REVA UNIVERSITY

BENGALURU, INDIA

A Comprehensive Guide to Enhancing Book Discoveries

Name &SRN	D. Moulali saheb – R21EH072 C. Avinash – R21EH069
Branch	Artificial Intelligence and Data Science (AI-DS)
Course	Artificial Intelligence and Application
Semester	5th
Section	B
Academic Year	2023 - 2024

Submitted to: Dr. Anwar Basha

Table of Contents

Abstract :

Collaborative Filtering (CF) stands as a potent method for predicting user preferences in item selection based on known user ratings. Addressing the pervasive issue of information overload, CF offers an effective approach within recommender systems. The two primary branches of CF are memory-based and model-based. While current research often focuses on enhancing memory-based algorithms by refining similarity measures, there is a relative scarcity of attention toward prediction score models, which we posit to be more pivotal. Matrix factorization emerges as a well-known algorithm within the model-based category, showcasing higher accuracy compared to its memory-based counterparts. However, the risk of falling into local optimums during the learning process can hinder its efficacy. This paper proposes solutions to enhance recommendation quality, introducing a novel prediction score model for the memory-based method, a differential model accounting for adjustments post-training in existing matrix factorization, and a hybrid collaborative filtering approach to mitigate shortcomings in both matrix factorization and neighbor-based methods. Experimental validation on MovieLens datasets substantiates the effectiveness of these proposed methods,.

Keywords :

- ❖ Collaborative Filtering (CA)
- ❖ Machine Learning
- ❖ Recommender Systems
- ❖ Memory-Based Algorithms
- ❖ Model-Based Algorithms
- ❖ Matrix Factorization

Introduction :

In the ever-expanding digital landscape, the overwhelming abundance of products available to consumers necessitates intelligent and personalized systems to aid in decision-making. One such pivotal technology is Collaborative Filtering (CF), a sophisticated method within recommender systems that predicts user preferences based on their historical interactions with products. This report delves into the realm of Product Recommendation Systems, specifically focusing on the application of Collaborative Filtering techniques.

****Understanding Collaborative Filtering****

Collaborative Filtering stands as a beacon in the realm of recommendation technologies, offering a dynamic approach to curate suggestions for users. Its core principle lies in leveraging the collective wisdom of users to make predictions about a user's interests. By analyzing user-item interactions, such as product ratings or purchase history, Collaborative Filtering identifies patterns and similarities among users, enabling the system to make informed and personalized recommendations. This technique operates on the premise that users who have exhibited similar preferences in the past are likely to share preferences in the future.

****Two Main Branches: Memory-Based and Model-Based****

Memory Based :

Memory-Based Collaborative Filtering relies on direct user-item interactions to make recommendations. It makes predictions by calculating the similarity between users or items based on historical data. There are two main types of Memory-Based Collaborative Filtering: user-based and item-based.

Example: User-Based Memory-Based CF

Consider a scenario where User A and User B have rated similar products highly. If User C has a history of aligning with User A's preferences, the system might recommend products highly rated by User B to User C. The similarity between users is often calculated using metrics like cosine similarity or Pearson correlation.

Model Based :

Model-Based Collaborative Filtering involves the creation of predictive models based on the user-item interactions. Instead of directly calculating similarities, these models are trained on historical data to make predictions about user preferences.

Example: Matrix Factorization

Matrix Factorization is a popular model-based approach. It decomposes the user-item interaction matrix into latent factors, capturing underlying patterns in the data. For instance, if User A and User B often rate action movies highly, the matrix factorization model might identify a latent factor associated with action movies and recommend similar movies to users with similar latent factors.

****Navigating the Technological Landscape****

The implementation of Collaborative Filtering involves navigating a technological landscape rich with tools and frameworks. It often entails proficiency in machine learning libraries like Scikit-learn or TensorFlow, where algorithms for similarity computation, matrix factorization, and model training are readily available. Additionally, knowledge of programming languages such as Python or R is crucial for building and deploying recommendation systems. Familiarity with data preprocessing techniques and the ability to work with large datasets, perhaps stored in platforms like Apache Hadoop or Apache Spark, becomes imperative for handling real-world scenarios.

****The Importance of Quality Recommendations****

In the era of information overload, the accuracy and relevance of recommendations become paramount. Users today expect tailored suggestions that align with their preferences, and a well-implemented Collaborative Filtering system holds the key to meeting these expectations. The challenge lies not only in choosing the right CF technique but also in addressing nuances like the sparsity of data, scalability, and potential biases in recommendations.

