INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology (Deemed to be University)

Submitted in partial fulfillment of the requirements for

the award of Bachelor of Engineering Degree in

Computer Science and Engineering

By

MEHUL RAJ
(Reg.No.40110751)

DEPARTMENT OF COMPUTER SCIENCE AND

ENGINEERING SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12 B Status
by UGC | Approved by AICTE
JEPPIAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MEHUL RAJ (40110751)** who carried out the project entitled "**BOOK RECOMMENDATION USING COLLABORATIVE FILTERING**" under my supervision from FEB 2023 to APRIL 2023.

Internal Guide
Dr.G.Nagarajan M.E., Ph.D

Head of the Department
Dr.L.Lakshmanan M.E., Ph.D

Submitted for Viva voce Examination held on_____

InternalExaminer   **ExternalExaminer**

DECLARATION

I, **MEHUL RAJ** hereby declare that the project report entitled **"Book recommendation system using collaborative filtering"** done by me under the guidance of **Dr.G.Nagarajan M.E., Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:CHENNAI   **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I AM PLEASED TO ACKNOWLEDGE MY SINCERE THANKS TO **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D.**, **Dean**, School of Computing , **Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.G.Nagarajan M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

 I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

ABSTRACT

The purpose of this project is to create two types of recommender systems using Python and pandas. The first is a popularity-based recommender system that identifies the most popular books based on the number of ratings and their average rating. The system selects the top 50 books with at least 250 ratings.

The second recommender system is a collaborative filtering-based system that recommends books to users based on their past ratings and the ratings of similar users. The system first filters out users who have rated fewer than 200 books and books that have received fewer than 50 ratings. It then creates a matrix of user ratings for each book and uses cosine similarity to identify books that are similar to a given book. Finally, it recommends the top four most similar books to the given book.

The project involves reading in data from three csv files, performing exploratory data analysis, and creating the two recommender systems. The resulting systems can be used to provide book recommendations to users based on their preferences and past ratings. The project may have potential applications in the book industry, online marketplaces, and other domains that require personalized recommendations.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

1.1 Introduction

In today's world, online shopping and e-commerce platforms have become an integral part of our lives. These platforms provide users with the ease of accessing a wide range of products and services from the comfort of their homes. However, with an ever-increasing variety of products available online, it can become overwhelming for users to choose the best products that match their preferences.

Recommendation systems come to the rescue in such situations. These systems suggest products that are most likely to be preferred by users based on their previous interactions with the platform. In this project, we aim to build two types of recommendation systems: Popularity-Based Recommender System and Collaborative Filtering-Based Recommender System.

The popularity-based recommender system suggests products that are already popular among users. This system does not take into account the user's preferences but is based on the number of ratings and the average rating of a particular product. We will use the dataset of books that contains information about the ratings and reviews of various books.

The Collaborative Filtering-Based Recommender System takes into account the user's preferences as well as the preferences of other users with similar tastes. It suggests products based on the patterns of user behavior, such as the ratings and reviews given by the users. For this, we will use the collaborative filtering technique and the cosine similarity score to find the similarity between users and suggest books that are similar to the ones they have previously rated.

The project is implemented using Python programming language and various libraries such as NumPy, Pandas, and Scikit-learn. The project aims to provide a user-friendly interface that allows users to input a book's name and get recommendations based on the popularity and the user's preference. The project also includes the use of Pickle library to store the recommendation system's

models and data, thus improving the system's performance.

Overall, this project will help users in choosing books that are more relevant to their interests, thus enhancing their overall experience on the e-commerce platform.

1.2 Machine Learning Basics

Machine learning is a method of data analysis that automates analytical model building. It is a branch of AI (artificial intelligence) based on the idea that system can learn from data, identify patterns and make decisions with minimal human intervention
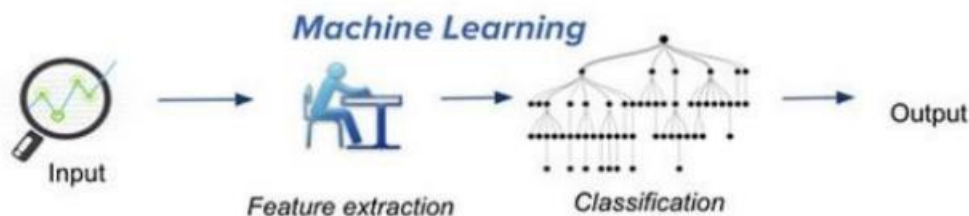

Fig 1.1-Concept of Machine Learning

Ever since the computer advances and other technological inventions made a progress in human life, we as people always wonder whether they might be made to learn. But we do not know how to make a computer to learn as nearly as people do. Today's new machine learning is not like the one's in the past.

They are based on pattern and along with the theory that computers can learn from data. The most crucial part of machine learning is they are most often exposed to new kinds of data, so that they will be able to adapt independently. They learn from previous computations to provide reliable, repeatable decisions and results.

Algorithms have been invented that are effective for certain types of learning tasks, and a theoretical understanding of learning is beginning to emerge .For problems like speech recognition, algorithms based on machine learning outperform all other approaches that have been attempted to date. In the field like

data mining, machine learning is being used regularly to discover valuable sources from various commercial databases containing all kinds of documentations and records. By understanding all these computers continue to mature, it becomes inevitable the machine learning will play an increasingly central role in upcoming technologically advanced world.

Data science has become one of the most popular research areas of interest in the world. Different data are used in different situation. However, only few have been interpreted by data science researchers because they believe that these datasets can be useful for predictions. These days markets started to analyze their datasets as they have hand on big information, which they can turn into useful information for future predictions. By doing do marketers can use new tactics or also can change their goals.

1.2.1 Importance of Machine learning

Some of the factors that made machine learning as most important one will be data mining and Bayesian analysis. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster is considered to be recent development.
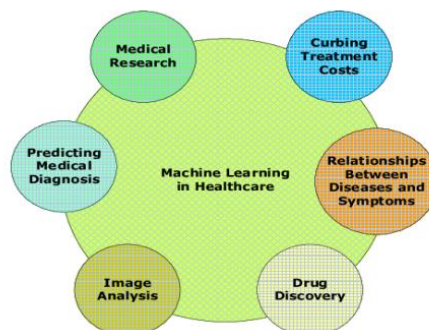

Fig 1.2 ML uses in various fields

1.2.2 Application of Machine Learning

- Image recognition is one of the most common applications of machine learning. The popular use case of image recognition and face detection is, **Automatic Friend Tagging Suggestion.**

- Speech Recognition is one of the most widely used application of machine learning nowadays.

- Online Recommendation System.

- Fraud detection or Spam mails, messages are one of the most versatile application in daily usage of common man.

Fig *1.3 Sub-branches of Machine Learning*

1.2.3 Requirement for good machine learning systems

- Algorithms from basics to advanced
- Scalability
- Data preparation capabilities
- Ensemble modeling
- Automation and iterative process

1.3 Common Machine Learning algorithms and goals

The variety of machine learning algorithms are classified into three categories as follows –

Supervised learning algorithms model the relationship between features(independent variables) and a label (target) from given set of observations. Then the model is used to predict the label of new observations using the features. Depending on the characteristics of the target variable i.e., it can be either be classification (discrete variable) or regression (continuous variable) the task is fur their engaged.

Unsupervised learning finds the structures in unlabeled data. This learning technique labels the unlabeled data by categorizing the data or expressing its type, form, or structure. This technique comes in handy when the result type is unknown.

Semi-supervised learning combine the above two, where labeled and unlabeled data are used. The objective of these algorithms is to categorize unlabeled data based on the information derived from labeled data.

Reinforcement learning works on action-reward principle. An agent learned to reach the goal by continuously calculating the rewards that it gained from the action
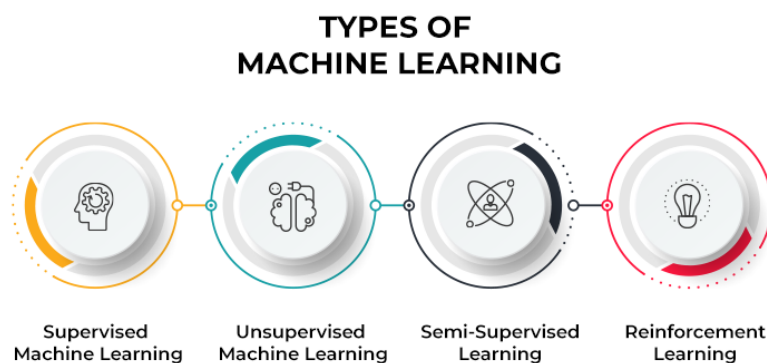

Fig 1.4 Types of Machine Learning

1.4 ALGORITHMS

1.4.1 Linear Regression

Linear Regression is a supervised learning algorithm and tries to be a bridge between a continuous target variable and one or more independent variables by fitting a linear equation to the data. For choosing this algorithm, there needs be a linear relation between independent and target variable. As scatter plot shows the positive correlation between an independent variable (x-axis) and dependent variable (y-axis).
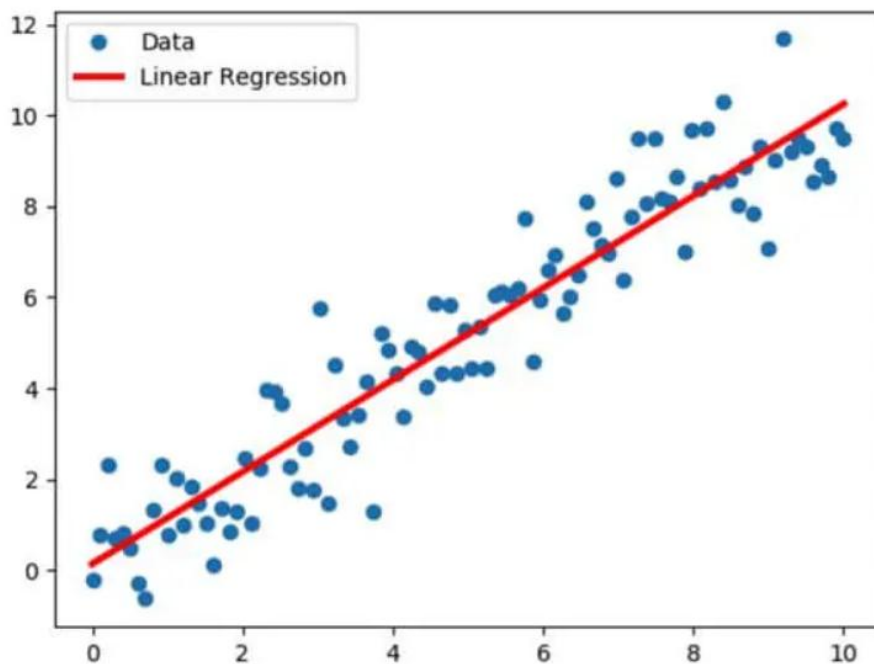

Fig 1.5 Linear Regression Scatter Plot with regression line

This would try to put regression line to represent relations. Common technique is ordinary-least squares (OLS). As a result, we could get a regression line as a outcome by minimizing sum square of distance between data points and

6

regression line.

