# Book Recommendation System

A

Project Report

Submitted for the partial fulfillment

of B. Tech Degree

in

COMPUTER SCIENCE & ENGINEERING

By

**Samarth Gautam (200520100050)**

**Shivam Pandey (200520100061)**

**Rohit Tiwari (2100520109002)**

*Under the Supervision of*

**Dr. Natthan Singh**

**Mr. Deepanshu Singh**


Department of Computer Science and Engineering

**Institute of Engineering & Technology**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh**

June, 2024

## Contents

# Declaration

We hereby declare that this submission is entirely original work of us, with the exception of any passages where acknowledgment is explicitly stated in the text. It also contains no material that has been previously published or written by someone else or that has, with a substantial error, been accepted for the award of any degree or diploma from a university or other higher education institution. We haven't turned in the project to fulfill any other degree requirements at any other institution.

Submitted by:- Date:05/06/2024

(1) Name: Samarth Gautam
Roll No.: 200520100050
Branch: Computer Science & Engineering
Signature:

(2) Name: Shivam Pandey
Roll No.: 200520100061
Branch: Computer Science & Engineering
Signature:

(3) Name: Rohit Tiwari
Roll No.: 2100520109002
Branch: Computer Science & Engineering
Signature:

## Certificate

This certifies that the work completed by Samarth Gautam, Shivam Pandey, and Rohit Tiwari at the Department of Computer Science and Engineering at the Institute of Engineering and Technology, Lucknow, under my supervision and guidance, is documented in the project report titled "Book Recommendation System," which they presented as a partial fulfillment for the award of a Bachelor of Technology in Computer Science and Engineering.

Additionally, it certifies that, to the best of my knowledge, this project has not been submitted to any other institute for the award of any other degrees.

(Dr. Natthan Singh)
Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow

(Mr. Deepanshu Yadav)
Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow

## Acknowledgement

We would like to express our gratitude to Dr. Girish Chandra, Head, CSE Department, IET Lucknow, and our humble Supervisors Dr. Natthan Singh and Ms. Deepanshu Yadav for their continued support and suggestions so far in this project. They have been very constructive, supportive, kind and made it easier to write up this thesis work. Their continuous support gave us the motivation to work on this write-up and helped me to gain knowledge in various fields. I would also like to thank them for the suggested changes. They allowed me to locate areas of improvement that helped me to write this Interim write-up. We would like to express gratitude to all our classmates and teammates who motivated us and helped us in time of need to complete this project. We shall be failing in duty if we do not acknowledge with gratitude thanks to the authors of the references and other literature referred to in this thesis work.

**Samarth Gautam (2000520100050)**

**Sign**

**Shivam Pandey (2000520100061)**

**Sign**

**Rohit Tiwari (2100520109002)**

**Sign**

## Abstract

The abundance of books available in the digital age makes it difficult for readers to find recommendations that suit their interests. This thesis compares and contrasts the advantages and disadvantages of several book recommendation system approaches, as well as their efficacy. Our goal is to create a sophisticated recommendation system that will improve customers' reading experiences by offering them accurate, tailored book recommendations. The project makes use of a multi-page Flask web application that incorporates a number of collaborative filtering approaches and machine learning algorithms.

Our system's main features include a showcase of various recommendation techniques, trending book suggestions, and personalized recommendations. We evaluate classical collaborative filtering techniques including Non-Negative Matrix Factorization (NMF) and Singular Value Decomposition (SVD). Based on user satisfaction, scalability, and accuracy, each approach is assessed, and the top-performing algorithms are incorporated into the final recommendation system. In addition, our system has a trending books area that shows you the most popular books according to recent user interactions.

Both frontend and backend integration are covered in the part on web application development, with an emphasis on using HTML/CSS for a visually appealing and responsive user interface and Flask for server-side operations. The homepage, an introduction of the methodology, the results display, demo sites for trending and customized recommendations, and a about page with information about the project's history and goals are important pathways. We perform thorough testing and user feedback sessions to confirm the efficacy of our recommendation system, and we find that it significantly outperforms current systems in terms of recommendation accuracy and user happiness.

This thesis concludes with the presentation of a strong book recommendation system that makes use of collaborative filtering techniques and sophisticated machine learning algorithms. This project represents a significant contribution to the field of recommendation systems because it includes a user-friendly web application together with a thorough study and implementation of numerous strategies.

# Chapter 1
# Introduction

The abundance of books available online in the current digital era offers readers both opportunities and challenges. It might be difficult to locate the ideal book to fit a person's tastes when there are millions of options available. Decision fatigue is a common result of this much availability, making it difficult for readers to choose books that genuinely suit their interests and likes. As a result, there is a great need for efficient recommendation systems that can help readers find books they will probably like.

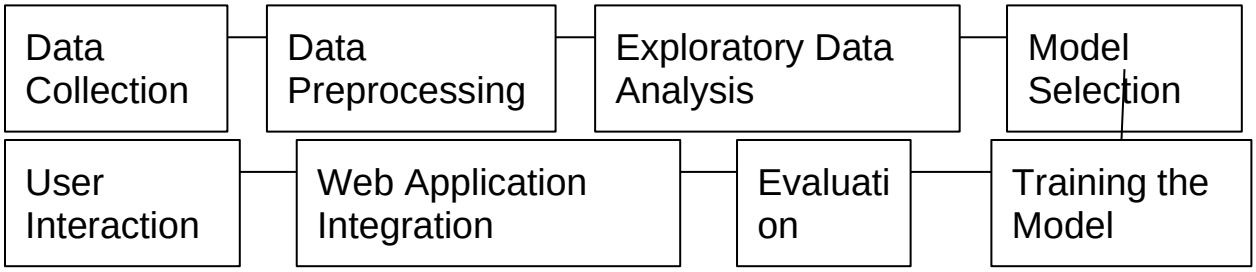
Recommendation systems, which offer tailored suggestions based on users' prior actions and interests, have become an essential component of many online platforms. When it comes to books, these algorithms are able to forecast and suggest books that a user would find interesting by examining their reading preferences, ratings, and other pertinent information. The goal of this project is to investigate various approaches to creating a strong book recommendation system, analyze their effectiveness, and pinpoint their advantages and disadvantages.

This project's main goal is to compare and apply many methodologies, such as matrix factorization models and collaborative filtering, in order to assess the effectiveness of different recommendation systems. By doing this, we hope to find out which techniques yield the most precise and tailored book recommendations. This thorough analysis will assist in comprehending the trade-offs related to each strategy and determining which is best for various situations.

This project's execution includes preparing data, choosing suitable algorithms, and assessing each method's effectiveness. The final recommendation system offers easy-to-use, rapid access to book recommendations through its intuitive design. With this project, we hope to advance the field of recommendation systems by providing a thorough study of current approaches and suggesting a solution that improves book discoverability and user experience.

The following sections of this report will delve into the detailed literature review, methodology, implementation, and evaluation of the proposed book recommendation system. The results obtained from the system will be discussed, followed by a conclusion that highlights the achievements of the project and suggests potential directions for future work.

## Workflow of the Book Recommendation System




## Structure of the report

This chapter gives an overview of the topic to be followed on the report.

- **Literature Review -** All of the important books, studies, and investigations related to this project are included in this chapter. It attempts to give a concise overview of the technical features of the project and the findings of the study that was done. This field is the subject of comparison, contrast, and evaluation.

- **Development -** This chapter will provide comprehensive information on the creation of the artifact, outline the development cycle, and provide appropriate rationale for the tools and methods used. This chapter's primary goal is to collect requirements, then design and create the system in accordance with those needs.

- **Testing and system Evaluation -** This chapter's goal is to test the developed system and make sure that all prerequisites are satisfied. The primary goal of the testing phase is to look for flaws and errors in the system and make the required adjustments.

- **Conclusion -** Summarizing the main conclusions and results of the Book Recommendation System project is the aim of the Conclusion chapter. It seeks to illustrate the efficacy of the applied recommendation algorithms, consider the project's overall impact, and offer suggestions for future improvements and possible uses. The goal of this chapter is to highlight the contributions made by this research and to synthesize the knowledge that has been acquired.

# Chapter 2

# Literature Review

This chapter provides an overview of the field's extensive historical past, which is the focus of this research. It seeks to give a concise overview of the technical components of the project and the findings of the carried out research. In this field, comparison, contrast, and evaluation are carried out.

## 2.1 Introduction :

The practice of assessing or filtering objects based on user opinions is known as collaborative filtering. Even if the phrase "collaborative filtering" (CF) has only been in use for a little more than ten years, the idea behind it has its roots in the custom of exchanging viewpoints with others. For generations, people have formed their own opinions based on discussions about the books they've read, the restaurants they've tried, the music they've listened to, and the movies they've seen.

For instance, John might choose to read a new book if a lot of his coworkers are raving about it. On the other hand, he might not read the book if a large number of people thought it was a disaster. John gradually comes to know whose advice he can trust; for example, he might frequently concur with Sam's suggestions, detest Arya's, and typically see Sansa's advice as trustworthy. John makes use of this data to assess an item's quality.

This process has been broadened by the internet, which now enables us to take into account the perspectives of thousands of individuals as opposed to just a select few. Real-time processing of these thoughts by computers allows us to not only learn what the general public thinks of a product, but also to provide customized suggestions based on the preferences of specific consumers.

**2.2 Core Concepts**:

All collaborative filtering techniques forecast or suggest new products that a specific user might appreciate based on their past rating activity. Assuming commonalities between people or items based on their past behavior or ratings is the primary premise behind collaborative filtering. This technique is used with book recommendation models, which take into account user behavior in the past, books that the user has already read, and other users' opinions on particular books.

Anybody who rates the system is referred to as a "user". This covers both the people who provide the data (ratings) that powers the recommendation engine and the people who utilize the system to obtain recommendations.

A collaborative filtering system generates one or more item predictions or recommendations for a specific user. Anything that can be rated, including music, movies, literature, art, DVDs, and travel locations, can be included in this category. A collaborative filtering system can have many kinds of ratings.

