iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.36025

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: 7.429
Volume 9 Issue VI June 2021- Available at www.ijraset.com

Book Recommendation System using Matrix Factorization

K. Venkata Ruchitha[1], V. Raviteja[2], P. Suresh[3], Mrs. N. Rajeswari[4]
[1, 2, 3]*Student, CSE Department & GEC, Gudlavalleru, Krishna*
[4]*Associate Professor, CSE department & GEC, Gudlavalleru, Krishna*

Abstract: ***In recent years, recommender systems became more and more common and area unit applied to a various vary of applications, thanks to development of things and its numerous varieties accessible, that leaves the users to settle on from bumper provided choices. Recommendations generally speed up searches and create it easier for users to access content that they're curious about, and conjointly surprise them with offers they'd haven't sought for. By victimisation filtering strategies for pre-processing the information, recommendations area unit provided either through collaborative filtering or through content-based Filtering. This recommender system recommends books supported the description and features. It identifies the similarity between the books supported its description. It conjointly considers the user previous history so as to advocate the identical book.***
Keywords: Recommender System, Collaborative Filtering, Book Recommendations.

I. INTRODUCTION

Recommendation systems, whether or not we tend to like them or not, are infiltrating each side of our lives. A recommender system, in easy terms, seeks to model a user's behavior relating to targeted things and/or product. That is, a recommender system leverages user information to higher perceive however they act with things. things here may well be books in a very book store.

Recommender systems operate via machine learning algorithms. Typically, these algorithms may be classified into 2 classes — content-based and collaborative filtering. Content-based ways live the similarity of item attributes and specialise in recommending alternative things almost like what the user likes, supported their previous actions or express feedback (i.e., via surveys or ratings). Although content-based ways square measure quite effective in some instances, they are doing have their drawbacks. collaborative filtering ways operate otherwise, and do their best to handle a number of the restrictions of content-based filtering. With collaborative filtering, algorithms use similarities between users and things at the same time to supply recommendations. primarily, the underlying models square measure designed to suggest associate degree item to User A supported interests of the same User B.

II. RELATED WORK

Francesco Ricci, LiorRokach and BrachaShapira define the recommender systems as software tools that make relevant suggestions to a user. Depending upon the user profile and the product profile, which are formed using various techniques and algorithms, suggestions are made.

According to Anna Gatzioura and Miquel Snchez, the aim of recommendation system is to provide effective and meaningful content (item) to the user which is active on the platform.

According to K. Shah, A.k Salunke, S. Dongare, and K. Antala, recommendation systems are a machine learning technology that comes under unsupervised learning machine learning models in which data is not labelled.

Ruiqin Wang et al, [4] discuss about Collaborative Filtering (CF) algorithms that have been widely used to provide personalized recommendations in e-commerce websites and social network applications and Among which, Matrix Factorization (MF) is one of the most popular and efficient techniques. However, most MF-based recommender models rely only on the past transaction information of users, hence there is inevitably a data sparsity problem. So they propose a novel recommender model based on matrix factorization and semantic similarity measure. Initially, a new semantic similarity measure is created based on the semantic information in the Linked Open Data (LOD) knowledge base, which is a hybrid measure that is based on feature and distancemetrics.

The paper [8] by Ms. Sushama Rajpurkar et al put forward a book recommendation system based on the combined features of content filtering, collaborative filtering and association rule mining respectively. This book recommendation has considered many parameters such as content of the book and quality of the book, by applying collaborative filtering. This recommender system also uses associative model to give stronger recommendations. This system does not have performance problem since it built the recommendations offline.