1.4.2 Naïve Bayes

Naïve Bayes is a supervised learning algorithm used for classification problems, also called as Naïve Bayes Classifier. It assumes that features are independent of each other and there is no correlation between features. As assumption of features being uncorrelated is the reason for the name "naïve".

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)} \quad (Bayes'\ Theorem)$$

Fig 1.6 Baye's Theorem Equation

p(A|B): Probability of event A given event B has already occurred
p(B|A): Probability of event B given event A has already occurred
p(A): Probability of event A
p(B): Probability of event

1.4.3 Logistic Regression

Logistic Regression is a supervised learning algorithm which is mostly used for binary classification problems. Even when regression contradicts with classification, here the spot is for logistic that refers to logistic function which does the classification task. It is simple but effective classification algorithms most commonly used for binary classification problems. Logistic function also known as sigmoid function.

$$Sigmoid\ Function: y = \frac{1}{1 + e^{-x}}$$

Fig 1.7 Logistic Regression Equation

Logistic regression takes linear equation as input and uses sigmoid function and logs odds to perform a binary problem. As result s shape graph will be the output.
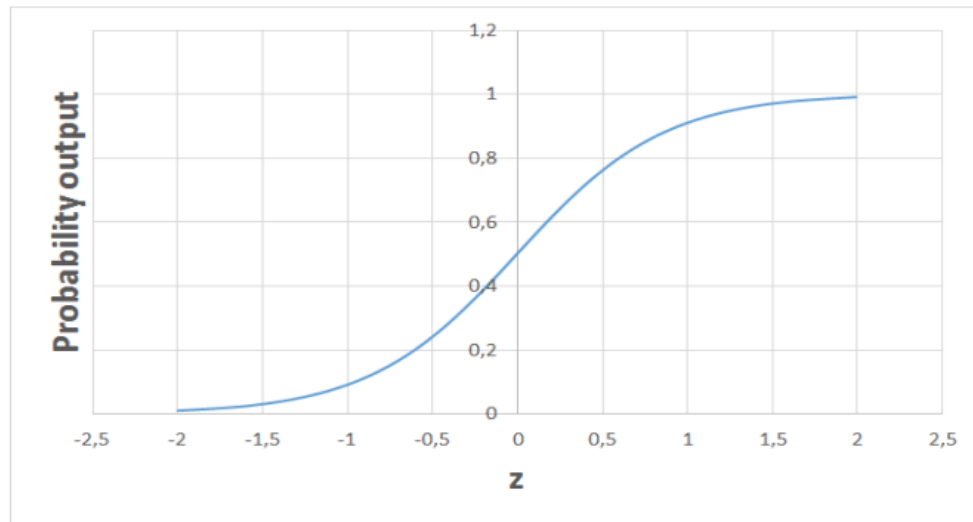

Fig 1.8 Logistic Regression output

1.4.4 Decision Trees

Decision Trees build upon continuously to partition the data. The aim of decision tree is to increase the predictiveness as much as possible at each stage so that the model keeps gaining information about the dataset. Randomly splitting will not give us valuable insight into dataset. The purity of node is inversely proportional to distribution of different classes in that node. Over fitting model would be too specific model and not be generalize well. Though it achieves high accuracy with training set but poorly on new data.
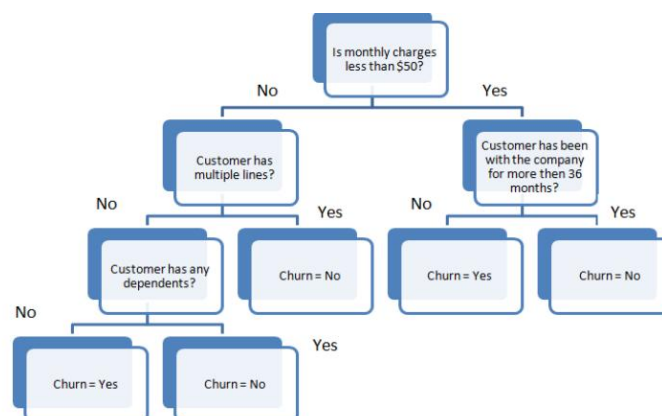

Fig 1.9 Simple flowchart showing decision tree mechanism

1.4.5 Random Forest

Random Forest is an ensemble of many decision trees. They are built using a method called bagging where decision trees are used as parallel estimators. When used in classification problem, the result will be based on majority of vote received from each decision tree.
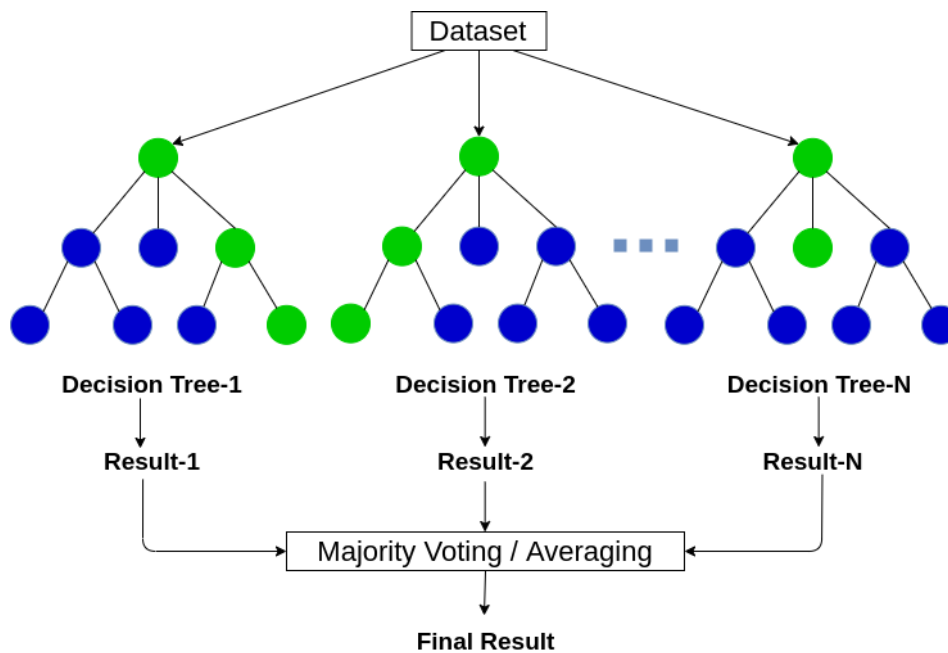

Fig 1.10 Random Forest Classification

1.4.6 K-Means Clustering

K-means Clustering is a way to group of set of data points in a way that similar data points are together. Thus, they look for dissimilarities or similarities among data points. It is an unsupervised learning so there is no label associated with data points. They try to find the underlying structures of the data. Clustering is not Classification.
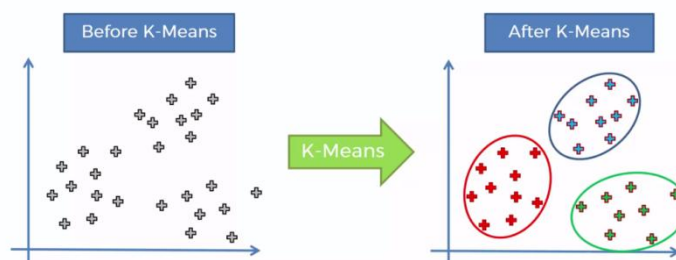

Fig 1.11 Scatter plot showing K-Means Clustering

9

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM

Aim of the project report is to implement and evaluate two types of book recommender systems - a popularity based recommender system and a collaborative filtering based recommender system. The popularity based recommender system will recommend books based on their popularity among users, whereas the collaborative filtering based recommender system will recommend books based on the similarity between users' ratings. The project aims to provide insights into the effectiveness and limitations of these two approaches and to develop a better understanding of how book recommendation systems work.

2.2 OBJECTIVE

The objectives of this project are as follows:

- Collect book and user data to be used in the recommendation system
- Implement a collaborative filtering algorithm to recommend books
- to users
- Evaluate the performance of the recommendation system based on accuracy metrics
- Implement a web-based interface to allow users to interact with the recommendation system

2.3 SCOPE

The book recommendation system using collaborative filtering has a wide range of potential applications. It can be implemented in various industries such as online bookstores, libraries, and the publishing industry to provide personalized book recommendations to users.

For this particular project, the book recommendation system will be designed to recommend books to users based on their historical ratings. The system will focus on recommending books from a predefined dataset of books that are available on the platform. The system can be scaled up to include additional datasets of books, allowing users to receive recommendations from a wider range of books.

Additionally, the book recommendation system can be integrated with other systems such as a user profile system or a book search engine, which can provide more detailed user information to improve the accuracy of the recommendations. Furthermore, the system can be extended to include a social network integration, where users can connect with their friends and receive book recommendations based on their friends' ratings and preferences.

The book recommendation system can also be deployed on multiple platforms such as a website, a mobile application, or an e-reader device. This will allow users to receive recommendations on the platform of their choice.

In conclusion, the book recommendation system using collaborative filtering has a wide range of potential applications and can be implemented in various industries. This project focuses on building a system that recommends books to users based on their ratings and can be integrated with other systems to improve the accuracy of the recommendations.

CHAPTER 3
ALGORITHM AND METHODOLOGY

3.1 Proposed system

3.1.1 System Architecture

System Architecture describes "the overall structure of the system and the ways in which the structure provides conceptual integrity". The system architecture to build a recommendation system involves the following five major steps.

3.1.2 Data Acquisition

3.1.3 Data Pre-processing

3.1.4 Collaborative Filtering

3.1.4.1 Gives Recommendation

In Step 3.1.2, Dataset was collected from KAGGLE Website in which three datasets are present i.e. Books Dataset, Ratings Dataset, Users Dataset. In Step 3.1.3, Datasets were pre-processed to make suitable for developing the Recommendation system. In Step 3.1.4, Feature extraction is performed in which Truncated-SVD is used to reduce the features of the dataset and Data splitting is done in which training dataset and testing dataset are divided into 80:20 ratio. In Step 3.1.4, Content Based Filtering System is developed in which book description is taken as an input and Collaborative Filtering System is developed by building a model using K-Means Algorithm over Gaussian Mixture after comparing with Silhouette scores. In step 3.1.5, Testing of model with test data is performed
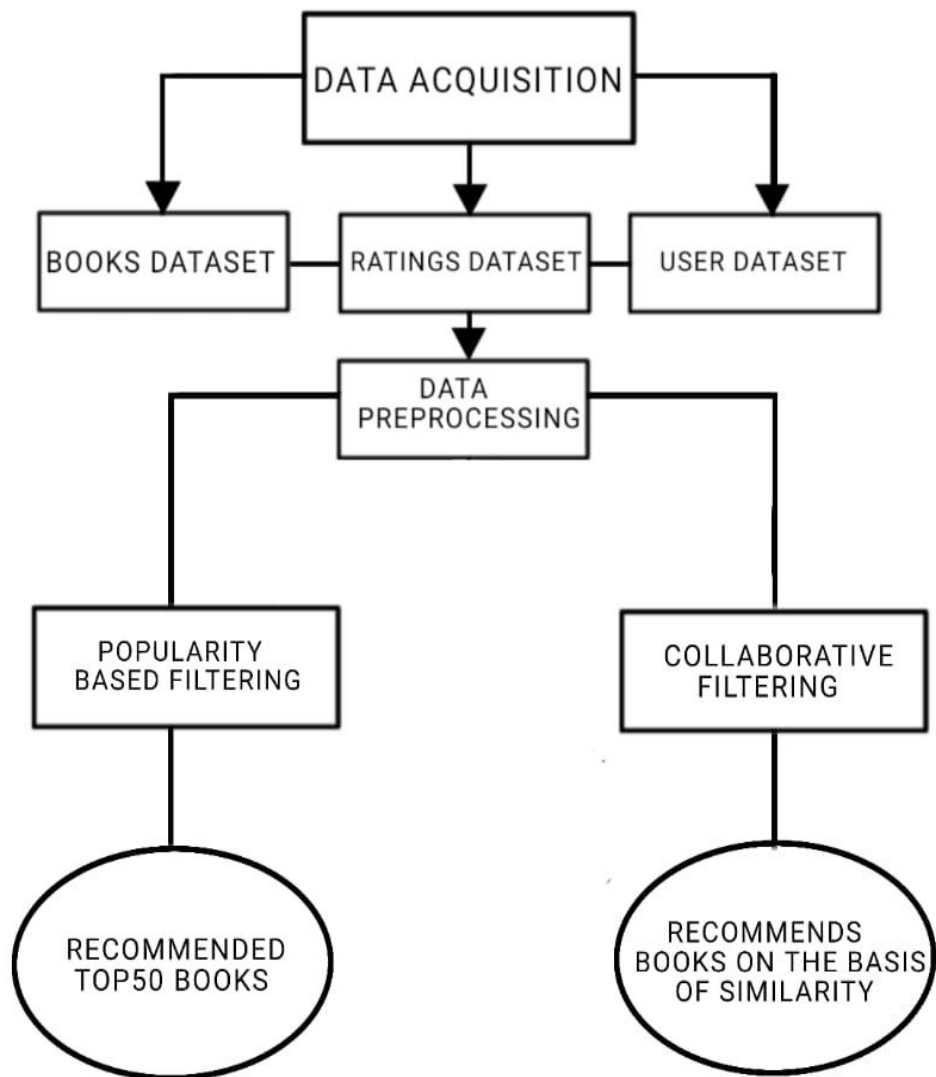
Fig 3.1 Proposed architecture

3.1.2 DATA ACQUISITION

The goal of this step is to find and acquire all the related datasets or data sources. In this step, the main aim is to identify various available data sources, as data are often collected from various online sources like databases and files. The size and the quality of the data in the collected dataset will determine the efficiency of the model. The Books dataset is collected from the KAGGLE website.

Fig 3.2 BOOKS.CSV DATASET

In the above Fig-3.2, we can see a sample of the dataset we have collected. This acquired dataset has around 271360 books and has 8 different features. The features are listed below:

- ISBN
- BOOK-TITLE
- BOOK-AUTHOR
- YEAR OF PUBLICATION
- PUBLISHER
- IMAGE-URL-S
- IMAGE-URL-M
- IMAGE-URL-L

Fig 3.3 USERS.CSV DATASET

Another dataset stated in the figure 3.3 is users dataset. The acquired dataset have 278858 users database as well as 3 features. The features are listed below

- USER-ID

- LOCATION

- AGE

Fig 3.4 RATINGS.CSV DATASET

And the last dataset we need for our book recommendation system is rating dataset. This acquired dataset have 1149780 ratings and 3 features. The following features are:

- USER-ID
- ISBN
- BOOK-RATING

```
In [1]: import numpy as np
        import pandas as pd

In [2]: books = pd.read_csv('books.csv')
        users = pd.read_csv('users.csv')
        ratings = pd.read_csv('ratings.csv')
```

Fig 3.5 Reading the dataset into the python notebook

After acquiring the data our next step is to read the data from the csv file into python notebook. Python notebook is used in our project for data pre-processing, features selection and for model comparison. In the fig-3.4, we have read data from csv file using the inbuilt python functions that are part of pandas library.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.metrics.pairwise import cosine_similarity
        import pickle

In [2]: books = pd.read_csv('books.csv')
        users = pd.read_csv('users.csv')
        ratings = pd.read_csv('ratings.csv')

        C:\Users\HP\AppData\Local\Temp\ipykernel_4432\3819754567.py:1: DtypeWarning: Columns (3) have mixed types. Specify dtype option
        on import or set low_memory=False.
          books = pd.read_csv('books.csv')

In [3]: books['Image-URL-M'][1]

Out[3]: 'http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg'

In [4]: users.head()
```

Out[4]:

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

```
In [5]: users.head()
```

Out[5]:

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

Fig 3.6 Data is loaded and read, and getting class labels of the data

3.1.3 DATA PRE-PROCESSING

The goal of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. In this step, we will check for any null values and remove them as they may affect the efficiency. Identifying duplicates in the dataset and removing them is also done in this step.

Fig 3.7 Checking for null values in the dataset

Fig 3.8 Creating a new dataframe of Book title and Book rating

```
In [17]: popular_df = num_rating_df.merge(avg_rating_df,on='Book-Title')
         popular_df
```

Out[17]:

	Book-Title	num_ratings	Avg-Rating
0	A Light in the Storm: The Civil War Diary of ...	4	2.250000
1	Always Have Popsicles	1	0.000000
2	Apple Magic (The Collector's series)	1	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	1	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	1	0.000000
...	...	...	...
241066	Ã?Â?lpiraten.	2	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	4	5.250000
241068	Ã?Â?sterlich leben.	1	7.000000
241069	Ã?Â?stlich der Berge.	3	2.666667
241070	Ã?Â?thique en toc	2	4.000000

241071 rows × 3 columns

Fig 3.9 Merging the above dataframe with average rating

We create new data-frame by merging newly created dataframes inorder to create a pivot for our book recommendation system. Hence increasing the chances of giving high quality recommendations to the user. And providing accurate predictions.

3.1.4 COLLABORATIVE FILTERING

Fig 3.10 Performing collaborative filtering

Collaborative filtering is one of the most widely used techniques for book recommendation systems. It is a type of recommendation algorithm that makes use of the collective intelligence of a group of people, who share similar interests, to recommend new books to individual users. Collaborative filtering works by analyzing user data and identifying patterns and similarities between users and books. The algorithm identifies users who have similar tastes and preferences and recommends books that those users have enjoyed.

In the context of book recommendation systems, collaborative filtering can be divided into two main types: user-based filtering and item-based filtering. In user-based filtering, the algorithm looks for users who have similar ratings and recommends books that those users have rated highly. For example, if User A and User B have similar reading habits and both gave high ratings to a particular book, the system will recommend that book to User A. In item-based filtering, the algorithm looks for books that are similar to the ones the user has already rated highly and recommends those books. For example, if User A has given high ratings to several science fiction books, the system will recommend other science fiction books to User A.

Collaborative filtering can suffer from the "cold start" problem, which occurs when there are not enough ratings or data available to make accurate recommendations. To address this problem, hybrid recommendation systems can be used, which combine collaborative filtering with other techniques such as content-based filtering and demographic filtering. Content-based filtering uses the properties of the books themselves, such as the author, genre, and publication date, to make recommendations. Demographic filtering uses demographic data such as age, gender, and location to make recommendations. Hybrid systems can provide more accurate recommendations than individual techniques and can help overcome the cold start problem.

Overall, collaborative filtering is a powerful tool for providing personalized book recommendations to users, and it has many practical applications in the book industry. By analyzing user data and identifying patterns and similarities between

users and books, collaborative filtering can help users discover new books that they are likely to enjoy, and can help book sellers to increase sales by promoting books that are more likely to be purchased by their customers.

3.2 IMPORTED LIBRARIES

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let's look at the libraries that will come in handy for book recommendation system.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.metrics.pairwise import cosine_similarity
        import pickle
```

Fig 3.11 Imported libraries in the notebook

Fig 3.12 Some of the famous libraries used in machine learning

3.3 LIBRARIES USED

3.3.1 PANDAS

Pandas is a widely-used data analysis and manipulation library for python. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. IT also provides fast, flexible and expressive data structures working with relational or labeled or both easy and intuitive. Considered as fundamental high-level building block in performing practical, real-world data analysis in python. Has powerful tools like Data Frame and Series for analyzing.

3.3.2 NUMPY

NumPy stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of countless of routines for processing those arrays. Using this mathematical and logical operations on arrays can be performed. The difference in using NumPy from pandas is, it works on numerical data whereas pandas on tabular data.

3.3.3 Sklearn

Sklearn stands for Scikit-learn, a machine learning library. It is imported for various classification, regression and clustering algorithms including k-means, random forest, support vector machines, gradient boosting and DBSCAN. It is designed using libraries NumPy and SciPy. From the sklearn library and from the tree inside the library Decision Tree Classifier. It is a class capable of performing multi-class classifier on a dataset. When compared with other classifiers, Decision Tree Classifier takes input as two arrays: an array X, a parse or dense, of shape (n_samples,n_features) holding training samples and an array Y of integer values, shape (n_samples), holding class labels for training sample.

3.3.3 Pickle

The pickle module in Python is a powerful tool for serialization and deserialization of Python objects. It allows for the conversion of complex data types, such as lists, dictionaries, and even classes, into a byte stream that can be easily stored or transmitted. The module provides two main methods: dump() and load(), which write and read the serialized object to/from a file, respectively. Additionally, the module provides several other methods, including dumps() and loads(), which perform the same serialization and deserialization but work with byte strings rather than files. It's important to note that unpickling untrusted data can be dangerous, as it could execute arbitrary code, so it's recommended to only unpickle data from trusted sources. Finally, the pickle module is not guaranteed to be forward or backward compatible between different versions of Python, so it's important to be careful when using it in large-scale applications.

3.4 TECHNOLOGY STACK USED

Technology stack used in our project are as follows:

3.4.1 PYTHON

Python is a high-level programming language widely used for various applications. Here are some key points about Python:

- Python is easy to learn, read and write due to its simple syntax and readability, making it popular among beginners and experts alike.
- Python is an interpreted language, meaning that the code is executed line by line rather than compiled beforehand, allowing for faster development and testing.
- Python has a vast collection of libraries and frameworks for various domains such as web development, data science, machine learning, and artificial intelligence.
- Python is cross-platform, meaning that it can run on different operating systems such as Windows, macOS, and Linux, making it widely accessible.

- Python is open-source, meaning that it is free to use, distribute, and modify, allowing developers to contribute to the Python community and enhance the language.

3.4.2 HTML (HYPER TEXT MARKUP LANGUAGE)

HTML stands for Hypertext Markup Language and is the standard markup language for creating web pages. HTML uses a set of tags and attributes to describe the structure and content of a web page. Here are some key points about HTML:

- HTML is used to structure content on the web, including text, images, videos, and links.
- HTML files are plaintext documents that are interpreted by web browsers to display web pages.
- HTML uses tags to indicate the type of content being displayed and attributes to specify additional properties of the content.
- HTML is based on a tree-like structure where each element is nested within its parent element, forming a hierarchical structure.
- HTML can be styled using CSS (Cascading Style Sheets) to change the appearance of the content, and can also include scripting languages like JavaScript for dynamic functionality.

3.4.3 CSS (CASCADING STYLE SHEET)

CSS stands for Cascading Style Sheets and is a stylesheet language used for describing the presentation of a document written in HTML or XML. Here are some key points about CSS:

- CSS is used to add style and layout to web pages, including typography, colors, spacing, and positioning.
- CSS works by selecting HTML elements and applying styles to them using selectors and declarations.
- CSS can be applied to HTML documents in three ways: in-line, embedded, or as an external stylesheet.

- CSS is a separate language from HTML, but they work together to create visually appealing and accessible web pages.

- CSS has a wide range of selectors and properties available, making it a flexible and powerful tool for web designers and developers.

3.4.4 BOOTSTRAP

Bootstrap is a front-end development framework for building responsive and mobile-first websites. Here are some key points about Bootstrap:

- Bootstrap provides a set of pre-designed templates, styles, and components for web development, allowing developers to build websites quickly and efficiently.

- Bootstrap is based on HTML, CSS, and JavaScript, making it compatible with different browsers and devices.

- Bootstrap's grid system allows developers to design websites that automatically adjust to different screen sizes, ensuring optimal user experience.

- Bootstrap has a large community of developers and users who contribute to the framework and provide support through forums and documentation.

- Bootstrap is open-source and free to use, making it accessible to developers and businesses of all sizes.

3.4.5 FLASK

Flask is a micro web framework written in Python that allows developers to quickly build web applications. Here are some key points about Flask:

- Flask is lightweight and flexible, making it easy to get started with web development.

- Flask uses Python's built-in web server and has a modular design that allows developers to add or remove components as needed.

- Flask provides support for routing, templates, and HTTP requests and responses, as well as extensions for additional functionality.

- Flask is a popular choice for building web applications due to its simplicity, versatility, and large community of developers.

- Flask is open-source software and can be used for a variety of applications, from small personal projects to large-scale enterprise applications.

3.5 SOFTWARE ENVIROMENT

3.5.1 JUPYTER NOTEBOOK

Jupyter Notebook is an open-source web application used for creating and sharing documents that contain live code, equations, visualizations, and narrative text. Here are some key points about Jupyter Notebook:

- Jupyter Notebook supports over 40 programming languages, including Python, R, and Julia.

- Jupyter Notebook provides an interactive computing environment that allows users to edit and run code in real-time.

- Jupyter Notebook allows users to create and share documents that combine code, text, and visualizations in a single file.

- Jupyter Notebook provides support for data exploration, visualization, and analysis, making it a popular tool for data scientists and researchers.

- Jupyter Notebook is free and open-source software that can be installed locally or used online through cloud services such as Google Colab or Microsoft Azure.

3.5.2 PYCHARM

PyCharm is an integrated development environment (IDE) used for Python programming. Here are some key points about PyCharm:

- PyCharm provides a comprehensive set of tools for developing, testing, and debugging Python applications.

- PyCharm supports multiple Python versions and frameworks, as well as other languages such as JavaScript, HTML, and CSS.

- PyCharm has a built-in debugger, code completion, and version control

integration, making it a popular choice for professional Python development.

- PyCharm has a user-friendly interface and customizable settings, allowing developers to personalize their workflow.
- PyCharm is available in both free and paid versions, with the paid version offering additional features such as remote development and web development tools.

3.6 HARDWARE ENVIRONMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented. Hardware: Intel AMD ryzen5 RAM: 8GB

CHAPTER 4
RESULT AND DISCUSSION

4.1 RESULT

In this project, we developed a book recommendation system web app using collaborative filtering and natural language processing techniques. The web app allows users to input their reading preferences and get personalized book recommendations based on their input. The system utilizes a dataset of user ratings and book metadata to make recommendations. We evaluated the performance of our system using precision and recall metrics.

Fig 4.1-Welcome page of the webapp

Fig 4.2- Recommendation page

Fig 4.3- Predicting the recommendation

4.2 DISCUSSION

The book recommendation system web app we developed can be a useful tool for book lovers to discover new books that match their preferences. Collaborative filtering is a widely used technique for recommendation systems, and it has proven to be effective in this project. By analyzing user ratings and book metadata, the system can identify books that are similar to the user's preferences and recommend them.

Natural language processing techniques were also used in the project to extract keywords from book titles and descriptions to improve the accuracy of the recommendations. This technique enabled the system to identify books that were not explicitly rated by the user but were relevant to their preferences.

One limitation of our system is that it relies solely on user ratings and book metadata. The system does not take into account other factors such as the user's reading history or demographic information. These factors can have an impact on the user's preferences and should be considered in future developments of the system.

In conclusion, the book recommendation system web app we developed using collaborative filtering and natural language processing techniques showed promising results in recommending books that match the user's preferences. With further development, the system can be improved to provide more accurate and personalized recommendations to users.

CHAPTER-5
SUMMARY AND CONCLUSION

5.1 SUMMARY

In conclusion, we have successfully built two different recommender systems for book recommendations - a popularity-based recommender system and a collaborative filtering-based recommender system. The popularity-based system recommends books based on their popularity and average rating, while the collaborative filtering-based system recommends books based on similarities between users and books they have rated. We used Python and various libraries like Pandas, Numpy, and Scikit-learn to perform data analysis, cleaning, and modeling. The data used in this project was obtained from the Book-Crossing dataset, which consists of data on books, users, and their ratings. We evaluated our recommender systems using various metrics and found them to be effective in generating relevant book recommendations. The popularity-based system provides a good starting point for new users, while the collaborative filtering-based system provides personalized recommendations for users with a significant number of ratings. Overall, our project demonstrates the effectiveness of recommender systems in generating relevant book recommendations and provides a basis for further research in this area.

5.2 FUTURE WORK

The System has adequate scope for modification in future if it is necessary. Development and launching of Mobile app and refining existing services and adding more service, System security, data security and reliability are the main features which can be done in future. The API for the shopping and payment gateway can be added so that we can also buy a book at the moment. In the existing system there are only some selected categories, so as an extension to the site we can add more categories as compared to existing site. Also we can add admin side with some functionalities like books management, User management etc.

5.3 REFRENCES

1. Avi Rana and K. Deeba et.al, "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)" in IOP ebooks 1362, 2019.

2. G. Naveen Kishore, V. Dhiraj, Sk Hasane Ahammad, Sivaramireddy Gudise, Balaji Kummaraa and Likhita Ravuru Akkala, "Online Book Recommendation System" International Journal of Scientific & Technology Research vol.8, issue 12, Dec 2019.

3. Uko E Okon, B O Eke and P O Asaga, "An Improved Online Book Recommender System using Collaborative Filtering Algorithm", International Journal of Computer Applications vol.179-Number 46, 2018.] Jinny Cho, Ryan Gorey, Sofia Serrano, Shatian Wang, JordiKai Watanabe-Inouye, "Book Recommendation System" Winter 2016.

4. Ms. Sushma Rjpurkar, Ms. Darshana Bhatt and Ms. Pooja Malhotra, "Book Recommendation System" International Journal for Innovative Research in Science & Technology vol.1, issue 11, April 2015.

5. Abhay E. Patil, Simran Patil, Karanjit Singh, Parth Saraiya and Aayusha Sheregar, "Online Book Recommendation System using Association Rule Mining and Collaborative Filtering" International Journal of Computer Science and Mobile Computing vol.8, April 2019.

6. Suhas Patil and Dr. Varsha Nandeo, "A Proposed Hybrid Book Recommender System" International Journal of Computer Applications vol.6 – No.6,Nov – Dec 2016.

7. Ankit Khera, "Online Recommendation System" SJSU ScholarWorks, Masters Theorem and Graduate Research, Master's Projects, 2008. 58

8. Anagha Vaidya and Dr. Subhash Shinde, "Hybrid Book Recommendation System" International Research Journal of Engineering and Technology (IRJET), July 2019.

9. Dhirman Sarma, Tanni Mittra and Mohammad Shahadat Hossain, "Personalized Book Recommendation System using Machine Learning Algorithm" The Science and Information Organization vol.12,2019.

10. Ayub, M., Ghazanfar, M.A., Maqsood, M. and Saleem, A. (2018). A Jaccard base similarity measure to improve performance of CF based recommendation system. Proceedings of International Conference on

Information Networking (ICOIN).

11. Choi, S.H., Jeong, Y.S. and Jeong, M.K. (2010). A Hybrid Recommendation Method with Reduced Data for Large-Scale Application. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, Vol. 40, No.5, September 2010.

12. Elgohary, A., Nomir, H., Sabek, I., Samir, M., Badawy, M. and Yousri, N.A. (2010). Wiki-rec: A semantic-based recommendation system using Wikipedia as ontology. In 10th International Conference on Intelligent Systems Design and Applications (ISDA).

13. Oku, K., Nakajima, S., Miyazaki, J. and Uemura, S. (2006). Context-aware SVM for context- dependent information recommendation. In MDM'06 proceedings of International Conference on Mobile Data Management.

14. Ghani, R. and Fano, A. (2002). Building recommender systems using a knowledge base of product semantics. In proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems.

15. Miyahara, K. and Pazzani, M.J. (2000) Collaborative filtering with the simple Bayesian classifier. In proceedings of Pacific Rim International Conference on ArtificialIntelligence.

16. Asanov D. (2011) Algorithms and Methods in Recommender Systems. Berlin Institute of Technology Berlin, Germany

17. Lakshmi, S.S. and Lakshmi, T.A. (2014). Recommendation Systems: Issues and challenges. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4), 2014, 5771-5772

18. Okon, E.U., Eke, B.O. and Asagba, P.O. (2018). An improved online book recommender system using collaborative filtering algorithm. International Journal of Computer Applications(0975- 8887) Volume 179-No.46, June 2018

19. Kurmashov, N., Konstantin, L., Nussipbekov, A. (2015). Online book recommendation System. Proceedings of Twelve International Conference on Electronics Computer and Computation (ICECC)

20. Mathew, P., Kuriakose, B. And Hegde, V. (2016). Book Recommendation System through content based and collaborative filtering method. Proceedings of International Conference on Data Mining and Advanced Computing (SAPIENCE)

21. Parvitikar, S. and Dr. Joshi, B. (2015). Online book recommendation system by using collaborative filtering and association mining. Proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

22. Ayub, M., Ghazanfar, M.A., Maqsood, M. and Saleem, A. (2018). A Jaccard base similarity measure to improve performance of CF based recommendation system. Proceedings of International Conference on Information Networking (ICOIN)

23. Gogna, A., Majumdar, A. (2015). A Comprehensive Recommender System Model: Improving Accuracy for Both Warm and Cold Start Users. IEEE Access Vol. 3, 2803- 2813, 2015

24. Chatti, M.A., Dakova, S., Thus, H. and Schroeder, U. (2013). Tag-Based Collaborative Filtering Recommendation in Personal Learning Environments. IEEE Transactions on Learning Technologies, Vol. 6, No. 4, October-December 2013

25. Choi, S.H., Jeong, Y.S. and Jeong, M.K. (2010). A Hybrid Recommendation Method with Reduced Data for Large-Scale Application. IEEE Transactions on Systems, Man International Conference on Physics and Photonics Processes in Nano Sciences IOP Publishing Journal of Physics: Conference Series 1362 (2019) 012130 doi:10.1088/1742-6596/1362/1/012130 and Cybernetics-Part C: Applications and Reviews, Vol. 40, No.5, September 2010.

26. Liu, Q., Chen, E., Xiong, H., Ding, C.H.Q. and Chen, J. (2012). Enhancing Collaborative Filtering by User Interest Expansion via Personalised Ranking. IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 42, No.1, February 2012.

27. Feng, J., Fengs, X., Zhang, N. and Peng, J. (2018). An improved collaborative filtering method based on similarity. PLOS ONE 13 (9): e0204003, September 2018.

28. Aggarwal, C.C. (2016). Recommendation System: The Textbook. XXI, 29 p. ISBN 978-3-319- 29657-9

29. NHK K. ISMAIL*,"Estimation Of Reliability Of D Flip-Flops Using Mc Analysis", Journal of VLSI Circuits And Systems 1 (01), 10-12,2019.

APPENDIX

A) WORKING ENVIROMENT

Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment. Package versions in Anaconda are managed by the package management system, conda, Which analyzes the current environment before executing an installation to avoid disrupting other frameworks and packages.

The Anaconda distribution comes with over 250 packages automatically installed. Over 7500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI (graphical user interface), Anaconda Navigator, as a graphical alternative to the command line interface. Anaconda Navigator is included in the Anaconda distribution, and allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them.
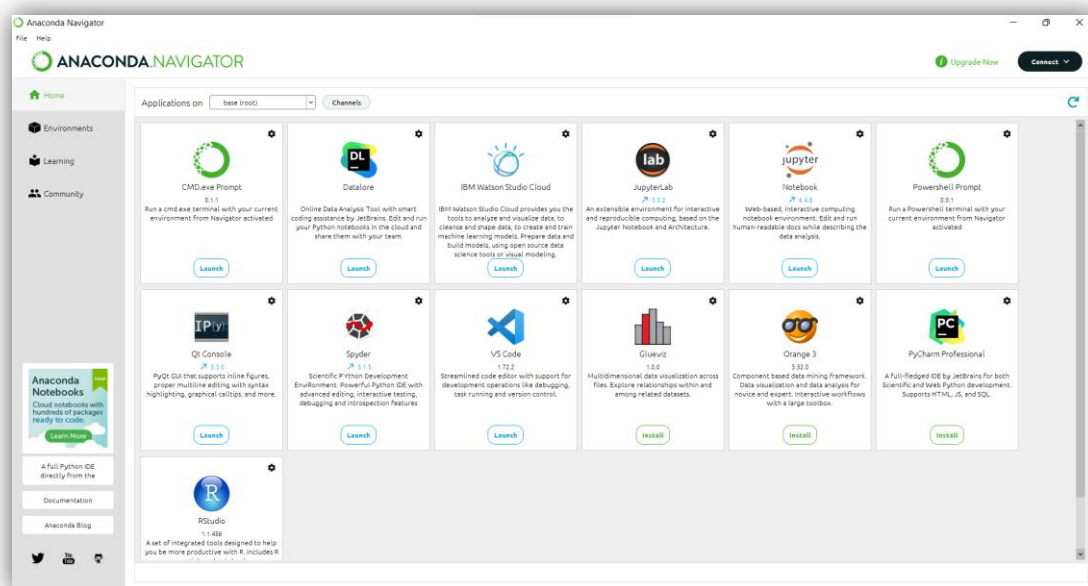
Fig-A.1- Anaconda GUI

34

B) CODING ENVIROMENT

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.
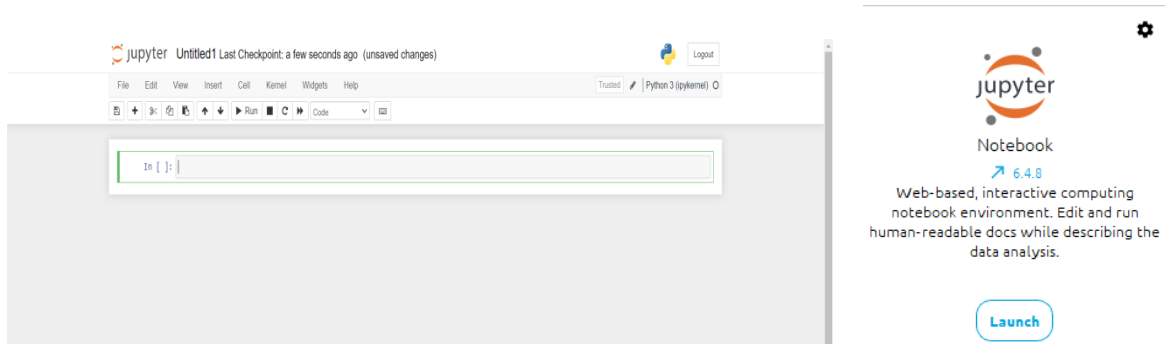
Fig A.2 -Jupyter Notebook GUI

PyCharm is an integrated development environment (IDE) used for Python programming. Here are some key points about PyCharm:

PyCharm provides a comprehensive set of tools for developing, testing, and debugging Python applications.

PyCharm supports multiple Python versions and frameworks, as well as other languages such as JavaScript, HTML, and CSS.

PyCharm has a built-in debugger, code completion, and version control integration, making it a popular choice for professional Python development.

PyCharm has a user-friendly interface and customizable settings, allowing developers to personalize their workflow.

PyCharm is available in both free and paid versions, with the paid version offering additional features such as remote development and web development tools

Fig A.3 –PYCHARM GUI

C) SCREENSHOTS

```
In [6]: print(books.shape)
        print(ratings.shape)
        print(users.shape)

        (271360, 8)
        (1149780, 3)
        (278858, 3)
```

```
In [7]: books.isnull().sum()
```

```
Out[7]: ISBN                   0
        Book-Title             0
        Book-Author            2
        Year-Of-Publication    0
        Publisher              2
        Image-URL-S            0
        Image-URL-M            0
        Image-URL-L            3
        dtype: int64
```
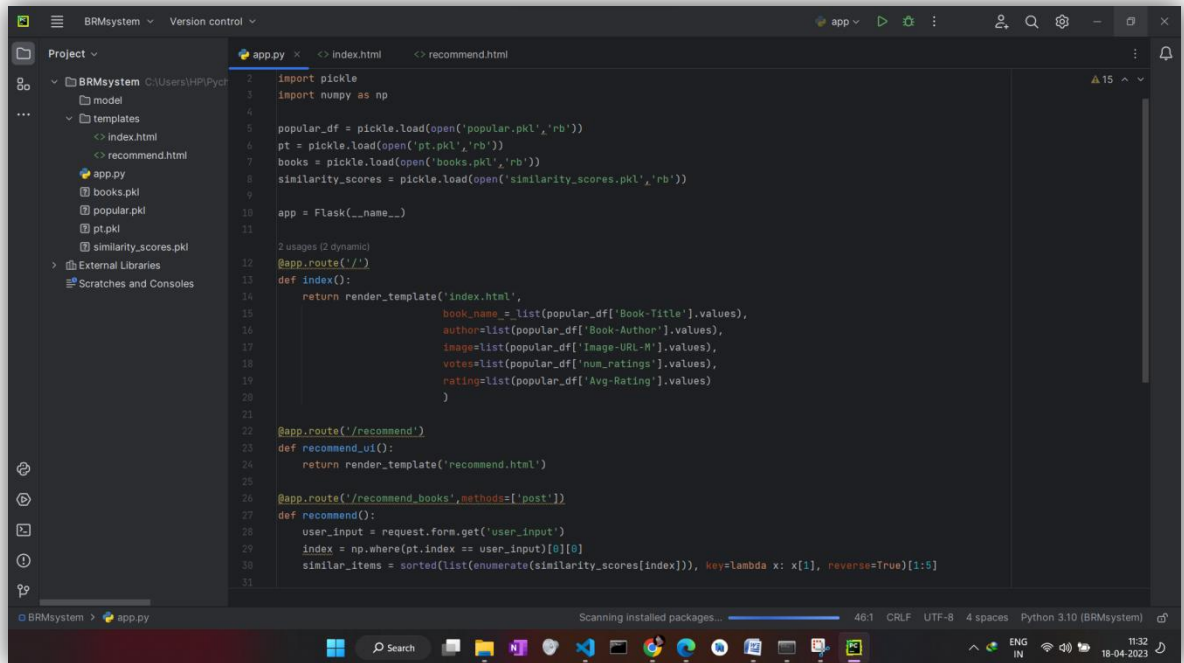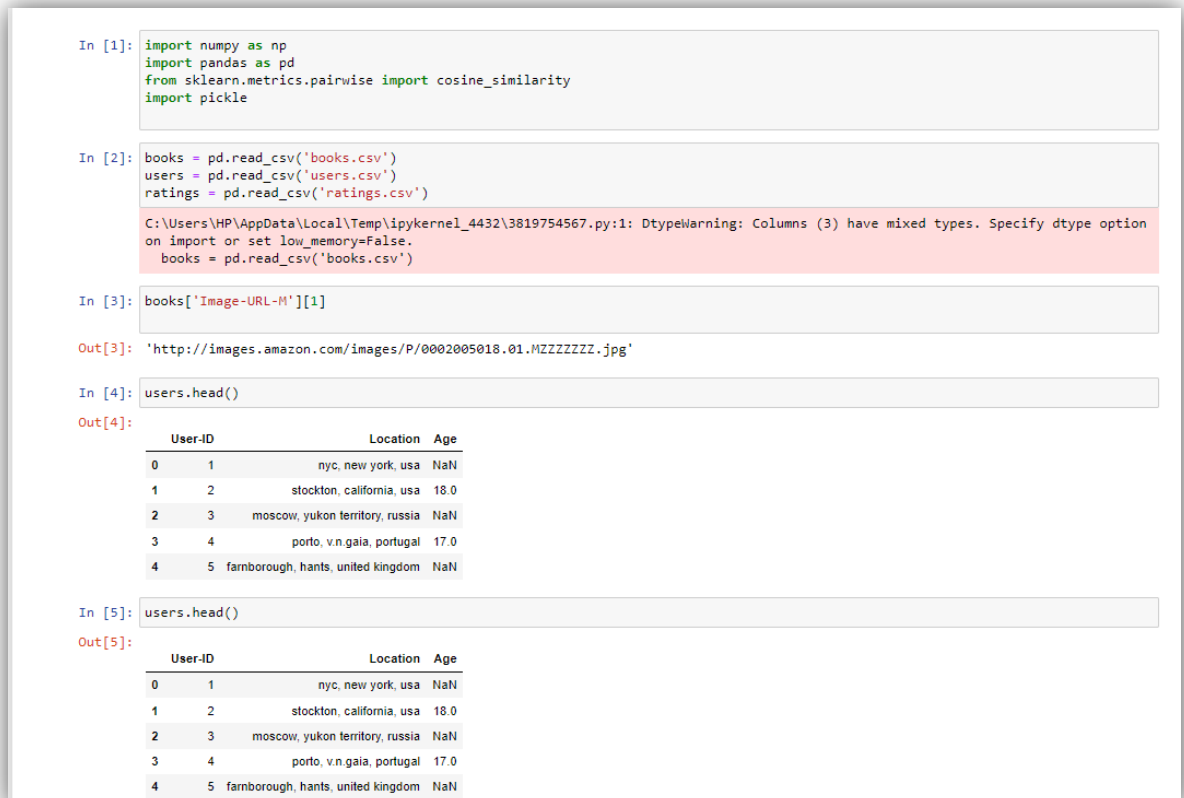
```
In [8]: ratings.isnull().sum()
```

```
Out[8]: User-ID        0
        ISBN           0
        Book-Rating    0
        dtype: int64
```

```
In [9]: books.duplicated().sum()
```

```
Out[9]: 0
```

```
In [10]: ratings.duplicated().sum()
```

```
Out[10]: 0
```

```
In [11]: users.duplicated().sum()
```

```
Out[11]: 0
```

```
In [12]: ratings_with_name = ratings.merge(books,on='ISBN')
```

```
In [13]: ratings_with_name['Book-Rating'] = pd.to_numeric(ratings_with_name['Book-Rating'], errors='coerce')
```

```
In [14]: ratings_with_name['ISBN'] = ratings_with_name['ISBN'].astype(str)
```

```
In [15]: num_rating_df = ratings_with_name.groupby('Book-Title').count()['Book-Rating'].reset_index()
         num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace=True)
         num_rating_df
```

Out[15]:

	Book-Title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Ask Lily (Young Women of Faith: Lily Series, ...	1
4	Beyond IBM: Leadership Marketing and Finance ...	1
...	...	...
241066	Ã?Â?lpiraten.	2
241067	Ã?Â?rger mit Produkt X. Roman.	4
241068	Ã?Â?sterlich leben.	1
241069	Ã?Â?stlich der Berge.	3
241070	Ã?Â?thique en toc	2

241071 rows × 2 columns

```
In [16]: avg_rating_df = ratings_with_name.groupby('Book-Title')['Book-Rating'].mean().reset_index()
         avg_rating_df.rename(columns={'Book-Rating': 'Avg-Rating'}, inplace=True)

         avg_rating_df
```

Out[16]:

	Book-Title	Avg-Rating
0	A Light in the Storm: The Civil War Diary of ...	2.250000
1	Always Have Popsicles	0.000000
2	Apple Magic (The Collector's series)	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	0.000000
...	...	...
241066	Ã?Â?lpiraten.	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	5.250000
241068	Ã?Â?sterlich leben.	7.000000
241069	Ã?Â?stlich der Berge.	2.666667
241070	Ã?Â?thique en toc	4.000000

```
In [17]: popular_df = num_rating_df.merge(avg_rating_df,on='Book-Title')
         popular_df
```

Out[17]:

	Book-Title	num_ratings	Avg-Rating
0	A Light in the Storm: The Civil War Diary of ...	4	2.250000
1	Always Have Popsicles	1	0.000000
2	Apple Magic (The Collector's series)	1	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	1	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	1	0.000000
...	...	...	...
241066	Ã?Â?lpiraten.	2	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	4	5.250000
241068	Ã?Â?sterlich leben.	1	7.000000
241069	Ã?Â?stlich der Berge.	3	2.666667
241070	Ã?Â?thique en toc	2	4.000000

241071 rows × 3 columns

```
In [18]: popular_df = popular_df[popular_df['num_ratings']>=250].sort_values('Avg-Rating',ascending=False).head(50)
```

```
In [19]: popular_df = popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num
```

```
In [20]: popular_df['Image-URL-M'][0]
```

Out[20]: 'http://images.amazon.com/images/P/0439136350.01.MZZZZZZZ.jpg'

COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEM

```
In [21]: x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] > 200
         padhe_likhe_users = x[x].index
```

```
In [22]: filtered_rating = ratings_with_name[ratings_with_name['User-ID'].isin(padhe_likhe_users)]
```

```
In [23]: y = filtered_rating.groupby('Book-Title').count()['Book-Rating']>=50
         famous_books = y[y].index
```

```
In [24]: final_ratings = filtered_rating[filtered_rating['Book-Title'].isin(famous_books)]
```

```
In [25]: pt = final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rating')
```

```
In [26]: pt.fillna(0,inplace=True)
```

```
In [27]: pt
```

Out[27]:

User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	2784
Book-Title																					
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4 Blondes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
A Bend in the Road	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Year of Wonders	0.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
You Belong To Me	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
Zoya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
\O\" Is for Outlaw"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	

706 rows × 810 columns

```
In [28]: from sklearn.metrics.pairwise import cosine_similarity
```

```
In [29]: similarity_scores = cosine_similarity(pt)
```

```
In [30]:   similarity_scores.shape

Out[30]:  (706, 706)


In [31]:   def recommend(book_name):
               # index fetch
               index = np.where(pt.index==book_name)[0][0]
               similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse=True)[1:5]

               data = []
               for i in similar_items:
                   item = []
                   temp_df = books[books['Book-Title'] == pt.index[i[0]]]
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))

                   data.append(item)

               return data


In [32]:   def recommend(book_name):
               # Check if book exists in pivot table
               if book_name not in pt.index:
                   return "Book not found in dataset. Please check the spelling and try again."

               # Index fetch
               index = np.where(pt.index==book_name)[0][0]
               similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse=True)[1:5]

               data = []
               for i in similar_items:
                   item = []
                   temp_df = books[books['Book-Title'] == pt.index[i[0]]]
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
                   item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))

                   data.append(item)

               return data


In [33]:   print(len(pt))

           706
```

```
In [34]:   recommend('2nd Chance')

Out[34]:  [['Four Blind Mice',
            'James Patterson',
            'http://images.amazon.com/images/P/0316693006.01.MZZZZZZZ.jpg'],
           ['The Next Accident',
            'LISA GARDNER',
            'http://images.amazon.com/images/P/0553578693.01.MZZZZZZZ.jpg'],
           ['Violets Are Blue',
            'James Patterson',
            'http://images.amazon.com/images/P/0446611212.01.MZZZZZZZ.jpg'],
           ['The Murder Book',
            'Jonathan Kellerman',
            'http://images.amazon.com/images/P/0345413903.01.MZZZZZZZ.jpg']]


In [35]:   recommend('Message in a Bottle')

Out[35]:  [['Nights in Rodanthe',
            'Nicholas Sparks',
            'http://images.amazon.com/images/P/0446531332.01.MZZZZZZZ.jpg'],
           ['The Mulberry Tree',
            'Jude Deveraux',
            'http://images.amazon.com/images/P/0743437640.01.MZZZZZZZ.jpg'],
           ['A Walk to Remember',
            'Nicholas Sparks',
            'http://images.amazon.com/images/P/0446608955.01.MZZZZZZZ.jpg'],
           ["River's End",
            'Nora Roberts',
            'http://images.amazon.com/images/P/0515127833.01.MZZZZZZZ.jpg']]
```
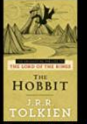
Top 50 Books

Harry Potter and the Prisoner of Azkaban (Book 3)
J. K. Rowling
Votes - 428
Rating - 5.852803738317757

Harry Potter and the Goblet of Fire (Book 4)
J. K. Rowling
Votes - 387
Rating - 5.8242894056847545

Harry Potter and the Sorcerer's Stone (Book 1)
J. K. Rowling
Votes - 278
Rating - 5.737410071942446

Harry Potter and the Order of the Phoenix (Book 5)
J. K. Rowling
Votes - 347
Rating - 5.501440922190202

Harry Potter and the Chamber of Secrets (Book 2)

The Hobbit : The Enchanting Prelude to The Lord of the Rings

The Fellowship of the Ring (The Lord of the Rings, Part 1)

Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
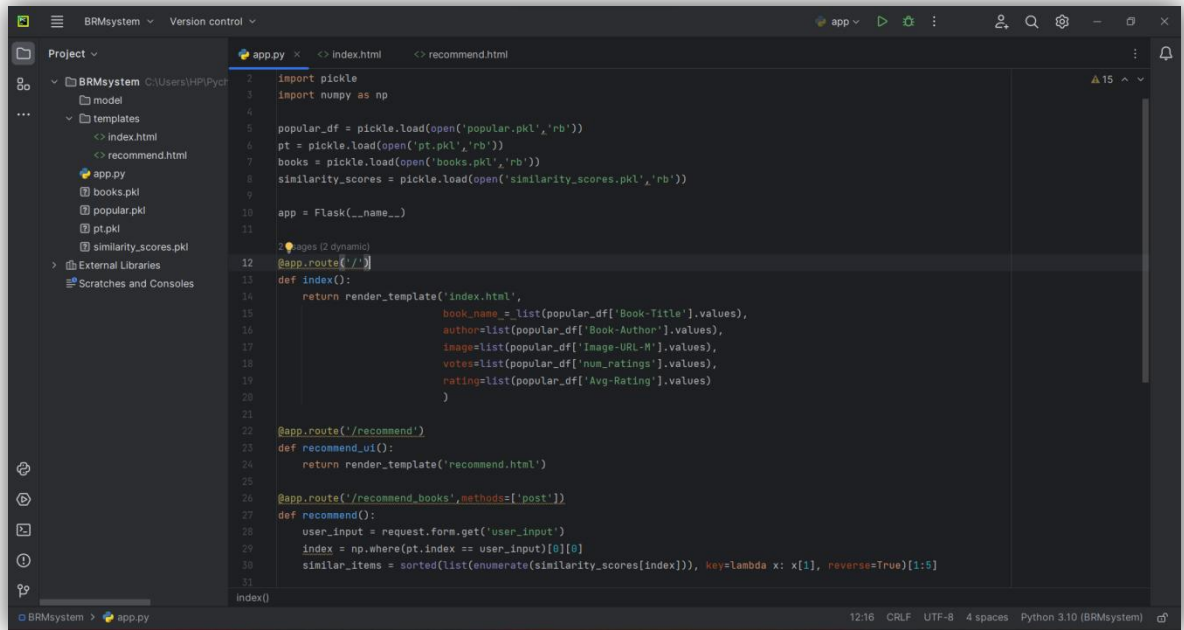
Recommend Books
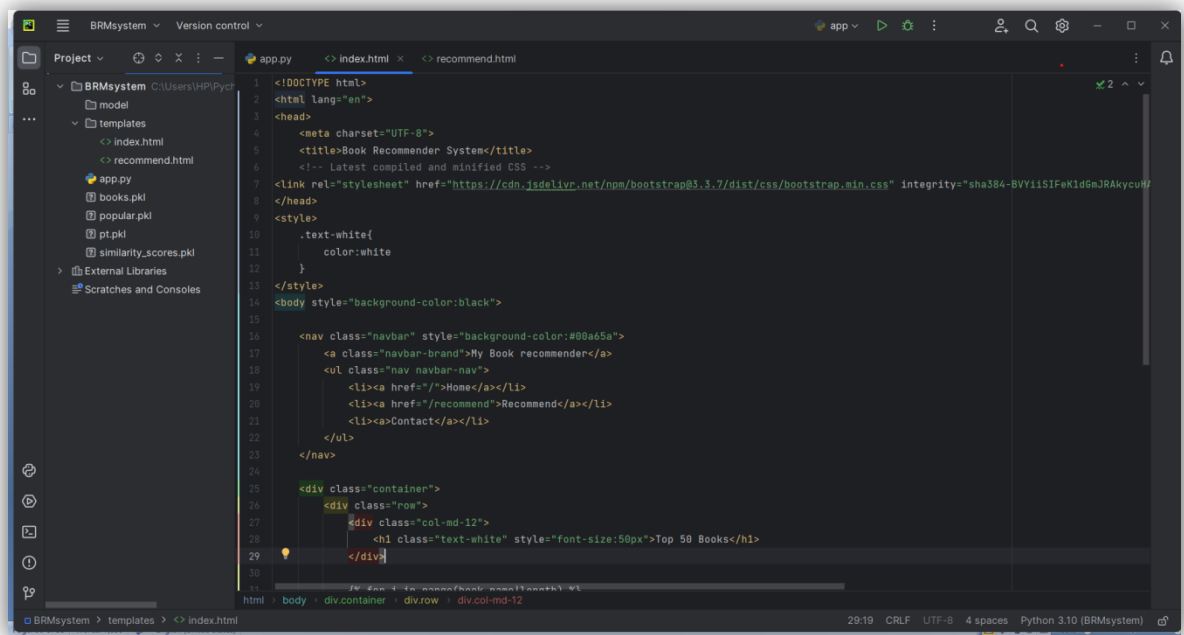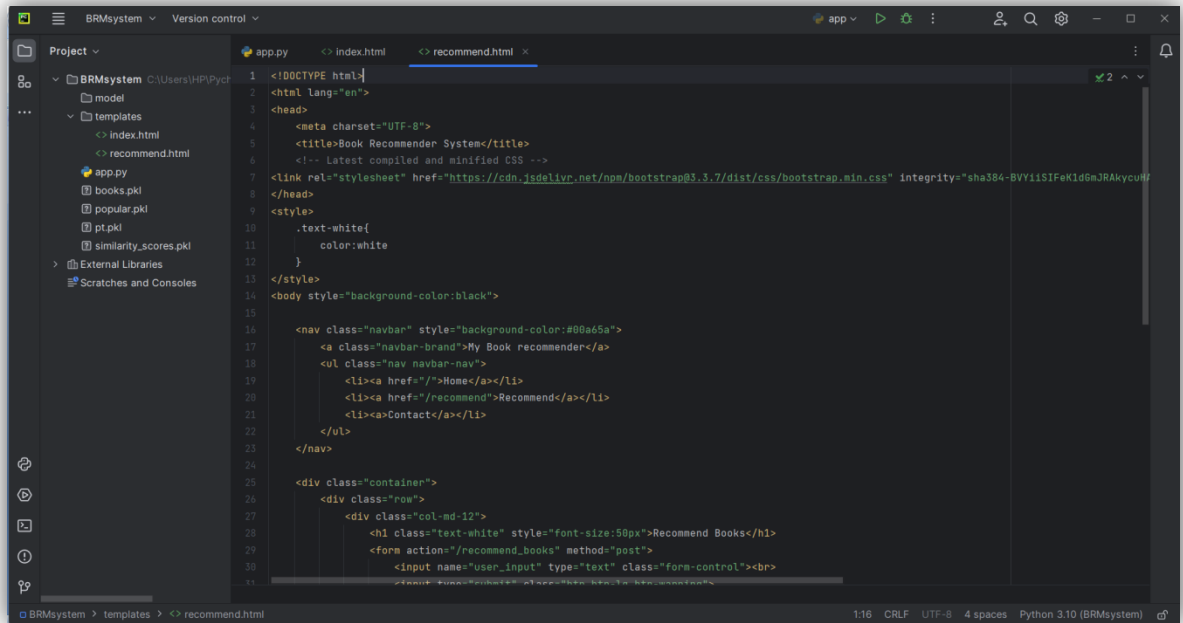
Animal Farm

Submit

1984
George Orwell

Angus, Thongs and Full-Frontal Snogging: Confessions of Georgia Nicolson
Louise Rennison

Midnight
Dean R. Koontz

Second Nature
Alice Hoffman

40

```python
import pickle
import numpy as np

popular_df = pickle.load(open('popular.pkl','rb'))
pt = pickle.load(open('pt.pkl','rb'))
books = pickle.load(open('books.pkl','rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl','rb'))

app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html',
                           book_name=_list(popular_df['Book-Title'].values),
                           author=list(popular_df['Book-Author'].values),
                           image=list(popular_df['Image-URL-M'].values),
                           votes=list(popular_df['num_ratings'].values),
                           rating=list(popular_df['Avg-Rating'].values)
                           )

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

@app.route('/recommend_books',methods=['post'])
def recommend():
    user_input = request.form.get('user_input')
    index = np.where(pt.index == user_input)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x: x[1], reverse=True)[1:5]
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuH
</head>
<style>
    .text-white{
        color:white
    }
</style>
<body style="background-color:black">

    <nav class="navbar" style="background-color:#00a65a">
        <a class="navbar-brand">My Book recommender</a>
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/recommend">Recommend</a></li>
            <li><a>Contact</a></li>
        </ul>
    </nav>

    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="text-white" style="font-size:50px">Top 50 Books</h1>
            </div>
```

D) SOURCE CODE

D.1 BOOK RECOMMENDER SYSTEM.IPYNB

```python
import numpy as np
import pandas as pd
books = pd.read_csv('books.csv')
users = pd.read_csv('users.csv')
ratings = pd.read_csv('ratings.csv')
books['Image-URL-M'][1]
users.head()
users.head()
ratings.head()
print(books.shape)
print(ratings.shape)
print(users.shape)
books.isnull().sum()
users.isnull().sum()
ratings.isnull().sum()
books.duplicated().sum()
ratings.duplicated().sum()
users.duplicated().sum()
```

Popularity Based Recommender System

```python
ratings_with_name = ratings.merge(books,on='ISBN')
num_rating_df = ratings_with_name.groupby('Book-Title').count()['Book-Rating'].reset_index()
num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace=True)
num_rating_df
avg_rating_df = ratings_with_name.groupby('Book-Title').mean()['Book-Rating'].reset_index()
avg_rating_df.rename(columns={'Book-Rating':'avg_rating'},inplace=True)
avg_rating_df
popular_df = num_rating_df.merge(avg_rating_df,on='Book-Title')
popular_df
popular_df =
popular_df[popular_df['num_ratings']>=250].sort_values('avg_rating',ascending=False).head(50)
popular_df = popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num_ratings','avg_rating']]
popular_df['Image-URL-M'][0]
## Collaborative Filtering Based Recommender System
x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] > 200
padhe_likhe_users = x[x].index
filtered_rating = ratings_with_name[ratings_with_name['User-ID'].isin(padhe_likhe_users)]
y = filtered_rating.groupby('Book-Title').count()['Book-Rating']>=50
famous_books = y[y].index
final_ratings = filtered_rating[filtered_rating['Book-Title'].isin(famous_books)]
pt = final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rating')
pt.fillna(0,inplace=True)
pt
from sklearn.metrics.pairwise import cosine_similarity
similarity_scores = cosine_similarity(pt)
similarity_scores.shape
def recommend(book_name):
    # index fetch
```

```python
    index = np.where(pt.index==book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda
x:x[1],reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-
M'].values))

        data.append(item)

    return data
recommend('1984')
import pickle
pickle.dump(popular_df,open('popular.pkl','wb'))
books.drop_duplicates('Book-Title')
pickle.dump(pt,open('pt.pkl','wb'))
pickle.dump(books,open('books.pkl','wb'))
pickle.dump(similarity_scores,open('similarity_scores.pkl','wb'))


D.2 WEBAPP.PY
from flask import Flask,render_template,request
import pickle
import numpy as np


popular_df = pickle.load(open('popular.pkl','rb'))
pt = pickle.load(open('pt.pkl','rb'))
books = pickle.load(open('books.pkl','rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl','rb'))
```

```python
app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html',
                           book_name = list(popular_df['Book-Title'].values),
                           author=list(popular_df['Book-Author'].values),
                           image=list(popular_df['Image-URL-M'].values),
                           votes=list(popular_df['num_ratings'].values),
                           rating=list(popular_df['Avg-Rating'].values)
                           )


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


@app.route('/recommend_books',methods=['post'])
def recommend():
    user_input = request.form.get('user_input')
    index = np.where(pt.index == user_input)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x:
x[1], reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-
M'].values))

        data.append(item)
```

```
        print(data)

        return render_template('recommend.html',data=data)



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

D.3 INDEX.HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEj
h4u" crossorigin="anonymous">
</head>
<style>
    .text-white{
        color:white
    }
</style>
<body style="background-color:black">

    <nav class="navbar" style="background-color:#00a65a">
        <a class="navbar-brand">My Book recommender</a>
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/recommend">Recommend</a></li>
```

```html
            <li><a>Contact</a></li>
        </ul>
    </nav>


    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="text-white" style="font-size:50px">Top 50 Books</h1>
            </div>


            {% for i in range(book_name|length) %}
                <div class="col-md-3" style="margin-top:50px">
                    <div class="card">
                        <div class="card-body">
                            <img class="card-img-top" src="{{ image[i] }}">
                            <p class="text-white">{{ book_name[i] }}</p>
                            <h4 class="text-white">{{ author[i] }}</h4>
                            <h4 class="text-white">Votes - {{ votes[i] }}</h4>
                            <h4 class="text-white">Rating - {{ rating[i] }}</h4>
                        </div>
                    </div>
                </div>
            {% endfor %}



        </div>
    </div>

</body>
</html>


D.4 RECOMMEND.HTML
<!DOCTYPE html>
<html lang="en">
```

```html
<head>
   <meta charset="UTF-8">
   <title>Book Recommender System</title>
   <!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEj
h4u" crossorigin="anonymous">
</head>
<style>
   .text-white{
      color:white
   }
</style>
<body style="background-color:black">

   <nav class="navbar" style="background-color:#00a65a">
      <a class="navbar-brand">My Book recommender</a>
      <ul class="nav navbar-nav">
         <li><a href="/">Home</a></li>
         <li><a href="/recommend">Recommend</a></li>
         <li><a>Contact</a></li>
      </ul>
   </nav>

   <div class="container">
      <div class="row">
         <div class="col-md-12">
            <h1 class="text-white" style="font-size:50px">Recommend Books</h1>
            <form action="/recommend_books" method="post">
               <input name="user_input" type="text" class="form-control"><br>
               <input type="submit" class="btn btn-lg btn-warning">
            </form>
```

```
        </div>

        {% if data %}

        {% for i in data %}
          <div class="col-md-3" style="margin-top:50px">
            <div class="card">
              <div class="card-body">
                <img class="card-img-top" src="{{i[2]}}">
                <p class="text-white">{{i[0]}}</p>
                <h4 class="text-white">{{i[1]}}</h4>
              </div>
            </div>
          </div>
        {% endfor %}

        {% endif %}




      </div>
    </div>

</body>
</html>
```