- Scalar Ratings: Scalar ratings can include numeric ratings, such as a scale from 1 to 10, or ordinal ratings, such as "agree," "strongly agree," "neutral," "disagree," and "strongly disagree."
- Binary Ratings: Binary ratings offer simple choices, such as "good/bad" or "agree/disagree."
- Unary Ratings: Unary ratings indicate that a user has observed, bought, or positively evaluated an item. If a rating is absent, it suggests that the system has no information about the user's interaction with the item or that the user may have acquired the product elsewhere.

Either explicit methods, implicit methods, or both can be used to gather ratings. When a user is asked to rate an item explicitly, they are asked to provide their opinion. Implicit ratings are those that are derived from a user's behavior. For instance, a user who visits the product page may be interested in it, but a user who ends up purchasing it may be even more interested.

**2.3 Theoretical Framework:**

Over the past ten years, collaborative filtering algorithms have progressed from simple research tools that record user preferences to intricate algorithms that could meet the demands of large-scale commercial applications. This section will cover several of the most popular non-probabilistic collaborative filtering methods.

Non-probabilistic algorithms are distinct from probabilistic ones, which are generated from underlying probabilistic models and represent probability distributions for grading items or making suggestion lists. One of the most used collaborative filtering methods is nearest neighbor. Item-based nearest neighbor and user-based nearest neighbor are the two types of them.

The following figure illustrates the schematic diagram for the collaborative filtering method.

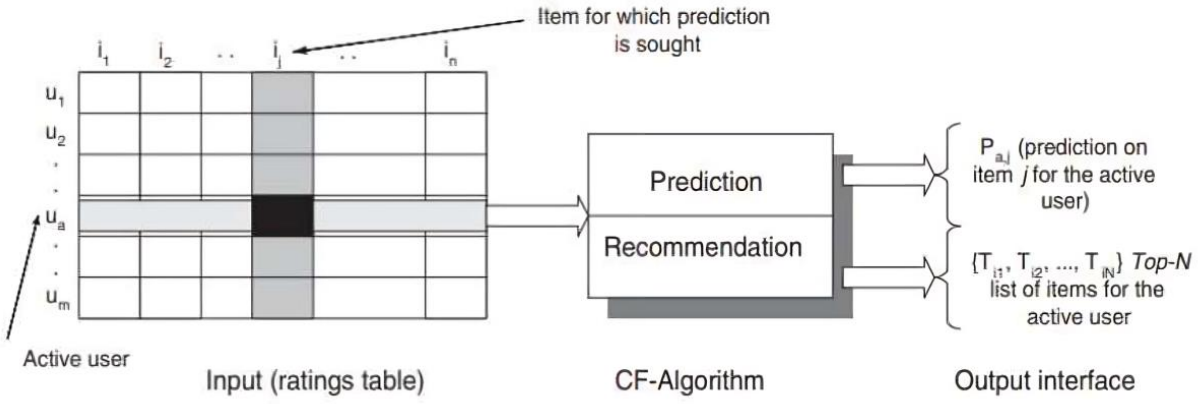



Fig 1(a) : Collaborative Based Filtering

**User-Based Collaborative Filtering**

Finding users who share a target user's preferences and utilizing their ratings to anticipate the target user's ratings for things is known as user-based collaborative filtering, or UBCF. The fundamental idea is that consumers are likely to continue agreeing if they have done so in the past.

Steps in User-Based Collaborative Filtering:

1. Similarity Calculation: Find users who rate in a similar way as the intended user. Measures like Pearson correlation or cosine similarity are frequently used to compute this similarity.
    - Cosine Similarity: This measures the cosine of the angle between two users' rating vectors.

$$\text{sim}(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

      where $I_{uv}$ is the set of items rated by both users u and v, $r_{ui}$ is the rating of user u for item i, and $r_{vi}$ is the rating of user v for item i.

    - Pearson Correlation: This measures the linear correlation between two users' rating vectors.

$$\text{sim}(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

      where $\bar{r}_u$ and $\bar{r}_v$ are the average ratings of users u and v respectively.

2. Neighborhood Selection: Select a set of users (neighbors) who have the highest similarity scores with the target user.
3. Prediction Computation: Predict the rating for an item that the target user has not rated yet, based on the ratings of the selected neighbors. A common approach is to use a weighted sum of the neighbors' ratings.
    - Weighted Sum: The predicted rating $r^{\wedge}_{ui}$ for user u on item i can be computed as:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u)} \text{sim}(u, v) \cdot (r_{vi} - \bar{r}_v)}{\sum_{v \in N(u)} |\text{sim}(u, v)|}$$

      where N(u) is the set of neighbors for user u.

Example:

Suppose we have the following ratings matrix:

| User | Item 1 | Item 2 | Item 3 | Item 4 |
|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 |
| Bob | 3 | 1 | 2 | 3 |
| Charlie | 4 | 3 | 4 | 3 |
| Dave | 3 | 3 | 1 | 5 |

Fig 1(b) : Rating in Matrix form

If we want to predict Alice's rating for Item 2, we would:

1. Calculate similarities between Alice and the other users (Bob, Charlie, Dave).
2. Select the top N similar users (neighbors).
3. Use their ratings for Item 2 to compute Alice's predicted rating using the weighted sum formula.

User-based collaborative filtering is a powerful technique, especially in environments where user preferences are stable and well-documented, making it a widely-used approach in recommendation systems.

**Item-Based Collaborative Filtering**

Instead of emphasizing users, item-based collaborative filtering (IBCF) concentrates on item similarity. The fundamental tenet is that users are more likely to like comparable things in the future if they have previously enjoyed them. This approach works especially well when user preferences don't change all that much over time.

## Steps in Item-Based Collaborative Filtering:

1. Similarity Calculation: Identify items that are similar to the item for which we want to predict a rating. Similarity between items can be calculated using measures such as cosine similarity or Pearson correlation.
    - **Cosine Similarity**: This measures the cosine of the angle between two items' rating vectors.

$$\text{sim}(i,j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

      where $U_{ij}$ is the set of users who have rated both items i and j, $r_{ui}$ is the rating of user u for item i, and $r_{uj}$ is the rating of user u for item j.

    - **Pearson Correlation**: This measures the linear correlation between two items' rating vectors.

$$\text{sim}(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

      where $\bar{r}_i$ and $\bar{r}_j$ are the average ratings of items i and j respectively.

    - **Neighborhood Selection**: Select a set of items (neighbors) that are most similar to the target item.
    - **Prediction Computation**: Predict the rating for the target item based on the ratings of the similar items. A common approach is to use a weighted sum of the

ratings of the similar items.

i. **Weighted Sum:** The predicted rating r^ui\hat{r}_{ui}r^ui for user uuu on item iii can be computed as:

$$\hat{r}_{ui} = \frac{\sum_{j \in N(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum_{j \in N(i)} |\text{sim}(i,j)|}$$

where N(i) is the set of neighbors for item i.

**Example:**

Suppose we have the following ratings matrix:

| User | Item 1 | Item 2 | Item 3 | Item 4 |
|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 |
| Bob | 3 | 1 | 2 | 3 |
| Charlie | 4 | 3 | 4 | 3 |
| Dave | 3 | 3 | 1 | 5 |

Fig 1(c) : Rating in Matrix form

To estimate Alice's score on Item 2, we would:

1. Calculate similarities between Item 2 and other items (Item 1, Item 3, Item 4).
2. Select the top N similar items (neighbors).
3. Utilizing the weighted sum formula, get Alice's anticipated rating for Item 2 based on the ratings of these comparable items.

Item-based collaborative filtering is a popular choice for recommendation systems in a variety of applications because it works especially well in situations where there are a lot of users and relatively consistent item properties.

**Singular Value Decomposition**

One potent method used in collaborative filtering for recommendation systems is Singular Value Decomposition (SVD). When dealing with sparse data—which is typical in recommendation systems where many users have only evaluated a small portion of the available items—SVD is especially useful.

**Steps in Singular Value Decomposition:**

1. **Matrix Factorization:**

   SVD decomposes the user-item rating matrix R into three matrices: U, Σ, and $V^T$.

   - R: The original user-item rating matrix.
   - U: A matrix where each row represents a user in the latent feature space.
   - Σ: A diagonal matrix containing singular values, which represent the strength of each latent feature.
   - $V^T$: A matrix where each column represents an item in the latent feature space.

   The decomposition is represented as:

$$R \approx U\Sigma V^T$$

2. **Dimensionality Reduction:**

   To reduce the complexity and remove noise, we can truncate Σ to keep only the top k singular values, resulting in reduced matrices $U_k$, $\Sigma_k$, and $V_k^T$.

3. **Prediction Computation:**

   The prediction rating can be computed by multiplying the truncated matrices.

$$\hat{R} = U_k \Sigma_k V_k^T$$

**Advantages of SVD:**

- **Latent Features**: SVD captures latent features that explain the relationships between users and items.
- **Noise Reduction**: By keeping only the top singular values, SVD reduces noise and focuses on the most significant patterns in the data.
- **Handling Sparsity**: SVD effectively handles the sparsity of the rating matrix, making it suitable for recommendation systems with large datasets.

SVD is widely used in recommendation systems for its ability to model complex user-item interactions and improve the accuracy of recommendations. It is particularly effective in capturing latent relationships that are not immediately apparent from the raw data.

**Non-Negative Matrix Factorization**

Another matrix factorization method used in collaborative filtering for recommendation systems is called Non-Negative Matrix Factorization (NMF). NMF forces the factors to be non-negative, in contrast to SVD, which permits both positive and negative values in its factorized matrices. Because of this, NMF is especially helpful in situations when non-negative values are included in the data by nature, like in user ratings.

**Steps in Non-Negative Matrix Factorization**

1. **Matrix Factorization:**

   NMF decomposes the user-item rating matrix R into two non-negative matrices W and H:

   $R \approx W H$

   - R: The original user-item rating matrix.
   - W: A matrix where each row represents a user in the latent feature space.
   - H: A matrix where each column represents an item in the latent feature space.

2. **Matrix Factorization:**

NMF aims to minimize the reconstruction error between the original matrix R and the product of the two factor matrices W and H. The objective function is often formulated as:

$$\min_{W,H} \|R - WH\|_F^2$$

3. **Optimization:**

   The optimization is typically performed using iterative algorithms such as multiplicative update rules, which ensure that W and H remain non-negative throughout the iterations.

