BOOKS RECOMMENDATION SYSTEM

A project report submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

BY

AKANKSHA SINGH

(205122008)

DEPARTMENT OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPALLI – 620015

JUNE 2024

BONAFIDE CERTIFICATE

This is to certify that the project "**Books Recommendation System**" is a project work successfully done by **Akanksha Singh (205122008)** in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications** from National Institute of Technology, Tiruchirappalli, during the academic year 2023-2024 (4th semester – CA749 Mini Project Work).

Dr. Vinay Raj	**Dr. Michael Arock**
(Project Guide)	(Head of the Department)

Project viva-voce held on………..

Internal Examiner	**External Examiner**

ABSTRACT

The Book Recommendation System is a cutting-edge application aimed at enhancing the reading experience by providing personalized book recommendations based on two distinct algorithms: Popularity-Based and Collaborative Filtering. The system harnesses the power of cosine similarity and nearest neighbour algorithms to offer users tailored suggestions.

The Popularity-Based algorithm recommends books solely based on their overall popularity among users. It serves as a baseline for comparison against the Collaborative Filtering algorithm.

In contrast, the Collaborative Filtering Algorithm utilizes cosine similarity to analyses user preferences and recommend books similar to those enjoyed by users with similar tastes. By leveraging nearest neighbour algorithms, the system identifies users with comparable preferences, enhancing the accuracy and relevance of recommendations.

ACKNOWLEDGEMENTS

We express our deep sense of gratitude to **Dr. G. Aghila**, Director National Institute of Technology, Tiruchirappalli for giving us an opportunity to work on this project.

I wish to extend my sincere gratitude to **Dr. Michael Arock**, Professor and Head, Department of Computer Applications, National Institute of Technology, Tiruchirappalli.

I sincerely thank my guide, **Dr. Vinay Raj**, Assistant Professor, Department of Computer Applications, National Institute of Technology, Tiruchirappalli, for supporting and encouraging me to carry out this project work and duly evaluatingmy progress.

I also thank the review panel for helping me identifying the shortages of the project and helping me to overcome those shortages by guiding me.

Finally, I would like to express my regards for all the faculty members of the Department of Computer Applications and others who have helped me develop this project directly or indirectly.

AKANKSHA SINGH

(205122008)

Table of Contents

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

A recommendation engine is a class of machine learning which offers relevant suggestions to the customer. Before the recommendation system, the major tendency to buy was to take a suggestion from friends. But Now Google knows what news you will read, YouTube knows what type of videos you will watch based on your search history, watch history, or purchase history.

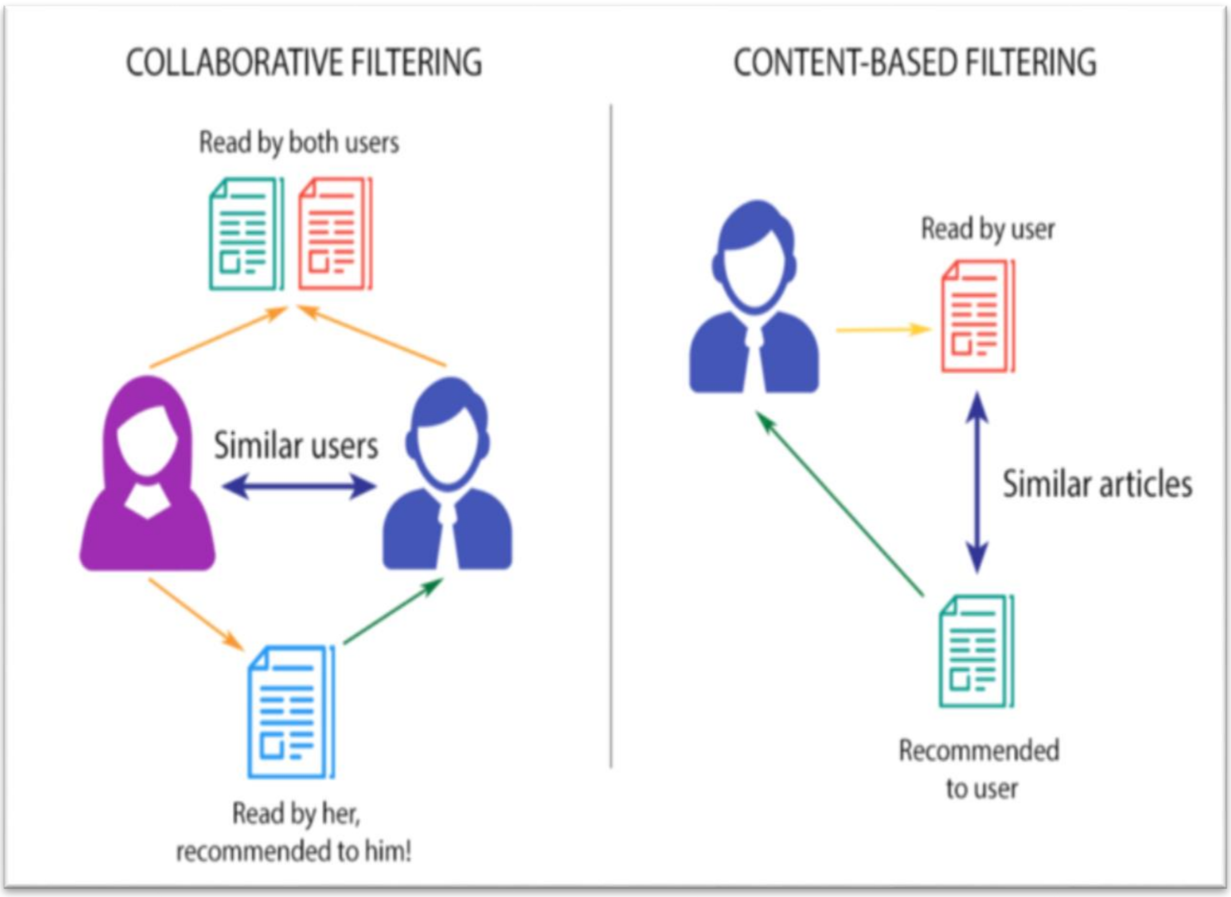


Figure 1.1

1) Content-Based Filtering

The algorithm recommends a product that is similar to those which used as watched. In simple words, In this algorithm, we try to find finding item look alike. For example, a person likes to watch Sachin Tendulkar shots, so he may like watching Ricky Ponting shots too because the two videos have similar tags and similar categories.

Only it looks similar between the content and does not focus more on the person who is watching this. Only it recommends the product which has the highest score based on past preferences.

2) Collaborative-based Filtering

Collaborative based filtering recommender systems are based on past interactions of users and target items. In simple words here, we try to search for the look-alike customers and offer products based on what his or her lookalike has chosen. Let us understand with an example. X and Y are two similar users and X user has watched A, B, and C movie. And Y user has watched B, C, and D movie then we will recommend A movie to Y user and D movie to X user.

YouTube has shifted its recommendation system from content-based to Collaborative based filtering technique. If you have experienced sometimes there are also videos which not at all related to your history but then also it recommends it because the other person similar you has watched it.

CHAPTER 2
Problem Statement and Motivation

Both the online entertainment and e-commerce companies are trying to retain their customers by taking their access to the website to more personalized manner. So, provide additional recommendations based on users past activity. Our project would be one of such system that recommends additional books that belongs to similar genre, author or publisher. Such systems result in increase in rate of purchase, these may also include unplanned purchases driven by surprise factor from the recommendations made.

Collaborative Filtering Systems: It is a common approach that provide recommendations of items based on patterns of sales of a product of other brands along which the one the user has searched for. This technique does not rely much on the information about items or users but make recommendations based on the user ratings. The proposed system collects the ratings from the user to predict the interest and analyses the item to find the features using collaborative filtering method. A correlation function is used to calculate the similarity between the movies in the dataset. Once the user selects a book, the books that have a cosine similarity closer to the selected book are recommended. In this process, we don't need to have the knowledge about the item specifications. The approaches taken are Item-Item based filtering, User-User based filtering and Alternating Least Squares Algorithm.

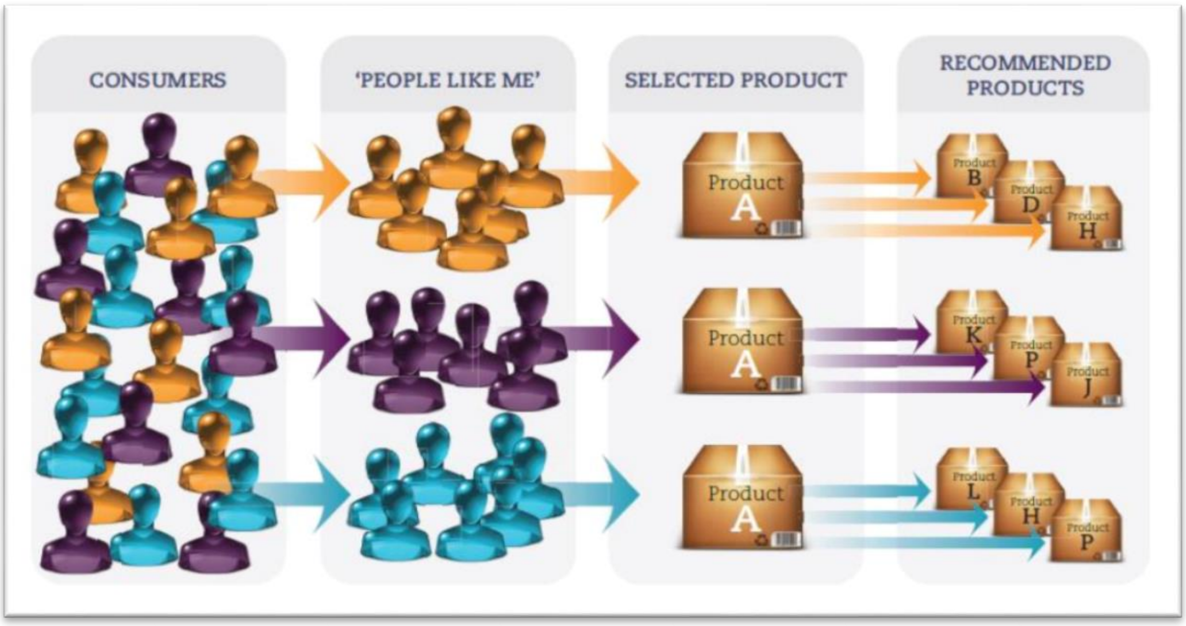


Figure 2.1

The Item-Item based filtering identifies the similarity between the items and uses this information to recommend books to the users based the previous ratings which the user has provided. The User-User based filtering identifies the similarity between the users and uses this information to recommend books to the users.

CHAPTER 3
Platform (Hardware & Software)

Hardware Requirements

- Processor: Intel Core i5 or equivalent AMD processor
- Memory: 8GB RAM or more
- Storage: 128GB SSD or more
- Networking: Ethernet or Wi-Fi connections

Software Requirements:

- Operating System: Windows 10, macOS, or Linux
- Programming Languages: Python
- Data Processing and Analysis Tools:
  - Pandas: A powerful data manipulation and analysis library in Python, essential for preprocessing and cleaning data.
  - NumPy: Fundamental package for scientific computing in Python, used for numerical operations and array manipulation.
- Other Machine Learning Libraries**:**
  - Matplotlib
  - Seaborn
  - Scikit-learn: A versatile library in Python for machine learning tasks such as classification, regression, and clustering.

CHAPTER 4

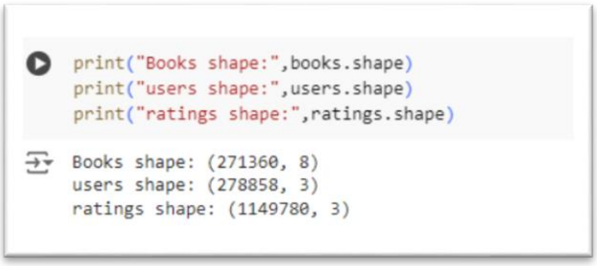
METHODOLOGIES

The methodology for building a book recommendation system involves several key steps, encompassing data collection, preprocessing, algorithm implementation, evaluation, and deployment. Below is a detailed outline of the methodology.

Data Collection:

- Gather a dataset of book titles, authors, genres, and user ratings. Utilize public datasets such as Goodreads, Book Crossing, or Amazon Reviews dataset.
- Dataset: https://www.kaggle.com/datasets/rxsraghavagrawal/book-recommender-system

Data set consist of three files.

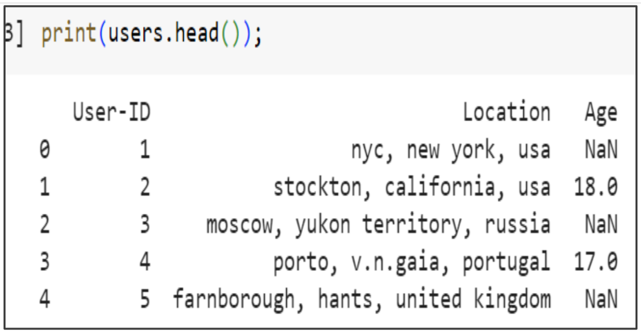


```
print("Books shape:",books.shape)
print("users shape:",users.shape)
print("ratings shape:",ratings.shape)
```

```
Books shape: (271360, 8)
users shape: (278858, 3)
ratings shape: (1149780, 3)
```

Figure 4.1

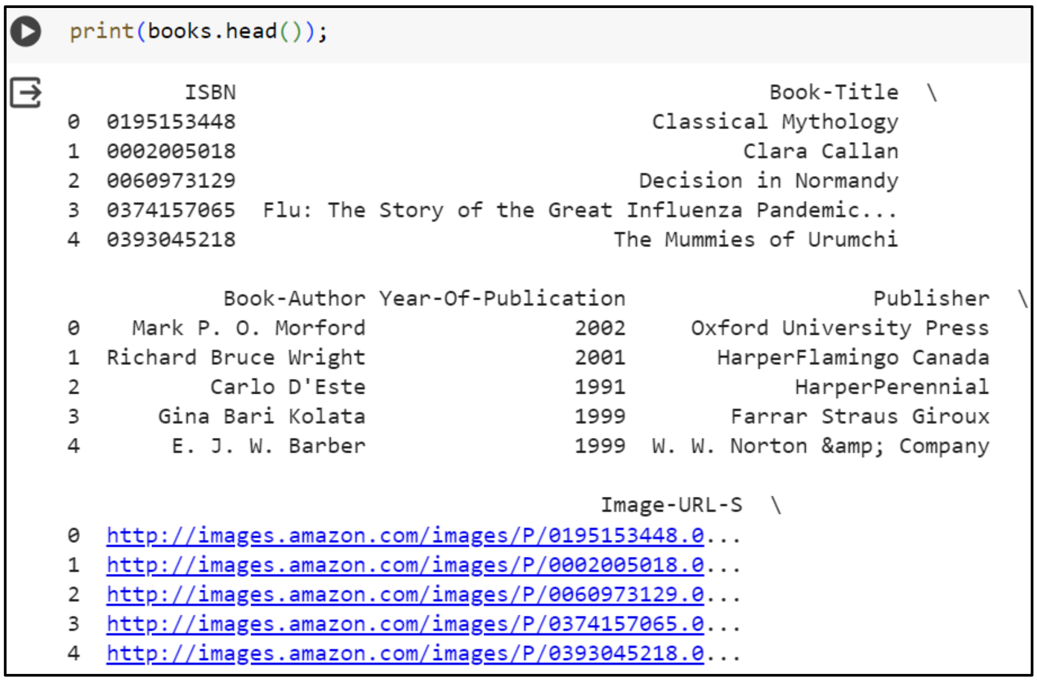
- Users.csv:
  Contains the users. User IDs (`User-ID`) and Demographic data (`Location`, `Age`)



```
3] print(users.head());

   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
```

Figure 4.2

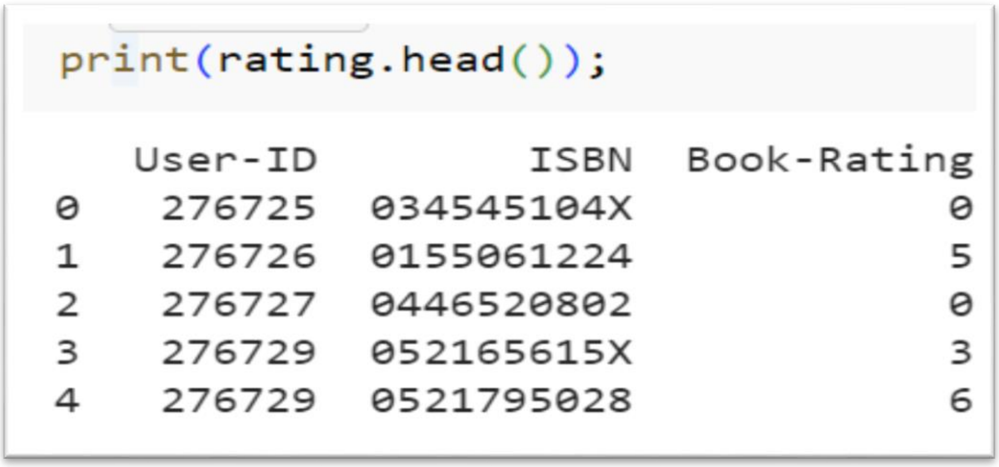
- Books.csv:
  Contains Books data (`Book-Title`, `Book-Author`, `Year-Of-Publication`, `Publisher`)



```
print(books.head());

         ISBN                                         Book-Title  \
0  0195153448                                Classical Mythology
1  0002005018                                       Clara Callan
2  0060973129                               Decision in Normandy
3  0374157065  Flu: The Story of the Great Influenza Pandemic...
4  0393045218                             The Mummies of Urumchi

            Book-Author Year-Of-Publication                    Publisher  \
0    Mark P. O. Morford                2002      Oxford University Press
1  Richard Bruce Wright                2001        HarperFlamingo Canada
2          Carlo D'Este                1991              HarperPerennial
3      Gina Bari Kolata                1999         Farrar Straus Giroux
4       E. J. W. Barber                1999  W. W. Norton &amp; Company

                                         Image-URL-S  \
0  http://images.amazon.com/images/P/0195153448.0...
1  http://images.amazon.com/images/P/0002005018.0...
2  http://images.amazon.com/images/P/0060973129.0...
3  http://images.amazon.com/images/P/0374157065.0...
4  http://images.amazon.com/images/P/0393045218.0...
```

Figure 4.3

- Ratings.csv:
  Contains the book rating information. Ratings (`Book-Rating`) are either explicit expressed on a scale from 1-10 (higher values denoting higher appreciation.



```
print(rating.head());

   User-ID        ISBN  Book-Rating
0   276725  034545104X            0
1   276726  0155061224            5
2   276727  0446520802            0
3   276729  052165615X            3
4   276729  0521795028            6
```

Figure 4.4

Data Analysis and Preprocessing:

- Clean and preprocess the dataset by removing duplicates, handling missing values, and ensuring data consistency.
- Normalize user ratings to a common scale to mitigate biases and discrepancies.
- Conduct exploratory data analysis (EDA) to gain insights into the dataset's characteristics, distribution of user ratings, and book genres.

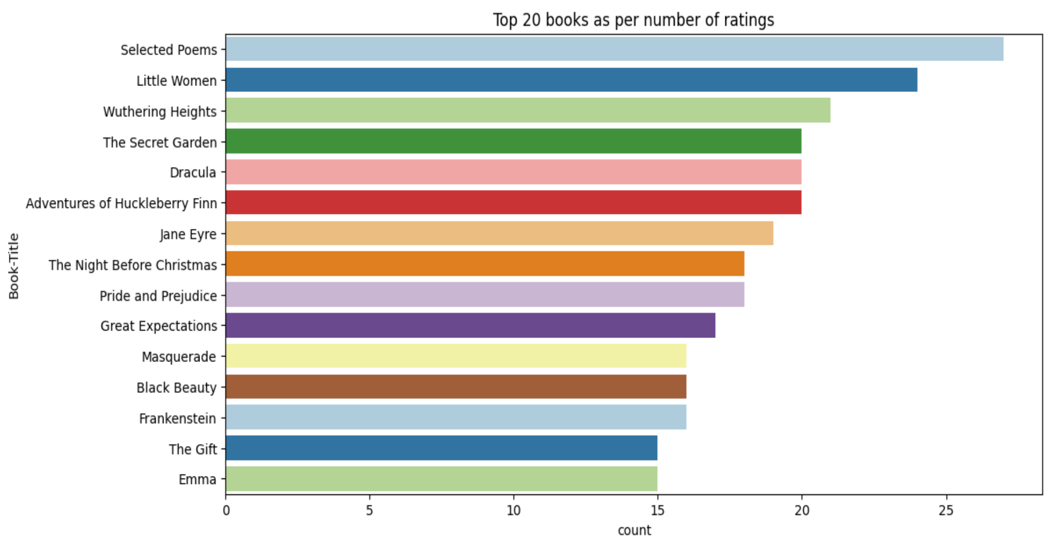


Figure 4.5

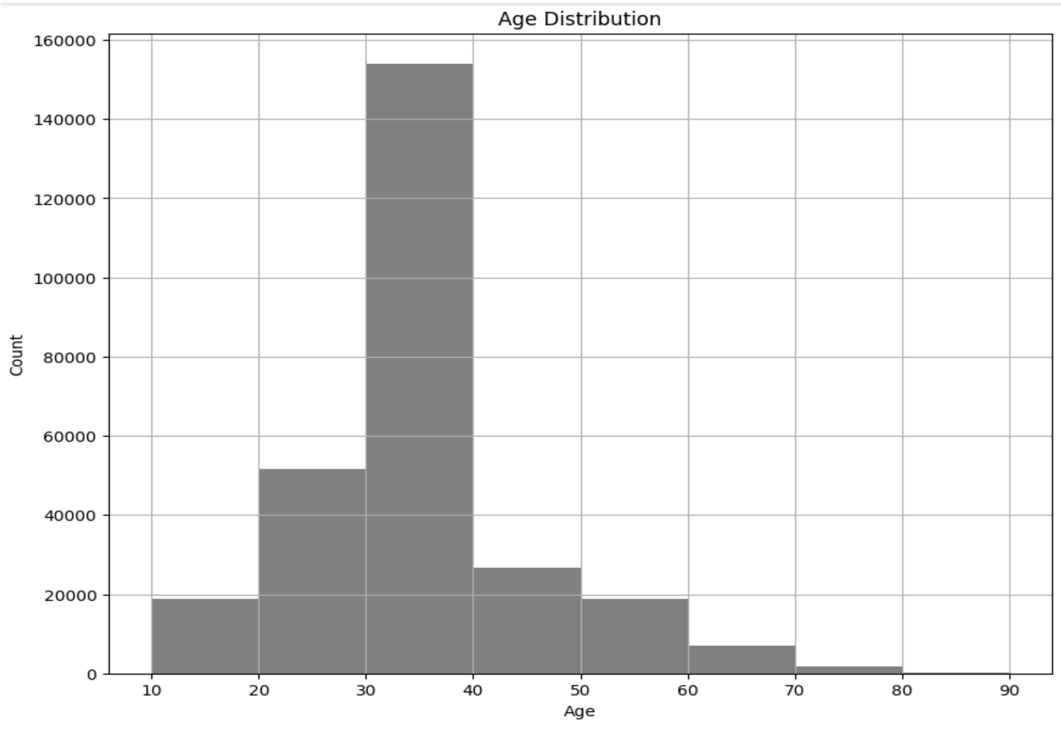


Figure 4.6

```
<Axes: xlabel='Book-Rating', ylabel='count'>
```

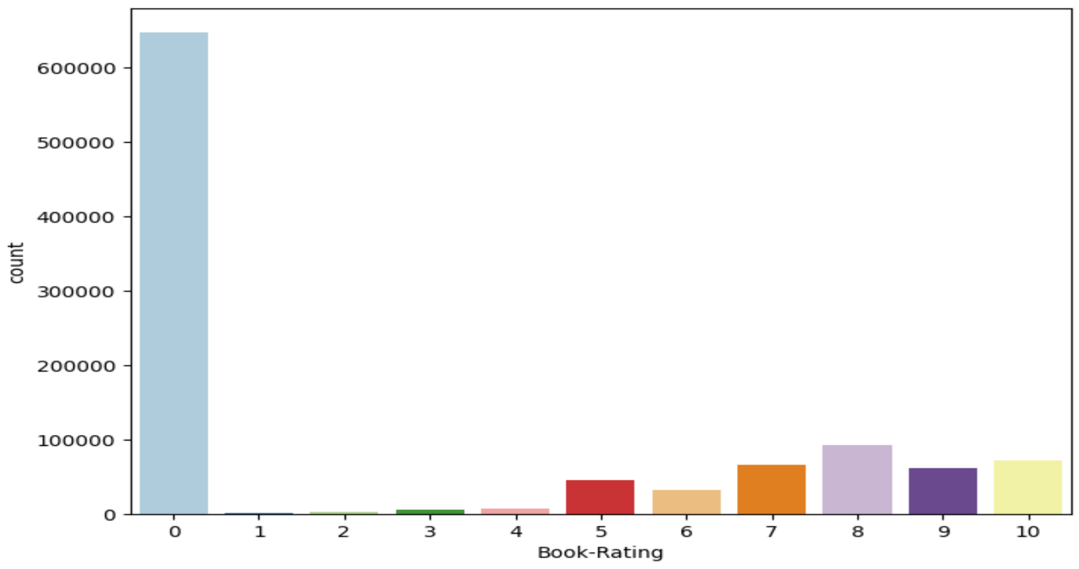


Figure 4.7

Algorithm Selection:

- Choose appropriate recommendation algorithms based on the nature of the dataset and the desired recommendation approach, such as Popularity-Based, Collaborative Filtering, Content-Based, or Hybrid methods.

Model Applied:

- **Cosine similarity**:
  Cosine similarity measures the similarity between two vectors of an inner product space.

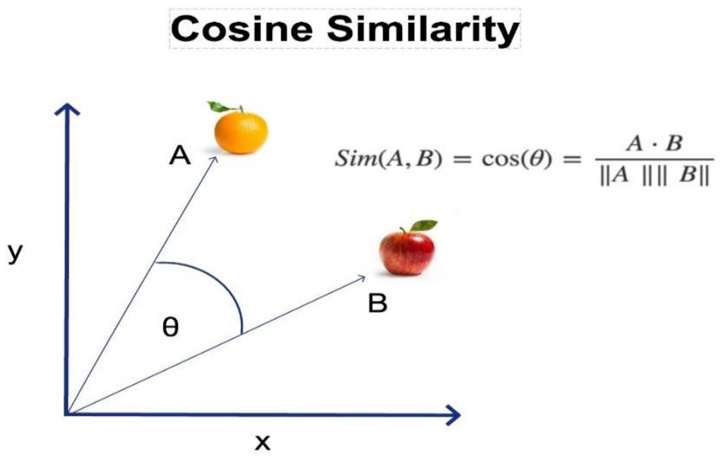
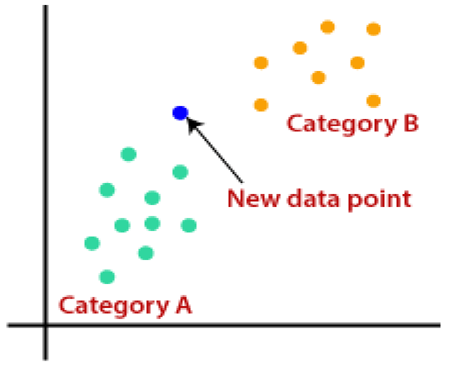
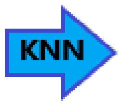


Figure 4.8

- **K-Nearest Neighbours (KNN):**
  Implement the K-Nearest Neighbours (KNN) algorithm to compute similarities between users based on their preferences and recommend items favoured by similar users.





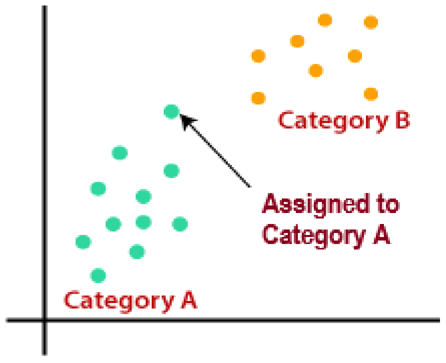


Figure 4.9 Figure 4.10

We do not want to find a similarity between users or books. we want to do that If there is user A who has read and liked x and y books, and user B has also liked this two books and now user A has read and liked some z book which is not read by B so we have to recommend z book to user B. This is what collaborative filtering is.

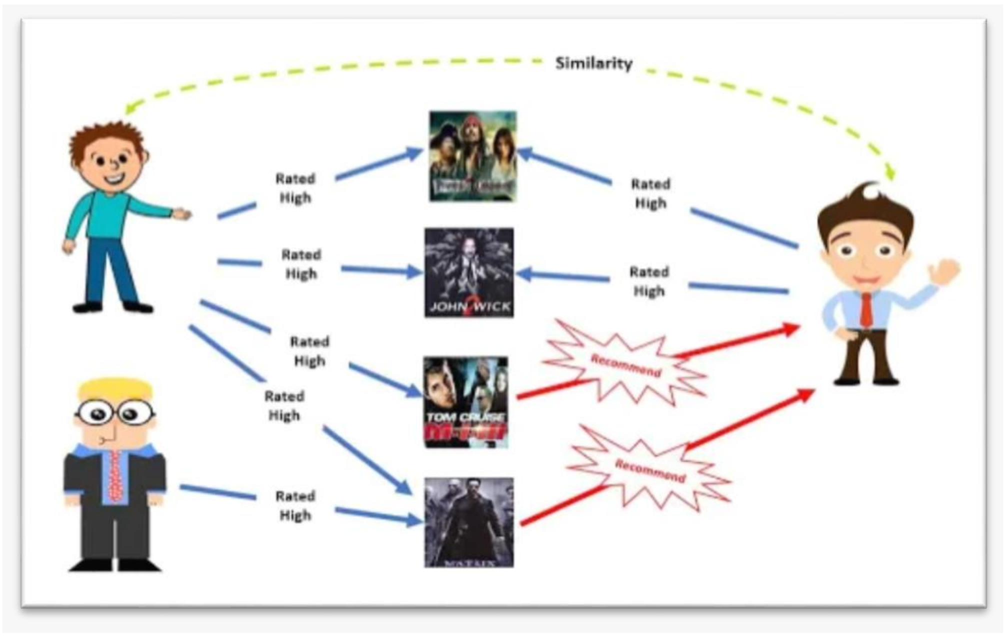
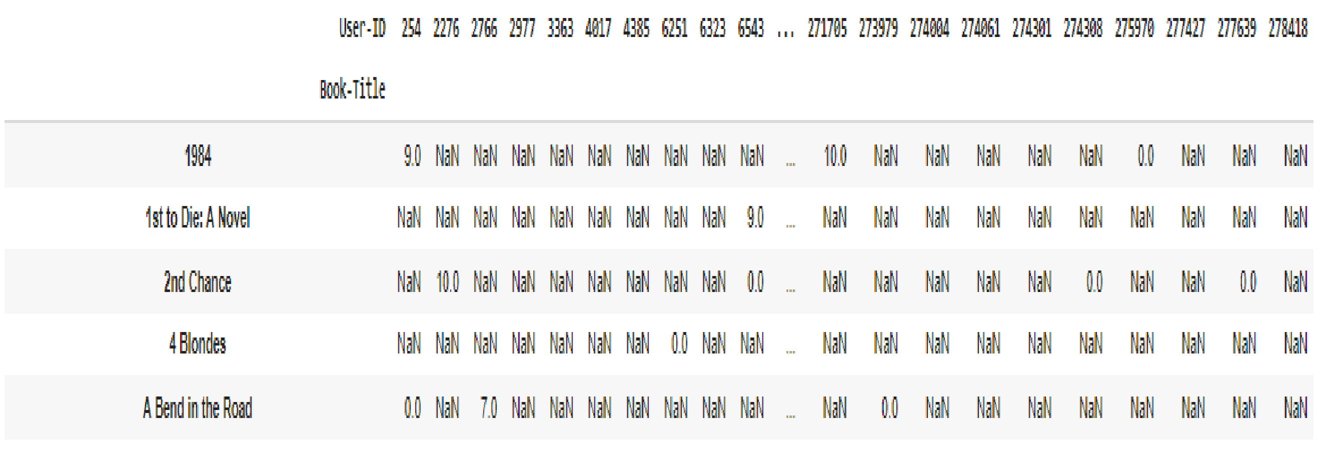


Figure 4.11

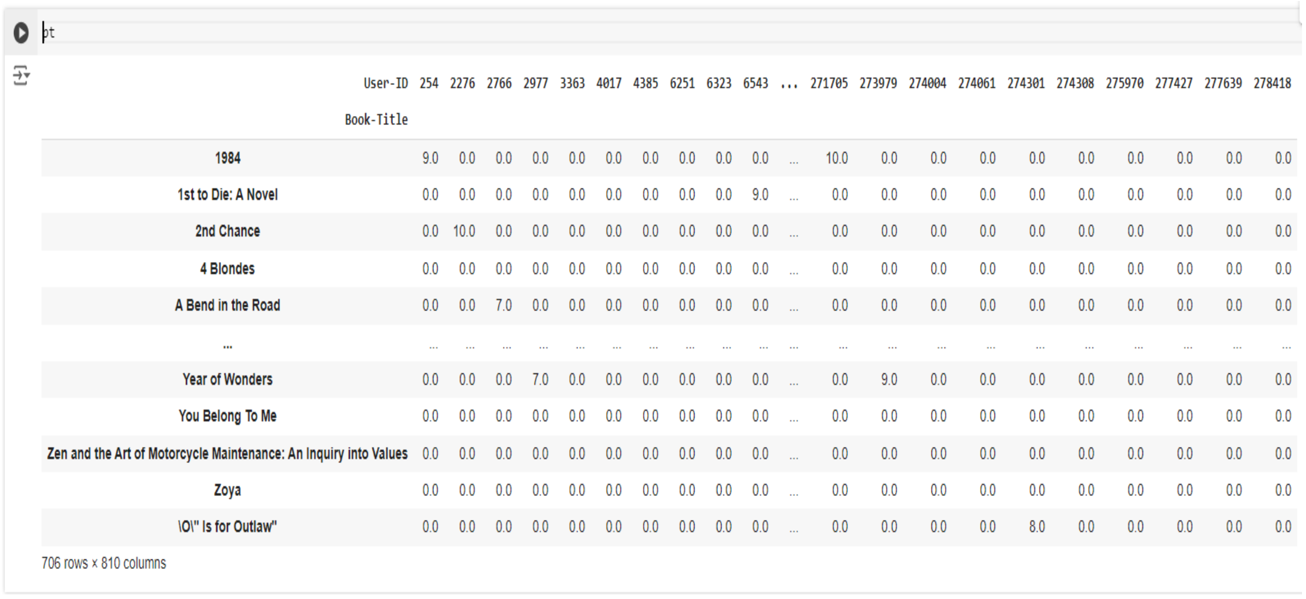
So this is achieved using Matrix Factorization, we will create one matrix where columns will be users and indexes will be books and value will be rating. Like we have to create a Pivot table.

Pivot Table of Matrix Factorization



User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	278418
Book-Title																					
1984	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	10.0	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
1st to Die: A Novel	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2nd Chance	NaN	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	0.0	NaN
4 Blondes	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
A Bend in the Road	0.0	NaN	7.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 4.12



```
pt
```

User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	278418
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	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	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	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	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	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	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	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	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	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	0.0

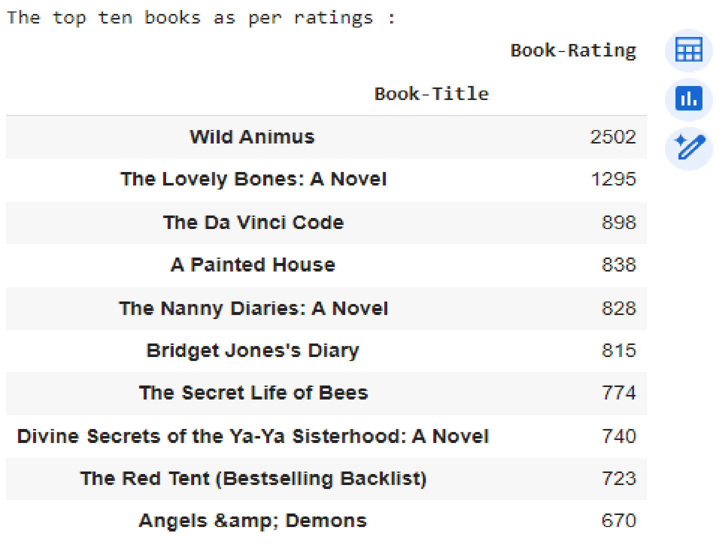
706 rows × 810 columns

Figure 4.13

CHAPTER 5

EXPERIMENTAL RESULTS

❖ Result of Popularity Based System



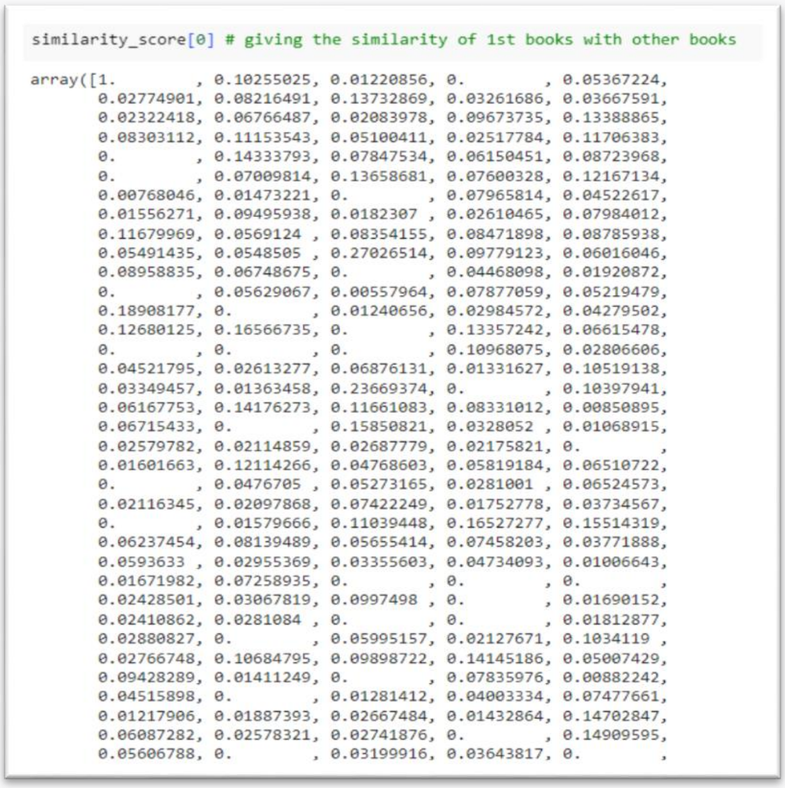
```
The top ten books as per ratings :
```

Book-Title	Book-Rating
Wild Animus	2502
The Lovely Bones: A Novel	1295
The Da Vinci Code	898
A Painted House	838
The Nanny Diaries: A Novel	828
Bridget Jones's Diary	815
The Secret Life of Bees	774
Divine Secrets of the Ya-Ya Sisterhood: A Novel	740
The Red Tent (Bestselling Backlist)	723
Angels &amp; Demons	670

Figure 5.1

❖ Result of Collaborative Based System:

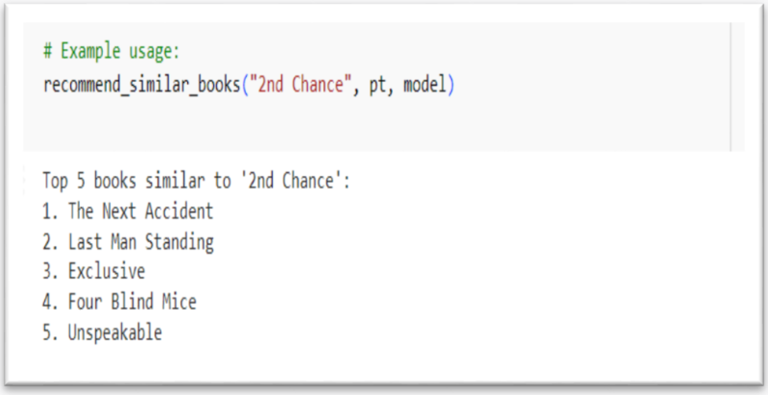
Let's make a prediction and see whether it is suggesting books or not. we will find the nearest neighbors to the input book id and after that, we will print the top 5 books which are closer to those books.



```
similarity_score[0] # giving the similarity of 1st books with other books
```

```
array([1.        , 0.10255025, 0.01220856, 0.        , 0.05367224,
       0.02774901, 0.08216491, 0.13732869, 0.03261686, 0.03667591,
       0.02322418, 0.06766487, 0.02083978, 0.09673735, 0.13388865,
       0.08303112, 0.11153543, 0.05100411, 0.02517784, 0.11706383,
       0.        , 0.14333793, 0.07847534, 0.06150451, 0.08723968,
       0.        , 0.07009814, 0.13658681, 0.07600328, 0.12167134,
       0.00768046, 0.01473221, 0.        , 0.07965814, 0.04522617,
       0.01556271, 0.09495938, 0.0182307 , 0.02610465, 0.07984012,
       0.11679969, 0.0569124 , 0.08354155, 0.08471898, 0.08785938,
       0.05491435, 0.0548505 , 0.27026514, 0.09779123, 0.06016046,
       0.08958835, 0.06748675, 0.        , 0.04468098, 0.01920872,
       0.        , 0.05629067, 0.00557964, 0.07877059, 0.05219479,
       0.18908177, 0.        , 0.01240656, 0.02984572, 0.04279502,
       0.12680125, 0.16566735, 0.        , 0.13357242, 0.06615478,
       0.        , 0.        , 0.        , 0.10968075, 0.02806606,
       0.04521795, 0.02613277, 0.06876131, 0.01331627, 0.10519138,
       0.03349457, 0.01363458, 0.23669374, 0.        , 0.10397941,
       0.06167753, 0.14176273, 0.11661083, 0.08331012, 0.00850895,
       0.06715433, 0.        , 0.15850821, 0.0328052 , 0.01068915,
       0.02579782, 0.02114859, 0.02687779, 0.02175821, 0.        ,
       0.01601663, 0.12114266, 0.04768603, 0.05819184, 0.06510722,
       0.        , 0.0476705 , 0.05273165, 0.0281001 , 0.06524573,
       0.02116345, 0.02097868, 0.07422249, 0.01752778, 0.03734567,
       0.        , 0.01579666, 0.11039448, 0.16527277, 0.15514319,
       0.06237454, 0.08139489, 0.05655414, 0.07458203, 0.03771888,
       0.0593633 , 0.02955369, 0.03355603, 0.04734093, 0.01006643,
       0.01671982, 0.07258935, 0.        , 0.        , 0.        ,
       0.02428501, 0.03067819, 0.0997498 , 0.        , 0.01690152,
       0.02410862, 0.0281084 , 0.        , 0.        , 0.01812877,
       0.02880827, 0.        , 0.05995157, 0.02127671, 0.1034119 ,
       0.02766748, 0.10684795, 0.09898722, 0.14145186, 0.05007429,
       0.09428289, 0.01411249, 0.        , 0.07835976, 0.00882242,
       0.04515898, 0.        , 0.01281412, 0.04003334, 0.07477661,
       0.01217906, 0.01887393, 0.02667484, 0.01432864, 0.14702847,
       0.06087282, 0.02578321, 0.02741876, 0.        , 0.14909595,
       0.05606788, 0.        , 0.03199916, 0.03643817, 0.        ,
```

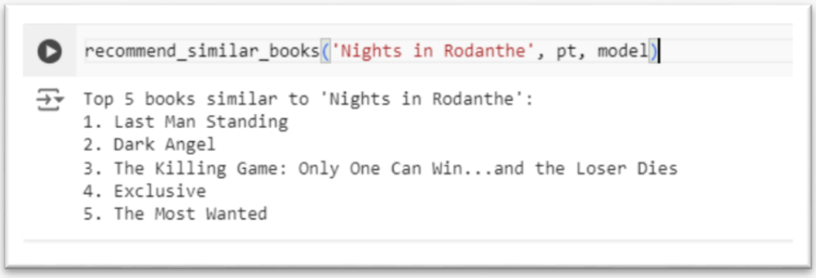
Figure 5.2



```
# Example usage:
recommend_similar_books("2nd Chance", pt, model)
```

```
Top 5 books similar to '2nd Chance':
1. The Next Accident
2. Last Man Standing
3. Exclusive
4. Four Blind Mice
5. Unspeakable
```

Figure 5.3



```
recommend_similar_books('Nights in Rodanthe', pt, model)
```

```
Top 5 books similar to 'Nights in Rodanthe':
1. Last Man Standing
2. Dark Angel
3. The Killing Game: Only One Can Win...and the Loser Dies
4. Exclusive
5. The Most Wanted
```

Figure 5.4

CHAPTER 6
DISCUSSIONS

Pros & Cons of Popularity based Approach:

Pros: There is no need for the user's historical data.

Cons: This would recommend the same books which are scary based on popularity to every other.

Pros & Cons of Collaborative Approach:

Pros: Collaborative Approach does not require much user profile like content-based recommendation approach.

Cons: This system suffers cold-start problem where the user/community is very new to the system/environment when there are no enough information and rating for the books to process and recommend item.

Challenges:

Book recommendation datasets often suffer from data sparsity, where users may have rated only a small fraction of available books, making it challenging to generate accurate recommendations for all users. If we take all the books and all the users for modelling, Don't you think will it create a problem? So what we have to do is we have to decrease the number of users and books because we cannot consider a user who has only registered on the website or has only read one or two books. On such a user, we cannot rely to recommend books to others because we have to extract knowledge from data.

Recommending books to new users or books with limited ratings poses a cold start problem, where traditional recommendation algorithms struggle to provide relevant suggestions due to lack of user data.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Conclusion:

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales.

Future Work:

Explore advanced recommendation algorithms such as deep learning models or reinforcement learning to improve recommendation accuracy and relevance. Integrate social features that allow users to connect with friends, share book recommendations, and see what books others in their network are reading. Enable users to create and curate personalized reading lists based on their interests, allowing for long-term planning and organization of reading materials.

We also can deploye our project in real time environment, we can make it as a website where user can search books based on there interest.

REFERENCES

[1]"Retracted: Research on Book Recommendation Algorithm Based on Collaborative Filtering and Interest Degree", *Wireless Communications and Mobile Computing*, vol. 2023, Article ID 9813784, 1 pages, 2023. https://doi.org/10.1155/2023/9813784

[2]https://www.analyticsvidhya.com/blog/2021/06/build-book-recommendation-system-unsupervised-learning-project/

[3] https://www.geeksforgeeks.org/collaborative-filtering-ml/

APPENDIX

```
#importing library
import numpy as np
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')

#importing all dataset
books = pd.read_csv('/content/drive/MyDrive/Book Recommendation
System/Books.csv')
users = pd.read_csv('/content/drive/MyDrive/Book Recommendation
System/Users.csv')
ratings = pd.read_csv('/content/drive/MyDrive/Book Recommendation
System/Ratings.csv')

books.head()
users.head()
ratings.head()

"""## Data Analysis"""
print("Books shape:",books.shape)
print("users shape:",users.shape)
print("ratings shape:",ratings.shape)
books.isnull().sum()
users.isnull().sum()
ratings.isnull().sum()
books.duplicated().sum()
users.duplicated().sum()
ratings.duplicated().sum()
books.info()
users.info()
ratings.info()

"""Analysing the book's data"""
#nan values in a particular column
books.loc[(books['Book-Author'].isnull()),:]
#nan value in a particular column
books.loc[(books['Publisher'].isnull()),:]
#getting unique value from 'Year of publication' feature
books['Year-Of-Publication'].unique()
books.loc[2216]
#replacing 'nan' value with 'no mention'
books.loc[(books['ISBN']=='193169656X'),'Publisher']='No Mention'
```

```
books
books.info()
books['Publisher'].isnull().sum()

"""lets see the users dataset"""
#unique value in age
users['Age'].unique()
#there is nan value we will replace it with mean of 'age'
users['Age'].fillna((users['Age'].mean()),inplace=True)
#unique value in age
users['Age'].unique()
#retrieving age data between 5 to 90
users.loc[(users['Age']>90) | (users['Age']<5)] = np.nan
#unique value in age
users['Age'].unique()
#there is nan value we will replace it with mean of 'age'
users['Age'].fillna((users['Age'].mean()),inplace=True)
#unique value in age
users['Age'].unique()
#unique ratings from 'ratings' dataset
ratings['Book-Rating'].unique()
# finding unique ISBNs from rating and book dataset
unique_ratings = ratings[ratings.ISBN.isin(books.ISBN)]
unique_ratings

"""## Data Visualition Analysis no. 1- Number of Books published in yearly."""
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import warnings; warnings.simplefilter('ignore')

publications = {}
for year in books['Year-Of-Publication']:
    if str(year) not in publications:
        publications[str(year)] = 0
    publications[str(year)] +=1
publications = {k:v for k, v in sorted(publications.items())}
fig = plt.figure(figsize =(12, 6))
plt.bar(list(publications.keys()),list(publications.values()), color = 'red')
plt.ylabel("Number of books published")
plt.xlabel("Year of Publication")
plt.title("Number of books published yearly")
plt.show()

"""Analysis no. 2-Which are the top Author with number of books ?"""
plt.figure(figsize=(12,6))
sns.countplot(y="Book-Author",palette = 'Paired',
data=books,order=books['Book-Author'].value_counts().index[0:20])
```

```
plt.title("Top 20 author with number of books")

"""Analysis no.3 -Which are top publishers with published books ?"""
plt.figure(figsize=(12,6))
sns.countplot(y="Publisher",palette = 'Paired',
data=books,order=books['Publisher'].value_counts().index[0:20])
plt.title("Top 20 Publishers with number of books published")

"""Analysis No. 4 What are top 20 books as per number of ratings ?"""
plt.figure(figsize=(12,6))
sns.countplot(y="Book-Title",palette = 'Paired',data= books,
order=books['Book-Title'].value_counts().index[0:15])
plt.title("Top 20 books as per number of ratings")

"""Analysis no.4 -Age distributions of users_data"""
plt.figure(figsize=(10,8))
users.Age.hist(bins=[10*i for i in range(1, 10)], color = 'grey')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()

"""it is obvious that most of the user books are from Age 30 to 40. Analysis
No. 5 What are top 20 books as per number of ratings ?"""
plt.figure(figsize=(12,6))
sns.countplot(y="Book-Title",palette = 'Paired',data= books,
order=books['Book-Title'].value_counts().index[0:15])
plt.title("Top 20 books as per number of ratings")
plt.figure(figsize=(8,6))
sns.countplot(x="Book-Rating",palette = 'Paired',data= unique_ratings)
"""This countplot shows users have rated 0 the most, which means they haven't
rated books at all.

## Popularity Based Recommender System"""
# we will display the books with highest average rating
ratings
books
# we will merge both books and ratings dataframe on the basis of ISBN
ratings.merge(books,on ='ISBN').shape
ratings_with_name =ratings.merge(books,on ='ISBN')
ratings_with_name.head()
ratings_with_name.groupby('Book-Title').count()
num_rating_df =ratings_with_name.groupby('Book-Title').count()['Book-
Rating'].reset_index()
num_rating_df
num_rating_df.rename(columns = {'Book-Rating':'num_of_rating'},inplace= True)
num_rating_df
```

```
ratings_with_name['Book-Rating'] = ratings_with_name['Book-
Rating'].astype(float)
ratings_with_name['Book-Rating'].dtype
avg_rating_df = ratings_with_name.groupby('Book-Title')['Book-
Rating'].agg(lambda x: x.astype(float).mean()).reset_index()
avg_rating_df.rename(columns = {'Book-Rating' : 'avg_rating'}, inplace = True)
avg_rating_df
# top ten books as per book ratings and recommendation
top_ten_books= pd.DataFrame(ratings_with_name.groupby('Book-Title')['Book-
Rating'].count().sort_values(ascending=False).head(10))
print('The top ten books as per ratings : ')
top_ten_books

"""#Memory-Based Collabrative Filtering"""
x=ratings_with_name.groupby('User-ID').count()['Book-Rating'] >200
#boolean indexing
x[x]
#fetching users by index
trustable_users = x[x].index
trustable_users
filtered_rating =ratings_with_name[ratings_with_name['User-
ID'].isin(trustable_users)]
filtered_rating
#now filtering this data based on book which has at least 50 rates by the
users
y = filtered_rating.groupby('Book-Title').count()['Book-Rating'] >=50
famous_books = y[y].index # which has more than fifty ratings
famous_books
#only those books are here which has more than 50 rating and those users who
have vote more than 200 times
final_rating = filtered_rating[filtered_rating['Book-
Title'].isin(famous_books)]
final_rating
#checking duplicates
final_rating.drop_duplicates()
#creating pivot table
pt =final_rating.pivot_table(index = 'Book-Title',columns = 'User-ID',values
='Book-Rating')
pt
pt.fillna(0,inplace=True)
pt
#our each book is represented as a vector so we have to find out similarity
between each vector
from sklearn.metrics.pairwise import cosine_similarity
similarity_score =cosine_similarity(pt)
similarity_score[0] # giving the similarity of 1st books with other books
#geting the books after sorting in reverse order
```

```
sorted(list(enumerate(similarity_score[0])),key = lambda x:x[1],reverse =
True)[1:6]
#getting the index of book by the name of book
np.where(pt.index=='Year of Wonders')[0][0]

#creating the function which will give suggegtions
def recommend(book_name):
    if book_name in pt.index:
        index = np.where(pt.index == book_name)[0][0]
        similar_books_list = sorted( list(enumerate(similarity_score[index])),
key=lambda x: x[1], reverse=True)[1:7]

        print(f'Recommendations for the book {book_name}:')
        print('-'*5)
        for book in similar_books_list:
            print(pt.index[book[0]])
        print('\n')
    else:
        print('Book Not Found')
        print('\n')
recommend('Harry Potter and the Chamber of Secrets (Book 2)')
recommend('Message in a Bottle')
recommend('A Walk to Remember')
recommend("Nights in Rodanthe")

""" **Implementing new model**"""
#convert the pivot table to the sparse matrix
from scipy.sparse import csr_matrix
book_sparse = csr_matrix(pt)
#training the model
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(algorithm='brute')
model.fit(book_sparse)
#let us pass harry potter which is at 237 indexes.
distances, suggestions = model.kneighbors(pt.iloc[237, :].values.reshape(1, -
1))
# to print the title of book
for i in range(len(suggestions)):
  print(pt.index[suggestions[i]])
def recommend_similar_books(book_name, pt, model):
    # Find the index of the book
    book_index = pt.index.get_loc(book_name)
    # Get distances and suggestions from the model
    distances, suggestions = model.kneighbors(pt.iloc[book_index,
:].values.reshape(1, -1), n_neighbors=6)
    # Print the recommended books
    print("Top 5 books similar to '{}':".format(book_name))
    for i in range(1, 6):  # Exclude the first suggestion (the book itself)
```

```
        similar_book_index = suggestions[0][i]
        similar_book_name = pt.index[similar_book_index]
        print("{}. {}".format(i, similar_book_name))
recommend_similar_books('A Walk to Remember', pt, model)
recommend_similar_books('Nights in Rodanthe', pt, model)

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
recommend('Year of Wonders')
```