A MINI PROJECT REPORT

Submitted by

AAKAASH K B	**(711620243001)**
ANISH SURESH TIWARI	**(711620243009)**
PRIYADHARSHINI B	**(711620243308)**

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KATHIR COLLEGE OF ENGINEERIING

"WISDOM TREE", NEELAMBUR, COIMBATORE – 641 062

ANNA UNIVERSITY: CHENNAI – 600 025

JANUARY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report

"BOOK RECOMMENDATION SYSTEM"

is the bonafide work of

"AAKAASH K B (711620243001),

ANISH SURESH TIWARI (711620243001) and

PRIYADHARSHINI B (711620243308) "

who carried out the mini project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M.RAJESH BABU, M.E., Ph.D.,	**Ms. P. SANGEETHA, B.E., M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor and Head	Assistant Professor
Department of	Department of
Artificial Intelligence and Data Science	Artificial Intelligence and Data Science
Kathir College of Engineering	Kathir College of Engineering
Coimbatore – 641 062	Coimbatore – 641 062

Project viva voce held on ____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our immense gratitude to Thiru E.S. KATHIR, Chairman, Kathir Institutions, Coimbatore for giving us an opportunity to study in their prestigious institution and to take up the project in Partial Fulfilment of the Regulation for the B.Tech Program.

We would like to express our deepest gratitude to Thirumathi LAVANYA KATHIR, Secretary, Kathir Institutions, Coimbatore for the soul support in our studies.

We would bound to express our gratitude to Dr.R.UDAIYAKUMAR, M.E.,Ph.D., Principal, Kathir College of Engineering, Coimbatore for their permission and constant encouragement throughout our course.

It is great pleasure to express our sincere and wholehearted gratitude to Professor Dr. M. RAJESH BABU, M.E., Ph.D., Head of the Artificial Intelligence And Data Science Department, and Ms. M KAVITHA, M.E.,(Ph.D.,), Mini Project coordinator of Artificial Intelligence And Data Science Department, for their constant suggestion and encouragement in the project work.

We also express our heartfelt thanks to Ms. M. KAVITHA, M.E.,(Ph.D), Assistant Professor and Project Guide, Department of Artificial Intelligence And Data Science for being supportive throughout the tenure of our project.

ABSTRACT

Recommender systems are the systems that are designed to recommend things to the user based on many different factors. The recommender system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user’s preference and interest. A book recommendation system is a type of recommendation system where we have to recommend similar books to the reader based on his interest. The books recommendation system is used by online websites which provide ebooks like google play books, open library, good Read's, etc.

As we used a Source platform Jupyter Notebook for running the recommendation engine, this platform helps to execute the machine learning algorithm and helps to suggest certain range of books depending upon various approaches.

Different Libraries are used in the python coding for loading and preprocessing the sample dataset which has 1,149,780 ratings from 278,858 users about 271,379 books. Multiple variants of filtering are done based on the input book name.

To provide efficient recommendations we have build unique filtering models that work based on

- Item – Based Collaborative Filtering
- Content – Based Filtering
- Custom Recommender

After processing the input text from the any of the available filters, the Tittles of the books are displayed on the screen along with the overall ratings from the users. The approximate accuracy turns out to be around 80%

LIST OF FIGURES

Table of Contents

CHAPTER 1
INTRODUCTION

1.1 Motivation of the Project

The explosive growth in the amount of available digital information and the number of visitors to the Internet have created a potential challenge of information overload which hinders timely access to items of interest on the Internet. Information retrieval systems, such as Google, DevilFinder and Altavista have partially solved this problem but prioritization and personalization (where a system maps available content to user's interests and preferences) of information were absent. This has increased the demand for recommender systems more than ever before.

Recommender systems are information filtering systems that deal with the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information according to user's preferences, interest, or observed behavior about item. Recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile.

Recommender systems are beneficial to both service providers and users. They reduce transaction costs of finding and selecting items in an online shopping environment. Recommendation systems have also proved to improve decision making process and quality. In e-commerce setting, recommender systems enhance revenues, for the fact that they are effective means of selling more products. In scientific libraries, recommender systems support users by allowing them to move beyond catalog searches. Therefore, the need to use efficient and accurate recommendation techniques within a system that will provide relevant and dependable recommendations for users cannot be over-emphasized.

Online book reading and selling websites like Kindle and Goodreads compete against each other on many factors. One of those important factors is their book recommendation system. A book recommendation system is designed to recommend books of interest to the buyer. The purpose of a book recommendation system is to predict buyer's interest and recommend books to them accordingly. A book recommendation system can take into account many parameters like book content and book quality by filtering user reviews.

On the whole, for any learner or casual reader, the main objective of their purpose of search of any book should be satisfied in order to uplift the quality of the customers. These book recommendation systems can be of immense help on e-libraries where the database of books is enormous. The manual search methodology or the logger search technique is highly biased and volatile as they don't much rely on any formulated aspects.

Hence , by using the constructive and concise recommendation engines, users are given a wide range of choice based on their similar interests. The accuracy turns out to be efficient as in terms of recommendation systems rather than any other forms of procedural workflow.

1.2 Objective of the Project

The Main Objective of this Project is to offer the users various kinds of filters which would help them customize their searches and allows them to focus on what path they wish to move ahead.

A complete algorithmic implementation of various kinds of filtering has been implemented so that it enriches the users choice. Variable classes are added separately which takes the user's input book name and suggests various books that are of similar interests and high ratings. Moreover to give the users the visual aid that gathers high response, the Tittle of the book is also displayed to enhance the quality of life.

Significant advances have been made in the field over the past few years, largely due to an unprecedented growth in variety of choices, opinions, etc. It now has countless applications, offering insights for library management, online book store, e-platforms, etc.

1.3 Machine Learning

Machine learning (ML) is the process of using mathematical models of data to help a computer learn without direct instruction. It's considered a subset of artificial intelligence (AI). Machine learning uses algorithms to identify patterns within data and those patterns are then used to create a data model that can make predictions.

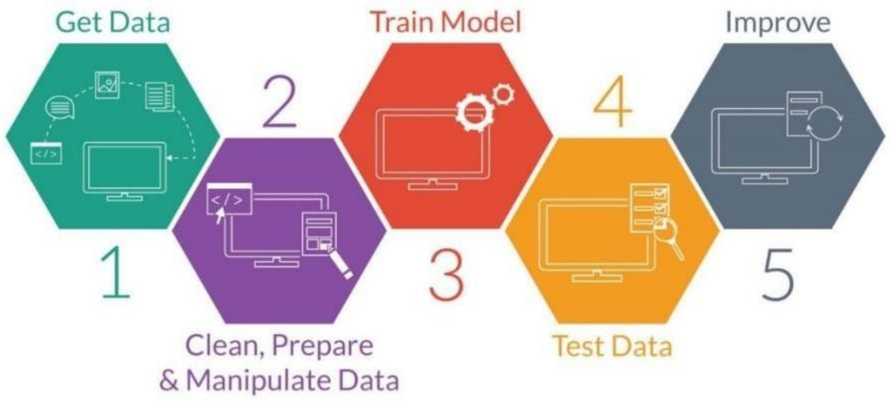


Fig 1.3 Machine Learning Pipeline

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain.

1.4 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner- level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games

CHAPTER – 2
LITERATURE SURVEY

1. Book Recommendation System using Machine learning [Fatima Ijaz (2020)].

Suggestion framework is a common and cold e-commerce issue. Recommendation system performs in many ways including faculty member base on quality, suggestion for reciprocal filtering, and hint for the mix technique. This article proposes a collective suggestion filtering system focused on naive Bayesian approach. The recommendation method does have a good performance, according to both the undertake experimentation, than numerous prior implementations, including the praised k-NN algorithm being used by suggestion especially at longer length.

2. Online Book Recommendation System [Nursultan Kurmashov, Konstantin Latuta, Abay Nussipbekov(2015)]

Moment of the quantum of information in the internet growth veritably fleetly and people need some instruments to find and pierce applicable information. Recommendation systems help to navigate snappily and admit necessary information. Generally they're used in Internet shops to ameliorate the profit. This paper proposes a quick and intuitive book recommendation system that helps compendiums to find applicable book to read. The overall armature is presented with it’s detailed description. We used a cooperative filtering system grounded on Stoner correlation factor. Eventually the results grounded on the online check are handed with some conversations**.**

3. The Design and Implementation of Books Recommendation System[Yongen Liang, ShimingWan (2018)]

Individualized recommendation technology is a new technology which can mine products by using stoner's information, and that match stoner's preferences through a series of algorithms, so as to achieve better recommendation effect. The number of books in university library is adding fleetly. How to find intriguing books from a large number of books is a problem that every anthology is concerned about. In order to help these druggies find the books that they're interested in, this author designs a books recommendation system grounded on cooperative filtering algorithm The system can principally meet the requirements of druggies to recommend functions, and achieved good results.

4. ***Research on Book Recommendation Algorithm Based on Collaborative Filtering and Interest Degree[Balakrishnan Nagaraj, 2021]***

With the increase of library collections, it is difficult for readers to quickly find the books they want when choosing books. Book recommendation system is becoming more and more important. Based on the previous research, this paper proposes a book recommendation algorithm based on collaborative filtering and interest. Take the interest of the book itself as an important measurement index, including the number of searches, borrowing time, borrowing times, borrowing interval, and renewal times. Through the analysis of MAE and RMSE experiments, the results show that the method proposed in this paper converges faster than the traditional method.

CHAPTER – 3
SYSTEM ANALYSIS

3.1 Problem Definition

For any recommendation system, there are basically three broad perspectives of filtering architectures. They are classified as collaborative filtering, content based filtering and hybrid filtering. Each architecture has its own use cases and features.

Users can use book recommendation systems to search and select books from a number of options available on the web or elsewhere electronic sources. They give the user a little bit selection of products that fit the description, given a large group of objects and a description of the user needs. Our system will simply provide recommendations. Recommendations are based on previous user activity, such as purchase, habits, reviews, and likes. These systems gain lot of interest.

In our project, the set-up is executed in Jupyter Notebook which paves a way to use the desired architecture. As each entity are encapsulated on their own domain, they do not interfere in the process of one another. The accuracy of each phase is maintained through this manner.

3.2 Existing System

3.2.1 Methodology

In general, a library has a Book Search System, both search books by title, author, publisher, and book subject. But with so many book search results displayed by the system, sometimes it makes users (library members) find it difficult to choose books that match their profile. For that, we need a system that can provide recommendations in the book search process to provide recommendations systematically based on previous user searches. There are existing book search recommendation system in a desktop-based library using the Python programming language and the MySQL database.

Recommendation system aims to reduce errors in getting the reference books needed. The book search recommendation system uses a user-based collaborative filtering method based on the similarity of one member to another member based on the lending pattern which is grouped based on the subject of the book being sought. This system will display the results of book search recommendations by ranking from highest to lowest and provide book title solutions to users according to their profile. The development of the book search recommendation system uses the System Development life cycle (SDLC) method to design a system with a case study research.

The first stage of system development is the Identification of Problems that occur in the University of Struggle library following the phenomena that occur in the field. The second stage is the analysis of software requirements, hardware, and information needed in developing a system. The third stage is the design of workflow system design recommendations for book search using Unified Modeling Language (UML), database design and designing the interface. The fourth stage is the implementation of the design results in real form using Python as a programming language and MySql as a database system. The last step is testing the book search recommendation system using the Blackbox testing method by Trial and Error.

The existing methodologies follows a destined path that is particular to the developed model which decreases the scalability. As the success of an recommendation system highly scales upon the user's reviews, opinions and responses, a narrower way of establishment brings forth a low scale response.

3.2.2 Disadvantages

- The main shortcoming of the Existing systems was that they are not an wide scaling model.
- The models limits the user's search to a particular extent where a user get same recommendations for different perspectives.
- Unavailability if multiple frameworks causes the issue of poor recommendation in scenarios where the user might what to filter out the cases with different ideologies.
- Limited Datasets tends to cause the problem of over fitting, where the recommendations are repeated regardless of user's opinion.
- The logger based system where the recommendations are based on previous user's usage rate are also not an optimal technique to implement the system.

3.3 Proposed System

In this system, the users have a wide way of options to choose from the following:

- Item-Based Collaborative Filtering
- Content-Based Collaborative Filtering using *Title, Author, Publisher, Category* as features
- Content-Based Collaborative Filtering using *Summary* as a feature
- Custom Recommender

The database file is loaded in the jupyter notebook console and necessary pre-processing is done. Each of these architectures are implemented in the encapsulated manner where the choice of the architecture can be abstract. For a single book instance, the user can have varied recommendations based on what architecture they prefer. The recommended books are displayed on the screen with their Tittles and average user ratings.

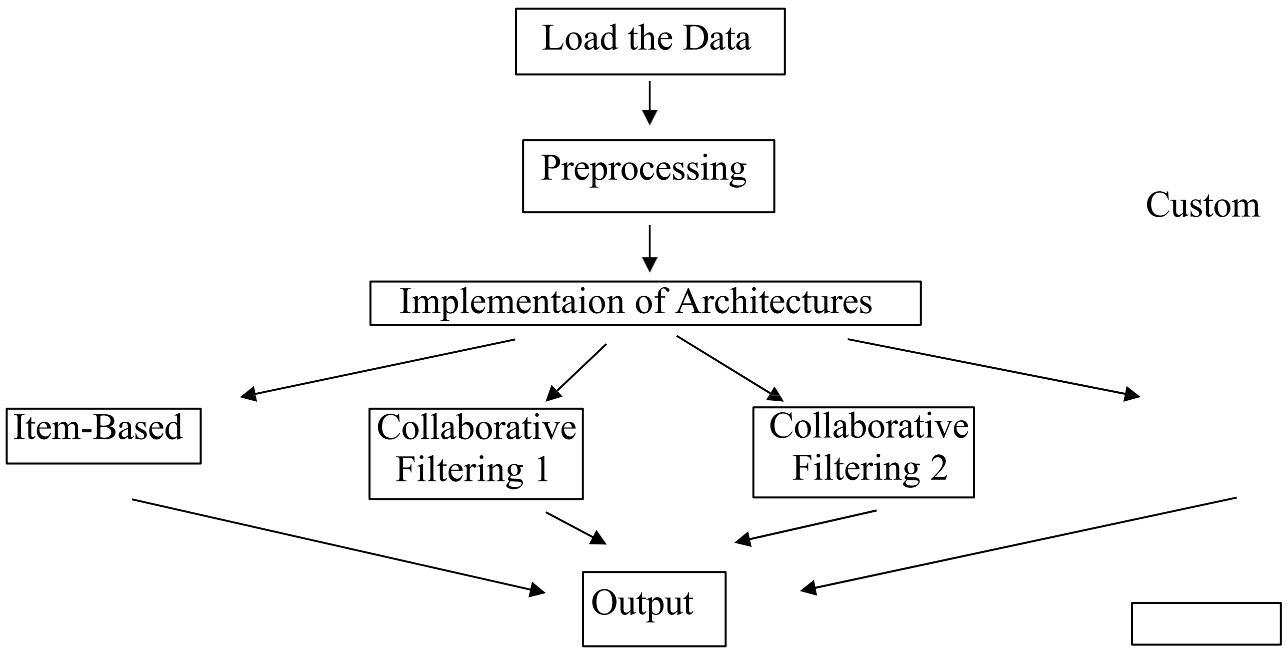

Load the Data
Preprocessing
Custom
Implementaion of Architectures
Item-Based
Collaborative Filtering 1
Collaborative Filtering 2
Output

Fig 3.3 Block Diagram of The Proposed System

3.4 Advantages

- The Main advantage of the Proposed System is that is improves the scalability of the recommendation paradigm as users are offered multiple ways of choosing the filtering method.
- The implementation is such that each method are encapsulated in their own domain, thereby user can take multiple outputs with all the methods without overloading of the process.
- The database is loaded and preprocessed at the start of the program flow because of which there cannot be a problem of over fitting, as updating the database does not hinder the procedural flow.
- Tittles of the books are displayed with the user ratings improvising the interface and quality of life for the users

CHAPTER – 4
SYSTEM DESIGN

4.1 System Architecture

The architecture of the system is designed in such a manner that the initial step highly influences the succeeding steps as preprocessing the data has an immense impact in the final output. This architecture helps in achieving high validation rate as updating the database will not affect the implementations as they are processed forehand, rather than relying on stocked data.

Each of the filtering architecture is unaffected by other and hence the collective output will maintain encapsulation and abstraction.

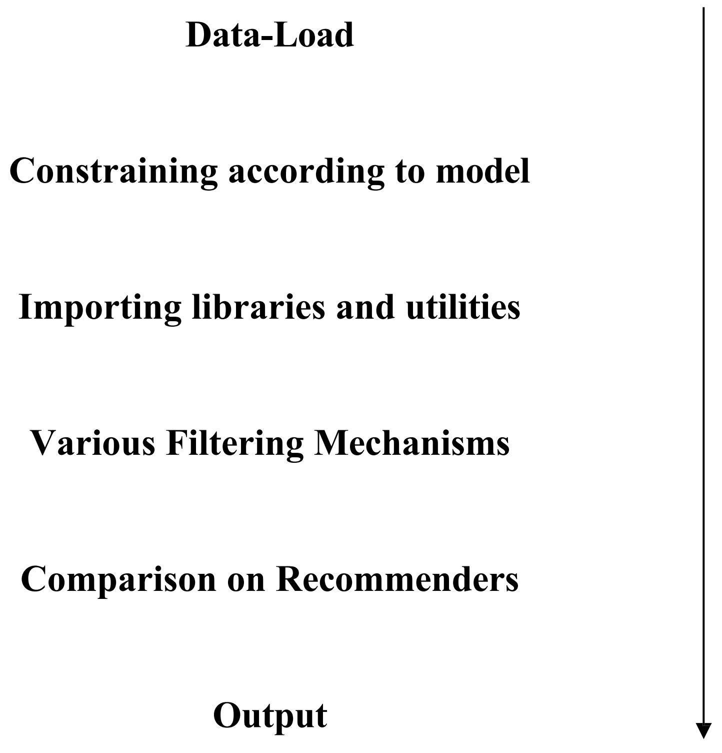

Data-Load
Constraining according to model
Importing libraries and utilities
Various Filtering Mechanisms
Comparison on Recommenders
Output

Fig 4.1System Architecture

The final output is displayed separately for each architecture class with individual interfaces.

4.2 Use case diagram

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roleswithin the system.

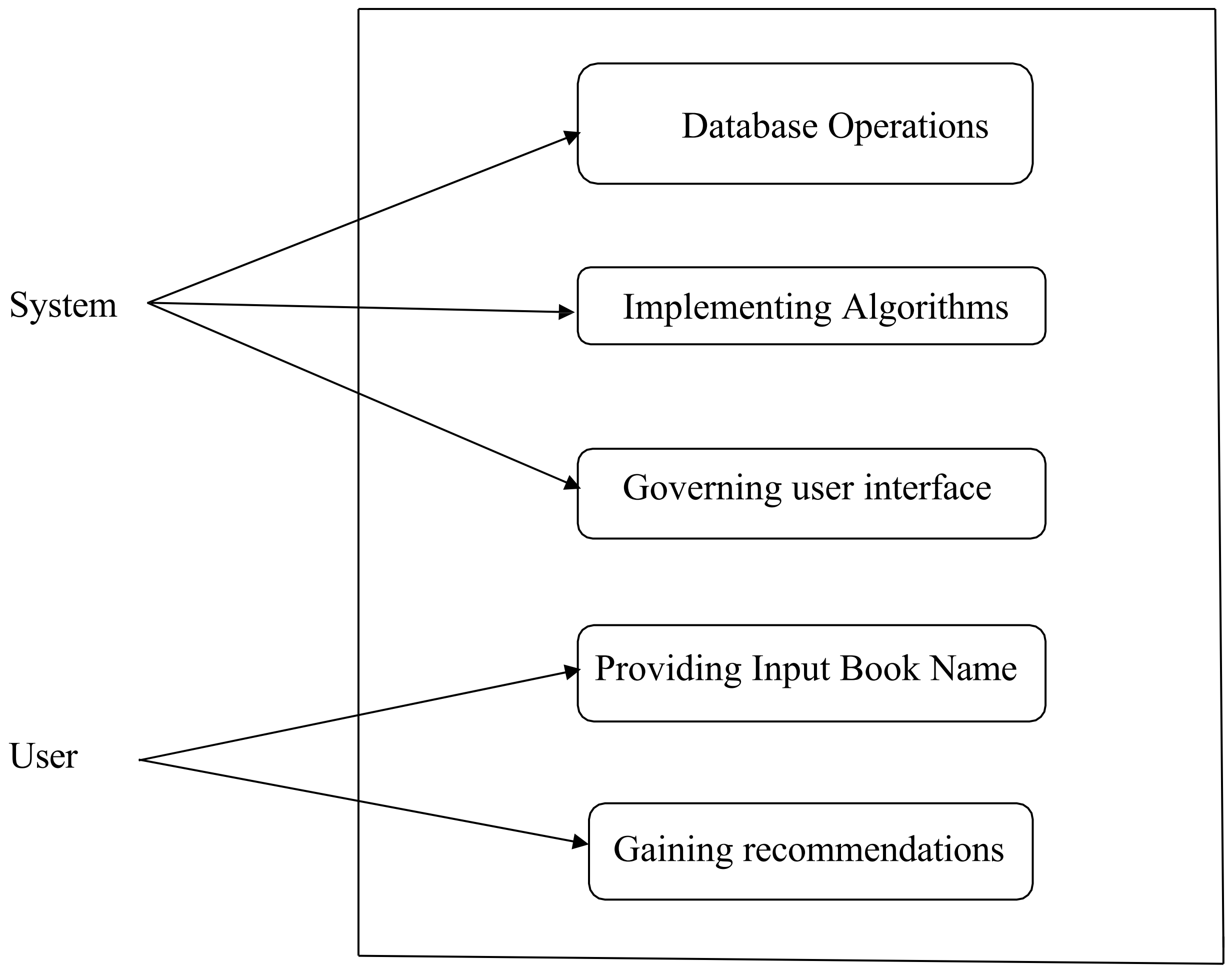


Fig 4.2 Use case diagram

The use-case diagram representing the actions performed by users and system itself is shown above.

4.3 Data flow diagram

Dataflow is often defined using a model or diagram in which the entire process of data movement is mapped as it passes from one component to the next within a program or a system, taking into consideration how it changes form during the process. The flow of data in the recommendation engine is given below

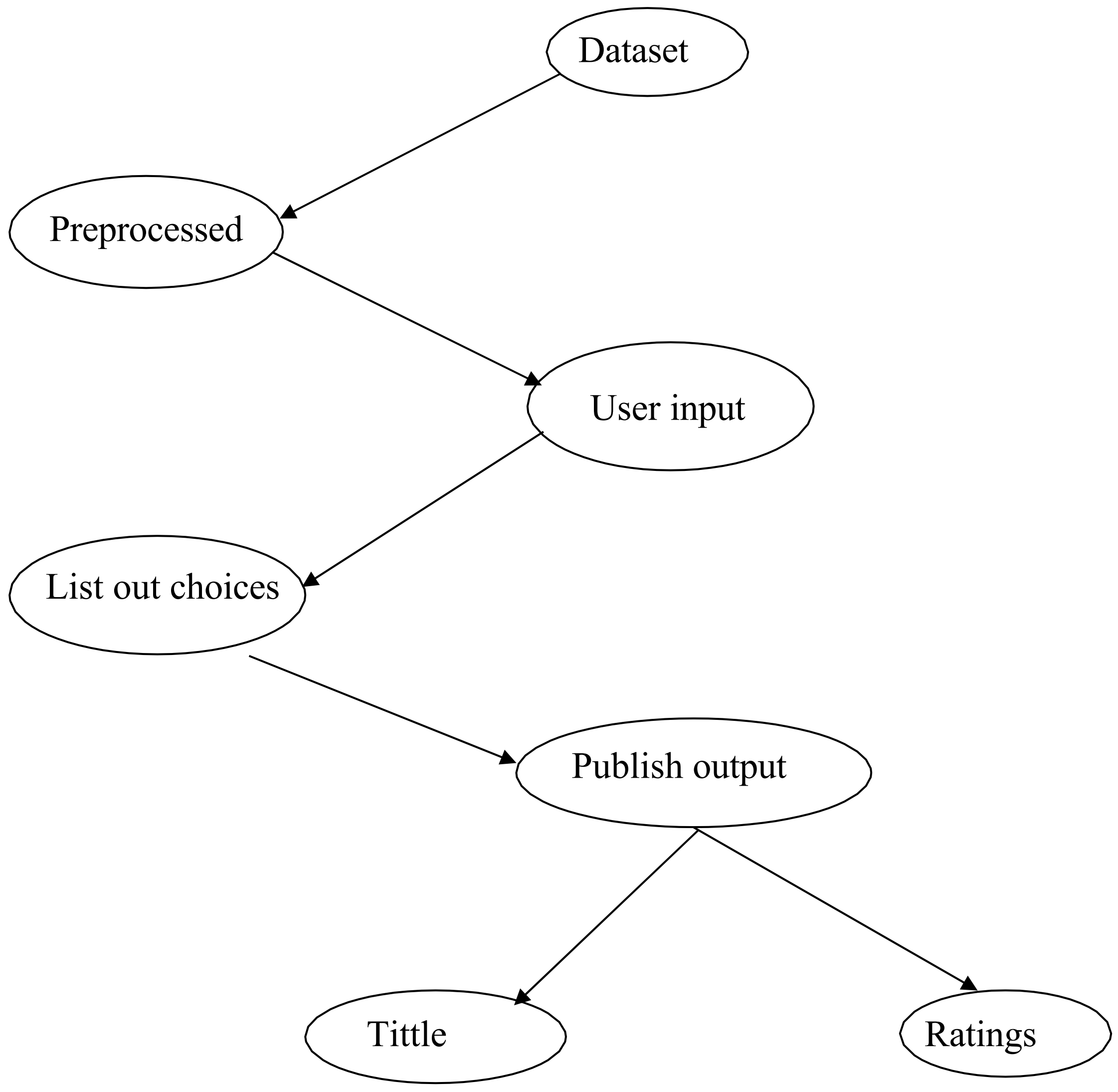


Fig 4.3 Data Flow

CHAPTER – 5
SYSTEM DESCRIPTION

5.1 Source Platform – Colab Notebook

Google Colab, short for Google Colaboratory, offers a cloud-based platform provided by Google that facilitates collaborative Python programming. This platform is particularly favored among data scientists, researchers, and students due to its accessibility and integration with popular libraries like TensorFlow, PyTorch, and OpenCV. Users can write and execute Python code in a collaborative environment, harnessing the power of cloud computing without the need for high-end local hardware.

One of the standout features of Google Colab is its free access. Anyone with a Google account can utilize Colab without any cost, making it an attractive option for individuals and teams on a budget. Moreover, Colab supports Jupyter notebooks, providing users with a familiar interface for writing and running code. These notebooks offer an interactive environment where users can combine code, text, and visualizations in a single document.

Another key advantage of Google Colab is its cloud-based execution. By leveraging Google's powerful cloud infrastructure, users gain access to high-performance computing resources, including GPUs and TPUs. This capability enables tasks such as training complex machine learning models and processing large datasets efficiently, even on relatively modest local machines. Additionally, Colab seamlessly integrates with Google Drive, allowing users to save and share notebooks directly from their Drive accounts. This integration facilitates collaboration and ensures that notebooks are easily accessible from any device with an internet connection, further enhancing the platform's usability and convenience.

5.2 Libraries used in Python

a. Numpy

Numpy is a general-purpose array-processing package. Itprovides a high- performance multidimensional array object, and toolsfor working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses,Numpy can also be used as an efficient multi-dimensional containerof generic data.

Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays canalso be created with the use of various data types such as lists, tuples,etc. The type of the resultant array is deduced from the type of the elements in the sequences.

Numpy is used for dealing with array operations of the database efficiently.

b. Pandas

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance &productivity for users.

Pandas Dataframe is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Dataframe consists of three principal components,the data, rows, and columns.

The entire management of database is done using pandas right away from loading till final processing.

c. nltk

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

d. Requests

Python requests is a library for making HTTP requests. It provides an easy-to-use interface that makes working with HTTP very simple, which means it simplifies the process of sending and receiving data from websites by providing a uniform interface for both GET and POST methods. Some of the benefits of using python requests are that they're fast, support multiple languages, and can be piped into other programs to make processing tasks easier.

CHAPTER – 6
MODULE DESCRIPTION

A. DataBase

In order to create a recommendation engine, the foremost part is the data collection phase where we gather huge amounts of data, in our case, data regarding books which forms the base of our design. The database contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books.

B. Pre-processing

The data is pre-processed in every initialization,hence the updated database can be used without any issues. Here we remove the unused columns, and do some explicit changes like removing 9 from categories and 0 from ratings etcetera, so that the logical constraints are maintained. Only the copy of the original dataset if used and the actual database is kept unmodified.

C. Collaborative Filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. For example, a collaborative filtering recommendation system for preferences in television programming could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes).[3] Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.

D. User Based Collaboative Filtering

As collaborative filtering procures its results from implicit data, it is able to retrieve information that users otherwise might not provide. The first class of collaborative filtering is the user-based approach. This approach narrows down users with the help of collaborative filtering that has similar behaviors, common contacts, and close demographics, and similar consumer behaviors. Social networking sites incorporate this approach to recommend users to other users based on their patterns of behavior. Moreover, this approach is also employed for targeted ads and suggested items based on other users who have similar choices and preferences. Among the various approaches of collaborative filtering, user-based collaborative filtering is the first approach that came into existence.

E. Item Based Collaborative Filtering

A class of collaborative filtering techniques, item-based collaborative filtering refers to the recommendation of items or products using collaborative filtering. By measuring similarity among products and inferring respective ratings, items are recommended to users based on their historical data and interactive history. With the statistical technique of Nearest Neighbour, the technique of Collaborative Filtering in this approach works effectively and presents users with legitimate recommendations that have only led to an increase in consumption.

F. Displaying the Result

The Final task of the recommendation engine is to display the recommendations of the books with their Tittles and ratings.

CHAPTER – 7
PERFORMANCE AND EVALUATION

In this project, we achieved a successful implementation, significant improvements can be made by addressing several key issues. First, a much larger dataset should be designed to improve the model's generality. While we achieved 80% accuracy in test condition, higher the scale of data, the more effective will be the algorithm to provide various accurate suggestions.

Moreover, there can be a separate form made to ask out the opinions of the users upon which the major success of the project depends on. The user satisfaction is a key perspective to keep in mind while analyzing the performance of the recommendation systems.

CHAPTER – 8
CONCLUSION AND FUTURE WORK

In this project, a highly scalable recommendation engine model was proposed where the user could have the liberty of selecting wide range of filtering architectures that includes item based and content based collaborative filtering. The added feature of custom recommender prioritizes the best possible outcomes of all filters. The end result turns out to be the books with their Tittles and ratings of the viewers which enhances the user system interaction. Depending upon what the user intends to filter out, the models works in that manner,

Future work will mainly focus on improving the user interface of the models as the user-friendly interface helps in enriching the model for common audience. A standalone application maybe developed which connects to the local database for data extraction. Also, the database should be made more secure, by using mysql, oracle etc. A web-based application can also be developed with streamlit, flask, etc.

APPENDIX

A. SOURCE CODE

Libraries and Utilities

```
import os
import re
import nltk
import requests
import
warnings
import pandas as pd
import numpy as np

from nltk.corpus import stopwords
nltk.download("stopwords")

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

from PIL import Image
warnings.filterwarnings('ignore')
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Load and Check Data

```
books = pd.read_csv('/content/data/Books Data with Category Language and
Summary/Preprocessed_data.csv')
books.head(3)
```

Preprocessing

```
df = books.copy()
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)

df.drop(columns = ['Unnamed: 0','location','isbn',
                   'img_s','img_m','city','age',
                   'state','Language','country',
                   'year_of_publication'],axis=1,inplace = True) #remove useless cols

df.drop(index=df[df['Category'] == '9'].index, inplace=True) #remove 9 in category

df.drop(index=df[df['rating'] == 0].index, inplace=True) #remove 0 in rating

df['Category'] = df['Category'].apply(lambda x: re.sub('[\W_]+',' ',x).strip())

df.head(2)
```

Item-Based Collaborative Filtering

```
def item_based_recommender(book_title):

    book_title = str(book_title)
    if book_title in df['book_title'].values:

        rating_counts = pd.DataFrame(df['book_title'].value_counts())
        rare_books = rating_counts[rating_counts['book_title'] <= 180].index
        common_books = df[~df['book_title'].isin(rare_books)]

        if book_title in rare_books:

            random = pd.Series(common_books['book_title'].unique()).sample(2).values
            print('There are no recommendations for this book')
            print('Try: \n')
            print('{}'.format(random[0]),'\n')
            print('{}'.format(random[1]),'\n')

        else:
            user_book_df = common_books.pivot_table(index=['user_id'],
                                        columns=['book_title'],
                                        values='rating')

            book = user_book_df[book_title]
            recom_data = pd.DataFrame(user_book_df.corrwith(book). \
                             sort_values(ascending=False)).reset_index(drop=False)

            if book_title in [book for book in recom_data['book_title']]:
                recom_data = recom_data.drop(recom_data[recom_data['book_title'] ==
book_title].index[0])

            low_rating = []
            for i in recom_data['book_title']:
                if df[df['book_title'] == i]['rating'].mean() < 5:
                    low_rating.append(i)

            if recom_data.shape[0] - len(low_rating) > 5:
                recom_data = recom_data[~recom_data['book_title'].isin(low_rating)]

            recom_data = recom_data[0:5]
            recom_data.columns = ['book_title','corr']

            print("Suggested Books: ")
                for i in range(len(recom_data['book_title'].tolist())):
                    rating = round(df[df['book_title'] ==
            recom_data['book_title'].tolist()[i]]['rating'].mean(),1)
                    print(f"--> {recom_data['book_title'].tolist()[i]}
            (Rating :{rating})")
```

```
 else:
       print('Cant find book in dataset, please check spelling')
```

Content-Based Collaborative Filtering

Title, Author, Publisher, Category

```
def content_based_recommender(book_title):

    book_title = str(book_title)
    if book_title in df['book_title'].values:
        rating_counts = pd.DataFrame(df['book_title'].value_counts())
        rare_books = rating_counts[rating_counts['book_title'] <= 100].index
        common_books = df[~df['book_title'].isin(rare_books)]

        if book_title in rare_books:

            random = pd.Series(common_books['book_title'].unique()).sample(2).values
            print('There are no recommendations for this book')
            print('Try: \n')
            print('{}'.format(random[0]),'\n')
            print('{}'.format(random[1]),'\n')

        else:

            common_books = common_books.drop_duplicates(subset=['book_title'])
            common_books.reset_index(inplace= True)
            common_books['index'] = [i for i in range(common_books.shape[0])]
            target_cols = ['book_title','book_author','publisher','Category']
            common_books['combined_features'] = ['
'.join(common_books[target_cols].iloc[i,].values) for i in
range(common_books[target_cols].shape[0])]
            cv = CountVectorizer()
            count_matrix = cv.fit_transform(common_books['combined_features'])
```

```
          cosine_sim = cosine_similarity(count_matrix)
          index = common_books[common_books['book_title'] == book_title]
['index'].values[0]
          sim_books = list(enumerate(cosine_sim[index]))
          sorted_sim_books = sorted(sim_books,key=lambda x:x[1],
                                reverse=True)[1:6]

          books = []
          for i in range(len(sorted_sim_books)):
             books.append(common_books[common_books['index'] ==
sorted_sim_books[i][0]]['book_title'].item())

          print("Suggested Books: ")
          for i in range(len(books)):
               rating = round(df[df['book_title'] == books[i]]
          ['rating'].mean(),1)
                print(f"--> {books[i]} (Rating :{rating})")
   else:

      print('Cant find book in dataset, please check spelling')
```

Content-Based Collaborative Filtering

Summary

```
def content_based_recommender_summary(book_title):

   book_title = str(book_title)
```

```
    if book_title in df['book_title'].values:
        rating_counts = pd.DataFrame(df['book_title'].value_counts())
        rare_books = rating_counts[rating_counts['book_title'] <= 100].index
        common_books = df[~df['book_title'].isin(rare_books)]

        if book_title in rare_books:

            random = pd.Series(common_books['book_title'].unique()).sample(2).values
            print('There are no recommendations for this book')
            print('Try: \n')
            print('{}'.format(random[0]),'\n')
            print('{}'.format(random[1]),'\n')

        else:
            common_books = common_books.drop_duplicates(subset=['book_title'])
            common_books.reset_index(inplace= True)
            common_books['index'] = [i for i in range(common_books.shape[0])]

            summary_filtered = []
            for i in common_books['Summary']:

                i = re.sub("[^a-zA-Z]"," ",i).lower()
                i = nltk.word_tokenize(i)
                i = [word for word in i if not word in set(stopwords.words("english"))]
                i = " ".join(i)
                summary_filtered.append(i)

            common_books['Summary'] = summary_filtered
            cv = CountVectorizer()
            count_matrix = cv.fit_transform(common_books['Summary'])
            cosine_sim = cosine_similarity(count_matrix)
            index = common_books[common_books['book_title'] == book_title]
['index'].values[0]
            sim_books = list(enumerate(cosine_sim[index]))
            sorted_sim_books = sorted(sim_books,key=lambda
x:x[1],reverse=True)[1:6]

            books = []
        for i in range(len(sorted_sim_books)):
            books.append(common_books[common_books['index'] ==
sorted_sim_books[i][0]]['book_title'].item())

        print("Suggested Books: ")
            for i in range(len(books)):
              rating = round(df[df['book_title'] == books[i]]
        ['rating'].mean(),1)
              print(f"--> {books[i]} (Rating :{rating})")
    else:

        print('Cant find book in dataset, please check spelling')
```

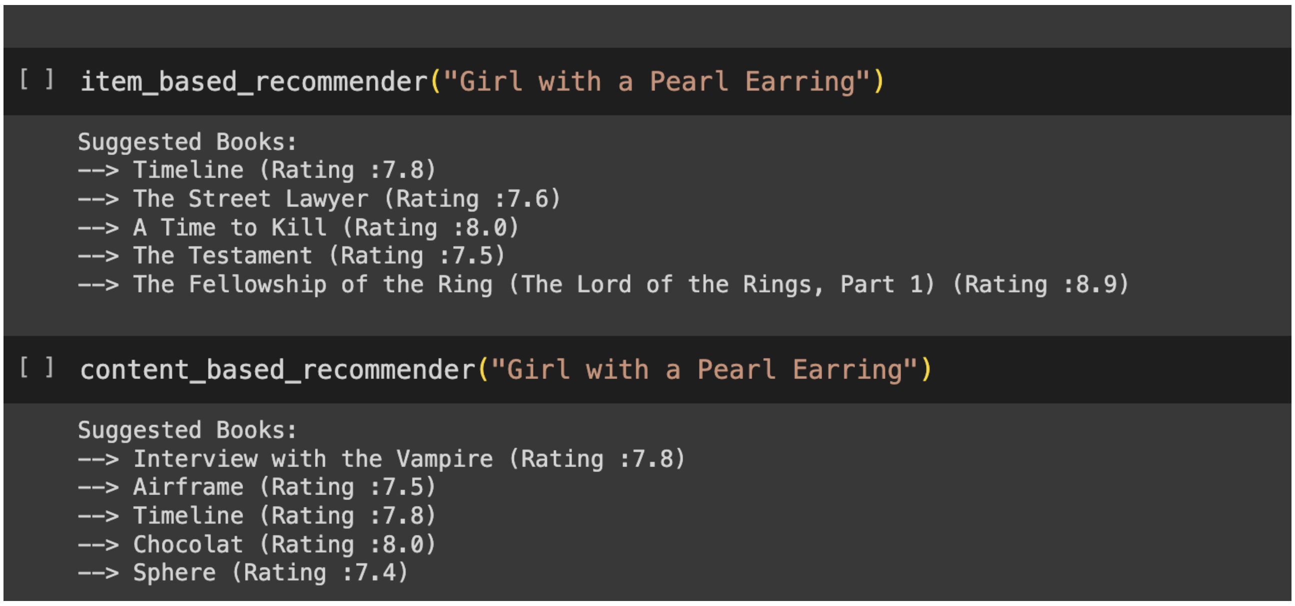
Custom Recommender

```
def custom_recommender(book_title):

    #ITEM-BASED
    book_title = str(book_title)
    if book_title in df['book_title'].values:

        rating_counts = pd.DataFrame(df['book_title'].value_counts())
        rare_books = rating_counts[rating_counts['book_title'] <= 180].index
        common_books = df[~df['book_title'].isin(rare_books)]

        if book_title in rare_books:
            random = pd.Series(common_books['book_title'].unique()).sample(2).values
            print('There are no recommendations for this book')
            print('Try: \n')
            print('{}'.format(random[0]),'\n')
            print('{}'.format(random[1]),'\n')

        else:
            user_book_df = common_books.pivot_table(index=['user_id'],
                                            columns=['book_title'], values='rating')

            book = user_book_df[book_title]
            recom_data = pd.DataFrame(user_book_df.corrwith(book). \
                                 sort_values(ascending=False)).reset_index(drop=False)

            if book_title in [book for book in recom_data['book_title']]:
                recom_data = recom_data.drop(recom_data[recom_data['book_title'] ==
book_title].index[0])

            low_rating = []
            for i in recom_data['book_title']:
                if df[df['book_title'] == i]['rating'].mean() < 5:
                    low_rating.append(i)

            if recom_data.shape[0] - len(low_rating) > 5:
                 recom_data = recom_data[~recom_data['book_title'].isin(low_rating)]

            recom_data = recom_data[0:1]
            recom_data.columns = ['book_title','corr']
            recommended_books = []
            for i in recom_data['book_title']:
                recommended_books.append(i)

            df_new = df[~df['book_title'].isin(recommended_books)]

            #CONTENT-BASED (Title, Author, Publisher, Category)
            rating_counts = pd.DataFrame(df_new['book_title'].value_counts())

            rare_books = rating_counts[rating_counts['book_title'] <= 100].index
            common_books = df_new[~df_new['book_title'].isin(rare_books)]
            common_books = common_books.drop_duplicates(subset=['book_title'])
            common_books.reset_index(inplace= True)
            common_books['index'] = [i for i in range(common_books.shape[0])]
            target_cols = ['book_title','book_author','publisher','Category']
            common_books['combined_features'] = ['
'.join(common_books[target_cols].iloc[i,].values) for i in
range(common_books[target_cols].shape[0])]
            cv = CountVectorizer()
            count_matrix = cv.fit_transform(common_books['combined_features'])
            cosine_sim = cosine_similarity(count_matrix)
            index = common_books[common_books['book_title'] == book_title]
['index'].values[0]
            sim_books = list(enumerate(cosine_sim[index]))
            sorted_sim_books = sorted(sim_books,key=lambda
x:x[1],reverse=True)[1:2]

            books = []
            for i in range(len(sorted_sim_books)):
                books.append(common_books[common_books['index'] ==
sorted_sim_books[i][0]]['book_title'].item())

            for i in books:
                recommended_books.append(i)

            df_new = df_new[~df_new['book_title'].isin(recommended_books)]

            #CONTENT-BASED (SUMMARY)
            rating_counts = pd.DataFrame(df_new['book_title'].value_counts())
            rare_books = rating_counts[rating_counts['book_title'] <= 100].index
            common_books = df_new[~df_new['book_title'].isin(rare_books)]

            common_books = common_books.drop_duplicates(subset=['book_title'])
            common_books.reset_index(inplace= True)
            common_books['index'] = [i for i in range(common_books.shape[0])]

            summary_filtered = []
            for i in common_books['Summary']:

                i = re.sub("[^a-zA-Z]"," ",i).lower()
                i = nltk.word_tokenize(i)
                i = [word for word in i if not word in set(stopwords.words("english"))]
                i = " ".join(i)
                summary_filtered.append(i)

            common_books['Summary'] = summary_filtered
            cv = CountVectorizer()
            count_matrix = cv.fit_transform(common_books['Summary'])
            cosine_sim = cosine_similarity(count_matrix)
            index = common_books[common_books['book_title'] == book_title]
['index'].values[0]
            sim_books = list(enumerate(cosine_sim[index]))
            sorted_sim_books2 = sorted(sim_books,key=lambda
x:x[1],reverse=True)[1:4]
            sorted_sim_books = sorted_sim_books2[:2]
            summary_books = []
            for i in range(len(sorted_sim_books)):
                summary_books.append(common_books[common_books['index'] ==
sorted_sim_books[i][0]]['book_title'].item())

            for i in summary_books:
                recommended_books.append(i)

            df_new = df_new[~df_new['book_title'].isin(recommended_books)]

            #TOP RATED OF CATEGORY
            category = common_books[common_books['book_title'] ==
book_title]['Category'].values[0]
            top_rated = common_books[common_books['Category'] ==
category].groupby('book_title').agg({'rating':'mean'}).reset_index()

            if top_rated.shape[0] == 1:
                recommended_books.append(common_books[common_books['index'] ==
sorted_sim_books2[2][0]]['book_title'].item())
```

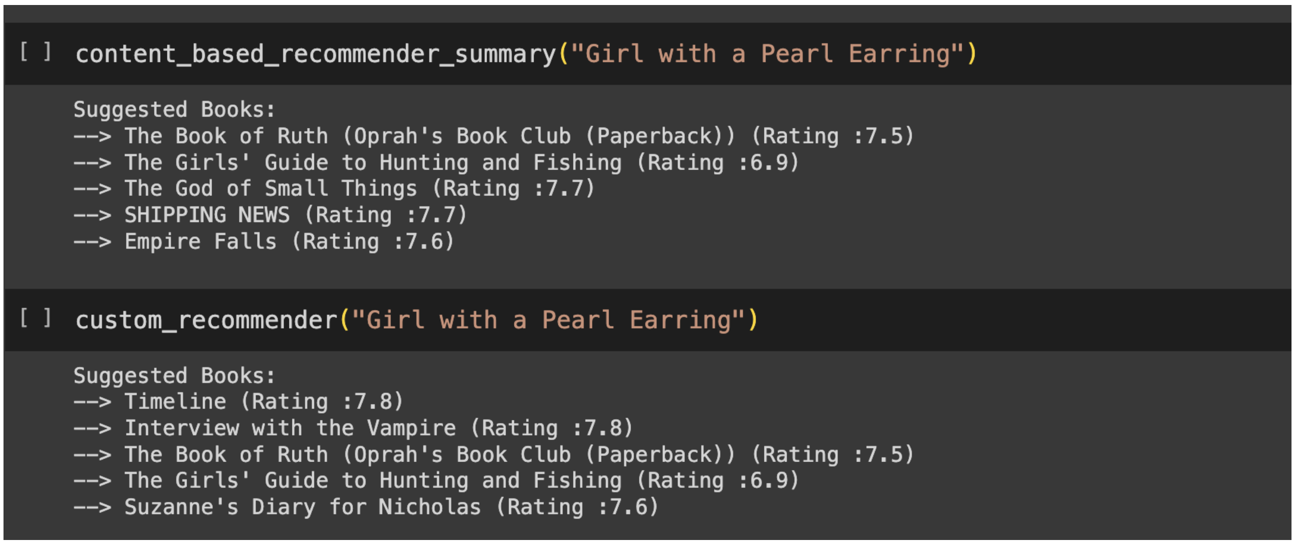
```
          else:
              top_rated.drop(top_rated[top_rated['book_title'] ==
book_title].index[0],inplace=True)
              top_rated = top_rated.sort_values('rating',ascending=False).iloc[:1]
['book_title'].values[0]
              recommended_books.append(top_rated)

          print("Suggested Books: ")
               for i in range(len(recommended_books)):
                 rating = round(df[df['book_title'] ==
          recommended_books[i]]['rating'].mean(),1)
                 print(f"--> {recommended_books[i]} (Rating :
          {rating})")
   else:
      print('Cant find book in dataset, please check spelling')
```

B. SCREENSHOTS

The below screenshots are the sample outcomes of the model we have proposed.



```
[ ] item_based_recommender("Girl with a Pearl Earring")

    Suggested Books:
    --> Timeline (Rating :7.8)
    --> The Street Lawyer (Rating :7.6)
    --> A Time to Kill (Rating :8.0)
    --> The Testament (Rating :7.5)
    --> The Fellowship of the Ring (The Lord of the Rings, Part 1) (Rating :8.9)

[ ] content_based_recommender("Girl with a Pearl Earring")

    Suggested Books:
    --> Interview with the Vampire (Rating :7.8)
    --> Airframe (Rating :7.5)
    --> Timeline (Rating :7.8)
    --> Chocolat (Rating :8.0)
    --> Sphere (Rating :7.4)
```



```
[ ] content_based_recommender_summary("Girl with a Pearl Earring")

    Suggested Books:
    --> The Book of Ruth (Oprah's Book Club (Paperback)) (Rating :7.5)
    --> The Girls' Guide to Hunting and Fishing (Rating :6.9)
    --> The God of Small Things (Rating :7.7)
    --> SHIPPING NEWS (Rating :7.7)
    --> Empire Falls (Rating :7.6)

[ ] custom_recommender("Girl with a Pearl Earring")

    Suggested Books:
    --> Timeline (Rating :7.8)
    --> Interview with the Vampire (Rating :7.8)
    --> The Book of Ruth (Oprah's Book Club (Paperback)) (Rating :7.5)
    --> The Girls' Guide to Hunting and Fishing (Rating :6.9)
    --> Suzanne's Diary for Nicholas (Rating :7.6)
```

REFERENCES

1. Algorithm Based on Collaborative Filtering and Interest Degree

   https://www.hindawi.com/journals/wcmc/2021/7036357/#abstract

2. Book Recommendation System using machine learning

   https://www.ijraset.com/research-paper/book-recommendation-system-using-mi

3. Source platform

   https://www.dominodatalab.com/data-science-dictionary/jupyter-notebook

4. Collaborative Filtering

   https://www.analyticsvidhya.com/blog/2022/02/introduction-to-collaborative-filtering/

5. Inferences

   https://nealanalytics.com/blog/how-to-build-a-personalized-recommender-using-data-science/

   https://thecleverprogrammer.com/2021/01/17/book-recommendation-system/

   https://towardsdatascience.com/my-journey-to-building-book-recommendation-system-5ec959c41847