**Advantages of NMF:**

- **Interpretability**: The non-negativity constraint makes the latent features more interpretable, as they can be seen as additive parts of the data.
- **Sparsity**: NMF can naturally handle sparse data, which is common in recommendation systems.
- **Non-Negativity**: Ensuring non-negativity aligns with the nature of user ratings, which are typically non-negative.

Because NMF can effectively retain the underlying structure of the data while guaranteeing that all components stay non-negative, it is especially helpful for recommendation systems. This is in line with the common use case, in which user ratings are whole numbers with no negative values.

NMF offers a computationally efficient and interpretable method of modeling the latent properties of users and things by factoring the rating matrix into non-negative matrices. Because of this, NMF is a useful technique for producing tailored recommendations in extensive recommendation systems.

**2.5 Major Findings and Citations:**

- Yiu-Kai Ng (2020) :CBRec: a book recommendation system for children using the matrix

factorisation and content-based filtering approaches.

- Pegah Sagheb Haghighi University of Louisville(2021) : Cosine-based explainable matrix factorization for collaborative filtering recommendation.

- Ashish Fatarphekar, Tejas Nashikkar, Vivek Patil, Gayatri Naik, “BOOK RECOMMENDATION SYSTEM BASED ON COMBINE FEATURES OF CONTENT BASED FILTERING, COLLABORATIVE FILTERING AND ASSOCIATION RULE MINING”, 2015, Advance Research in Social science and Humanities (ISSN: 2208-2387).

- Ms. Praveena Mathew, Ms. Bincy Kuriakose, Mr.Vinayak Hegde, “Book Recommendation System through Content Based and Collaborative Filtering Method”, 2016, Department of Computer Science Amrita Vishwa Vidyapeetham Mysore Campus Mysore, Karnataka, India.

# Chapter 3

# Methodology

This chapter will provide a thorough explanation of the development cycle's steps as well as the appropriate rationale for the tools and methods used.

This chapter's primary goal is to create the system in accordance with the requirements that have been gathered.

## 3.1 Development Tools and Frameworks to be used

The following tools and frameworks will be utilized for developing the system.

- **Python as the core Programming Language**

  Python is an object-oriented programming language that is easy to use and powerful, allowing developers to concentrate on the main features of their applications. because a large range of packages and libraries, like matplotlib, NumPy, and Pandas, are available. Python is widely used in the machine learning domain. It facilitates the learning and application of machine learning techniques by developers. It offers a structure that makes it possible to program clearly at both local and big scales. Its syntax is simple to learn and it's an open source programming language.

- **Flask as the Web Framework**

  Flask is a Python framework which is used to create web applications. It is developed by Armin Ronacher, the head of Pocco, an international community of Python fans. The Jinja2 template engine and Werkzeug WSGI toolkit serve as the foundation for Flask. Projects from Pocco are both.

  Flask was made with ease of use and extensibility in mind. Building a strong foundation for web applications of varying complexity is the goal behind Flask. You are then free to connect any extensions you believe you will require. You are also allowed to create your own modules. Flask works well for a wide range of projects. It works particularly well for prototyping. The Werkzeug WSGI toolkit and the Jinja2 template engine are the two external libraries that Flask requires.

- **Google Colab**

  Colab is a hosted Jupyter Notebook service that doesn't require setup and provides free access to computing resources, including GPUs and TPUs. Colab is especially useful in the domains of machine learning, data science, and education. Colab notebooks are stored on a Google Drive account and can be shared with other users just like any other Google Drive file. Although the notebooks have an autosave function as well, they cannot accommodate simultaneous editing, therefore cooperation must be serial as opposed to parallel.

## 3.2 Implementation

### 3.2.1 Data Sources and Preprocessing

- **Data Sources**

  We have used the Book-Crossing Dataset which is available on Kaggle.

  There are three files in the Book-Crossing dataset.

  - Users:

    Includes the users. Keep in mind that user IDs (User-ID) transfer to integers after being anonymised. If available, demographic information (age, location) is given. These fields have NULL-Values in them otherwise.

| | 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 2(a) : User Data Table

  - Books:

    Books can be recognized by their unique ISBN. The dataset has previously been cleared of invalid ISBNs. Additionally, certain content-based data (Book-Title,

Book-Author, Year-of-Publication, Publisher) is provided, which was acquired from Amazon Web Services. Keep in mind that just the first author is given when there are many authors. Additionally, three distinct variants (Image-URL-S, Image-URL-M, and Image-URL-L), or small, medium, and big, are provided for the URLs linking to the cover images. These web addresses lead to the Amazon website.

| | ISBN | Book-Title | Book-Author | Year-Of-Publication | Publisher | Image-URL-S |
|---|---|---|---|---|---|---|
| 0 | 0195153448 | Classical Mythology | Mark P. O. Morford | 2002 | Oxford University Press | http://images.amazon.com/images/P/0195153448.0... |
| 1 | 0002005018 | Clara Callan | Richard Bruce Wright | 2001 | HarperFlamingo Canada | http://images.amazon.com/images/P/0002005018.0... |
| 2 | 0060973129 | Decision in Normandy | Carlo D'Este | 1991 | HarperPerennial | http://images.amazon.com/images/P/0060973129.0... |
| 3 | 0374157065 | Flu: The Story of the Great Influenza Pandemic... | Gina Bari Kolata | 1999 | Farrar Straus Giroux | http://images.amazon.com/images/P/0374157065.0... |
| 4 | 0393045218 | The Mummies of Urumchi | E. J. W. Barber | 1999 | W. W. Norton & Company | http://images.amazon.com/images/P/0393045218.0... |

Fig 2(b) : Book Data Table .

- Ratings:

  includes the rating information for the book. Book ratings can be explicitly stated as a number between 1 and 10, with higher numbers indicating greater appreciation, or implicitly conveyed as 0.

| | 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 |

Fig 2(c) : User and their Rating.

- Dimension of Dataset:
  Book_df shape is (271360, 8)
  Ratings_df shape is (1149780, 3)
  Users_df shape is (278858, 3)

- **Data Cleaning and Exploration**

  One of the most important steps in machine learning (ML) is data cleaning, which includes finding and eliminating any duplicate, irrelevant, or missing data. Ensuring that the data is reliable, consistent, and error-free is the aim of data cleaning, since inconsistent or inaccurate data can have a detrimental effect on the performance of the machine learning model.

  **Users Table Preprocessing:**

```
missing_values(Users)
```

| | index | Missing Values | % of Total Values | Data_type |
|---|---|---|---|---|
| 0 | Age | 110762 | 39.72 | float64 |
| 1 | User-ID | 0 | 0.00 | int64 |
| 2 | Location | 0 | 0.00 | object |

Fig 2(d) : Finding Missing Values.

👉 **Age column has 39.72% missing values**

**Let’s check the outliers in the group**

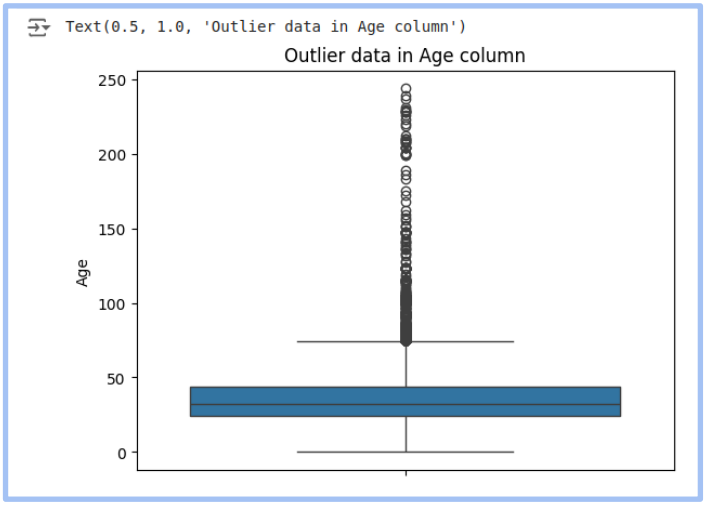
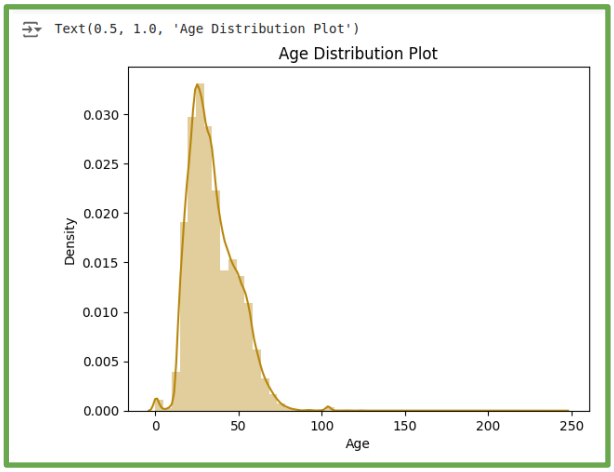



Fig 2(e) : Outlier Data

After replacing age values below 5 and above 100 with NaNs, we get a total of 112010 Null Values in the Age Column since these values do not make much sense for our book rating instance.

Age has positive Skewness (right tail) so we can use median to fill Nan values, but for this we don't like to fill Nan values just for one range of age. To handle this we'll use the country column to fill Nan.




**<u>Books Table Preprocessing</u>**

A few entries in the Year-Of-Publication field are not correct. Due to certain inconsistencies in the csv file, it appears that the publisher names "DK Publishing Inc." and "Gallimard" were put into the dataset wrongly as the Year-Of-Publication.

| | ISBN | Book-Title | Book-Author | Year-Of-Publication |
|---|---|---|---|---|
| **209538** | 078946697X | DK Readers: Creating the X-Men, How It All Beg... | 2000 | DK Publishing Inc |
| **221678** | 0789466953 | DK Readers: Creating the X-Men, How Comic Book... | 2000 | DK Publishing Inc |

Here bookAuthor is incorrectly loaded with bookTitle, hence we have to make required corrections.

After checking the unique values in Year-Of-Publication we found 0 value entries in some of the entries. The value 0 for Year-Of-Publication is invalid and as this dataset was published in 2004, We have assumed that the years after 2006 to be invalid and set invalid years as NaN.

Exploring the Publisher's column we found two missing values in the Publisher's Column, So we filled 'other' for NaN.

Same with exploring the Book-Author column, we found two missing values in the Book-Author Column, So we filled 'other' for NaN.

*Top 10 Authors which have written the most books.*

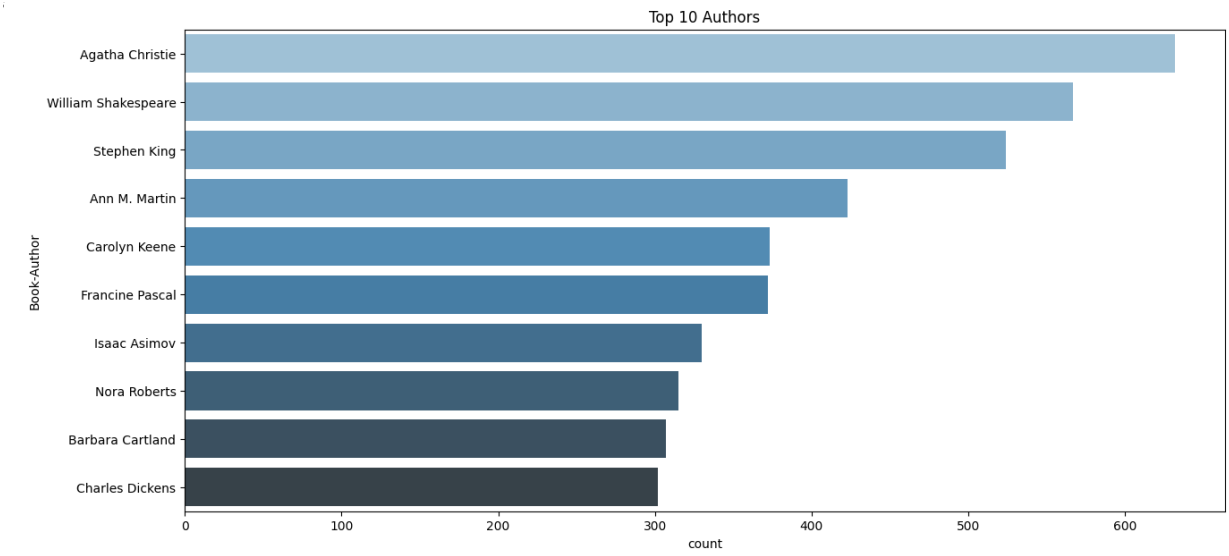



Fig 2(f) : Top 10 Authors

*Top 10 Publishers which have written the most books.*

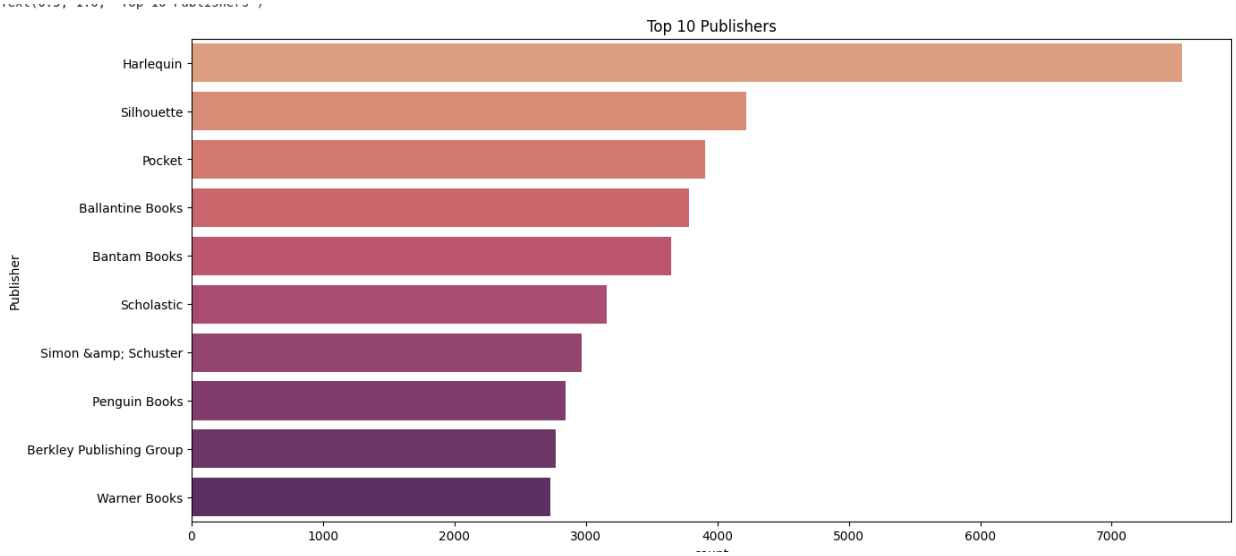



Fig 2(f) : Top 10 Publishers

**Ratings Table Preprocessing**

Books in the ratings dataset should only be those that are included in our book dataset. We discovered that a large number of rows with book ISBNs that are not included in the books dataset are removed.

The distribution of the ratings is wildly asymmetrical, with the great majority of ratings being 0.The book rating data is contained in BX-Book-Ratings, as stated in the dataset description. Either explicit or implicit ratings are indicated by 0 or on a scale of 1 to 10, with higher numbers indicating greater appreciation.Thus, separating the datasets for implicit and explicit ratings.

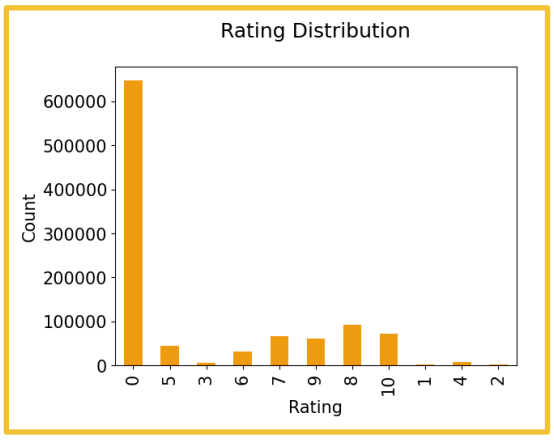



Fig 2(g) : Rating Distribution

*It is evident that consumers tend to give higher ratings, with rating 8 receiving the maximum amount of ratings.*

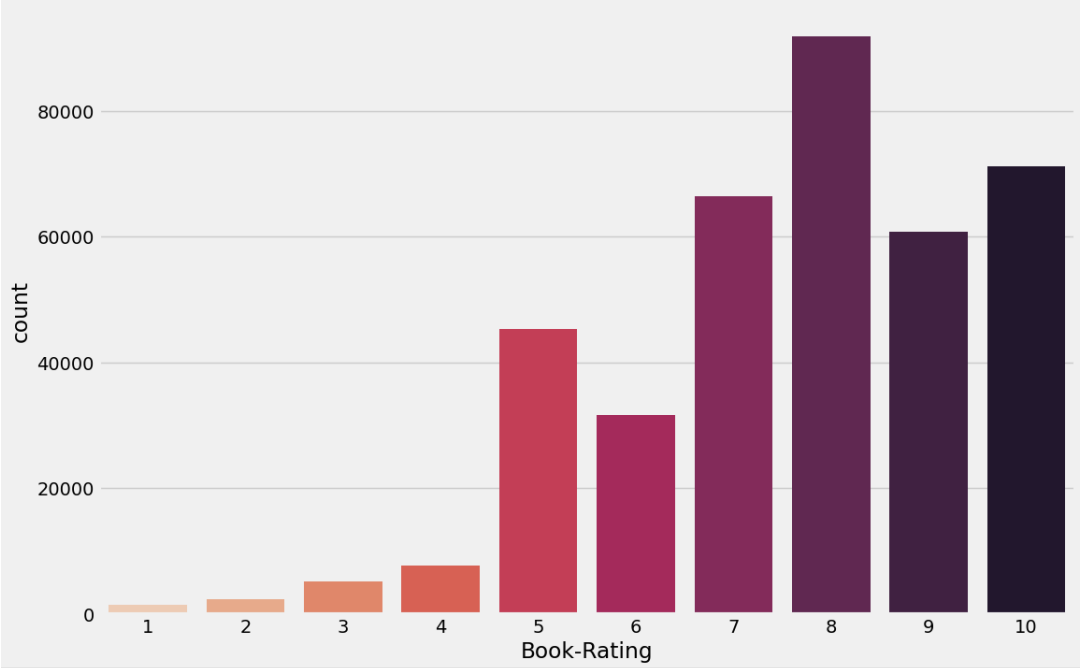



Fig 2(h) : Counting of Books And Rating

**Merging All Dataset**

Merging all three dataset into on Final Dataset

```
[ ] Final_Dataset.head()
```

| | User-ID | Age | Country | ISBN | Book-Rating | Avg_Rating | Total_No_Of_Users_Rated | Book-Title | Book-Author | Year-Of-Publication | Publisher |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 8 | 33.0 | canada | 0002005018 | 5 | 7.666667 | 9 | Clara Callan | Richard Bruce Wright | 2001.0 | HarperFlamingo Canada |
| 1 | 11676 | 28.0 | nan | 0002005018 | 8 | 7.666667 | 9 | Clara Callan | Richard Bruce Wright | 2001.0 | HarperFlamingo Canada |
| 2 | 67544 | 30.0 | canada | 0002005018 | 8 | 7.666667 | 9 | Clara Callan | Richard Bruce Wright | 2001.0 | HarperFlamingo Canada |
| 3 | 116866 | 32.0 | other | 0002005018 | 9 | 7.666667 | 9 | Clara Callan | Richard Bruce Wright | 2001.0 | HarperFlamingo Canada |
| 4 | 123629 | 33.0 | canada | 0002005018 | 9 | 7.666667 | 9 | Clara Callan | Richard Bruce Wright | 2001.0 | HarperFlamingo Canada |

Fig 2(i) : Final Data After Merging

### 3.2.2 Algorithm Development Process

- **Popularity Based Filtering**

  Recommendation algorithms based on popularity follow the current trend. In essence, it makes use of current fashion products. For instance, there's a likelihood that the user who recently signed up will be recommended a book if it's one that every new user purchases.

  Book weighted average formula:

  Weighted Rating(WR) = [vR/(v+m)]+[mC/(v+m)]

  Where,

  V is the total number of votes cast for the books,

M is the minimum number of votes needed for the book to be included in the chart,

R is the book's average rating, and

and C is the mean vote for the entire report.

```
[ ]  C= Final_Dataset['Avg_Rating'].mean()
     m= Final_Dataset['Total_No_Of_Users_Rated'].quantile(0.90)
     Top_Books = Final_Dataset.loc[Final_Dataset['Total_No_Of_Users_Rated'] >= m]
     print(f'C={C} , m={m}')
     Top_Books.shape
```

Here, the 90th percentile served as our cutoff point. Put another way, a book needs to receive more votes than at least 90% of the other books on the list in order to be included in the charts.

We discovered that 38570 books meet the requirements to be included in this list. It is now necessary for us to determine our metric for every eligible book. In order to accomplish this, we will create a new feature score and write a function called weighted_rating(). We will then apply this function to our DataFrame of qualifying books in order to determine its value.

```
[ ]  def weighted_rating(x, m=m, C=C):
         v = x['Total_No_Of_Users_Rated']
         R = x['Avg_Rating']
         return (v/(v+m) * R) + (m/(m+v) * C)


     Top_Books['Score'] = Top_Books.apply(weighted_rating,axis=1)


     #Sorting books based on score calculated above
     Top_Books = Top_Books.sort_values('Score', ascending=False)
```

Applying this function, we finally get our top trending books.

| | Book-Title | Total_No_Of_Users_Rated | Avg_Rating | Score |
|---|---|---|---|---|
| 0 | Harry Potter and the Goblet of Fire (Book 4) | 137 | 9.262774 | 8.741835 |
| 1 | Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback)) | 313 | 8.939297 | 8.716469 |
| 2 | Harry Potter and the Order of the Phoenix (Book 5) | 206 | 9.033981 | 8.700403 |
| 3 | To Kill a Mockingbird | 214 | 8.943925 | 8.640679 |
| 4 | Harry Potter and the Prisoner of Azkaban (Book 3) | 133 | 9.082707 | 8.609690 |
| 5 | The Return of the King (The Lord of the Rings, Part 3) | 77 | 9.402597 | 8.596517 |
| 6 | Harry Potter and the Prisoner of Azkaban (Book 3) | 141 | 9.035461 | 8.595653 |
| 7 | Harry Potter and the Sorcerer's Stone (Book 1) | 119 | 8.983193 | 8.508791 |
| 8 | Harry Potter and the Chamber of Secrets (Book 2) | 189 | 8.783069 | 8.490549 |
| 9 | Harry Potter and the Chamber of Secrets (Book 2) | 126 | 8.920635 | 8.484783 |
| 10 | The Two Towers (The Lord of the Rings, Part 2) | 83 | 9.120482 | 8.470128 |
| 11 | Harry Potter and the Goblet of Fire (Book 4) | 110 | 8.954545 | 8.466143 |
| 12 | The Fellowship of the Ring (The Lord of the Rings, Part 1) | 131 | 8.839695 | 8.441584 |
| 13 | The Hobbit : The Enchanting Prelude to The Lord of the Rings | 161 | 8.739130 | 8.422706 |
| 14 | Ender's Game (Ender Wiggins Saga (Paperback)) | 117 | 8.837607 | 8.409441 |
| 15 | Tuesdays with Morrie: An Old Man, a Young Man, and Life's Greatest Lesson | 200 | 8.615000 | 8.375412 |
| 16 | Charlotte's Web (Trophy Newbery) | 68 | 9.073529 | 8.372037 |
| 17 | Dune (Remembering Tomorrow) | 75 | 8.973333 | 8.353301 |
| 18 | A Prayer for Owen Meany | 181 | 8.607735 | 8.351465 |
| 19 | Fahrenheit 451 | 164 | 8.628049 | 8.346969 |

Fig 2(j) : Trending Books

All users are given access to a generic chart of recommended books via the popularity-based recommender. They don't take into account the preferences and interests of a specific user.

- **Memory based Collaborative Filtering**

  It uses the memory of previous user interactions to compute users similarities based on items they've interacted with (user-based approach) or compute items similarities based on the users that have interacted with them (item-based approach).

I. Train-Test Split:

We will be using the train_test_split function of model_selection class from scikit-learn to create the train test dataset. Train test split 80 : 20 with training set length : 307073 and test set length : 76769.

II. User-Item Matrix

```
n__users = test_dt['u__unique'].nunique()
n__books = test_dt['i__unique'].nunique()

test_matrix = np.zeros((n__users, n__books))

for entry in test_data.itertuples():
   test_matrix[entry[1]-1, entry[2]-1] = entry[3]
```

```
Test_matrix.shape : (25849, 46812)
```

III. Calculating the Cosine Similarity

Cosine similarity is a distance measure that is frequently employed in recommender systems. In this system, ratings are viewed as vectors in n-dimensional space, and the similarity between these vectors is determined by their angle.

To calculate the cosine distance we have used the pairwise_distances from metrics.pairwise class of the scikit-learn framework.

To make Item-Item similarity we need to take the the transpose of the matrix

```
item_predictions=predict_books(train_matrix,item_similarity,type='it
em')
user_predictions=predict_books(train_matrix,user_similarity,type='us
er')
```

IV. Evaluation Metric

We have used MSE(mean squared error) as the evaluation metric

```
#Evaluation metric by mean squared error
from sklearn.metrics import mean_squared_error
from math import sqrt

def rmse(prediction, test_matrix):
   prediction = prediction[test_matrix.nonzero()].flatten()
   test_matrix = test_matrix[test_matrix.nonzero()].flatten()
   return sqrt(mean_squared_error(prediction, test_matrix))

print(f'Item-based       CF       RMSE:       {rmse(item_prediction,
test_matrix_small)}')
print(f'User-based       CF       RMSE:       {rmse(user_prediction,
test_matrix_small)}')
```

```
Item-based CF RMSE: 7.960372815566727
User-based CF RMSE: 7.959470472772056
```

By cosine similarity in the recommendation system it gives a 7.94 RMSE score.

We can make improvements in this score by using another method. Let's use the Single Value Decomposition model (SVD) model to implement.

- **Singular Value Decomposition**

We are using a python framework ‘surprise’ for SVD implementation.

```
model_svd = SVDpp()
model_svd.fit(trainset)
```

On evaluation we found the SVD’s RMSE (root mean squared error) and MAE (mean absolute error) are 1.6368 and 1.2631 respectively.

```
# svd
algo = SVD()
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=1, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.6269  1.6419  1.6391  1.6443  1.6319  1.6368  0.0065
MAE (testset)     1.2544  1.2671  1.2663  1.2670  1.2604  1.2631  0.0050
Fit time          19.63   9.51    14.23   10.04   9.14    12.51   4.00
Test time         1.09    1.47    1.11    0.58    1.37    1.12    0.31
{'test_rmse': array([1.62687291, 1.64192355, 1.63906613, 1.64434404, 1.63189345]),
 'test_mae': array([1.25444761, 1.26709053, 1.2662902 , 1.26702847, 1.26040368]),
 'fit_time': (19.62594747543335,
  9.513878345489502,
  14.23366117477417,
  10.039339303970337,
  9.137583017349243),
 'test_time': (1.0883121490478516,
  1.4707691669464111,
  1.1144611835479736,
  0.5763239860534668,
  1.3657135963439941)}
```

- **Non-Negative Matrix Factorization**

  NMF is also a matrix factorization technique used in recommendation systems.

```
model_nmf = NMF()
model_nmf.fit(trainset)
```

NMF's RMSE and MAE scores are 2.1811 and 1.7517 respectively.

```
Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    2.2043  2.1525  2.1516  2.1976  2.1994  2.1811  0.0238
MAE (testset)     1.7672  1.7299  1.7325  1.7636  1.7656  1.7517  0.0168
Fit time          1.91    1.89    2.22    3.36    2.83    2.44    0.57
Test time         0.02    0.02    0.04    0.20    0.04    0.07    0.07
```

### 3.2.3 Web Application Development

The website will feature multiple pages including a homepage, an "About" page, a "Methods" page detailing the different recommendation algorithms explored, a "Results" page presenting our findings and a "Demo" page where users can see the book recommendation model in action. Flask

has been used to integrate the frontend with the backend, where our trained recommendation models are deployed.

Flask is a lightweight and flexible web framework for Python that is used to build web applications quickly and efficiently. It provides the essential tools and features needed for web development, such as routing, request handling, and template rendering, without the overhead of more complex frameworks. Flask's simplicity and modularity allow developers to choose and integrate only the components they need, making it an ideal choice for both small projects and large-scale applications. Its extensive documentation and active community support further enhance its appeal, ensuring that developers have access to a wealth of resources and examples to guide their development process. In our project, Flask is employed to seamlessly connect the frontend user interface with the backend where our recommendation models are deployed, ensuring smooth and efficient operation of the book recommendation system.

**Project Structure:**

The project is organized into a well-structured hierarchy, ensuring a clean separation of concerns and facilitating ease of development and maintenance. Below is an overview of the project structure:

**Parent Directory**

- **app.py**: This is the main application script that sets up the Flask server, defines routes, and handles HTTP requests. It is the entry point of the application.
- **model.py**: This script contains the BookRecommender class, which includes various functions for handling user data and generating book recommendations.
- **CSV Files**: All necessary data files (such as user ratings and book information) are stored directly in the parent directory. These files are loaded and processed by the application as needed.

**Templates Directory**

- **HTML Files**: This directory contains multiple HTML files, each corresponding to different pages of the website. The templates are used to render dynamic content and provide a consistent layout across the application.
    - **index.html**: The homepage of the website.
    - **about.html**: Provides information about the project.
    - **methods.html**: Describes the various recommendation methods explored.
    - **results.html**: Displays the results and findings of the project.
    - **demo.html**: A live demo page where users can interact with the recommendation system.
    - **contact.html**: A page for user feedback and contact information.

**Static Directory**

- **CSS Folder**: Contains all CSS files for styling the HTML pages. The stylesheets ensure the website is visually appealing and user-friendly.
    - **style.css**: The main stylesheet for the application.
    - **Demo.css:** stylesheet for the demo page
- **Images Folder**: Holds image files used throughout the website, such as logos, background images, and other visual elements.
- **JS Folder**: Contains JavaScript files that add interactivity to the website. These scripts handle client-side logic and enhance the user experience.
    - **script.js**: The main JavaScript file for the application.

This organized structure helps in maintaining the project efficiently and makes it easier to manage different components. The separation of HTML, CSS, and JavaScript files ensures a clean codebase, allowing for easy updates and scalability. The main application logic is contained within `app.py` and `model.py`, providing a clear distinction between the web server functionalities and the recommendation model operations.

# Chapter 4

# Experimental Result

## Front-End Integration

The frontend of the book recommendation system is integrated with the backend using Flask to provide a seamless and dynamic user experience. The integration process involves the following key aspects:

### Dynamic Content Rendering

- Flask's template rendering engine, Jinja2, is used to dynamically insert data into HTML templates. This allows for the generation of personalized recommendations and the display of relevant information to the user.
- **Example**

```
@app.route("/demo/personalized/<int:userid>")
def demoPersonalized(userid):
    users = Recommender.getUsers()
    read = Recommender.read_history(userid)
    recommended_nmf = Recommender.nmf(userid, 10)
    recommended_svd = Recommender.svd(userid, 10)
    return render_template("demopersonalized.html", read = read, svd=recommended_svd, nmf=recommended_nmf, users = users, userid = userid)
```

## Back-End Integration

The backend of the book recommendation system is implemented using Flask, a lightweight and flexible web framework for Python. The backend handles routing, processing user inputs, and serving dynamic content to the frontend. Below is a detailed description of the key routes implemented in the app.py file.

#### 1. Index Route

The index route serves the homepage of the website. It provides an overview of the project and guides users to other sections of the site.

```
from flask import Flask, render_template, redirect
import warnings
from model import Book_Recommender
warnings.filterwarnings('ignore')
Recommender = Book_Recommender()

app = Flask(__name__)
@app.route("/")
def index():
    return render_template("index.html")
```

**2. About Route**

The about route displays information about the project, including the motivation, goals, and team members involved.

```
@app.route("/about")
def about():
    return render_template("about.html")
```

**3. Demo/Trends Route**

The trends route showcases the trending books. It fetches the top trending books using the `trending_books` method from the `BookRecommender` class and displays them on the demo page.

```
@app.route("/demo/trends")
def demoTrends():
    a = Recommender.trending_books()
```

```
    return render_template("demotrends.html", recommended=a)
```

#### 4. Demo/Personalized Route

The personalized route allows users to see personalized book recommendations. It typically requires user identification to generate tailored recommendations.

```
@app.route("/demo/personalized")
def demoPersonalizedDefault():
    users = Recommender.getUsers()
    read = False
    recommended_nmf = False
    recommended_svd = False
    return    render_template("demopersonalized.html",    read    =    read,
svd=recommended_svd, nmf=recommended_nmf, users=users, userid = False)
```

#### 5. Demo/Personalized/UserID Route

This route handles the personalized recommendations for a specific user. It uses the user ID to fetch recommendations via the `nmf` or `svd` method from the `BookRecommender` class.

```
@app.route("/demo/personalized/<int:userid>")
def demoPersonalized(userid):
    users = Recommender.getUsers()
    read = Recommender.read_history(userid)
    recommended_nmf = Recommender.nmf(userid, 10)
    recommended_svd = Recommender.svd(userid, 10)
    return    render_template("demopersonalized.html",    read    =    read,
svd=recommended_svd, nmf=recommended_nmf, users = users, userid = userid)
```

#### 6. Methods Route

The methods route describes the different recommendation algorithms explored in the project. It explains their strengths, weaknesses, and use cases.

```
@app.route("/methods")
def methods():
   return render_template("methods.html")
```

### BookRecommender Class

The `BookRecommender` class in `model.py` encapsulates the core functionalities of the recommendation system. Here is a brief overview of the class:

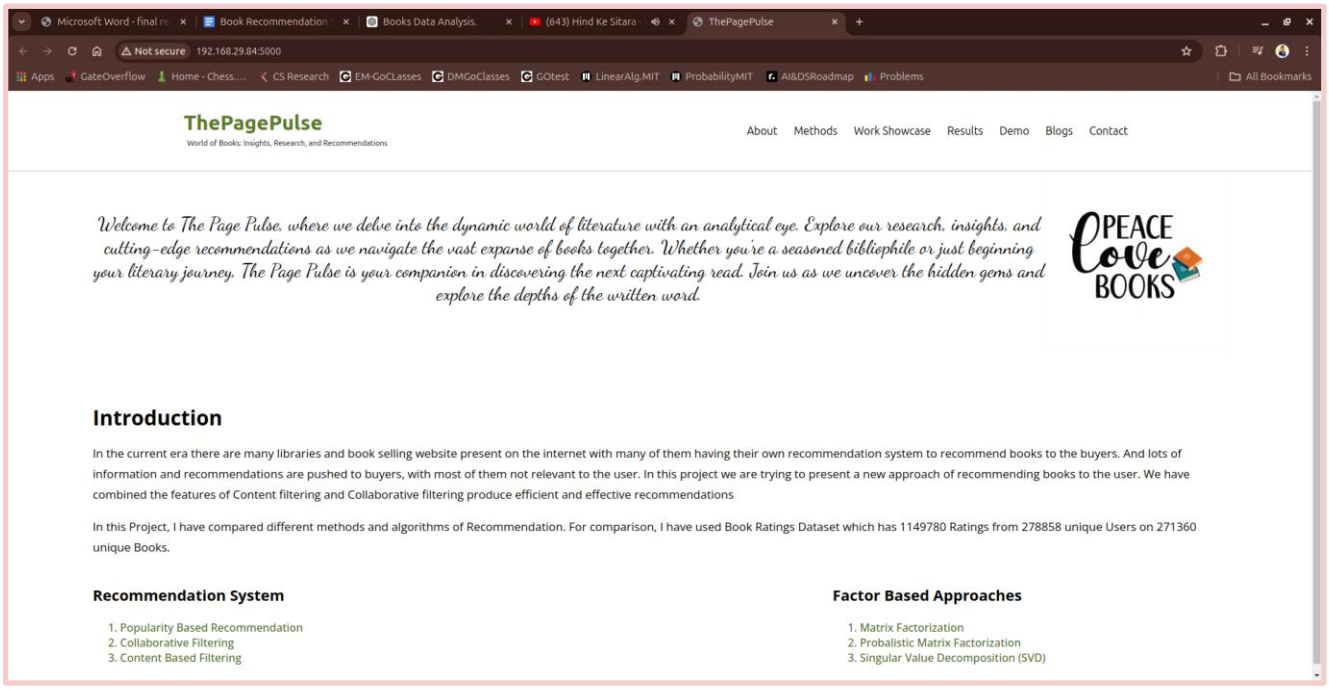
#### BookRecommender Class in model.py

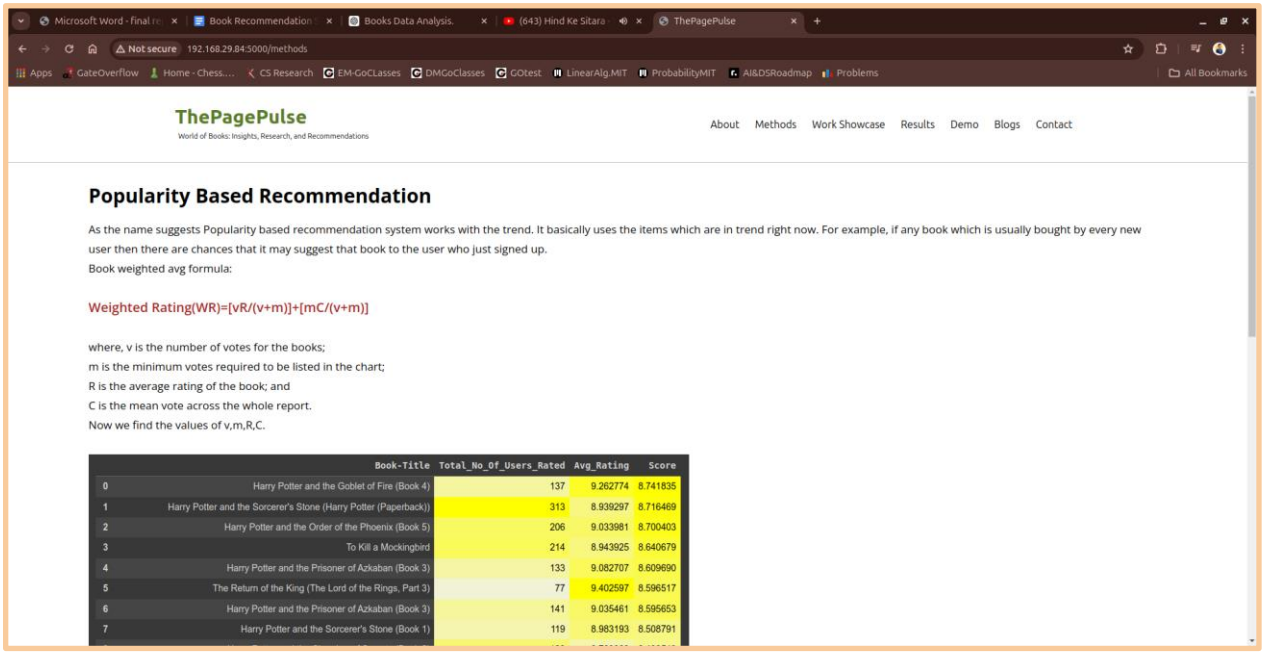
```
class Book_Recommender:

      def __init__(self):

            self.user_list = self.usersList(10)

      def getUsers(self):

            return self.user_list

      def setUsers(self):

            self.user_list = self.usersList(10)
      def svd(self, user-id, n=5):
           // Singular Value Decomposition implementation
      def nmf(self, user-id, n=5):
           // Non-Negative Matrix Factorization implementation
      def read_history(self, user-id):
           // users book read history
      def trending_books(self, n=20):
           // top 20 trending books
      def usersList(self, n=20):
            // random 20 users id
```

The backend implementation of the book recommendation system is designed to be modular and scalable. Each route serves a specific purpose, from displaying general project information to providing personalized recommendations. The `BookRecommender` class encapsulates the core recommendation algorithms, making the code easy to manage and extend. By integrating these

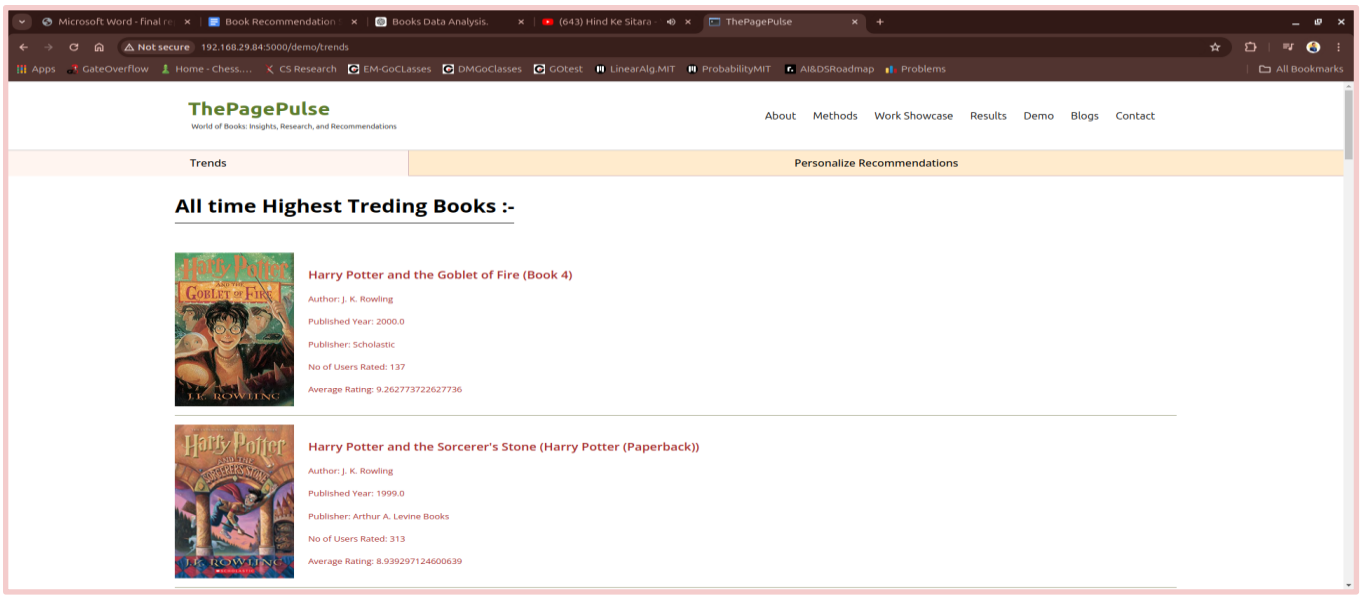
routes with dynamic HTML templates, the backend ensures a seamless and interactive user experience.

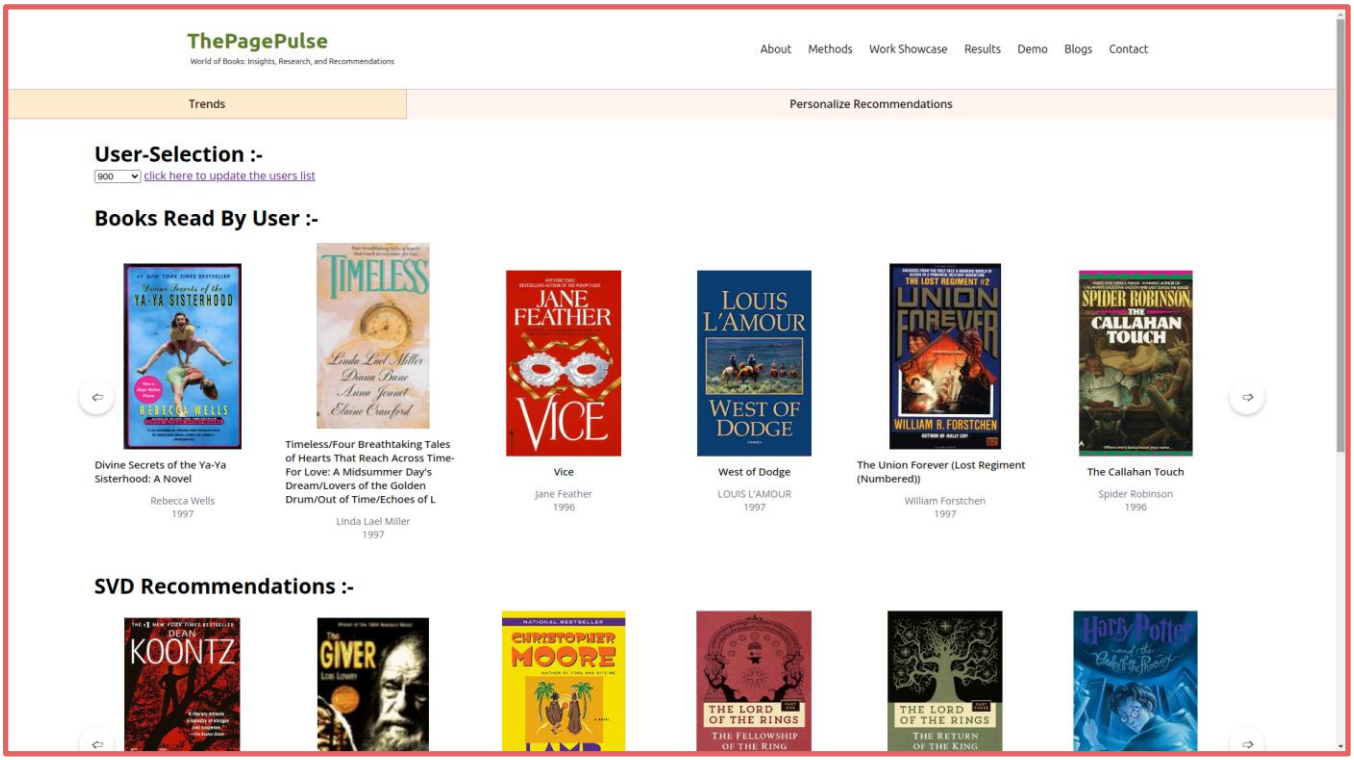
## Web Application Showcase




| | Book-Title | Total_No_Of_Users_Rated | Avg_Rating | Score |
|---|---|---|---|---|
| 0 | Harry Potter and the Goblet of Fire (Book 4) | 137 | 9.262774 | 8.741835 |
| 1 | Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback)) | 313 | 8.939297 | 8.716469 |
| 2 | Harry Potter and the Order of the Phoenix (Book 5) | 206 | 9.033981 | 8.700403 |
| 3 | To Kill a Mockingbird | 214 | 8.943925 | 8.640679 |
| 4 | Harry Potter and the Prisoner of Azkaban (Book 3) | 133 | 9.082707 | 8.609690 |
| 5 | The Return of the King (The Lord of the Rings, Part 3) | 77 | 9.402597 | 8.596517 |
| 6 | Harry Potter and the Prisoner of Azkaban (Book 3) | 141 | 9.035461 | 8.595653 |
| 7 | Harry Potter and the Sorcerer's Stone (Book 1) | 119 | 8.983193 | 8.508791 |

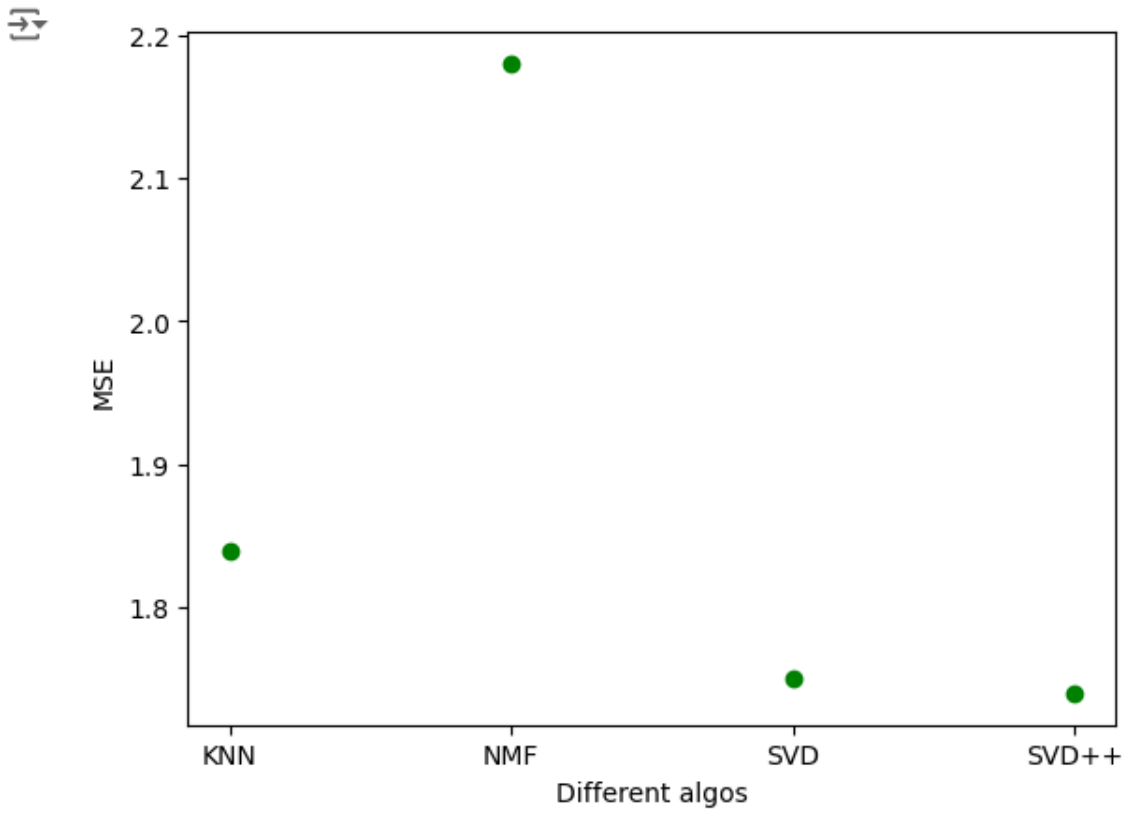

ThePagePulse
World of Books: Insights, Research, and Recommendations
About
Methods
Work Showcase
Results
Demo
Blogs
Contact
Trends
Personalize Recommendations
All time Highest Treding Books :-
Harry Potter and the Goblet of Fire (Book 4)
Author: J. K. Rowling
Published Year: 2000.0
Publisher: Scholastic
No of Users Rated: 137
Average Rating: 9.262773722627736
Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
Author: J. K. Rowling
Published Year: 1999.0
Publisher: Arthur A. Levine Books
No of Users Rated: 313
Average Rating: 8.939297124600639


ThePagePulse
World of Books: Insights, Research, and Recommendations
About
Methods
Work Showcase
Results
Demo
Blogs
Contact
Trends
Personalize Recommendations
User-Selection :-
900
click here to update the users list
Books Read By User :-
Divine Secrets of the Ya-Ya Sisterhood: A Novel
Rebecca Wells
1997
Timeless/Four Breathtaking Tales of Hearts That Reach Across Time-For Love: A Midsummer Day's Dream/Lovers of the Golden Drum/Out of Time/Echoes of L
Linda Lael Miller
1997
Vice
Jane Feather
1996
West of Dodge
LOUIS L'AMOUR
1997
The Union Forever (Lost Regiment (Numbered))
William Forstchen
1997
The Callahan Touch
Spider Robinson
1996
SVD Recommendations :-

# Chapter 5

# Conclusions

Using a variety of approaches, we were able to create a strong book recommendation system for this project that improves the relevance and accuracy of book recommendations. Through the investigation and use of multiple recommendation algorithms, such as matrix factorization and collaborative filtering methods, we evaluated each algorithm's capabilities, drawbacks, and overall effectiveness. Our approach illustrates the usefulness of modern data science techniques in improving user experiences by offering consumers personalized book recommendations based on their individual interests and reading histories.



Using Flask, the development process created a multi-page web application that seamlessly integrated the frontend and backend. This platform provides a live demonstration of the recommendation system so that people may interact with it and see its operation in real time, in addition to showcasing our findings and methods. The system's capabilities are guaranteed to be benefited from by non-technical users because of its user-friendly interface.

Our experiment emphasizes how crucial trustworthy and accurate recommendation systems are in a society where there is an abundance of options and information. We have made a contribution to the field of recommendation systems by addressing the shortcomings of the current systems and offering more practical solutions, opening the door for further advancements and developments. This work lays the groundwork for future research and development, with the goal of continuously improving suggestion quality by integrating increasingly advanced algorithms and data sources.

## 5.2 References


- Yiu-Kai Ng (2020) :CBRec: A BOOK RECOMMENDATION SYSTEM FOR A CHILDREN USING THE MATRIX FACTORIZATION AND CONTENT BASED FILTERING APPROACHES.

- Pegah Sagheb Haghighi University of Louisville(2021) : COSINE-BASED EXPLAINABLE MATRIX FACTORIZATION FOR COLLABORATIVE FILTERING RECOMMENDATION.

- Ashish Fatarphekar, Tejas Nashikkar, Vivek Patil, Gayatri Naik, "BOOK RECOMMENDATION SYSTEM BASED ON COMBINE FEATURES OF CONTENT BASED FILTERING, COLLABORATIVE FILTERING AND ASSOCIATION RULE MINING", 2015, Advance Research in Social science and Humanities (ISSN: 2208-2387).

- Ms. Praveena Mathew, Ms. Bincy Kuriakose, Mr.Vinayak Hegde, "Book Recommendation System through Content Based and Collaborative Filtering Method", 2016, Department of Computer Science
  Amrita Vishwa Vidyapeetham Mysuru Campus Mysuru, Karnataka, India

| 21% | 10% | 9% | 16% |
| --- | --- | --- | --- |
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
| --- | --- | --- |
| 1 | Submitted to University of Wolverhampton<br>Student Paper | 1% |
| 2 | Submitted to University of Bolton<br>Student Paper | 1% |
| 3 | Submitted to University of Portsmouth<br>Student Paper | 1% |
| 4 | Submitted to Uttar Pradesh Technical University<br>Student Paper | 1% |
| 5 | Submitted to California State University, San Bernadino<br>Student Paper | 1% |
| 6 | nb.recohut.com<br>Internet Source | 1% |
| 7 | Submitted to Kamla Nehru Institute of Technology Sultanpur<br>Student Paper | 1% |
| 8 | Submitted to University of North Texas<br>Student Paper | 1% |

| | | |
|---|---|---|
| 10 | Submitted to Delhi Metropolitan Education<br>Student Paper | 1 % |
| 11 | www.ijirset.com<br>Internet Source | 1 % |
| 12 | origin.tutorialspoint.com<br>Internet Source | 1 % |
| 13 | Submitted to Anna University<br>Student Paper | 1 % |
| 14 | Submitted to Liverpool John Moores University<br>Student Paper | <1 % |
| 15 | www.techtarget.com<br>Internet Source | <1 % |
| 16 | Submitted to Australian National University<br>Student Paper | <1 % |
| 17 | Ton Duc Thang University<br>Publication | <1 % |
| 18 | Submitted to Universiteit van Amsterdam<br>Student Paper | <1 % |
| 19 | www-cse.ucsd.edu<br>Internet Source | <1 % |
| 20 | Submitted to University of Hertfordshire | |

| | | |
|---|---|---|
| 21 | Submitted to bannariamman<br>Student Paper | <1 % |
| 22 | Recommender Systems Handbook, 2011.<br>Publication | <1 % |
| 23 | Submitted to The British College<br>Student Paper | <1 % |
| 24 | Submitted to Bahrain Polytechnic<br>Student Paper | <1 % |
| 25 | J. Ben Schafer. "Collaborative Filtering Recommender Systems", Lecture Notes in Computer Science, 2007<br>Publication | <1 % |
| 26 | medium.com<br>Internet Source | <1 % |
| 27 | vdocuments.net<br>Internet Source | <1 % |
| 28 | digilib.esaunggul.ac.id<br>Internet Source | <1 % |
| 29 | Submitted to Southampton Solent University<br>Student Paper | <1 % |
| 30 | SongJie Gong, HongWu Ye, YaE Dai. "Combining Singular Value Decomposition and Item-based Recommender in | <1 % |

International Workshop on Knowledge Discovery and Data Mining, 2009
Publication

| | | |
|---|---|---|
| 31 | arxiv.org<br>Internet Source | <1 % |
| 32 | Submitted to Banaras Hindu University<br>Student Paper | <1 % |
| 33 | Submitted to Letterkenny Institute of Technology<br>Student Paper | <1 % |
| 34 | Submitted to Aligarh Muslim University, Aligarh<br>Student Paper | <1 % |
| 35 | Submitted to Middle Tennessee State University<br>Student Paper | <1 % |
| 36 | Submitted to South Bank University<br>Student Paper | <1 % |
| 37 | Submitted to Universiti Kebangsaan Malaysia<br>Student Paper | <1 % |
| 38 | Submitted to University of Bedfordshire<br>Student Paper | <1 % |
| 39 | ecc.isc.ac<br>Internet Source | <1 % |

| | | |
|---|---|---|
| 41 | koreascience.or.kr<br>Internet Source | <1 % |
| 42 | 5dok.net<br>Internet Source | <1 % |
| 43 | Ke Ji, Hong Shen. "Jointly modeling content, social network and ratings for explainable and cold-start recommendation", Neurocomputing, 2016<br>Publication | <1 % |
| 44 | Gediminas Adomavicius, Jingjing Zhang. "Improving Stability of Recommender Systems: A Meta-Algorithmic Approach", IEEE Transactions on Knowledge and Data Engineering, 2015<br>Publication | <1 % |
| 45 | Submitted to Higher Education Commission Pakistan<br>Student Paper | <1 % |
| 46 | link.springer.com<br>Internet Source | <1 % |
| 47 | pubsonline.informs.org<br>Internet Source | <1 % |
| 48 | Aryudha Hattu, Yeremia Susetyo. "Development of Operational Application | <1 % |

and Computer Science (IJSECS), 2024
Publication

| | | |
|---|---|---|
| 49 | Submitted to Sreenidhi International School<br>Student Paper | <1 % |
| 50 | Submitted to University of Computer Studies<br>Student Paper | <1 % |
| 51 | www.globalknowledge.com<br>Internet Source | <1 % |
| 52 | www2.mdpi.com<br>Internet Source | <1 % |
| 53 | "Applied Assistive Technologies and Informatics for Students with Disabilities", Springer Science and Business Media LLC, 2024<br>Publication | <1 % |
| 54 | Huazhen Liu, Wei Wang, Yihan Zhang, Renqian Gu, Yaqi Hao. "Neural Matrix Factorization Recommendation for User Preference Prediction Based on Explicit and Implicit Feedback", Computational Intelligence and Neuroscience, 2022<br>Publication | <1 % |
| 55 | Li-Zhu Zhou. "Similarity measure and instance selection for collaborative filtering", Proceedings of the twelfth international | <1 % |

Publication

| | | |
|---|---|---|
| 56 | github.com<br>Internet Source | <1 % |
| 57 | pure.eur.nl<br>Internet Source | <1 % |
| 58 | repositories.lib.utexas.edu<br>Internet Source | <1 % |
| 59 | studyres.com<br>Internet Source | <1 % |
| 60 | topic.alibabacloud.com<br>Internet Source | <1 % |
| 61 | www.mdpi.com<br>Internet Source | <1 % |
| 62 | www.ncbi.nlm.nih.gov<br>Internet Source | <1 % |
| 63 | Sarabjot Singh Anand. "Intelligent Techniques for Web Personalization", Lecture Notes in Computer Science, 2005<br>Publication | <1 % |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 6 words |
| Exclude bibliography | On | | |