III. METHODOLOGY

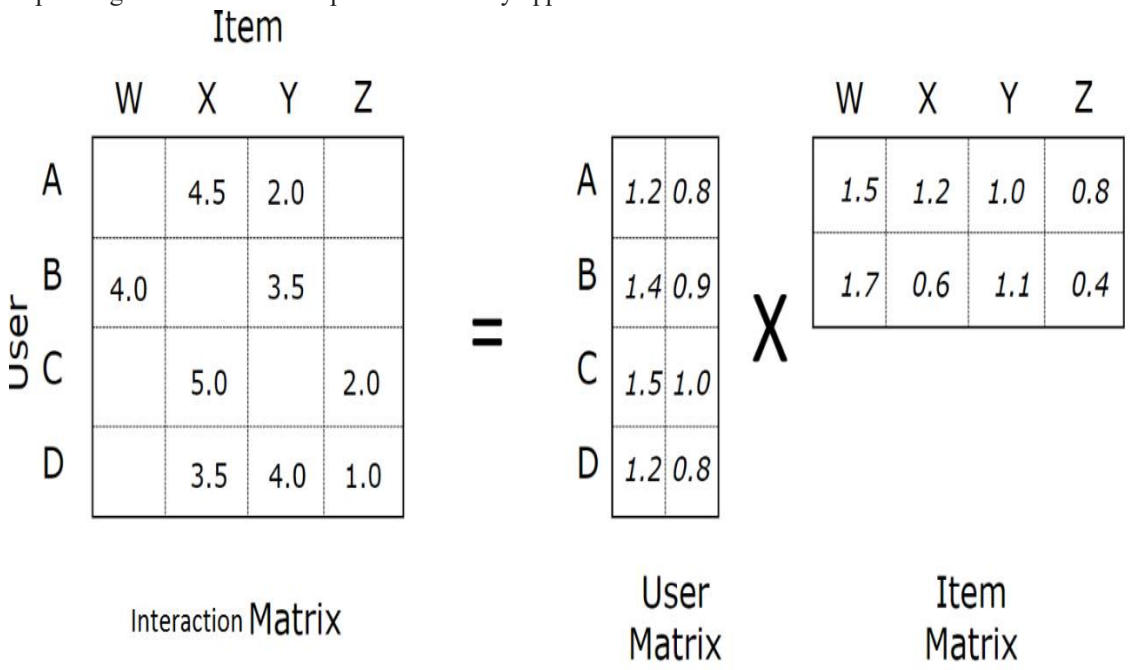
For developing the system, certain tools are used. Our system utilizing Flask, PostgreSQL, Databricks and PySpark distributive computing capabilities. The methodology used for our project is matrix factorization under collaborative filtering.

A. Collaborative Filtering

The usual technique analyzes the character of every item. In our case, recommending books to a user by playing language process on the content of every book. Collaborative Filtering, on the opposite hand, doesn't need any info regarding the things or the users themselves. It recommends books supported users past behaviour. Among the varied forms of cooperative filtering techniques, this technique uses model-based method. AN economical model-based CF technique is matrix factorisation.

1) Matrix Factorization

a) Collaborative filtering may be achieved in myriad ways that, however, one common methodology is to use matrix resolution and spatiality reduction techniques. Algorithms like Alternating method least squares (ALS), Singular Value Decomposition (SVD), and Stochastic Gradient Decent (SGD) try this quite well, and whereas operating with massive volumes of information (i.e., 1M+ knowledge points), these algorithms in conjunction with distributive computing and data processing square measure sensible operating solutions to create production-ready applications with massive volumes of information.



b) Here, we used ALS(Altering Least Square) alogorithm for book recommendations to usrs based on their ratings.

2) ALS (Altering Least Square): Alternating least square method is an algorithm to factorize a matrix. We can see how collaborative filtering uses ALS algorithm. As in the below figure, we see that a matrix being factorized into 2 smaller matrices. Consider the first matrix as the set of user-item interaction. Thus, the factorized matrices would be user features and item features.

a) A new user inputs their needed books, then system creates new user book interaction samples for model.
b) System retains ALS model on data with the user ratings.
c) System make rating predictions on all books for that user.
d) System outputs top N book recommendations for that user based on the ranking of book rating predictions.

Once we implemented the ALS recommender system in a python script as a small **PySpark** program, we can submit our spark application to a cluster with Client Deploy Mode or Cluster Deploy Mode and enjoy the power of distributed computing.

B. Implementation

The steps involved to build this model:

Flow Diagram

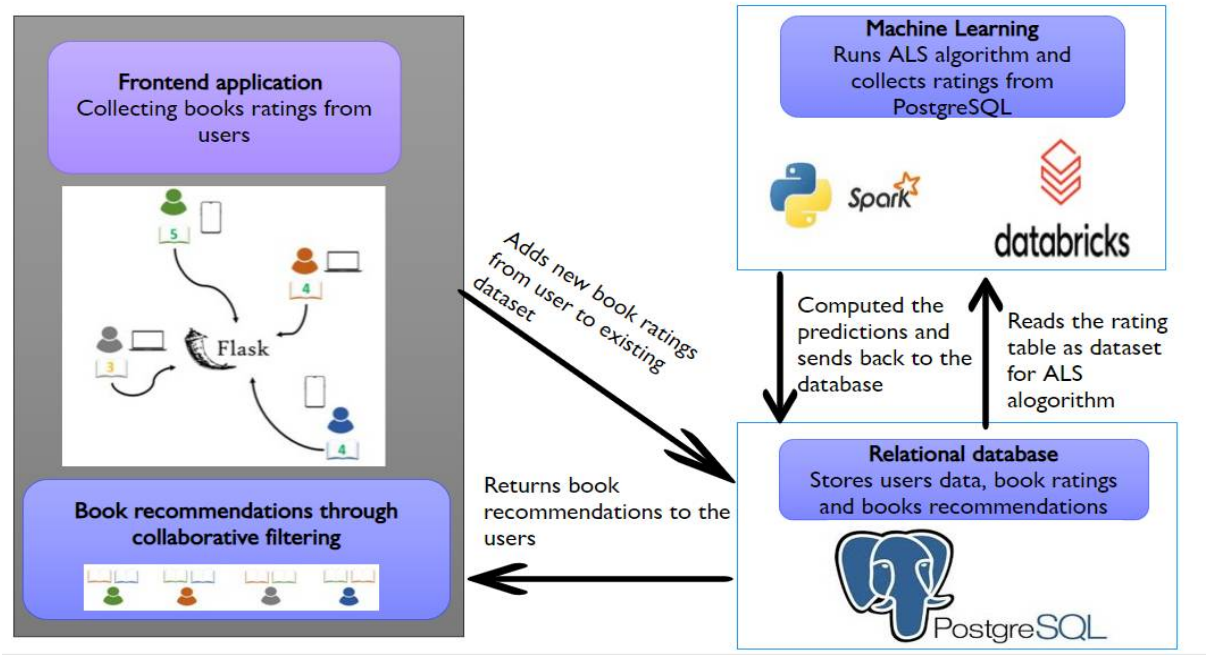


Fig 3.2: Flow diagram of our system

1) *Identifying and Collecting Dataset for ML Computing:* As a first step, it was imperative to find a large user-book dataset that would work well with my choice of algorithms and would provide reliable predictions and recommendations to users. Therefore, we did some exploratory analysis and decided to use the **goodbooks-10k dataset.** According to Kaggle, this dataset contains close to 1 million ratings across 10,000 different books located on a popular book platform, **Goodreads**. The dataset helped to formulate initial recommendations to users, and over time, will include additional books and ratings as more people use the application. Therefore, we initially downloaded the data and performed some exploratory analysis to get a better sense of its components. The Kaggle dataset contains a few different files, some of which we will use for this project, and others we can hope to utilize in later iterations of the recommendation model.
2) *Building And Deploying Full Production Flask App In Python:* After doing some exploratory data analysis, we would like to build out a simple python flask application. The functionality of flask application is :

a) Allows a user to register/sign-in under a specific username
b) Loads a profile page that displays a history of a user's rated books and provides navigation options to either rate more books or view recommended books
c) Has a search feature that interacts with the Goodreads API to search for a book by title name. The user can then rate that book on a scale of 1-5.
d) A user can also find their book recommendations in a separate page

To set up the application, we first created a basic Flask application, created my routes and page templates, and connected to a local PostgreSQL database. Next, after testing to make sure that we are able to properly register/sign-in each user, obtain specific userids to link back to particular book ratings, then integrated the Goodreads API into the application to help users search for books that they'd like to rate. By ensuring that any rating that was generated was linked back to the correct bookid and attributed to the correct user, we did a fair bit of testing during this stage. After this, we focused in on creating my recommender engine in PySpark. By using Databrick's built-in interface, this was quite simple. We used ALS to serve recommendations to users. Once the model was fine-tuned, We are able to connect the new recommendations table that we created in remote database to Flask application and render the information for each recommended book utilizing the same Goodreads API used earlier.

3) *Building Recommender Engine in Databricks:* With the application functionality set up and deployed, then used Databricks and Pyspark to build out an Alternating Least Squares (ALS) algorithm that absorbs my ratings dataset from my PostgreSQL database, computes the updated predictions based on matrix factorization techniques established in Databricks, and then recommends books to each user based on the highest predicted scores per user. With our Spark dataframe loaded into Databricks, we then can start to set up our data to run our ALS model. First, we can split the data frame into a training, validation and testing dataset -- 60% training, 20% validation and 20% testing. The validation dataset will be used to test the fine-tuning of our model parameters, and will allow us to have a hold-out test set that we can use to compute the RMSE on our final, optimized model. After caching the training and test datasets for later, which hopefully will cut down on some processing time, we imported the ALS, evaluation, and tuning functionality from the pyspark.ml package. For a base ALS model, we'll set the maximum iterations at 5, our initial regularization parameter at 0.25, and assign our user, book, and rating components to their corresponding column names to set up our matrix. The ALS() function conveniently sets up our matrix. The user ratings from the application were then incorporated back into the robust goodbooks dataset and eventually into the ALS algorithm. By setting up a PostgreSQL database, we are able to match new user ids, their ratings and the book ids to this original dataset. Therefore, whenever a new rating comes in from a user, it is appended to the ratings dataset which we used later for our recommender engine.

IV. RESULTS

Altering Least Square algorithm in matrix factorization method is used to build the model for book recommendation system. In this application The dataset contained a total of 53,425 users that have supplied ratings for at least 10 books**.** The average book rating was about 3**.**85, with 4 being the most common rating on a scale of 1 to 5. Therefore, most ratings were quite positive. As expected, the dataset was very sparse **(**~99.82% of the ratings file was blank**).** Alternating Least Squares (ALS) algorithm that absorbs my ratings dataset from my PostgreSQL database, computes the updated predictions based on matrix factorization technique in collaborative filtering established in Databricks, and then recommends books to each user based on the highest predicted scores per user.

let's take a look at some of the most highly rated books. This may provide some insight into how our recommendations will be served later on:

	Top 20 Highest Rated Books
0	ESV Study Bible
1	The Indispensable Calvin and Hobbes
2	The Days Are Just Packed: A Calvin and Hobbes ...
3	Attack of the Deranged Mutant Killer Monster S...
4	The Divan
5	There's Treasure Everywhere: A Calvin and Hobb...
6	Harry Potter Boxed Set, Books 1-5 (Harry Potte...
7	The Calvin and Hobbes Lazy Sunday Book
8	The Authoritative Calvin and Hobbes: A Calvin ...
9	It's a Magical World: A Calvin and Hobbes Coll...
10	A Court of Mist and Fury (A Court of Thorns an...
11	The Revenge of the Baby-Sat
12	Harry Potter Collection (Harry Potter, #1-6)
13	The Absolute Sandman, Volume One
14	The Calvin and Hobbes Tenth Anniversary Book
15	Preach My Gospel: A Guide To Missionary Service
16	Locke & Key, Vol. 6: Alpha & Omega
17	The Way of Kings, Part 1 (The Stormlight Archi...
18	Saga, Vol. 2 (Saga, #2)

Some of the most highly rated books in the dataset were household favorites, including parts of the Calvin and Hobbes collection**,** Harry Potter**,** and a few books with religious principles**.** Recommender system using Pyspark's collaborative filtering and an Alternating Least Squares algorithm to find similarities between users. Below table shows that new recommended books for the users.

userid	book_id	prediction
53429	10026	3.7028527
53427	10034	4.4262066
53426	6590	5.2477336
53426	5580	5.1231766
53426	6361	5.1833167
53428	5580	4.003819
53427	10036	4.4262066
53428	6361	4.0357094
53426	6920	5.243292
53428	10036	4.064948
53428	6920	4.0739856
53429	10020	4.065512
53428	10034	4.064948
53427	10020	4.4985204
53426	7254	5.21943
53426	9566	5.3054423

In the end, able to create a full production application that serves up user-specific book recommendations based on user ratings of past books. Given that the overall approach was to use collaborative filtering to serve up recommendations. This demonstrates that the collaborative filtering and ALS calculations are working effectively, and the recommender system is functioning as expected.

V. CONCLUSION

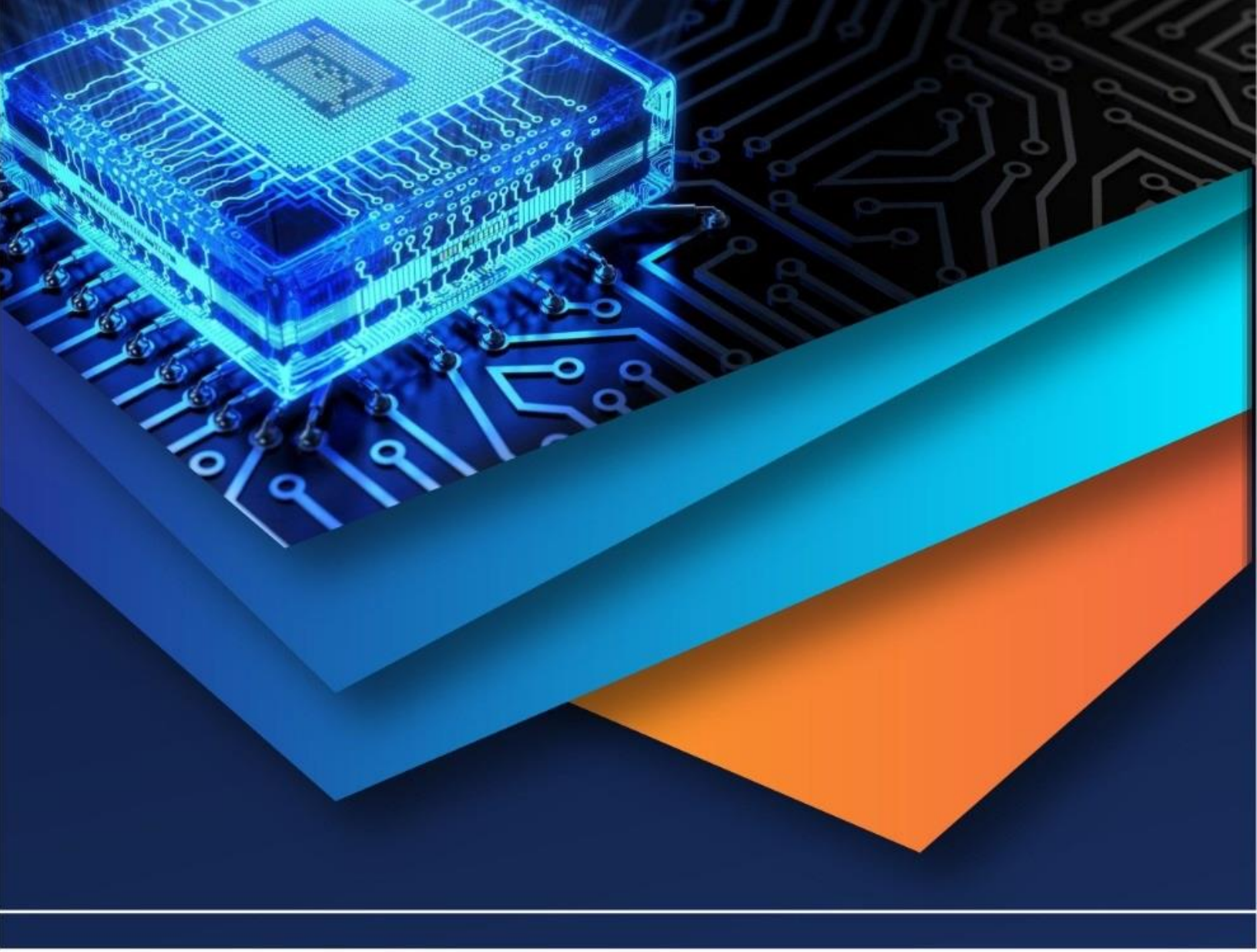
Since the data is huge and complex it is difficult to get useful information from it. Recommender System are effective software techniques to overcome this problem. Based on the user's and books information available, these techniques provides recommendations to users in their area of interest. In our project, we are going to develop a machine learning model by using matrix factorization. Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. It automatically recommends books to a user by taking similarity of books. This recommender system recommends books based on the description or features. It identifies the similarity between the books based on its description. It also considers the user previous history and ratings given by them in order to recommend a similar book. After rating a few books, the user can then see their individualized recommendations on a separate page.

VI. FUTURE WORK

As we are dealing with the book recommendation system using matrix factorization, need to consider every possible parameter that changes the recommendations for an user. Since this matrix factorization uses ALS algorithm , each user should signup properly for matching user IDs and books ratings given by the users to recommend new books. Even though here no constraint for users like limited number of users ,if number of users are increased it might not give the results quickly. So, there is need to use cloud platform instead of local databases and servers to store the data and produce fast results. Adding online books buying feature for recommended books to the users will make our application more effective with high robustness.

REFERENCES

[1] M, I. (2020). Predicting Books' Overall Rating Using Artificial Neural Network. International Research Journal of Engineering and Technology .
[2] Shah, K. (2019). Book Recommendation System using Item based Collaborative Filtering .International Research Journal of Engineering and Technology .
[3] Parvatikar, S. (2015). Online Book Recommendation System by using Collaborative filtering and Association Mining . IEEE.
[4] Gao, Y. (2019). Research on Book Personalized Recommendation Method Based on Collaborative Filtering Algorithm. IEEE.
[5] Dara, S. (2019). A survey on group recommender systems. Journal of Intelligent Information Systems.
[6] Tan, K. W. (2019). A New Collaborative Filtering Recommendation Approach Based on Naive Bayesian Method. Beijing: ResearchGate.
[7] Ramakrishnan, G. (2020). Collaborative filtering for book recommendation system. SocProS.
[8] Sohail, S. S. (2013). Book Recommendation System Using Opinion Mining Technique . IEEE.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY