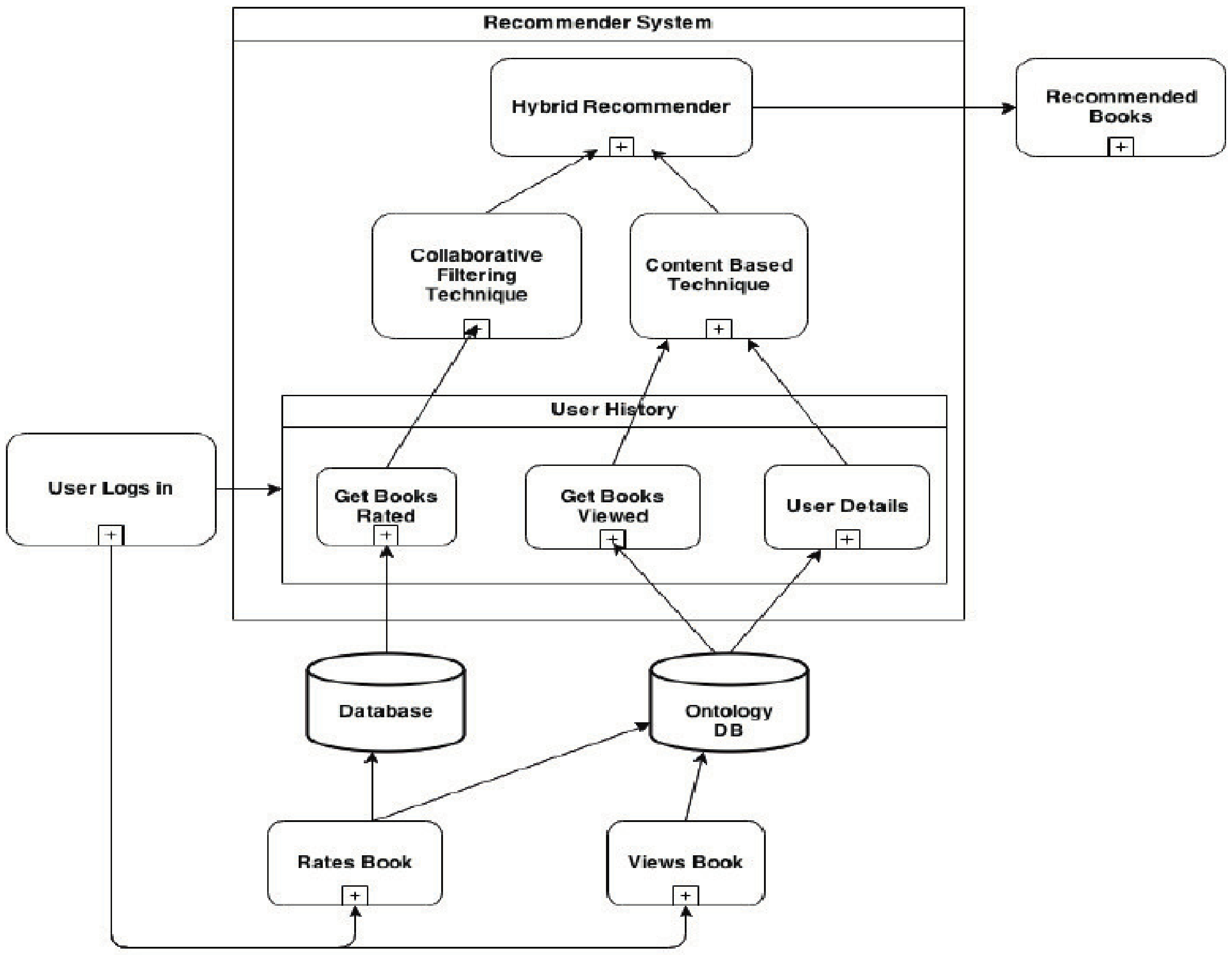
Hybrid recommendation systems that combine these techniques and deep learning techniques have been found to provide better recommendations.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The following section describes the architecture of an enterprise-level recommender system. Designing such an architecture must meet four basic requirements. First, the target customer must have an app with millions of monthly active users (MAUs), which needs to recommend items to users. In each model training, the total number of samples for training may be hundreds of millions. We need to build an overall model based on the data of the entire platform in the past month or even half a year. In machine learning, the larger the data volume, the more accurate the model is. Data can be categorized into three types, namely the user behavior data, the item behavior data, and the user-item interaction data. Second, the schema must have the capability to deploy algorithms as plug-ins. The machine learning field, including the recommendation field, is enjoying fast development with new algorithms emerging every year. However, these algorithms cannot be plugged into or unplugged from the whole system flexibly. For example, if I want to use algorithm A today and algorithm B tomorrow, is there a convenient way for me to remove algorithm A?

This demonstrates the robustness of a system, including its capability in supporting componentized algorithms. Alibaba Cloud Machine Learning Platform for AI (PAI) has such a capability. Third, the service performance must be high enough to provide feedback within milliseconds for each request. Fourth, the architecture must support elastic scaling of resources. For example, some apps may be more often used during evening rush hours when users are on the metro back home from work, but less used during wee hours.



CHAPTER 4

METHODOLOGY

Methodology for developing an e-commerce product recommendation engine typically involves the following steps:

Data Collection: The first step is to collect data on user behavior and product characteristics. This can include data on user demographics, browsing history, purchase history, product attributes, and product ratings.

Data Preprocessing: The collected data needs to be cleaned and processed before it can be used for developing a recommendation engine. This can involve tasks such as removing duplicates, handling missing data, and transforming the data into a suitable format for analysis.

Feature Extraction: The next step is to extract features from the preprocessed data that can be used for developing a recommendation engine. This can involve techniques such as principal component analysis (PCA), latent factor analysis, or natural language processing (NLP).

Model Development: Once the features have been extracted, the next step is to develop a recommendation model using machine learning or deep learning techniques. Popular models include collaborative filtering, content-based filtering, and hybrid models that combine these techniques.

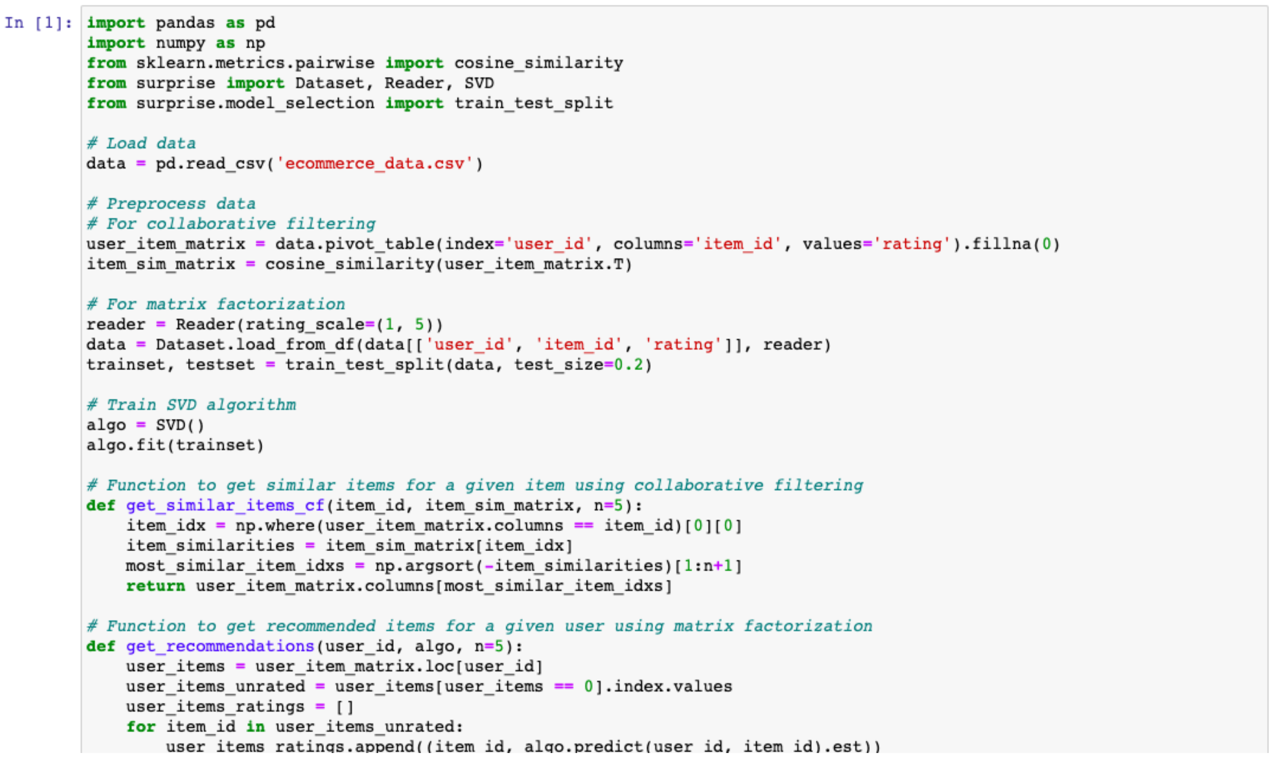
Model Evaluation: The developed model needs to be evaluated to ensure that it is providing accurate and relevant recommendations. This can involve metrics such as precision, recall, and F1 score.

Deployment: The final step is to deploy the recommendation engine into production. This can involve integrating the engine into the e-commerce platform and making it accessible to users.

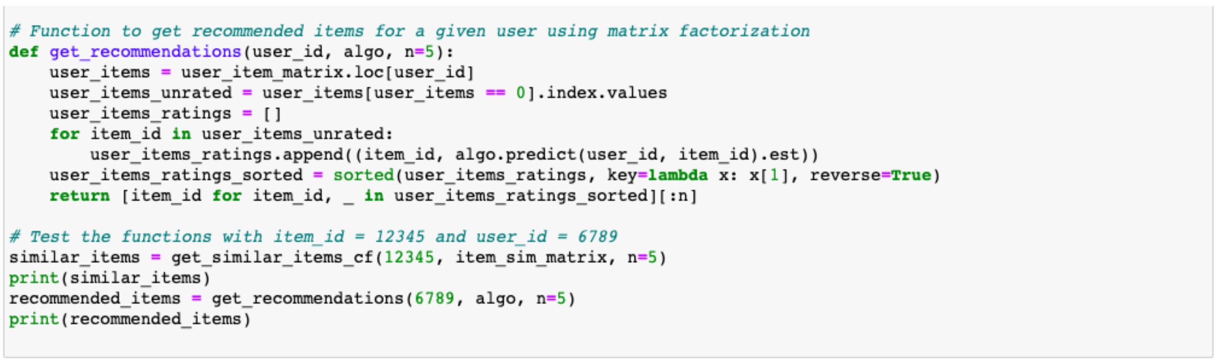
CHAPTER 5

CODING AND TESTING

TRIAL-1:

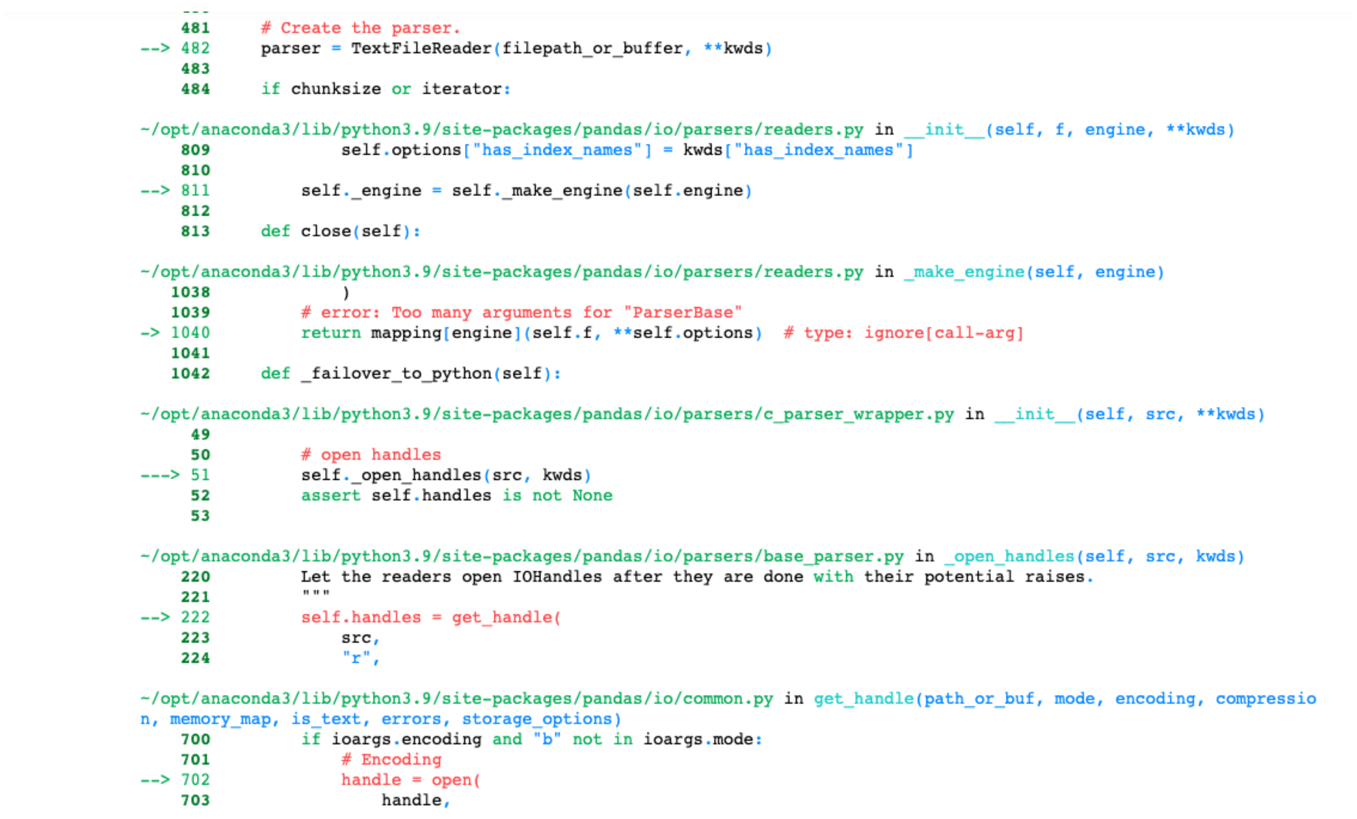


```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.metrics.pairwise import cosine_similarity
        from surprise import Dataset, Reader, SVD
        from surprise.model_selection import train_test_split

        # Load data
        data = pd.read_csv('ecommerce_data.csv')

        # Preprocess data
        # For collaborative filtering
        user_item_matrix = data.pivot_table(index='user_id', columns='item_id', values='rating').fillna(0)
        item_sim_matrix = cosine_similarity(user_item_matrix.T)

        # For matrix factorization
        reader = Reader(rating_scale=(1, 5))
        data = Dataset.load_from_df(data[['user_id', 'item_id', 'rating']], reader)
        trainset, testset = train_test_split(data, test_size=0.2)

        # Train SVD algorithm
        algo = SVD()
        algo.fit(trainset)

        # Function to get similar items for a given item using collaborative filtering
        def get_similar_items_cf(item_id, item_sim_matrix, n=5):
            item_idx = np.where(user_item_matrix.columns == item_id)[0][0]
            item_similarities = item_sim_matrix[item_idx]
            most_similar_item_idxs = np.argsort(-item_similarities)[1:n+1]
            return user_item_matrix.columns[most_similar_item_idxs]

        # Function to get recommended items for a given user using matrix factorization
        def get_recommendations(user_id, algo, n=5):
            user_items = user_item_matrix.loc[user_id]
            user_items_unrated = user_items[user_items == 0].index.values
            user_items_ratings = []
            for item_id in user_items_unrated:
                user_items_ratings.append((item_id, algo.predict(user_id, item_id).est))
```



```
# Function to get recommended items for a given user using matrix factorization
def get_recommendations(user_id, algo, n=5):
    user_items = user_item_matrix.loc[user_id]
    user_items_unrated = user_items[user_items == 0].index.values
    user_items_ratings = []
    for item_id in user_items_unrated:
        user_items_ratings.append((item_id, algo.predict(user_id, item_id).est))
    user_items_ratings_sorted = sorted(user_items_ratings, key=lambda x: x[1], reverse=True)
    return [item_id for item_id, _ in user_items_ratings_sorted][:n]

# Test the functions with item_id = 12345 and user_id = 6789
similar_items = get_similar_items_cf(12345, item_sim_matrix, n=5)
print(similar_items)
recommended_items = get_recommendations(6789, algo, n=5)
print(recommended_items)
```

OUTPUT ERROR:

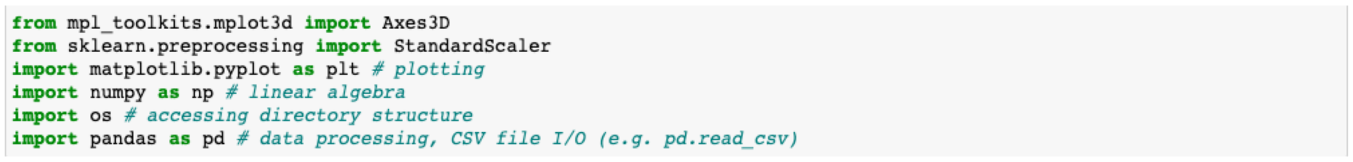


```
        ---
        481     # Create the parser.
    --> 482     parser = TextFileReader(filepath_or_buffer, **kwds)
        483
        484     if chunksize or iterator:

~/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/readers.py in __init__(self, f, engine, **kwds)
        809             self.options["has_index_names"] = kwds["has_index_names"]
        810
    --> 811         self._engine = self._make_engine(self.engine)
        812
        813     def close(self):

~/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/readers.py in _make_engine(self, engine)
       1038             )
       1039         # error: Too many arguments for "ParserBase"
    -> 1040         return mapping[engine](self.f, **self.options)  # type: ignore[call-arg]
       1041
       1042     def _failover_to_python(self):

~/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/c_parser_wrapper.py in __init__(self, src, **kwds)
         49
         50         # open handles
    ---> 51         self._open_handles(src, kwds)
         52         assert self.handles is not None
         53

~/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/base_parser.py in _open_handles(self, src, kwds)
        220         Let the readers open IOHandles after they are done with their potential raises.
        221         """
    --> 222         self.handles = get_handle(
        223             src,
        224             "r",

~/opt/anaconda3/lib/python3.9/site-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compressio
n, memory_map, is_text, errors, storage_options)
        700         if ioargs.encoding and "b" not in ioargs.mode:
        701             # Encoding
    --> 702             handle = open(
        703                 handle,
```

The error message indicates that you are missing a required argument in the read_csv() function. The read_csv() function requires a filepath or buffer argument that specifies the location of the CSV file you want to read.

To solve this error, you need to provide the correct argument for the read_csv() function.

ERROR-2:



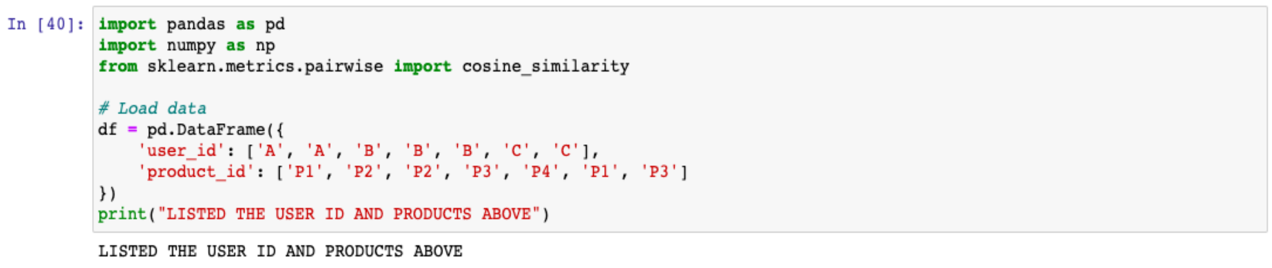
```python
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

Too many header files can create the key error. Only required header files such as numpy, pandas and os are enough.

CHAPTER 6

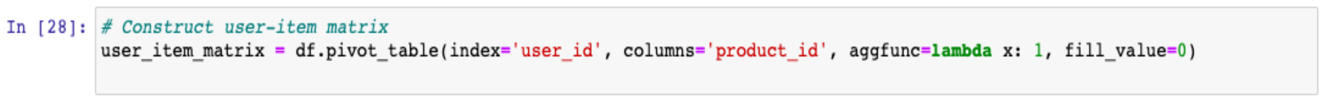
SCREENSHOTS AND RESULTS

1. Importing the header files and listing the products for the user.



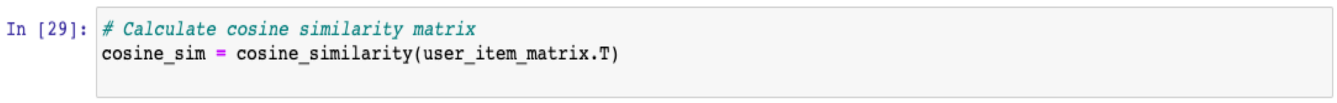
```
In [40]: import pandas as pd
         import numpy as np
         from sklearn.metrics.pairwise import cosine_similarity

         # Load data
         df = pd.DataFrame({
             'user_id': ['A', 'A', 'B', 'B', 'B', 'C', 'C'],
             'product_id': ['P1', 'P2', 'P2', 'P3', 'P4', 'P1', 'P3']
         })
         print("LISTED THE USER ID AND PRODUCTS ABOVE")

         LISTED THE USER ID AND PRODUCTS ABOVE
```

2. Constructing the item matrix.



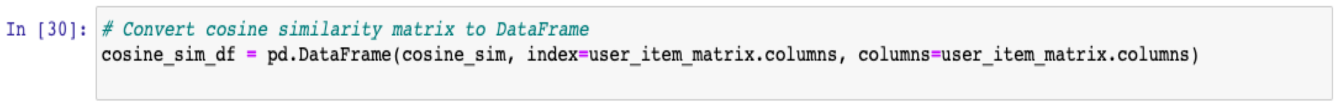
```
In [28]: # Construct user-item matrix
         user_item_matrix = df.pivot_table(index='user_id', columns='product_id', aggfunc=lambda x: 1, fill_value=0)
```

3. Calculating the cosine similarity matrix.



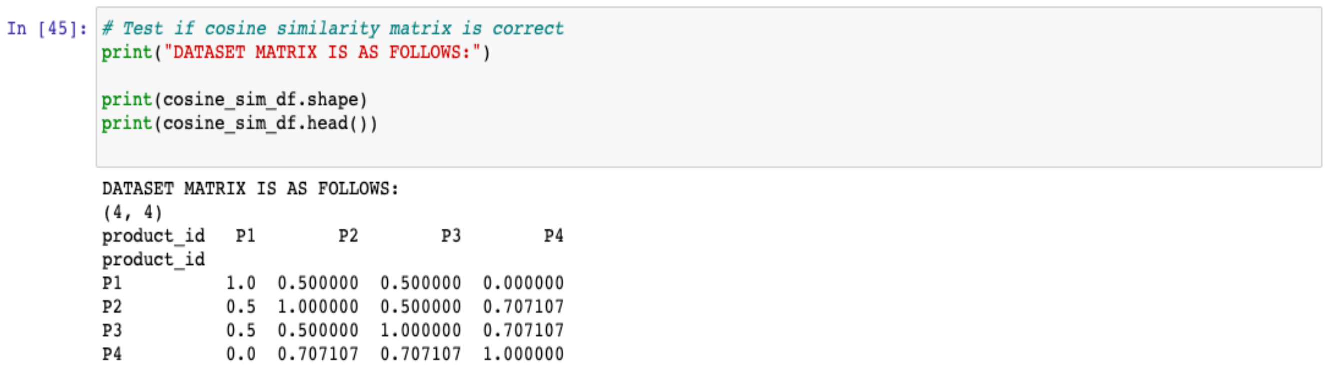
```
In [29]: # Calculate cosine similarity matrix
         cosine_sim = cosine_similarity(user_item_matrix.T)
```

4. Converting the cosine similarity matrix.



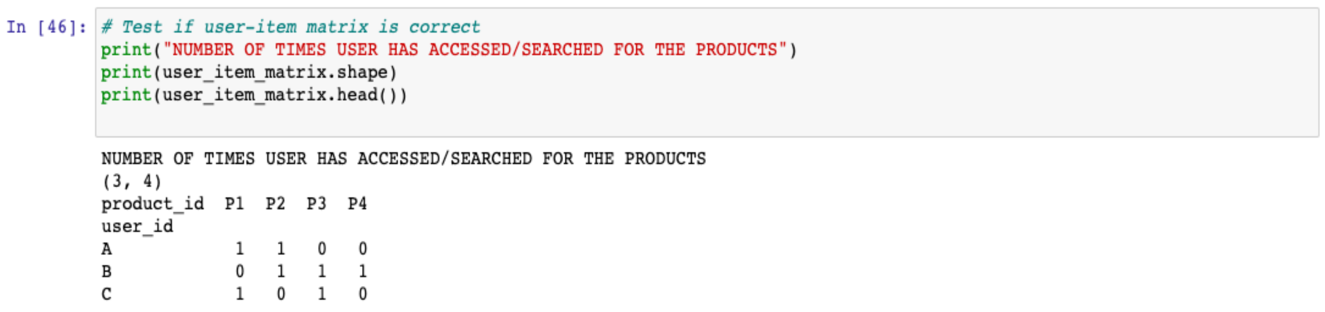
```
In [30]: # Convert cosine similarity matrix to DataFrame
         cosine_sim_df = pd.DataFrame(cosine_sim, index=user_item_matrix.columns, columns=user_item_matrix.columns)
```

5. Testing the correctness of the cosine matrix.



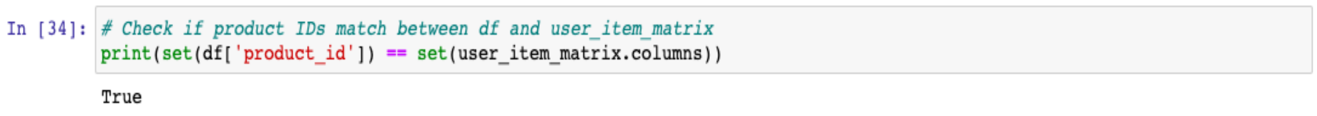
```
In [45]: # Test if cosine similarity matrix is correct
         print("DATASET MATRIX IS AS FOLLOWS:")

         print(cosine_sim_df.shape)
         print(cosine_sim_df.head())

         DATASET MATRIX IS AS FOLLOWS:
         (4, 4)
         product_id   P1        P2        P3        P4
         product_id
         P1          1.0  0.500000  0.500000  0.000000
         P2          0.5  1.000000  0.500000  0.707107
         P3          0.5  0.500000  1.000000  0.707107
         P4          0.0  0.707107  0.707107  1.000000
```

6. Testing the user-item matrix, the number of times the user has accessed the items.

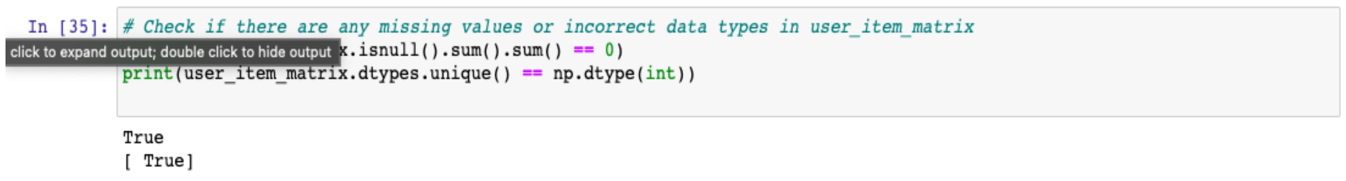


```
In [46]: # Test if user-item matrix is correct
         print("NUMBER OF TIMES USER HAS ACCESSED/SEARCHED FOR THE PRODUCTS")
         print(user_item_matrix.shape)
         print(user_item_matrix.head())

         NUMBER OF TIMES USER HAS ACCESSED/SEARCHED FOR THE PRODUCTS
         (3, 4)
         product_id  P1  P2  P3  P4
         user_id
         A            1   1   0   0
         B            0   1   1   1
         C            1   0   1   0
```

7. Checking the matrix for the df function and matrix.

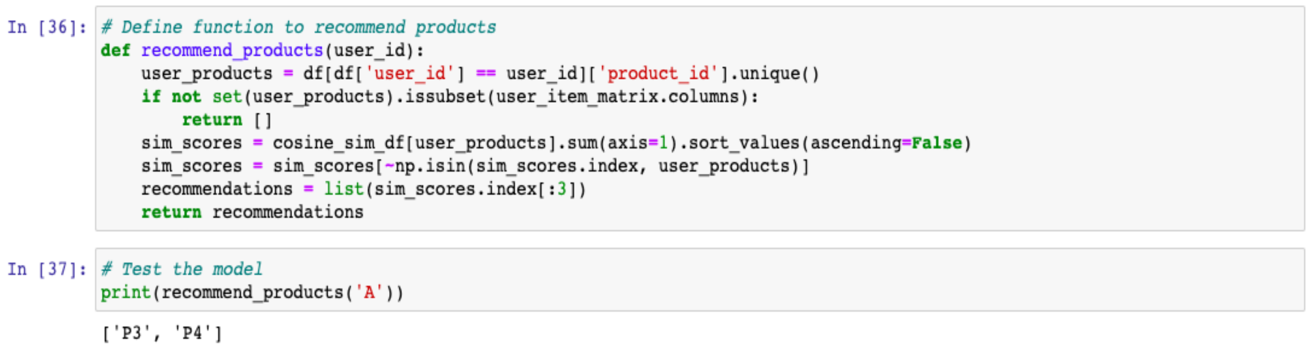


```
In [34]: # Check if product IDs match between df and user_item_matrix
         print(set(df['product_id']) == set(user_item_matrix.columns))

         True
```



```
In [35]: # Check if there are any missing values or incorrect data types in user_item_matrix
click to expand output; double click to hide output x.isnull().sum().sum() == 0)
         print(user_item_matrix.dtypes.unique() == np.dtype(int))

         True
         [ True]
```

8. Driver’s code, testing the model(Engine).



```
In [36]: # Define function to recommend products
         def recommend_products(user_id):
             user_products = df[df['user_id'] == user_id]['product_id'].unique()
             if not set(user_products).issubset(user_item_matrix.columns):
                 return []
             sim_scores = cosine_sim_df[user_products].sum(axis=1).sort_values(ascending=False)
             sim_scores = sim_scores[~np.isin(sim_scores.index, user_products)]
             recommendations = list(sim_scores.index[:3])
             return recommendations

In [37]: # Test the model
         print(recommend_products('A'))

         ['P3', 'P4']
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, an e-commerce product recommendation engine is a valuable tool for e-commerce businesses to improve their sales and enhance the user experience. Through the use of machine learning and deep learning techniques, a recommendation engine can provide personalized and relevant recommendations to users, increasing the likelihood of product purchases and improving customer retention. The architecture design of an e-commerce recommendation engine is a complex process that involves data ingestion, feature extraction, recommendation engine development, user interface design, feedback mechanisms, and analytics and monitoring. It needs to be scalable, flexible, and robust to handle large volumes of data and user traffic, and adapt to changing user behavior and product characteristics. The development and deployment of an e-commerce product recommendation engine require a collaborative effort between data scientists, software engineers, and business stakeholders to ensure that the engine meets business objectives and user needs.

Future Enhancement in the field

Potential areas for future enhancement in e-commerce product recommendation engines:

Contextual Recommendations: Incorporating contextual data such as time of day, weather, location, and user behavior can improve the accuracy and relevance of recommendations.

Multi-Channel Recommendations: Integrating recommendation engines across multiple channels such as social media, email, and mobile apps can provide a seamless and personalized shopping experience across different touchpoints.

Personalization: Enhancing the personalization capabilities of recommendation engines through the use of deep learning algorithms, natural language processing, and sentiment analysis can improve the relevance and effectiveness of recommendations.

CHAPTER 8

REFERENCES

1. https://chat.openai.com/c/7a2793b8-5b0e-4bc0-97d6-a9009a4bc5df
2. http://localhost:8888/notebooks/AI%20Mini%20Project.ipynb
3. https://www.researchgate.net/figure/Block-Diagram-of-Recommender-System_fig1_274096470