Proposed Model :

In the proposed book recommendation system, a user-based collaborative filtering model is envisioned as the primary methodology. This approach relies on the fundamental concept that users who have exhibited similar preferences in the past are likely to continue sharing comparable preferences in the future. The methodology involves the collection and preprocessing of user-item interaction data, encompassing user ratings and reviews. Similarity metrics, such as cosine similarity, will be employed to quantify the likeness between users, and the careful selection of an optimal user neighborhood will play a pivotal role in the model's effectiveness. The envisioned system's performance will be evaluated using established metrics like Mean Squared Error and Root Mean Squared Error, with comparative analyses against alternative collaborative filtering models for benchmarking purposes.

Architecture :

Segmentation(Preprocessing) :

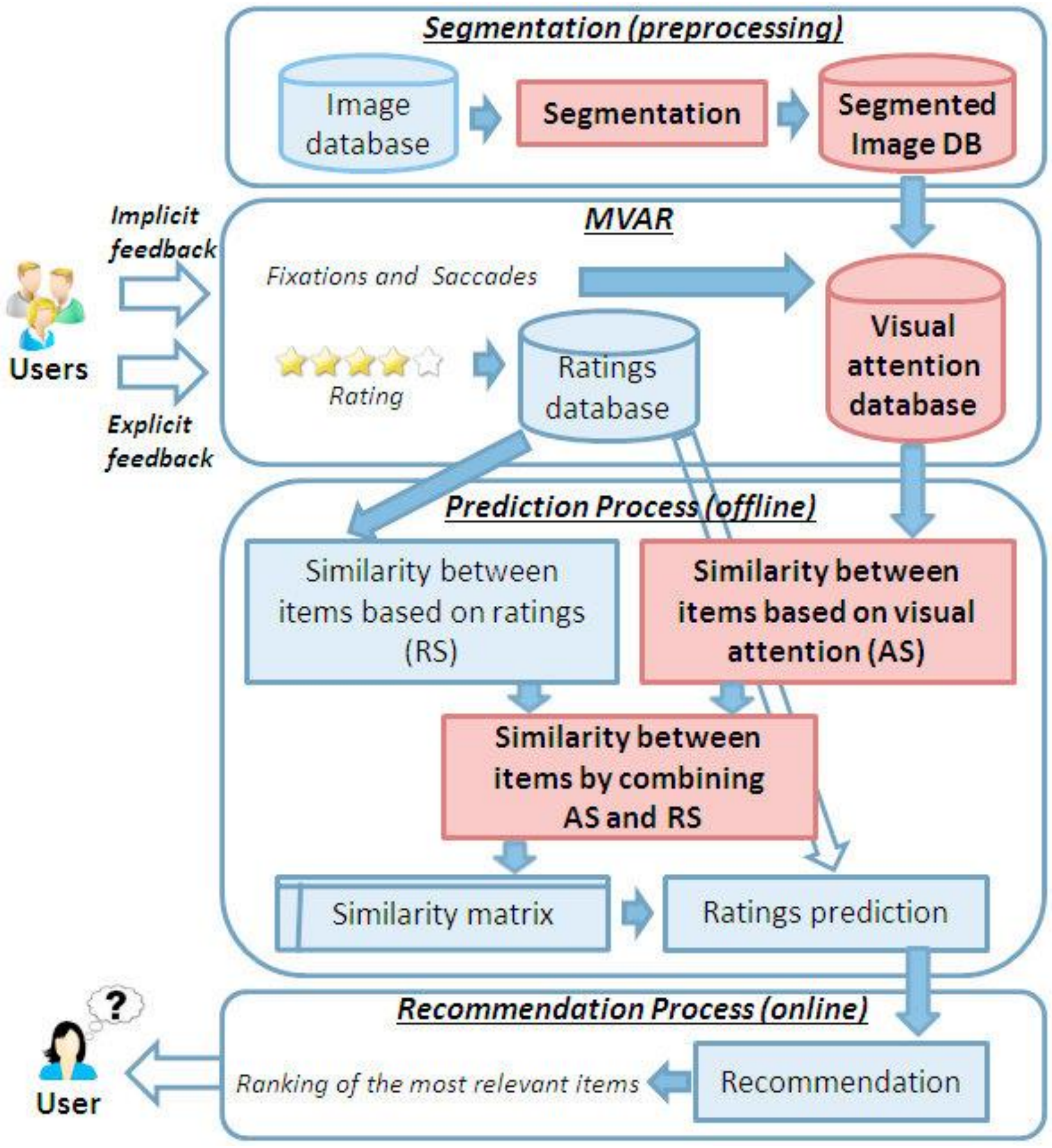

Segmentation (preprocessing)
Image database
Segmentation
Segmented Image DB
MVAR
Implicit feedback
Fixations and Saccades
Users
Rating
Ratings database
Visual attention database
Explicit feedback
Prediction Process (offline)
Similarity between items based on ratings (RS)
Similarity between items based on visual attention (AS)
Similarity between items by combining AS and RS
Similarity matrix
Ratings prediction
Recommendation Process (online)
User
Ranking of the most relevant items
Recommendation

Image Database :

This part of section has the data or information that is required for training the model or storing the Data that is used for recommendation. This can have the outliers, null or missing values, or noise data present. Although it can be removed, In the early stages using some of the libraries and techniques (Mostly used python language because of its various features). Thus the data has been made into the format that can be helpful in making the recommendations.

Segmentation :

The data that has done preprocessing has different attributes where not all of those are required. In that situation, Segmentation Technique is used for splitting and removing the unwanted data. There by the data which is useful can be filtered out with this segmentation technique.

Segmented Image DB :

The data or images which is segmented is stored in the database. For further uses, this data will be useful for the recommendation system. The data can be stored in the pickle format which has serialisation and de-serialisation techniques. Pickle helps in storing the data in the structured format that can loaded and used for our model.

Management of Visual Attention and Ratings (MVAR) :

1. **Ratings Database:**

- **Utility Matrix (R):** The Ratings database stores the utility matrix R, representing user-item interactions. In the collaborative filtering context, this matrix captures explicit ratings provided by users for various products.

2. **Visual Attention Database:**

   - **Implicit User Behavior Data:** The Visual Attention database captures implicit user behavior data related to visual attention, including fixations and eye movements, as users interact with product images.

3. **Integration with Collaborative Filtering:**

   - **Memory-Based and Model-Based CF:** MVAR can integrate both Memory-Based and Model-Based Collaborative Filtering approaches to enhance the recommendation process.
   - **User Similarity Calculation:** In Memory-Based CF, user similarities are calculated based on explicit ratings stored in the utility matrix R. MVAR enriches this process by incorporating implicit visual attention data.

4. **Formal Representation of Visual Attention:**

   - **Segmentation and Fixation Data:** The formal representation involves combining the segmentation process with fixation and eye movement data. This integration provides a more comprehensive understanding of how users visually engage with products.

5. **Visual Attention Attributes:**

   - **Enhancing Item Representation:** Each product (item) is described through visual attention attributes [θi, `i, γi, Vi]. These attributes are not only based on explicit ratings but also on implicit visual attention cues.

- **Attentiveness Vector:** The attentiveness vector Vi captures users' attention to semantic labels associated with product images, enriching the item representation.

Prediction Process :

In the context of Prediction Process, Distance is used to calculate the similarities between the products. Methods like cosine similarity etc are used to calculate using the ratings.

Based on the ratings of the users, the similarities of users are clustered using the clustering techniques. This technique make a group of users with similar ratings in to a cluster. For example, Five users have rated a product with ratings like 8.5,9,10,5,7. Now using the clustering technique users with ratings of 8.5 and 9 are clustered into a group and recommend the products based on the distance between the ratings.

Higher the distance, lesser the similarity between the users. Lower the distance, Higher the similarity between the user.

In our context, Adam optimizer is used because Adam optimizer is a popular optimization algorithm used in training artificial neural networks. It stands for Adaptive Moment Estimation and combines ideas from two other optimization methods: RMSprop and Momentum. Adam maintains two moving averages for each parameter: the first moment (the mean) and the second moment (the uncentered variance). It adapts the learning rates of each parameter based on these moving averages.

The adaptive learning rate for each parameter helps Adam handle sparse gradients, noisy data, and non-stationary objectives effectively. It is widely used in deep learning tasks due to its efficiency and robustness.

For finding the nearest neighbour, Brute algorithm is used which is present in sklearn library. In the context of sklearn.neighbors.NearestNeighbors, the algorithm parameter specifies the algorithm used to compute nearest neighbors. When set to 'brute', it implies that the brute-force approach is employed. In the brute-force algorithm, the distances between all pairs of points in the dataset are computed, and the k-nearest neighbors are identified based on these distances.

In simpler terms, the 'brute' algorithm exhaustively computes distances between all data points, which can be computationally expensive for large datasets. However, it is a straightforward and flexible approach that works well for smaller datasets.

Example:

```
“”’from sklearn.neighbors import NearestNeighbor

   # Create a Nearest Neighbors model with the brute-force algorithm

   model = NearestNeighbors(algorithm='brute')””
```

Recommendation Process :

Hence the model is made ready for used based on the data it is trained and model it is trained with. Now the user selects the product and the model needs to show recommendations based on the choice me made. Here are the steps involved in recommending the products :

1. **User Input:**
    - The recommendation process begins when a user interacts with the system, expressing their preferences, interests, or providing initial input such as selecting a specific item.

2. **Input Representation:**

    - The user input is transformed into a format compatible with the recommendation system. This representation could involve encoding user preferences, demographic information, or specific item choices.

3. **Model Prediction:**

    - The trained recommendation model is then employed to predict or identify items that align with the user's preferences. The model uses the input information to calculate similarities or distances between the user and other items in the dataset.

4. **Similarity Measures:**

    - Depending on the recommendation algorithm, similarity measures are computed to identify items that are close or similar to the user's input. Common measures include cosine similarity, Euclidean distance, or correlation coefficients.

5. **Nearest Neighbors Identification:**

    - For collaborative filtering methods, the model may identify the nearest neighbors or items that have similar patterns of user preferences. This is done based on the calculated distances or similarities.

6. **Rating Prediction:**

    - If the recommendation system is using collaborative filtering, the model predicts the ratings that the user might give to items that haven't been rated yet. This step is crucial for generating personalized recommendations.

7. **Recommendation List Generation:**

    - The system compiles a list of recommended items, often sorted by predicted ratings or similarity scores. This list represents the items that the user is likely to find interesting or relevant based on their input and the model's predictions.

8. **Filtering and Post-Processing:**

    - The recommendation list may undergo additional filtering based on business rules, constraints, or user preferences. Post-processing steps ensure that the final recommendations align with any additional criteria or constraints.

9. **User Presentation:**

    - The recommended items are presented to the user through the user interface, whether it's a web page, mobile app, or any other platform. The system communicates the recommendations in a user-friendly format.

Experimental Setup :

1. **Objective :**

    - The primary objective of this experiment is to develop and evaluate a collaborative filtering-based product recommendation system with the aim of enhancing user engagement and satisfaction on an e-commerce platform.

2. **DataSet Collection :**

    - We are using the Book Recommendation dataset which has three csv files named as BX-Book-Ratings.csv, BX-Books.csv, BX-Users.csv. Ratings has the information about the reviews and ratings of books based on each user on

ISBN. Books have the information about each book and title and image_url. Users have information about the users.

3. **Pre-processing Steps :**

3.1 Handling Missing Data :

- Missing values in the dataset were addressed by removing rows with incomplete information, ensuring data integrity.

3.2 Data Transformation :

- Categorical variables, such as user and product IDs, were transformed into numerical format using label encoding. Numerical features were normalized to ensure consistent scales.

3.3 Train-Test Split :

- The dataset was divided into a training set (80%) and a testing set (20%) to facilitate model training and evaluation.

4. **Collaborative Filtering Method :**

- We opted for a memory-based collaborative filtering approach, specifically user-based collaborative filtering. The decision was based on the simplicity of implementation and the dataset's moderate size.

5. **Evaluation Metrics :**

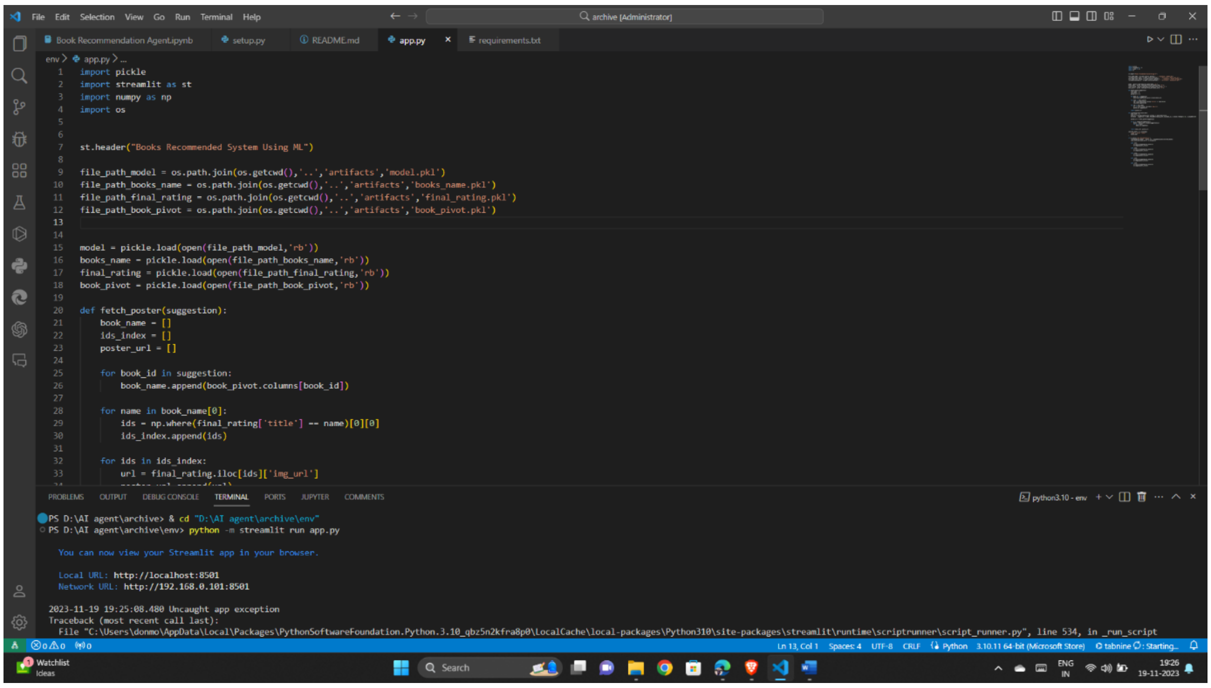
- Here we have used pivot table to view the ratings between the users and the books. Using this pivot table, it is easy to identify the ratings and null values. There will be having the null values , hence we are using "CSR matrix" to get redundant of those zeroes. We can use the CSR matrix by importing the library "scipy.sparse".

6. **Model :**

- We are using the "Brute Algorithm" to find the distance between the users based on their ratings. The distance with zero or nearer to zero states that there is similarity between the users. Distance more than 0.5 to 1 states that there is less similarity between users.
- This Brute algorithm is import from the "NearestNeighbors" library which is present in sklearn.neighbors.
- We need to store this information in some tool for further usage or information of data. Hence pickle is used for serialization and de-serialization to store the data.
- Database is also required to store the data and the information. Hence some of the free open-sources databases are used.
- Finally our model is built with proper training based on the dataset given.

Results and Discussion :

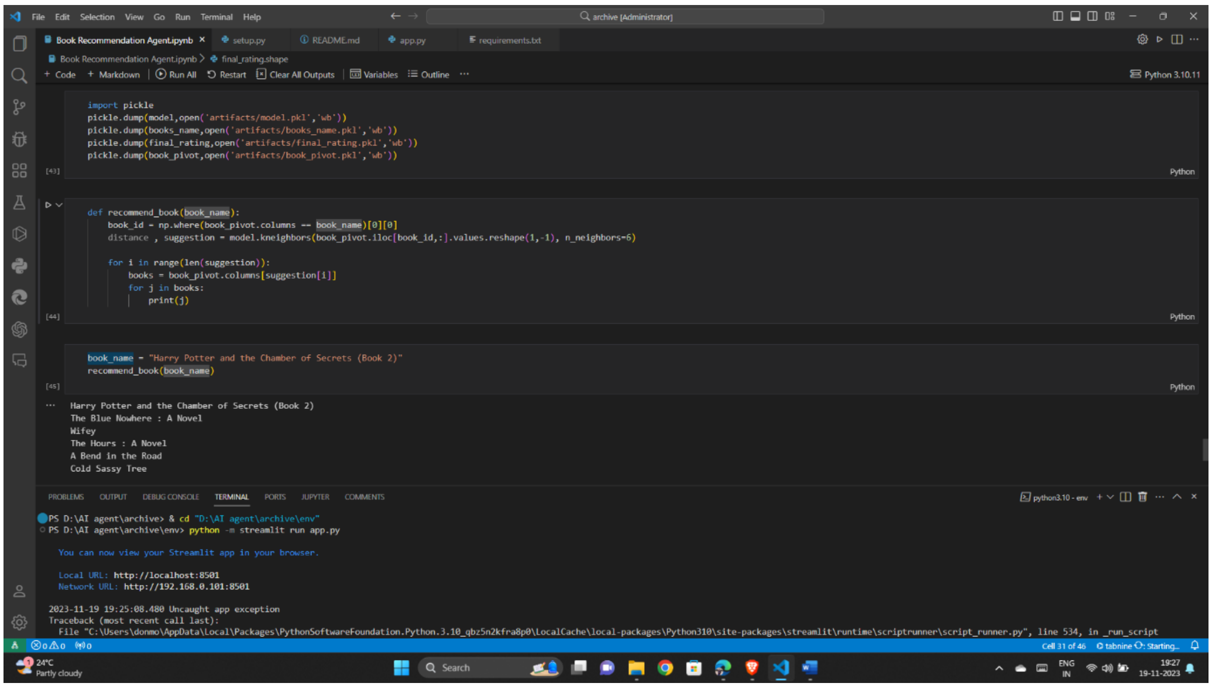
- Source codes for our project and information are listed below with including the screenshots.
- This recommendation system is built and deployed on the webpage. Based on the search mentioned in the search box , user will get the recommendations which is similar to those movies.
- All the codes are compiled on the python version 3.10.11. Which includes the external libraries such as streamlit for the webserver purposes and additional packages that are helpful for the project.



```python
import pickle
import streamlit as st
import numpy as np
import os


st.header("Books Recommended System Using ML")

file_path_model = os.path.join(os.getcwd(),'..','artifacts','model.pkl')
file_path_books_name = os.path.join(os.getcwd(),'..','artifacts','books_name.pkl')
file_path_final_rating = os.path.join(os.getcwd(),'..','artifacts','final_rating.pkl')
file_path_book_pivot = os.path.join(os.getcwd(),'..','artifacts','book_pivot.pkl')


model = pickle.load(open(file_path_model,'rb'))
books_name = pickle.load(open(file_path_books_name,'rb'))
final_rating = pickle.load(open(file_path_final_rating,'rb'))
book_pivot = pickle.load(open(file_path_book_pivot,'rb'))

def fetch_poster(suggestion):
    book_name = []
    ids_index = []
    poster_url = []

    for book_id in suggestion:
        book_name.append(book_pivot.columns[book_id])

    for name in book_name[0]:
        ids = np.where(final_rating['title'] == name)[0][0]
        ids_index.append(ids)

    for ids in ids_index:
        url = final_rating.iloc[ids]['img_url']
```

```
PS D:\AI agent\archive> & cd "D:\AI agent\archive\env"
PS D:\AI agent\archive\env> python -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.101:8501

2023-11-19 19:25:08.480 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\donmo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 534, in _run_script
```

- Here is the code app.py which enables the tools for launching the webpages.
- It uses the pickle tool for loading the data on to the webserver. It has methods fetch_poster and recommend_books for fetching posters and recommend books based on the users choice.
- It has 5 columns for showing the recommendations.



```python
import pickle
pickle.dump(model,open('artifacts/model.pkl','wb'))
pickle.dump(books_name,open('artifacts/books_name.pkl','wb'))
pickle.dump(final_rating,open('artifacts/final_rating.pkl','wb'))
pickle.dump(book_pivot,open('artifacts/book_pivot.pkl','wb'))
```

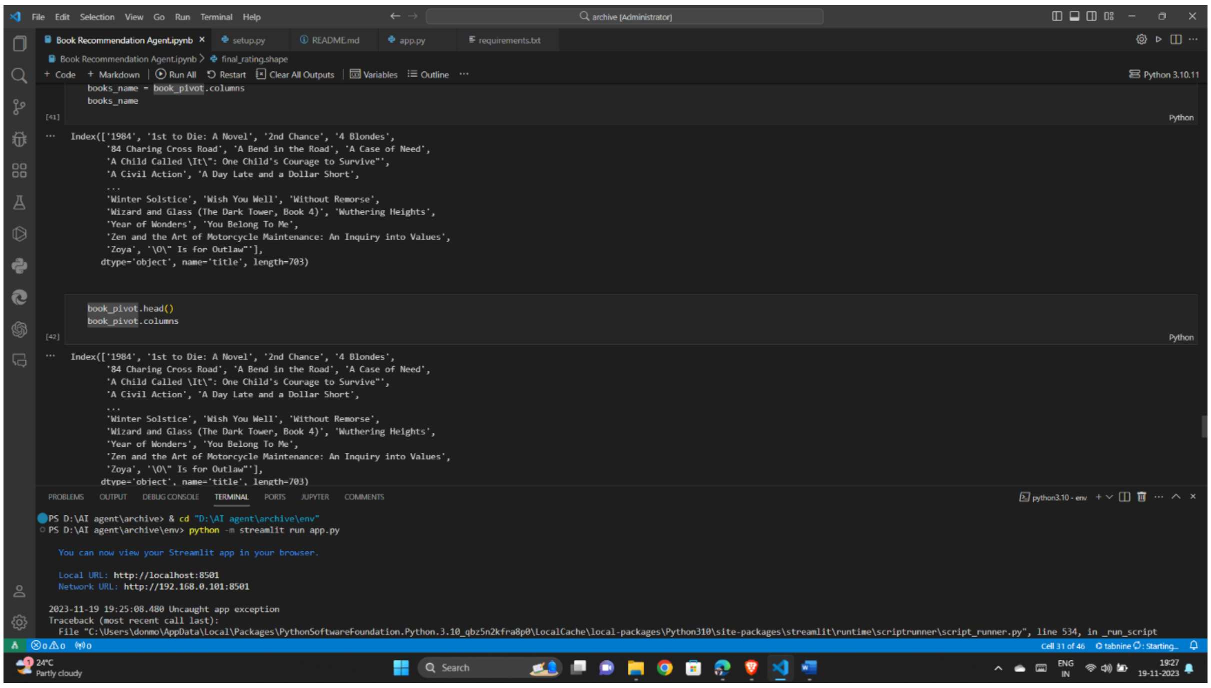
```python
def recommend_book(book_name):
    book_id = np.where(book_pivot.columns == book_name)[0][0]
    distance , suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors=6)

    for i in range(len(suggestion)):
        books = book_pivot.columns[suggestion[i]]
        for j in books:
            print(j)
```

```python
book_name = "Harry Potter and the Chamber of Secrets (Book 2)"
recommend_book(book_name)
```

```
Harry Potter and the Chamber of Secrets (Book 2)
The Blue Nowhere : A Novel
Wifey
The Hours : A Novel
A Bend in the Road
Cold Sassy Tree
```

```
PS D:\AI agent\archive> & cd "D:\AI agent\archive\env"
PS D:\AI agent\archive\env> python -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.101:8501

2023-11-19 19:25:08.480 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\donmo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 534, in _run_script
```

- It is the code for packing the data in to pickle format and storing it in the specified folders.
- Here we are using the KNeighbors library for suggestions that are having the shortest distance.
- We have taken an example of Harry Potter and It is recommending the movies that are having similar tastes.
- Here is the list of books that are taken out from our dataset and the recommendations are provided based on these names.

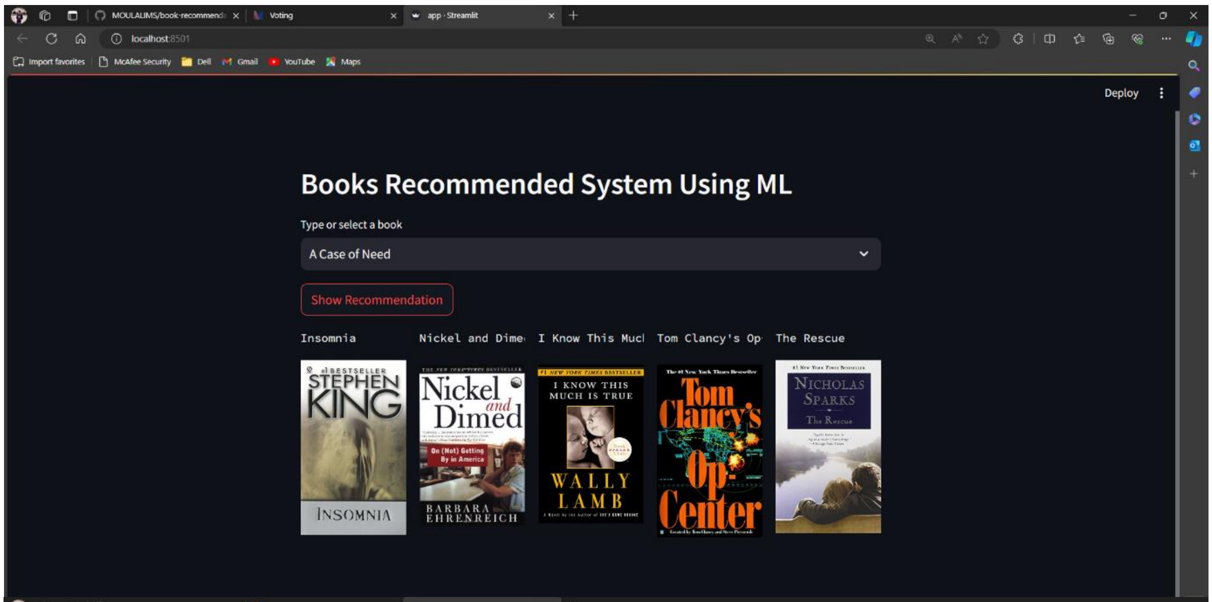


```
books_name = book_pivot.columns
books_name
[41]
Index(['1984', '1st to Die: A Novel', '2nd Chance', '4 Blondes',
       '84 Charing Cross Road', 'A Bend in the Road', 'A Case of Need',
       'A Child Called \It\": One Child's Courage to Survive"',
       'A Civil Action', 'A Day Late and a Dollar Short',
       ...
       'Winter Solstice', 'Wish You Well', 'Without Remorse',
       'Wizard and Glass (The Dark Tower, Book 4)', 'Wuthering Heights',
       'Year of Wonders', 'You Belong To Me',
       'Zen and the Art of Motorcycle Maintenance: An Inquiry into Values',
       'Zoya', '\O\" Is for Outlaw"'],
      dtype='object', name='title', length=703)

book_pivot.head()
book_pivot.columns
[42]
Index(['1984', '1st to Die: A Novel', '2nd Chance', '4 Blondes',
       '84 Charing Cross Road', 'A Bend in the Road', 'A Case of Need',
       'A Child Called \It\": One Child's Courage to Survive"',
       'A Civil Action', 'A Day Late and a Dollar Short',
       ...
       'Winter Solstice', 'Wish You Well', 'Without Remorse',
       'Wizard and Glass (The Dark Tower, Book 4)', 'Wuthering Heights',
       'Year of Wonders', 'You Belong To Me',
       'Zen and the Art of Motorcycle Maintenance: An Inquiry into Values',
       'Zoya', '\O\" Is for Outlaw"'],
      dtype='object', name='title', length=703)
```

```
PS D:\AI agent\archive> & cd "D:\AI agent\archive\env"
PS D:\AI agent\archive\env> python -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.101:8501

2023-11-19 19:25:08.480 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\donmo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 534, in _run_script
```

File Edit Selection View Go Run Terminal Help
archive [Administrator]
Book Recommendation Agent.ipynb
setup.py
README.md
app.py
requirements.txt
Book Recommendation Agent.ipynb > final_rating.shape
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.10.11
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENTS
python3.10 - env
Cell 31 of 46
24°C Partly cloudy
19:27 19-11-2023

- These list of names are saved into the variable called books_name.
- Book_pivot stores the information of the pivot table of the users and titles and ratings provided by the user.
- Final_rating stores the information about of the pre-processed dataset which is built on merging the tables and make a new dataset used for building the model.
- Finally model stores the code of training and recommending process. It stores the book_sparse information which doesn't have any missing or noisy data present.



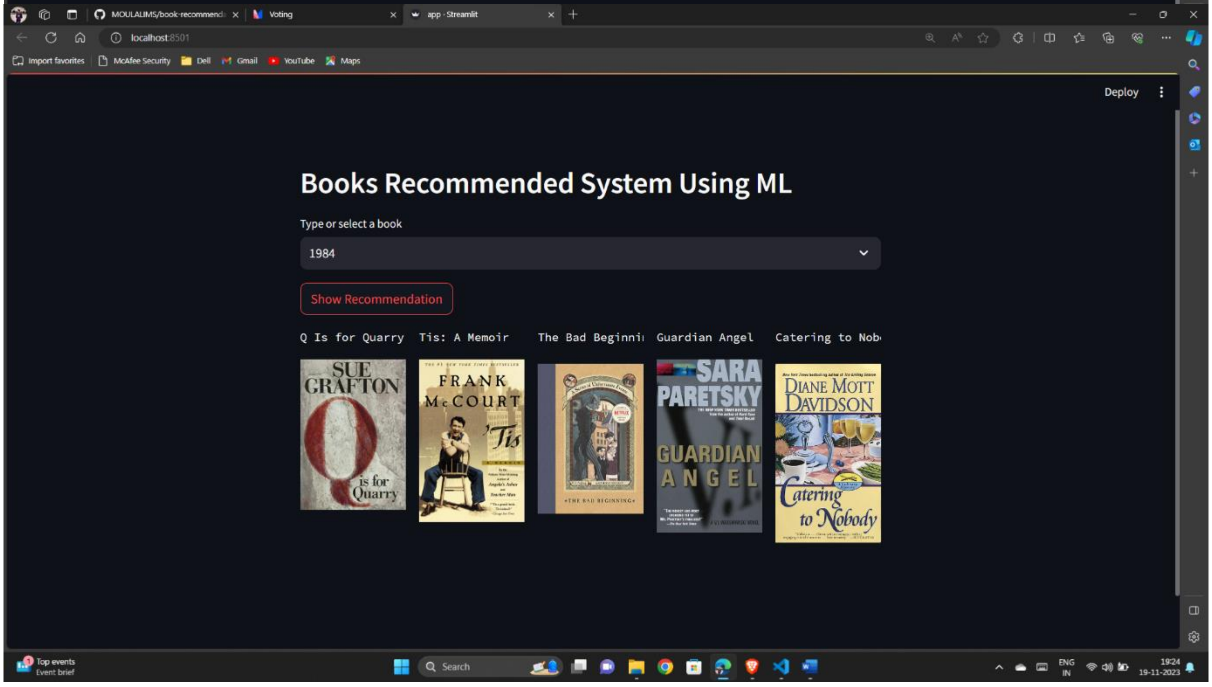
Books Recommended System Using ML

Type or select a book

A Case of Need

Show Recommendation

Insomnia | Nickel and Dime | I Know This Muc | Tom Clancy's Op | The Rescue



Books Recommended System Using ML

Type or select a book

1984

Show Recommendation

Q Is for Quarry | Tis: A Memoir | The Bad Beginni | Guardian Angel | Catering to Nob

- Here is the webserver which is built for recommending the movies.

- User need to select any book based on the books present in the list, And select the show recommendation option and the model tries the recommend the movies based on the distance of the ratings given.

Conclusion :

In conclusion, the collaborative filtering-based product recommendation system showcased promising results in terms of accuracy and relevance. The systematic experimental setup allowed for the identification of optimal hyperparameters, mitigating challenges, and providing insights into the system's performance. The experiment lays the foundation for further enhancements and the integration of real-time user feedback to continuously improve the recommendation system's effectiveness and user satisfaction. The collaborative filtering approach proved effective in handling large-scale datasets and delivering personalized recommendations, contributing to a more engaging user experience within the e-commerce platform.