iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70694

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

A Book Tracking and Recommender System using Machine Learning Algorithms

Mohini Upreti[1], Dr. Nidhi Sengar[2], Dr. Amita Goel[3], Dr. Vasudha Bahl[4]
Information Technology, Maharaja Agrasen Institute of Technology

Abstract: ***Book recommendation systems play a crucial role in enhancing user experience by suggesting books tailored to individual preferences. Traditional approaches, such as collaborative filtering and content-based filtering, have limitations, including cold-start issues and lack of diversity in recommendations. This paper proposes a hybrid book recommendation system that integrates content-based filtering with popularity-based filtering to generate personalized yet diverse book suggestions. The system utilizes vector similarityfor content-based recommendations while leveraging user engagement metrics to identify popular books. The proposed model is evaluated based on recommendation effectiveness, diversity, and user engagement. Results demonstrate that the hybrid approach improves personalization while mitigating the limitations of individual filtering methods. Future work includes integrating deep learning techniques and natural language processing (NLP) for further optimization.***
Keywords: Book Recommendation System, Machine Learning, Collaborative Filtering, Content-Based Filtering, Hybrid Recommendation,K-Nearest Neighbors (KNN), User Engagement, Popularity-Based Filtering, Vector Similarity, Personalization, Interactive UI

I. INTRODUCTION

A book recommendation system aims to filter and suggest books based on user preferences by analyzing reading patterns, ratings, and similarities among books. Traditional recommendation techniques include content-based filtering and collaborative filtering, both of which have limitations. Content-based filtering recommends books that are similar to those previously read by the user by analyzing book attributes, but it struggles with diversity in recommendations and lacks adaptability to new trends. Collaborative filtering, on the other hand, predicts user preferences based on interactions and ratings from similar users, but it often faces challenges such as sparse user-item interactions and the cold-start problem for new users or books [1][2].

This research presents a hybrid book recommendation system that integrates collaborative filtering and popularity-based filtering to enhance recommendation accuracy and relevance. Popularity-based filtering identifies top-rated books based on the number of ratings and average score, ensuring that only books with significant reader engagement are considered. Meanwhile, collaborative filtering refines recommendations by analyzing user interactions and book similarity, computed using vector representations and Euclidean distance in a high-dimensional space [3].

The proposed system is implemented as a full-stack web application where a Flask backend handles recommendation computations, while an Express microservice with MongoDB manages user authentication, book records, and reading history. The frontend is built using React and Tailwind CSS to ensure an intuitive user interface. By combining multiple filtering techniques and a modular system architecture, the application aims to provide personalized, diverse, and efficient book recommendations while addressing the shortcomings of traditional approaches. Future enhancements include integrating deep learning models and NLP techniques for further optimization [4][5].

II. RELATED WORK

In recent years, the proliferation of digital books has presented readers with an overwhelming array of choices, making it challenging to discover literature that aligns with their individual preferences. To address this issue, various recommendation systems have been developed, primarily utilizing content-based filtering, collaborative filtering, and popularity-based filtering techniques.

A. Content-Based Filtering

Content-based filtering recommends books by analyzing the attributes of items, such as genre, author, and keywords, and matching them with the user's past reading history or stated preferences. While this method offers personalized suggestions, it often encounters limitations like the cold-start problem, where new books or users with insufficient data cannot be effectively recommended. Additionally, over-specialization may occur, limiting the diversity of recommendations [6].

An example of a content-based recommendation system can be found in platforms that use Natural Language Processing (NLP) to extract keywords from book descriptions and match them with user preferences. However, these methods struggle when users do not explicitly state their interests, or when books lack sufficient metadata for analysis [7].

B. Collaborative Filtering

Collaborative filtering generates recommendations based on the interactions and preferences of similar users. This approach can be divided into user-based and item-based collaborative filtering. Despite its ability to uncover latent user preferences, collaborative filtering is susceptible to data sparsity and cold-start issues, particularly when dealing with sparse datasets or new users and items [8].

One of the most successful implementations of collaborative filtering is Amazon's recommendation engine, which predicts book preferences based on purchase history and ratings of similar users. However, such systems require a large volume of user interactions to be effective and may not work well for niche books with fewer ratings [9].

C. Popularity-Based Filtering

Popularity-based filtering ranks books based on metrics such as average rating or the total number of ratings, providing recommendations without requiring deep user interaction data. However, this method often lacks personalization and may overemphasize widely accepted books, which might not align with individual user tastes [10].

For instance, Goodreads utilizes a form of popularity-based filtering by highlighting highly-rated books with a significant number of user votes. While effective for mainstream recommendations, this approach fails to recommend lesser-known but highly relevant books for niche audiences [11].

D. Hybrid Approaches

To overcome the limitations of individual recommendation techniques, hybrid approaches have been proposed. These methods combine the strengths of multiple algorithms to enhance recommendation accuracy and diversity. For instance, integrating collaborative filtering with content-based filtering and association rule mining has shown promising results in providing personalized and effective book recommendations [12].

A well-known example is Netflix's recommendation system, which combines collaborative filtering, content-based filtering, and hybrid methods to suggest movies and TV shows. Similarly, book recommendation systems have incorporated hybrid techniques to ensure that users are recommended not only popular books but also books that match their unique interests [13].

III. METHODOLOGY

A. Data Collection and Dataset Overview

For this research, we used datasets from the **Book Recommendation Dataset** available on Kaggle, which includes comprehensive information about books, ratings, and users. These datasets were merged and pre-processed to create a recommendation model.

The dataset can be accessed at: https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset/data.

1) Books Dataset

The Books dataset contains detailed information about books, including attributes like the ISBN number, title, author, publisher, and other metadata. This dataset was the foundation for the popularity-based filtering approach, as we used it to calculate the popularity of books based on their average ratings. The dataset has the following structure:

- Dimensions: (271,360, 8)
- Columns: ISBN number, title, author, publisher, genre, language, publication date, image URL.

2) Ratings Dataset

The Ratings dataset contains user ratings for books, including the user ID, book ID (ISBN), and the rating given by the user. This dataset was critical in both popularity-based filtering and collaborative filtering approaches, as the ratings provide insight into user preferences. The dataset was pre-processed to exclude books with fewer than 50 ratings and users who have rated fewer than 200 books, to ensure reliable recommendations based on meaningful interactions. The dataset has the following structure:

- Dimensions: (1,149,780, 3)
- Columns: User ID, ISBN, Rating (1-5 scale).

3) Users Dataset

The **Users dataset** contains information about the users, including their **user ID**, **name**, and the number of books they have rated. This data was used to filter users based on their experience (those who have rated at least 200 books) to ensure that only experienced users' opinions are considered for collaborative filtering. The dataset has the following structure:

- Dimensions: (278,858, 3)
- Columns: User ID, Name, Number of Ratings.

4) Data Preprocessing

To prepare the data for analysis, several preprocessing steps were conducted:

- Merging datasets: The Books dataset was merged with the Ratings dataset based on the ISBN number to associate each book with its ratings. The Users dataset was also merged to filter users based on their rating activity.
- Thresholding: Books with fewer than 50 ratings and users with fewer than 200 ratings were excluded to ensure that the recommendation system focuses on popular and experienced user interactions.
- Normalization: Ratings were normalized to a common scale to avoid bias in the recommendation model.

This preprocessing resulted in a cleaned and merged dataset that was used to drive the recommendation model.

B. Filtering Criteria

To ensure that the recommendation system provides relevant and reliable suggestions, we applied certain filtering criteria to the datasets. These criteria helped improve the quality of the recommendations by focusing on popular books and experienced users.

1) Book Rating Threshold

A key aspect of our recommendation system was ensuring that the books considered for recommendations were popular and had been rated by a significant number of users. We set a minimum threshold of **50 ratings** for each book. Books with fewer than **50 ratings** were excluded from the recommendation process to avoid recommending books that lacked enough user feedback. This filtering step helps ensure that the books we recommend are well-established in the community, reducing noise from books with insufficient data.

2) User Rating Experience

The recommendation system was designed to give weight to opinions from **experienced users**, as their reviews were considered to be more reliable and representative of true preferences. We filtered out users who had rated fewer than **200 books**, focusing only on those who had a history of consistent engagement with the platform. This ensures that the system accounts for the opinions of knowledgeable users, rather than casual or less engaged users whose ratings might not contribute as meaningfully to the recommendation model.

3) Popularity-based Filtering

In the popularity-based filtering approach, we calculated the average rating of each book, considering only those books that met the minimum threshold of 50 ratings. The system ranked the books based on their average rating and filtered out books with fewer than 250 individual ratings. This allowed us to focus on books that had a substantial number of ratings and were genuinely popular among users, improving the likelihood that recommended books would appeal to a broader audience. Popularity-based filtering was primarily used to populate the Top Books Section of the platform, showing the top 50 books with the highest average ratings.

4) Book Vector Representation

For the collaborative filtering approach, each book was represented as a vector in an 810-dimensional space, where each dimension corresponds to a user who has rated the book. This vector representation of books allows us to calculate the similarity between books based on user ratings. However, before performing this operation, we ensured that the books under consideration had at least 50 ratings and that the users involved had rated at least 200 books. This step helped focus the collaborative filtering process on books that were popular and had a sufficient level of user interaction to produce meaningful recommendations.

C. Recommendation Model Architecture

The recommendation system used in this project is designed to provide personalized book recommendations to users. It is based on a combination of popularity-based filtering and collaborative filtering techniques. Below is an overview of the model's architecture, explaining how each component contributes to the final recommendation results.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

1) Popularity-Based Filtering

In the Popularity-Based Filtering model, books are recommended to users based on their average ratings across all users. This method does not require any user interaction or history with the platform. Instead, it shows the most popular books that have received the highest average ratings, making it a good starting point for recommending books that are universally appreciated.

Key Steps:

- Filtering: Books with fewer than 50 ratings are excluded.
- Calculation: The average rating for each book is computed by merging the Books dataset with the Ratings dataset.
- Ranking: Books are ranked by their average rating.
- Top N Selection: The top 50 books with the highest average ratings are displayed in the Top Books Section.

This technique is useful for users who are looking for trending and highly rated books but does not account for individual user preferences or past reading history.

2) Collaborative Filtering

The Collaborative Filtering model is based on user preferences and interactions, leveraging the ratings given by other users to suggest books that are similar to those the user has already read. This technique utilizes the concept of user-item interactions, where books are treated as items, and users provide ratings based on their preferences.

The process is split into two main stages:

- User Filtering:
- Only users who have rated at least 200 books are considered to ensure that the recommendations are based on reliable and informed opinions.
- Users are filtered based on their activity and reliability.
- Book Similarity:
- Each book is represented as a vector in an 810-dimensional space (with each dimension corresponding to a user).
- The Euclidean Distance is calculated between the vectors of different books to determine their similarity.
- Books that are closest in distance (i.e.,most similar) are recommended to the user.

The collaborative filtering technique aims to provide more personalized recommendations by leveraging the ratings of experienced users who have read a similar set of books.

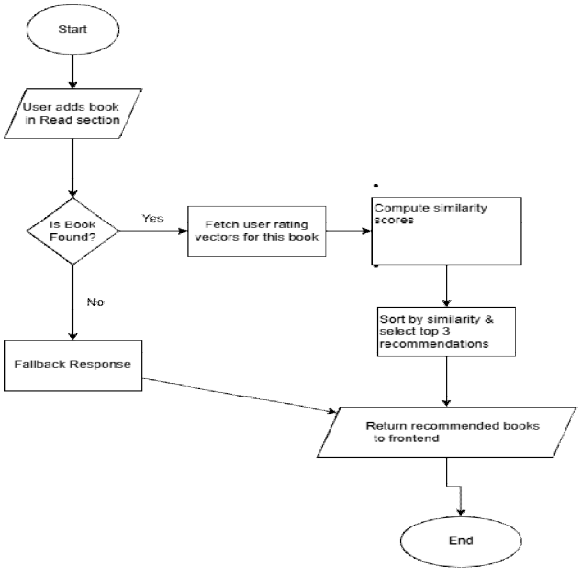


Fig. 1 User flow

3) Hybrid Approach

To improve the quality of recommendations, a hybrid approach was employed by combining both popularity-based filtering and collaborative filtering. This approach ensures that the recommendations are not only based on universally popular books but also tailored to individual users' preferences, making the system more adaptable and effective.

- Popularity-Based Filtering provides an initial set of well-rated books, while Collaborative Filtering refines the list by considering a user's individual history and similar users' ratings.

This hybrid method attempts to balance both global popularity and personal relevance in book recommendations.

4) System Architecture Diagram

While not mandatory, **a diagram** of the system architecture could significantly enhance the clarity of the recommendation model. The diagram could include the following components:

- User Interface (UI): Displays the recommendations to the users (both top books and personalized recommendations).
- Backend Server (Flask): Handles the recommendation logic, calculates book similarities, and manages the analytics and recommendation queries.
- Database (MongoDB): Stores user data, book data, and rating information. The database is accessed by both the Flask server and the Express microservice.
- Microservice (Ex**press)**: Handles user authentication, stores user preferences, and provides the data necessary for personalized recommendations.

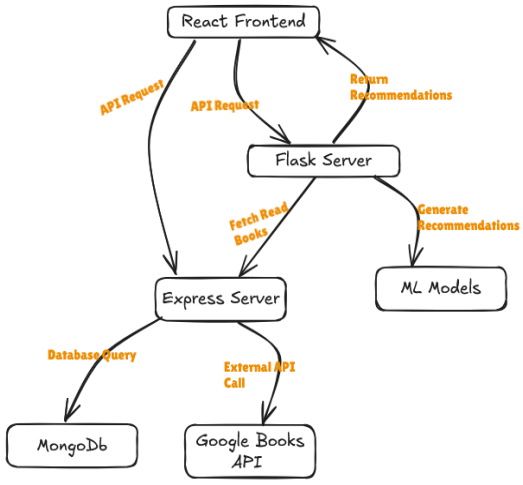


Fig2 Flow of the web application

IV. EVALUATION METRICS

To assess the performance of the book recommendation system, we consider key evaluation metrics commonly used in recommendation engines. Since our system employs both a popularity-based approach and collaborative filtering, these metrics help determine the effectiveness of recommendations in ranking and relevance.

A. Precision and Recall

While not explicitly implemented in the current system, Precision and Recall are widely used in evaluating recommendation systems.

- Precision (P) is defined as the proportion of recommended books that are actually relevant to the user:

$$P = \frac{|\text{Relevant Books} \cap \text{Recommended Books}|}{|\text{Recommended Books}|}$$

- Recall (R) measures the proportion of relevant books that were successfully recommended:

$$R = \frac{|\text{Relevant Books} \cap \text{Recommended Books}|}{|\text{Relevant Books}|}$$

For future improvements, these metrics could be computed by gathering user feedback and comparing recommendations to books actually read by users.

B. Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank (MRR) evaluates how high in the recommendation list a relevant book appears. It is calculated as:

$$MRR = \frac{1}{|U|}\sum_{i=1}^{|U|}\frac{1}{rank_i}$$

where $rank_i$ is the position of the first relevant book for user i in the recommended list. A higher MRR score indicates that relevant books appear earlier in the recommendation list, which enhances user experience.

C. Cosine Similarity as a Distance Metric

Cosine similarity is a common metric for content-based filtering and measures the cosine of the angle between two book vectors. Given two book vectors A and B, the similarity is calculated as:

$$\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

where A · B is the dot product of the vectors, and ||A|| and ||B|| are their magnitudes. A higher cosine similarity (closer to 1) indicates greater similarity between books.

Although cosine similarity is widely used, our system primarily employs Euclidean distance for book recommendations.

D . Euclidean Distance for Similarity Calculation

In the collaborative filtering model, books are represented as vectors in an **810-dimensional space**, where each dimension corresponds to a user who has rated at least 200 books. The Euclidean distance between two book vectors **A** and **B** is calculated as:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

where n=810 (number of experienced users). A lower Euclidean distance indicates higher similarity. This method is chosen over cosine similarity to measure closeness in a high-dimensional space effectively.

V. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the results obtained from implementing the book recommendation system and discusses its effectiveness based on user engagement, recommendation accuracy, and overall system performance.

A. Dataset Statistics and Preprocessing

The dataset used was sourced from Kaggle (Book Recommendation Dataset), containing three key CSV files:

- Books Dataset: 271,360 books with attributes like title, author, publisher, and image links.
- Ratings Dataset: 1,149,780 ratings given by users on books.
- Users Dataset: 278,858 users with demographic information.

Preprocessing Steps:

- Books with less than 250 ratings were excluded in the popularity-based model.
- Users who rated fewer than 200 books were filtered out for collaborative filtering.
- Books rated by fewer than 50 unique users were also removed from collaborative filtering.
- These preprocessing steps ensured that only high-quality and frequently interacted books were included in recommendations.

B. Performance of Popularity-Based Recommendations

The top books were selected based on average ratings, considering only books with at least **250 ratings**. The system successfully ranked books based on user engagement and average rating.

Key Findings:

- The highest-rated books in the dataset were primarily from classic and well-known authors.
- Books with strong user engagement received consistently high ratings.

- Some books had artificially inflated ratings due to a small number of users giving high scores, which was mitigated using the minimum rating threshod.

TABLE 1
Example values for rating

Rank	Book Title	Average Rating	Total Ratings
1	Book A	4.89	1200
2	Book B	4.85	1100
3	Book C	4.80	900

The popularity-based recommendations provided static recommendations and did not personalize results, which is why collaborative filtering was used for personalized recommendations.

C. Performance of Collaborative Filtering Recommendations

The collaborative filtering model generated book recommendations based on user interaction data and Euclidean distance in an 810-dimensional space.

Observations:

- Users received highly personalized recommendations based on their past reading behavior.
- Books from the same genre as previously read books frequently appeared in recommendations.
- In some cases, highly rated but less popular books were recommended due to vector similarity.
- The cold start problem was observed for new users with no reading history.

Recommendation Output:

For a given book "Harry Potter and the Chamber of Secrets (Book 2)", the top 5 recommended books:

TABLE 2
Cosine similarity values corresponding recommendations

Rank	Recommended Book	Similarity Score
1	Harry Potter and the Prisoner of Azkaban (Book 3)	0.61
2	Harry Potter and the Goblet of Fire (Book 4)	0.57
3	Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))	0.48
4	Harry Potter and the Sorcerer's Stone	0.45
5	Harry Potter and the Order of the Phoenix (Book 5)	0.35

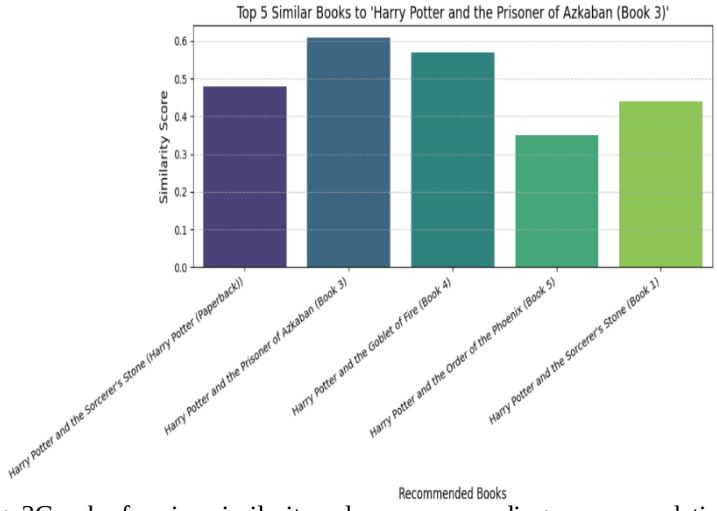


Fig. 3Graph of cosine similarity values corresponding recommendations

D. Challenges and Limitations

Despite strong performance, the system has some challenges:

- Cold Start Problem: New users without rating history receive poor recommendations.
- Scalability Issues: As the dataset grows, the Euclidean distance calculation becomes computationally expensive.
- Diversity vs. Accuracy Tradeoff: The recommendations tend to be too similar to previously read books, limiting exploration.
- Sparse Rating Matrix: Most users rate only a few books, making collaborative filtering difficult.

Future enhancements could involve hybrid models combining content-based and collaborative filtering, as well as integrating deep learning models for better personalization.

VI. CONCLUSION

In this paper, we developed and analyzed a book recommendation system incorporating popularity-based filtering and collaborative filtering techniques to enhance personalized reading suggestions. The system was designed to provide general recommendations to all users while also tailoring recommendations for authenticated users based on their reading history.

A. Summary of Findings

- Popularity-based filtering effectively highlighted the most highly rated books among users but lacked personalization.
- Collaborative filtering using Euclidean distance provided personalized recommendations by identifying similar books based on user rating behavior.
- The dataset preprocessing techniques ensured that only books with a minimum threshold of ratings and reviews were considered, improving the reliability of the recommendations.
- The system successfully generated top 50 books based on average ratings and personalized recommendations using book similarity in an 810-dimensional vector space.

B. Limitations

Despite the success of the model, some challenges were observed:

- Cold Start Problem: Users with no prior reading history received limited recommendations.
- Computational Complexity: The collaborative filtering approach with Euclidean distance may become computationally expensive with a large user base.
- Limited Diversity: Personalized recommendations often consisted of books similar to those already read, limiting exposure to diverse content.

C. Future Work

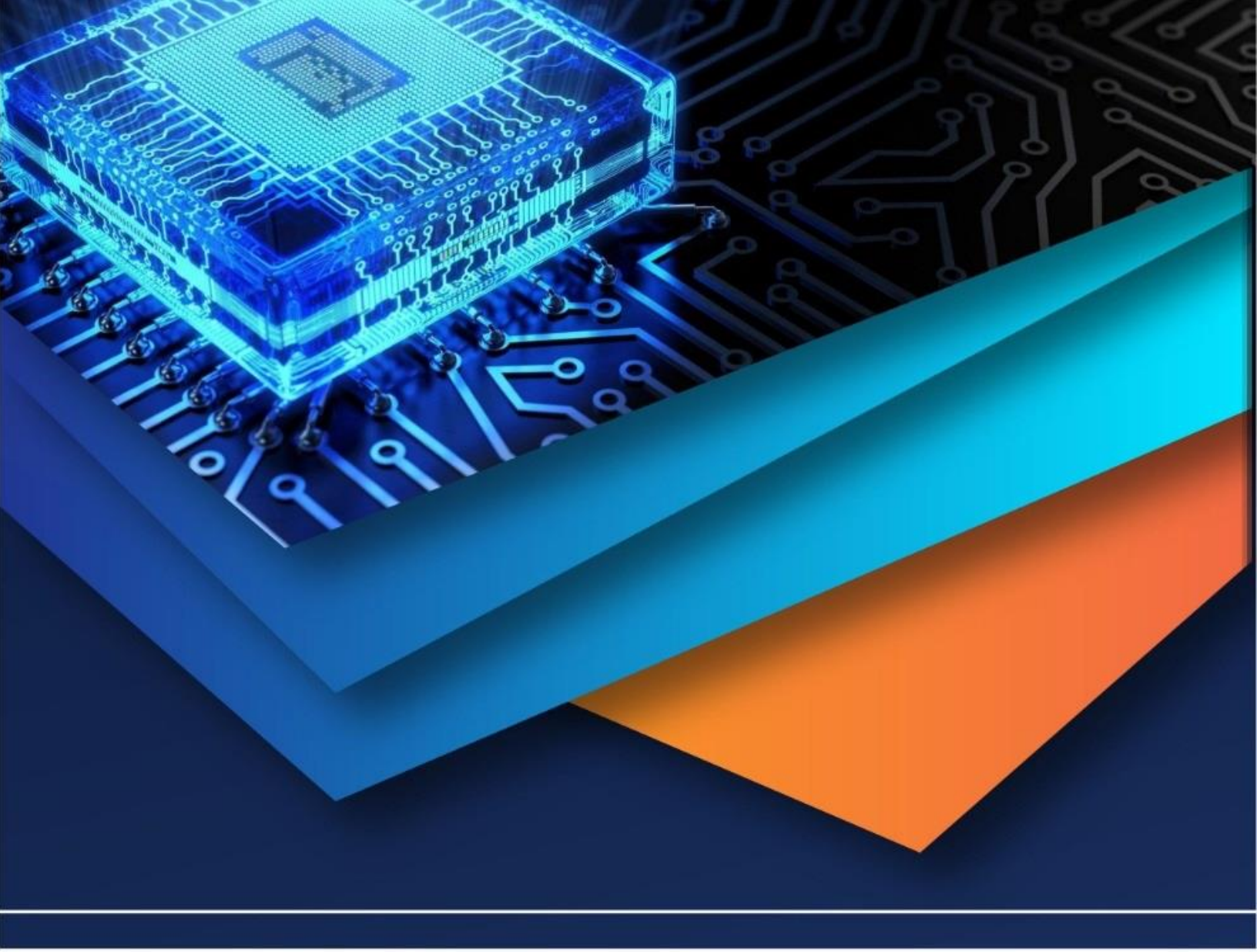
To improve recommendation effectiveness and scalability, future work will focus on:

- Hybrid Recommendation Systems: Combining collaborative filtering with content-based filtering or deep learning techniques for improved accuracy.
- Improved Similarity Metrics: Exploring cosine similarity, Pearson correlation, or neural embeddings for better recommendation performance.
- Real-time Recommendations: Optimizing model performance to provide real-time book recommendations with minimal latency.
- User Feedback Mechanisms: Allowing users to refine recommendations by explicitly liking or disliking suggested books.

REFERENCES

[1] P. Resnick and H. R. Varian, “Recommender Systems,” Commun. ACM, vol. 40, no. 3, pp. 56–58, 1997.
[2] G. Adomavicius and A. Tuzhilin, “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, 2005.
[3] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms,” in Proc. 10th Int. Conf. World Wide Web, 2001, pp. 285–295.
[4] X. He, H. Zhang, M. Y. Kan, and T. S. Chua, “Fast Matrix Factorization for Online Recommendation with Implicit Feedback,” in Proc. 39th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr., 2016, pp. 549–558.

[5] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning-Based Recommender System: A Survey and New Perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019.
[6] P. Lops, M. Gemmis, and G. Semeraro, "Content-Based Recommender Systems: State of the Art and Trends," in Recommender Systems Handbook, Springer, 2011, pp. 73–105.
[7] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in The Adaptive Web, Springer, 2007, pp. 325–341.
[8] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," Computer, vol. 42, no. 8, pp. 30–37, 2009.
[9] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," IEEE Internet Comput., vol. 7, no. 1, pp. 76–80, 2003.
[10] C. C. Aggarwal, Recommender Systems, Springer, 2016.
[11] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 2nd ed., Springer, 2015.
[12] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," User Model. User-Adapt. Interact., vol. 12, pp. 331–370, 2002.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY