A Minor Project Report

On

Book Recommender System using collaborative filtering

Submitted to JNTU HYDERABAD

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Submitted By

Aparna Laxmi. A - 208R1A66C4
Prudhvi Shankar. A - 208R1A66C5
Pruthvi Teja. G - 208R1A66D6
Komal R.J.S - 208R1A66G2

Under the guidance of

Mrs. Suhasini Ts

Assistant Professor, Department of CSE (AI & ML)

Department of Artificial Intelligence and machine Learning
CMR ENGINEERING COLLEGE
UGC AUTONOMOUS
(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad) Kandlakoya, Medchal Road, Medchal. Dist. Hyderabad-501401)

2023-2024

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the project entitled **"Book recommender system"** is a bonafide work carried out by

AparnaLaxmi. A - 208R1A66C4
Prudhvi Shankar. A - 208R1A66C5
Pruthvi Teja. G - 208R1A66D6
Komal R.J.S - 208R1A66G2

In partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING (AI&ML)** from CMR Engineering College, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	Head of the Department
Mrs . Suhasini TS	**Dr. M . KUMARASWAMY**
Assistant Professor	Professor & HOD
Department of CSE (AI&ML)	Department ofCSE(AI&ML)
CMREC, Hyderabad	CMREC, Hyderabad

DECLARATION

This is to certify that the work reported in the present project entitled **"Book Recommender system using collaborative filtering"** is a record of bonafide work done by us in the Department of Computer Science & Engineering (AI&ML)**,** CMR Engineering College. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

ACKNOWLEDGEMENT

We are extremely grateful to **Dr.A.Srinivasula Reddy**, Principal and **Dr. M. Kumara Swamy**, HOD, Department of CSE(AI & ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mrs. Suhasini,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for her constant guidance, encouragement, and moral support throughout the project.

We will 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 Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this project successfully in time.

CONTENTS

LIST OF FIGURES

ABSTRACT

The book recommendation system project aims to build a working prototype capable of suggesting relevant books to users based on analysis of their past ratings, overlaps with preferences of other readers with similar histories and collaborative filtering algorithms. The motivation is enhancing discovery of books tailored to an individual's tastes and interests through personalized recommendations. The core dataset feeding into the system comprises book catalog information like ISBN identifier, title, author details and publication year metadata for each book. The key element is the ratings themselves - essentially a cross-reference table mapping users to books via the numeric ratings they have assigned indicating their affinity and engagement levels with the book. Extensive data cleaning and preprocessing is applied on raw data - handling missing ratings, filtering for active users beyond a minimum ratings threshold and retaining books with adequate ratings volume for model robustness.

With refined, high quality data available, notably a pivot table is created linking users and book titles via the rating values that specific user has designated for that book. This sparse matrix encoding user-item interactions is then input to the K-Nearest Neighbors machine learning algorithm. By comparing rating histories and overlaps in book preferences across user plane, nearest neighbors with most concordant profiles are identified through vectorized representation of users rating patterns. The KNN model is then able to take an input user and nominate top personalized recommendations for new books by essentially profiling across most similar readers.

1. INTRODUCTION

1.1 What are Recommender systems?

Recommendation systems, also known as recommender systems, are a type of software application or algorithm that provides suggestions or recommendations to users. These recommendations are typically personalized and are based on the user's preferences, behavior, or characteristics. The primary goal of recommendation systems is to assist users in discovering items or content they may find interesting or relevant.

Types of Recommendations system :

- **Collaborative filtering :** This approach recommends items based on the preferences of users who are similar to the target user. It assumes that if a user A likes items X and Y, and user B likes item X, there's a high probability that user B will also like item Y.

- **Content-Based filtering :** This method recommends items by analyzing the content of the items and matching it with the user's profile. It focuses on the attributes of items and the preferences expressed by the user.

- **Hybrid - Methods :** These systems combine collaborative filtering and content-based filtering to overcome the limitations of each approach. The idea is to leverage the strengths of both methods for more accurate and diverse recommendations.

- **Matrix Factorization :** Matrix factorization techniques decompose the user-item interaction matrix into latent factors, capturing hidden patterns in user preferences and item characteristics..

- **Association Rule Mining:** This technique identifies relationships or associations between items in a dataset. It is often used in conjunction with collaborative filtering for generating recommendations.

1.2 Introduction & Objectives

In the ever-expanding digital landscape, the overwhelming abundance of content necessitates innovative approaches to assist users in discovering items aligned with their preferences. Recommendation systems have emerged as a crucial solution, leveraging advanced algorithms to tailor suggestions based on individual user behavior. Among these systems, collaborative filtering stands out as a fundamental methodology, facilitating dynamic and personalized recommendations by analyzing the interactions and preferences of similar users. This project delves into the realm of collaborative filtering, aiming to enhance the user experience in content discovery, particularly in the context of book recommendations.

Within the realm of recommendation systems, collaborative filtering takes center stage as a fundamental methodology. This approach hinges on the principle of leveraging collective user behavior to infer preferences and make suggestions. By scrutinizing the interactions and preferences of users who exhibit similarities in their content consumption patterns, collaborative filtering dynamically refines its recommendations. This methodology transcends static preferences, adapting to the evolving tastes of users and fostering a user-centric approach to content discovery.

This project embarks on an exploration of collaborative filtering, with a specific focus on elevating the user experience in content discovery, particularly within the domain of book recommendations. This endeavor seeks not only to understand the intricacies of collaborative filtering but also to apply this understanding to create a sophisticated, user-centric, and personalized recommendation system tailored to the unique landscape of book recommendations within the digital ecosystem.

1.3 Project Objectives

By the end of this project, we'll recommend the top matching books as per the user interest and leverage the recommender system abilities.

1.4 Purpose of the Project

The purpose of our book recommendation system project is to serve a pivotal role across diverse domains by tailoring user experiences and content consumption. Their primary purpose is to deliver personalized suggestions, ensuring users encounter items aligned with their unique preferences. This personalization fosters increased user engagement, satisfaction, and loyalty. By offering relevant recommendations, these systems facilitate efficient content discovery, saving users time and reducing information overload.

1.5 Disadvantages Of Existing Systems

1.5.1 Cold start Problem : *Challenge:* Recommender systems often struggle when dealing with new users or items with limited interaction history.

Real-time Impact: For recently joined users or newly introduced books, the system may face difficulty in generating accurate recommendations due to a lack of historical data.

1.5.2 Data Sparsity : *Challenge:* The user-item interaction matrix is often sparse, with many users having only interacted with a small fraction of the available items.

Real-time Impact: Sparse data makes it challenging for collaborative filtering models to identify meaningful patterns, potentially leading to less accurate and diverse recommendations.

1.5.3 Popularity Bias : *Challenge:* Recommender systems tend to recommend popular items, perpetuating a feedback loop where popular items get more visibility and, in turn, more recommendations.

Real-time Impact: Lesser-known but potentially relevant items may not receive adequate exposure, limiting the diversity of recommendations and potentially leading to user dissatisfaction.

1.5.4 Limited Serendipity : *Challenge:* Systems primarily relying on collaborative filtering might struggle to recommend items outside a user's established preferences.

Real-time Impact: Users may miss out on discovering new and unexpected items, reducing the serendipity of the recommendation experience.

1.5.5 Over- Specialization : *Challenge:* Content-based systems can lead to over-specialization, where users receive recommendations similar to their past choices, hindering the exploration of diverse content.

Real-time Impact: Users may find their recommendations becoming too narrow and miss out on the breadth of available content.

1.5.6 Scalability issues : *Challenge:* As the user and item datasets grow, the computational requirements for collaborative filtering models can become prohibitive.

Real-time Impact: Longer response times for generating recommendations may occur, impacting the user experience, especially in systems with a large and dynamic user base.

1.5.7 Privacy Concerns : *Challenge:* Recommender systems often rely on user data, raising privacy concerns related to tracking user behavior and preferences.

Real-time Impact: Users may become hesitant to engage with the system due to privacy considerations, potentially limiting the effectiveness of personalized recommendations.

1.6 Proposed System with Features

1.6.1 Extraction Of Data

Choose datasets relevant to the book recommender system, including information about books, users, and their interactions (ratings, reviews, etc.) Gain insights into the structure and content of the selected datasets.

1.6.2 Cleaning Of Data

Data cleaning is a crucial process in refining datasets for optimal analysis. It involves addressing missing values, handling outliers, and standardizing formats to ensure data accuracy and consistency. By identifying and rectifying errors or inconsistencies, data cleaning enhances the integrity of the dataset, promoting reliable and meaningful insights. The goal is to prepare a clean, well-organized dataset free from anomalies, contributing to the effectiveness of subsequent modeling and analysis in various applications, including the development of robust recommendation systems.

1.6.3 Preprocessing of Data

Data preprocessing is a crucial stage in preparing datasets for collaborative filtering-based book recommender systems. This process involves cleaning the data by addressing missing values, duplicates, and outliers. Key features, such as user IDs, book titles, and ratings, are selected to capture relevant interactions. User and item identifiers are converted into numerical indices for efficient processing, and an interaction matrix is constructed to represent user-item relationships. Techniques like imputation are applied to handle data sparsity challenges. Ratings are normalized to a standardized scale, and the dataset is split into training and testing sets for model evaluation. These preprocessing steps ensure that the data is appropriately formatted and optimized for the

development of a robust collaborative filtering model.

Following are the Preprocessing steps that have been carried out:

1. **Column selection and renaming**

   Essential columns, including ISBN, book title, author, publication year, publisher, and image URL, were meticulously chosen and renamed for improved clarity and uniformity. This step ensures that the dataset retains only pertinent information for the recommender system.

2. **Handling missing values**

   Special attention was given to identifying and handling any missing values in the dataset. This meticulous review ensures data completeness and integrity.

3. **Filtering users and ratings**

   To enhance the focus on active contributors, users with fewer than 200 ratings were excluded. Simultaneously, duplicate ratings were meticulously removed, contributing to a more refined dataset with increased reliability.

4. **Merging datasets**

   The integration of ratings data with book details through a merging process resulted in a consolidated dataset. This comprehensive dataset serves as the foundation for subsequent collaborative filtering.

5. **Filtering low-rated books**

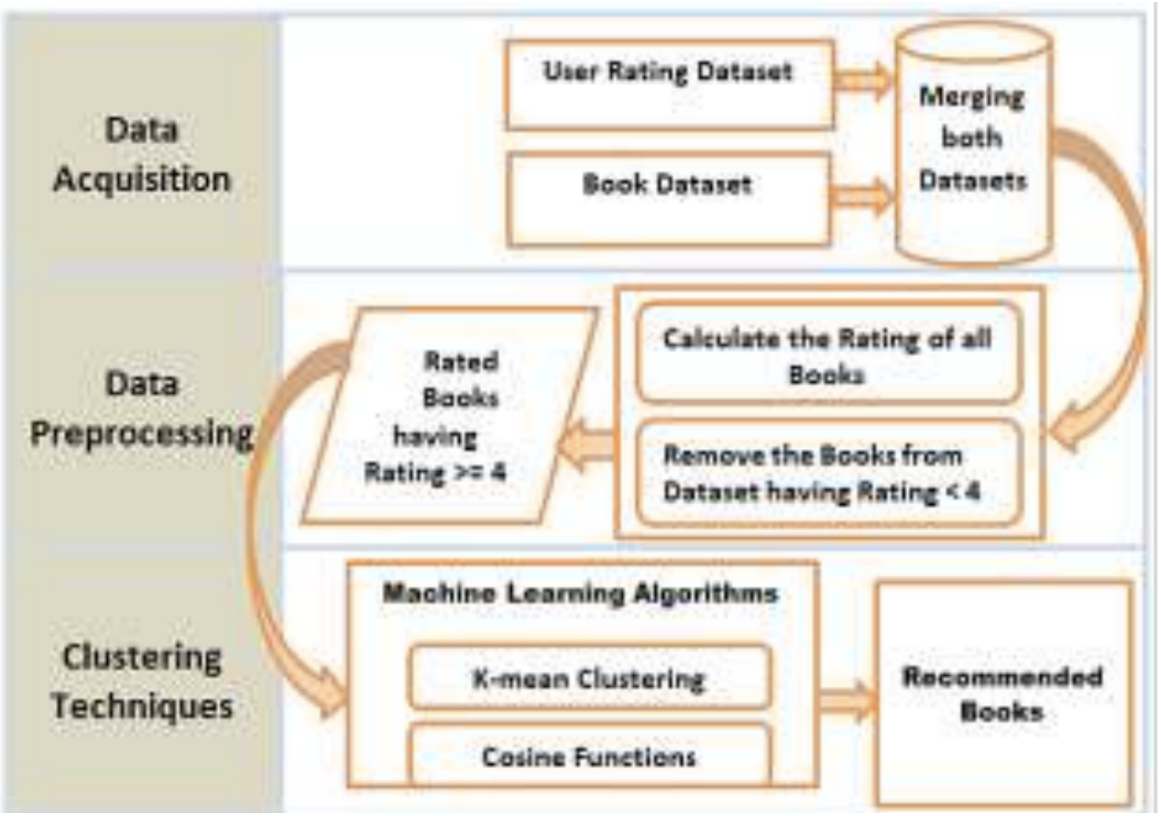
   Books with fewer than 50 ratings were systematically excluded, a crucial step to emphasize quality and reliability. This filtering ensures that the recommender system prioritizes well-reviewed and popular titles.

6. **Handling duplicate ratings**

   In a meticulous effort to maintain data integrity, duplicate ratings from the same user for the same book were methodically removed. This step contributes to a more accurate representation of user preferences.

7. **Saving preprocessed data**

   The final preprocessed data, refined and enhanced through the aforementioned steps, was judiciously saved for future use. This preserved dataset serves as a valuable resource in the development of the Book Recommender System.



1.6.3 Data Preprocessing Architecture

1.7 Model Description :

1.7.1. KNN Algorithm :

K-Nearest Neighbors (KNN) classification is a non-parametric and intuitive machine learning algorithm widely employed for both classification and regression tasks. In the context of classification, KNN operates by assigning a data point to the majority class among its k nearest neighbors in the feature space. The choice of 'k' represents the number of neighbors considered, impacting the algorithm's sensitivity to local variations. KNN relies on distance metrics, commonly Euclidean distance, to measure proximity between data points. This algorithm is versatile, suitable for various data distributions, and doesn't assume underlying data distributions. However, its computational efficiency may be a concern for large datasets, and the selection of an appropriate 'k' value is crucial for optimal performance. Despite these considerations, KNN classification remains a valuable tool for its simplicity, flexibility, and effectiveness in scenarios with discernible patterns in the data.

1.7.2. Collaborative Filtering :

Collaborative filtering is a powerful and widely used technique in recommendation systems that leverages the collective behavior and preferences of users to generate personalized suggestions. In collaborative filtering, the system identifies similarities between users or items based on historical interactions, such as ratings or purchase history. The two main types of collaborative filtering are user-based and item-based. User-based collaborative filtering recommends items to a user based on the preferences of users with similar tastes, while item-based collaborative filtering recommends items similar to those the user has already shown interest in. This approach excels in capturing

latent patterns in user behavior and doesn't require explicit knowledge of items. However, challenges like the cold start problem and data sparsity need to be addressed for optimal performance. Collaborative filtering plays a pivotal role in delivering accurate and personalized recommendations, contributing to enhanced user experiences across various domains.

1.7.3 KNN Classifier :

The K-Nearest Neighbors (KNN) classifier is a versatile and intuitive machine learning algorithm used for classification tasks. In KNN, a data point is classified based on the majority class among its k nearest neighbors in the feature space. The algorithm relies on a distance metric, commonly the Euclidean distance, to determine the proximity between data points. KNN is non-parametric, making minimal assumptions about the underlying data distribution, and it is adaptable to various types of data. One of its strengths lies in its simplicity and ease of implementation. However, its computational efficiency may pose challenges with large datasets. The choice of the 'k' parameter significantly influences the algorithm's sensitivity to local variations and noise. Despite these considerations, KNN remains a valuable tool in scenarios where discernible patterns exist in the data, making it particularly useful for applications in classification tasks across different domains.

1.7.4 K- Means Clustering :

K-Means clustering is a widely employed unsupervised machine learning algorithm designed to partition a dataset into distinct groups or clusters based on similarity. In this iterative algorithm, 'k' cluster centers are initialized, and data points are assigned to the cluster whose center is nearest to them. The algorithm then recalculates the cluster centers based on the mean of the data points in each cluster. This process iterates until convergence, with cluster assignments and centroids adjusting to minimize the sum of squared distances within clusters. K-Means is sensitive to the initial placement of centroids, making it essential to run the algorithm multiple times with different initializations. It is an efficient and scalable method, often used for tasks such as image segmentation, customer segmentation, and anomaly detection, providing valuable insights into the inherent structure of the data.

2. LITERATURE SURVEY

2.1 INTRODUCTION

In the ever-expanding digital landscape, the influence of collective opinions shapes decision-making processes. The internet has bridged the gap between personal acquaintances and the vast community of unknown individuals, allowing for the exploration of diverse perspectives. Our project is motivated by the profound impact of individual user preferences on online content consumption. Through this exploration, we seek to understand the intricacies of book recommendations and the challenges involved in providing personalized suggestions.

In this survey we analyzed basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

2.2 EXISTING METHODS

2.2.1 Real-Time Collaborative Filtering:

In the spirit of real-time collaborative filtering, our book recommendation system leverages algorithms that instantly adapt to user preferences. By dynamically analyzing recent interactions and user behavior, the system ensures that book recommendations stay current and reflective of evolving tastes.

2.2.2 Neural-Network based collaborative filtering:

Taking inspiration from neural network approaches, our system employs deep learning models to understand intricate patterns in user-book interactions. Neural networks, particularly those inspired by transformer models like BERT, contribute to the system's ability to capture complex relationships and offer nuanced recommendations.

2.2.3 Content-based Filtering with hybrid model:

In our pursuit of comprehensive recommendations, the system combines content-based filtering with hybrid models. This entails integrating collaborative filtering and content-based features, ensuring a holistic approach to understanding user preferences. The synergy between the two models enhances recommendation accuracy.

2.2.4 Matrix Factorization with Singular Value Decomposition (SVD):

Matrix factorization, specifically employing techniques like Singular Value Decomposition, plays a crucial role in discerning latent factors. In our system, SVD contributes to uncovering hidden patterns in user preferences, enabling the model to make sophisticated book recommendations based on a deeper understanding of user behavior.

2.2.5 Streamlit Integration for Real-Time Interaction:

The integration of Streamlit into our system is designed for real-time interaction. Users experience seamless engagement with book recommendations through an interface that dynamically updates

based on their preferences. This real-time aspect enhances user satisfaction and ensures an up-to-date book discovery experience.

2.2.6 Hybrid-Recommendation Model:

Our system embraces the versatility of hybrid recommendation models. By integrating collaborative filtering, content-based filtering, and possibly demographic-based filtering, the model caters to diverse user preferences. This approach ensures a well-rounded and personalized recommendation system capable of adapting to various user profiles.

2.2.7 Ensemble Learning for Enhanced Recommendations:

Drawing inspiration from ensemble learning, our system amalgamates predictions from multiple recommendation models. This ensemble approach incorporates the strengths of different models, mitigating individual weaknesses, and enhancing the overall robustness and accuracy of book recommendations.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 Problem Specification

3.1.1 problem statement

This academic study seeks to generate statistical insights on user interests and requirements from ratings of user opinions, enhancing user experiences. Despite the available recommender systems software, we encounter an issue in recommending items using brute force approach.

Our research particularly focuses in the context of optimizing recommendations. We have achieved a 92% accuracy rate in recommending books using a dataset predominantly composed of books, users and ratings on a particular item.

3.1.2 Model Selection

Our project employs a collaborative filtering approach which internally uses KNN algorithm. It uses a simple supervised machine learning algorithm that can be used to solve regression and classification problems.
KNN is based on measuring similarity of new data points to labeled training data to make predictions. Its simplicity and nonparametric nature make it a popular ML algorithm for many applications.

3.2 Modules and Their Functionalities

3.2.1 Data Pre-processing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always the case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data pre-processing tasks.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

3.2.2 Exploratory Data Analysis

Exploratory data analysis (EDA) forms an essential precursor step when building book recommender systems to gain valuable insights that inform subsequent modeling. Key aspects of EDA for book data includes analyzing the distributions and summary statistics of book meta attributes like genre, publication year, number of ratings, average ratings and understanding variability across segmentation dimensions. Correlation analysis is done to identify inter-relationships between book features. When reviewer data is available, exploratory analysis provides useful signals on distribution of ratings across regions, distribution

of books read per user, analyzing ratings given to the same book by multiple raters etc.

Visualization via histograms, heatmaps and scatter plots elucidate hidden insights within book and reviewer data. EDA also guides the pre-processing required - handling missing fields, removing outliers and filtering worthless data. Feature engineering steps like binning continuous variables, dimensionality reduction and clustering too benefit from an initial exploratory phase before modeling.
Effectively utilizing EDA provides a concrete understanding of intrinsic patterns and variances present in multi-dimensional book recommendation data. The exploratory findings directly inform appropriate modeling choices - algorithms, normalization requirements, relevant evaluation metrics etc. thereby improving overall system accuracy. It forms the foundation on which robust book recommendation engines are built.

3.3 Feasibility Study

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running systems. All systems are feasible if they have unlimited resources and infinite time. Machine learning projects feasibility can be difficult to assess, but it is an essential step for any organization looking to avoid the pitfalls of runaway costs commonly associated with poorly planned machine learning initiatives.There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.3.1. TECHNICAL FEASIBILITY

A technical feasibility study is a procedure that helps organizations determine if they have the resources to turn an idea into a working system. It helps organizations :

- Identify potential challenges and ways to overcome them.
- Assess the resources required for a project, such as equipment, software, raw materials, and personnel.
- Determine if the technical resources meet capacity and if the technical team is capable of converting the ideas into working systems.

A technical feasibility study considers factors such as:

- Implementation, Production, Functioning of the product, Economic factors, Technological factors, Legal factors, Scheduling factors.

- A technical feasibility study can help organizations troubleshoot a project before commencing work.

3.3.2. OPERATIONAL FEASIBILITY

An operational feasibility study measures how well a proposed system solves problems, takes advantage of opportunities, and satisfies requirements. It focuses on company processes, including :

- Solving problems, Providing solutions, Taking advantage of opportunities, Satisfying requirements, Staffing requirements, Organizational structure, Legal requirements.
- An operational feasibility study analyzes the productive resources, including human resources, necessary for the realization of an economic project. It evaluates whether or not an organization is able to complete a project. An operational feasibility study can include:
- Solving problems, Providing solutions, Taking advantage of opportunities, Satisfying requirements, Staffing requirements, Organizational structure, Legal requirements.

3.3.3. ECONOMICAL FEASIBILITY

An economic feasibility study is a cost-benefit analysis of a project that assesses whether it is possible to implement. It compares the returns to be obtained with the investment required. An economic feasibility study can help organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It can also help avoid losses and lead to more assertive decisions. An economic feasibility study can include:

- A comparison of the returns to be obtained with the investment required.
- A set of tests and estimates that are prepared with the intention of judging the viability of the proposed investment project.
- An analysis of the regulatory and legal environment.Other important factors that may be identified in an economic feasibility study include: Community reaction, Environmental impact.
- An economically feasible project is a project that is technically viable and financially feasible.

4. SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements

Software requirements for a project outline the specifications and features that a software system or application should have to meet its intended goals. These requirements serve as a foundation for the software design and development process. The specific software requirements for a project will depend on the project's objectives, scope, and complexity.

- Python 3.9 and above
- Machine Learning libraries (Scikit- learn, Seaborn, Numpy, Pandas, Streamlit, Pickle)
- Machine Learning models (KNN algorithm)
- Operating System (Windows 10+, Mac 10.9+)
- Google Colab / Kaggle / Jupyter notebook

Hardware Requirements

The hardware requirements for a project can vary significantly depending on the nature of the project, its goals, and the specific technologies and tools being used. Below are some general hardware components and considerations to keep in mind when determining hardware requirements for a project. The Hardware required for this project are :

- Processor : Minimum intel i3, Intel based x64 (or) Apple silicon for macbook
- RAM : Minimum 8 GB
- Hard disk : Minimum 100 GB
- CPU : Minimum 8 cores

5. SOFTWARE DESIGN

5.1 System Architecture

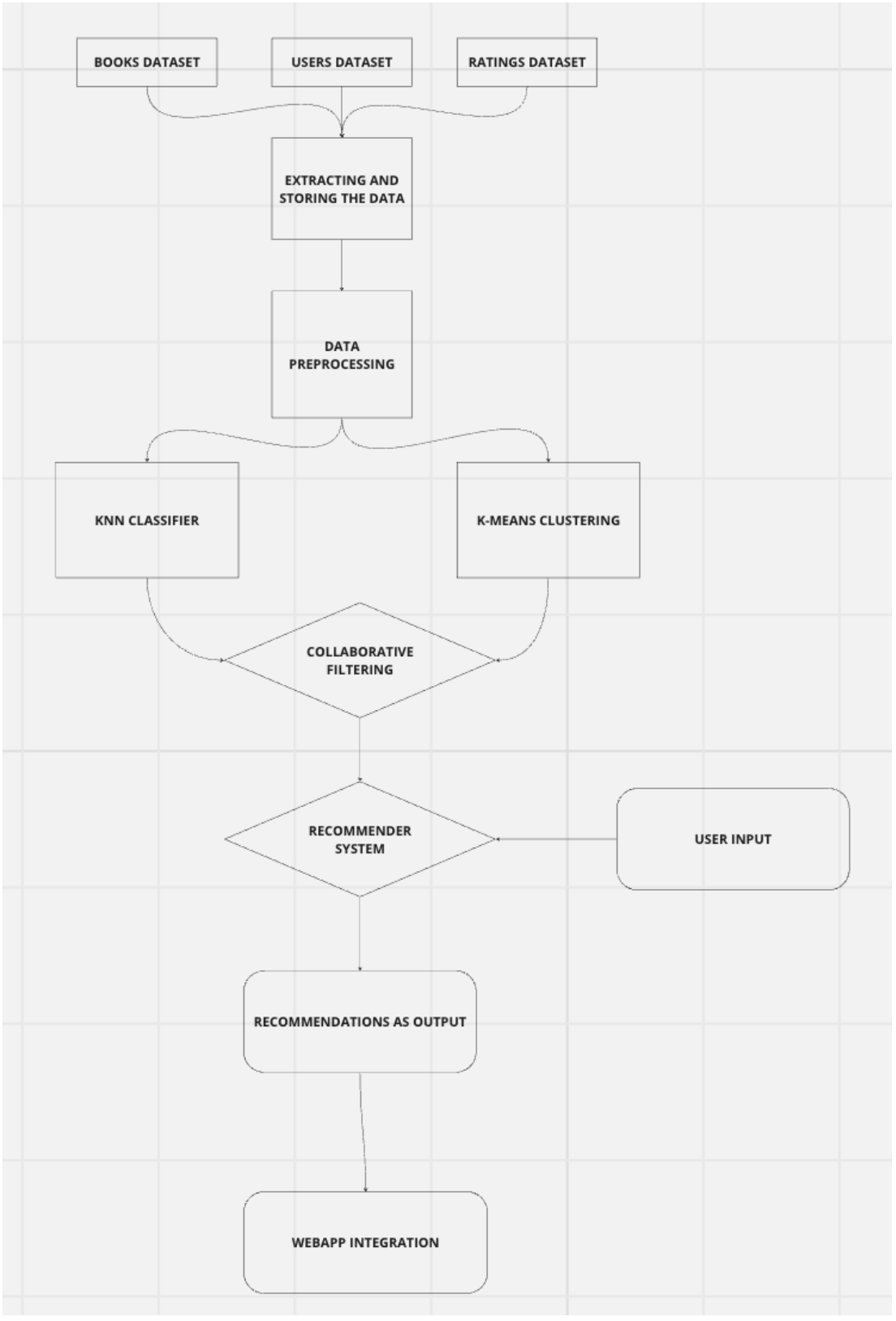


Fig. 5.1 System Architecture

5.2 UML Diagrams

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.
There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation). UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard. The two broadest categories that encompass all other types are:

1. Behavioral UML diagram
2. Structural UML diagram.

As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas others describe the behavior of the system, its actors, and its building components.

The different types are broken down as follows:

1. Sequence diagram
2. Use case Diagram
3. Activity diagram

5.2.1 Sequence diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

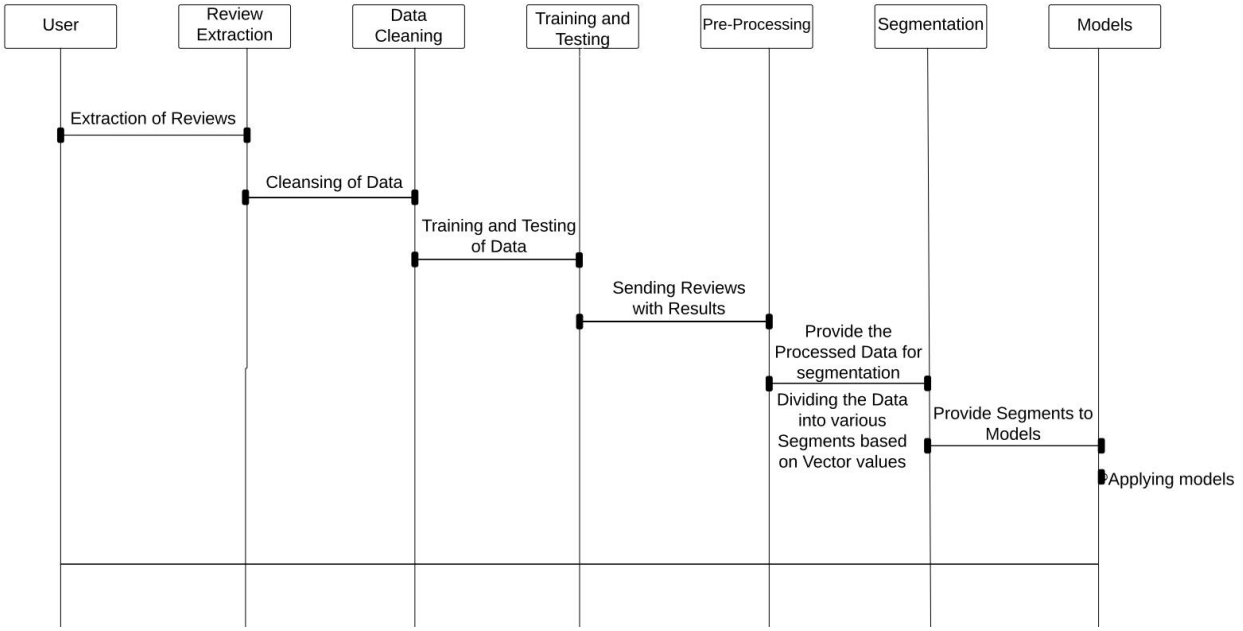


Fig. 5.2.1 Sequence diagram

List of actions User:

Users need to press any of the given three (i.e., Prediction data, prediction Skills, Jobsearch) then they will get the output accordingly.

System:

System gives the output when the user enters the text.

Results:
As per user entered text it provides the output.

5.2.2 Use case diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Users need to give the data. Then the System will give the results. System Consist of data set that will preprocess the data and then split the data and apply the train and test the data then it will predict the results.

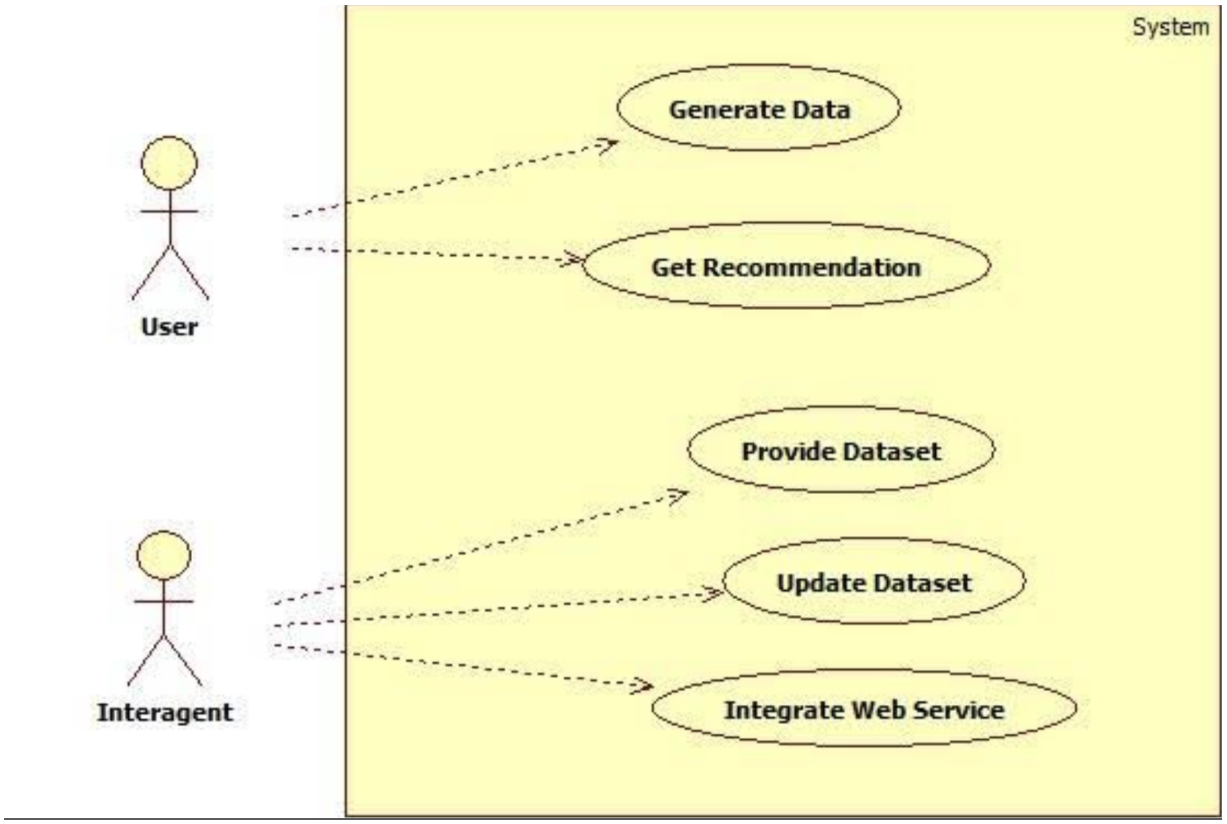


Fig. 5.2.2 Use case diagram

5.2.3 Activity diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc.

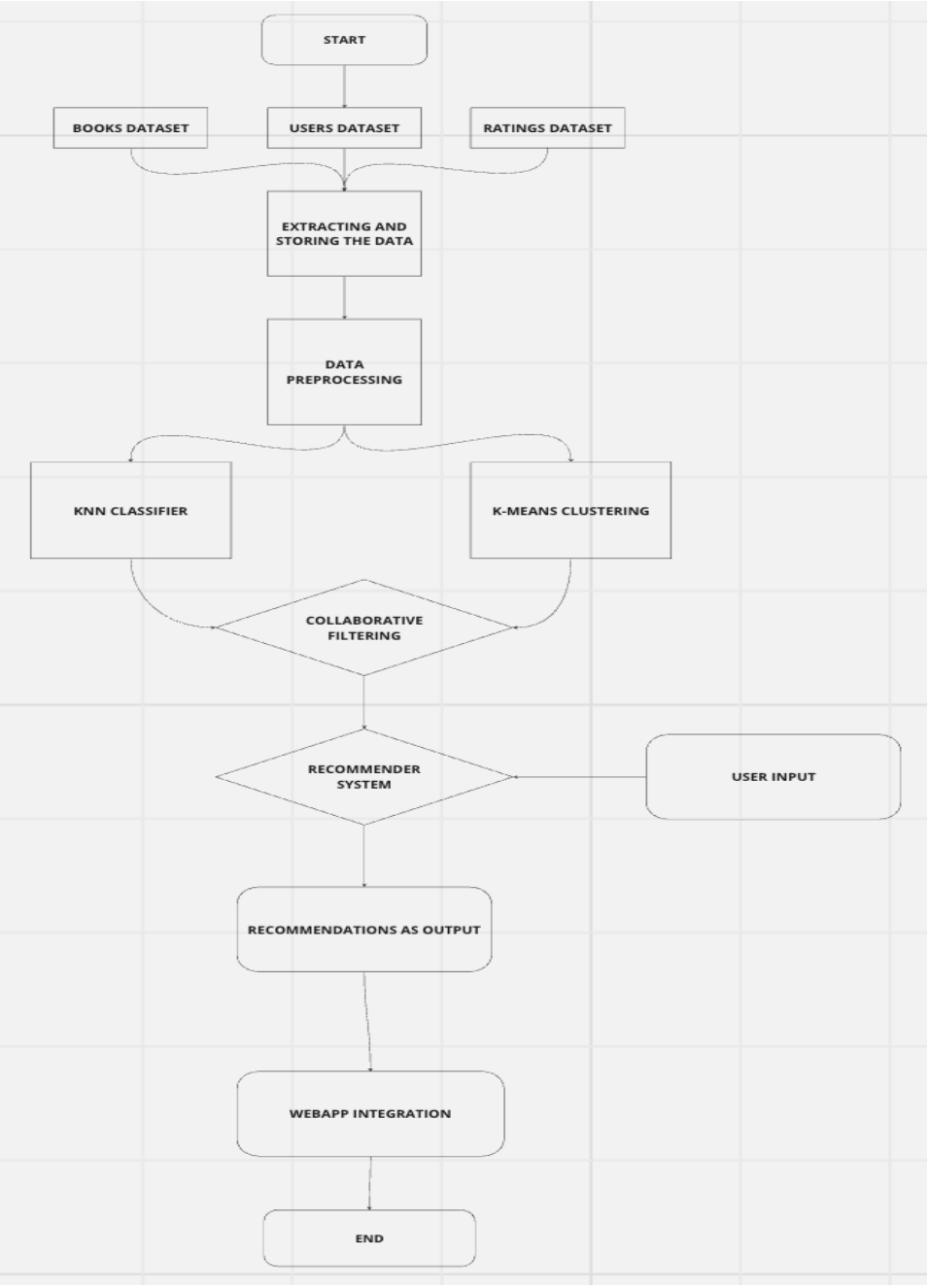


Fig. 5.2.3 Activity diagram

6. IMPLEMENTATION

6.1 Languages

Python

Python is a popular, versatile, and easy-to-read programming language known for its extensive standard library and large community. It's open source, works on various platforms, and is commonly used in web development, data analysis, machine learning, and more.

6.2 Modules

Streamlit

Streamlit is an open- source python library for creating web applications for data science and machine learning with minimal effort. It allows you to turn data scripts to shareable web apps quickly.

NumPy

NumPy is a Python library for numerical computing. It offers efficient multi-dimensional arrays, mathematical functions, and broadcasting for faster and easier numerical operations. It's widely used in scientific computing and data analysis.

Pandas

Pandas is a Python library for data manipulation and analysis. It offers data structures like DataFrames and Series, and tools for importing, cleaning, and analyzing data from various sources. It's widely used in data science and analysis.

Matplotlib

Matplotlib is a popular Python library for creating a wide variety of high-quality plots and charts. It's known for its versatility, customization options, and ability to produce publication-quality figures.

Scikit- learn:

Scikit learn is a python library which includes wide range of machine learning algorithms, such as linear and logistic regression, support vector machines, decision trees, k-nearest neighbors, clustering algorithms and more.

Seaborn :

Seaborn is a Python data visualization library that simplifies the creation of attractive and informative statistical plots. It offers high-level functions, color palettes, and seamless integration with Pandas for visualizing data relationships and distributions.

6.3 Source code

recommender.ipynb :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

books = pd.read_csv('BX-Books.csv', sep = ";", on_bad_lines = 'skip', encoding = 'latin-1')
books.head()
books.shape
books.columns


books = books[['ISBN','Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
'Image-URL-L']]
books.head()
books.shape

books.rename(columns = {
   "Book-Title" : "title",
   "Book-Author" : "author",
   "Year-Of-Publication" : "year",
   "Publisher" : "publisher",
   "Image-URL-L" : "img_url"}, inplace = True)
books.head()

users = pd.read_csv('BX-Users.csv', sep = ";", on_bad_lines = 'skip', encoding = 'latin-1')
users.head()
users.shape


ratings = pd.read_csv('BX-Book-Ratings.csv', sep = ";", on_bad_lines = 'skip', encoding = 'latin-1')
ratings.head()
ratings.shape
print(books.shape)
print(users.shape)
print(ratings.shape)


ratings.rename(columns = {
   "User-ID" : "user_id",
   "Book-Rating" : "rating",}, inplace = True)

ratings.head()
ratings['user_id'].value_counts()
ratings['user_id'].unique().shape
x = ratings['user_id'].value_counts()> 200
```

```
x[x].shape
y = x[x].index
y

ratings = ratings[ratings['user_id'].isin(y)]
ratings.head()
ratings.shape
books.head()


ratings_with_books = ratings.merge(books, on = "ISBN")
ratings_with_books.head()


ratings_with_books.shape
num_rating = ratings_with_books.groupby('title')['rating'].count().reset_index()
num_rating.head()


num_rating.rename(columns = {"rating" : "num_of_rating"}, inplace = True)
num_rating.head()


final_ratings = ratings_with_books.merge(num_rating, on = 'title')
final_ratings.head()
final_ratings.shape
final_ratings = final_ratings[final_ratings['num_of_rating'] >= 50]
final_ratings.head()


final_ratings.sample(10)
final_ratings.shape
final_ratings.drop_duplicates(['user_id', 'title'], inplace = True)
final_ratings.shape




book_pivot = final_ratings.pivot_table(columns = 'user_id', index = 'title', values = 'rating')
book_pivot
book_pivot.shape
book_pivot.fillna(0, inplace = True)
book_pivot


from scipy.sparse import csr_matrix
book_sparse = csr_matrix(book_pivot)
book_sparse
```

```
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(algorithm = 'brute')
model.fit(book_sparse)
distance, suggestion = model.kneighbors(book_pivot.iloc[237,:].values.reshape(1,-1), n_neighbors
= 6)
print(distance)
print(suggestion)


for i in range(len(suggestion)):
   print(book_pivot.index[suggestion[i]])
book_pivot.index[237]
books_name = book_pivot.index

# User item interaction matrix

plt.figure(figsize=(5, 5))
sns.heatmap(book_pivot, cmap='viridis', cbar=False)
plt.title('User-Item Matrix')
plt.xlabel('User ID')
plt.ylabel('Book Title')
plt.show()


#Number of ratings per user

plt.figure(figsize=(5, 5))
ratings['user_id'].value_counts().hist(bins=50, color='green', edgecolor='black')
plt.title('Number of Ratings per User')
plt.xlabel('Number of Ratings')
plt.ylabel('Users')
plt.show()




def recommend_book(book_name):
   book_id = np.where(book_pivot.index == book_name)[0][0]
   distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1),
n_neighbors = 6)
   for i in range(len(suggestion)):
      books = book_pivot.index[suggestion[i]]
      for j in books:
         print(j)
book_name = 'Harry Potter and the Chamber of Secrets (Book 2)'
recommend_book(book_name)


import pickle
```

```
pickle.dump(model, open('artifacts/model.pkl','wb'))
pickle.dump(books_name , open('artifacts/books_name.pkl','wb'))
pickle.dump(final_ratings, open('artifacts/final_ratings.pkl','wb'))
pickle.dump(book_pivot, open('artifacts/book_pivot.pkl','wb'))
```

main.py :

```
import pickle
import streamlit as st
import numpy as np


st.header("Book Recommender ")
model = pickle.load(open('artifacts/model.pkl', 'rb'))
books_name = pickle.load(open('artifacts/books_name.pkl', 'rb'))
final_rating = pickle.load(open('artifacts/final_ratings.pkl', 'rb'))
book_pivot = pickle.load(open('artifacts/book_pivot.pkl', 'rb'))


def fetch_poster(suggestion):
    book_name = []
    ids_index = []
    poster_url = []

    for book_id in suggestion:
        book_name.append(book_pivot.index[book_id])

    for name in book_name[0]:
        ids = np.where(final_rating['title'] == name)[0][0]
        ids_index.append(ids)

    for idx in ids_index:
        url = final_rating.iloc[idx]['img_url']
        poster_url.append(url)

    return poster_url


def recommend_book(book_name):
    book_list = []
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(
        book_pivot.iloc[book_id, :].values.reshape(1, -1), n_neighbors=6
    )

    poster_url = fetch_poster(suggestion)
```

```
    for i in range(len(suggestion)):
        books = book_pivot.index[suggestion[i]]
        for j in books:
            book_list.append(j)

    return book_list, poster_url


selected_books = st.selectbox(
    "Type or select a book",
    books_name
)


if st.button("Show Recommendation"):
    recommendation_books, poster_url = recommend_book(selected_books)
    col1, col2, col3, col4, col5 = st.columns(5)

    with col1:
        st.text(recommendation_books[1])
        st.image(poster_url[1])

    with col2:
        st.text(recommendation_books[2])
        st.image(poster_url[2])

    with col3:
        st.text(recommendation_books[3])
        st.image(poster_url[3])

    with col4:
        st.text(recommendation_books[4])
        st.image(poster_url[4])

    with col5:
        st.text(recommendation_books[5])
        st.image(poster_url[5])
```

7. SYSTEM TESTING

7.1. Introduction to System Testing

System testing is a crucial software testing phase that evaluates the entire software system to verify functionality, performance, and compliance with requirements. It follows integration testing and ensures that the software works as intended in real-world scenarios, covering both functional and non-functional aspects.

7.1.1 Importance in the Software Development Life Cycle (SDLC)

In a sentiment analysis project using Transformers, key SDLC phases are as follows:
Requirement Analysis : Critical for understanding project needs.
Development : Very important for code implementation.
Testing : Critical for model accuracy.
Deployment : Important for real-world application.
Maintenance and Support : Ongoing for system health.
Documentation : Important for knowledge transfer.
Project Management : Very important for efficient progress.
User Training : Important for effective system use.
Feedback and Iteration : Continuous for system enhancement.

7.1.2. Objectives of System Testing

In our project the system testing objectives is to verify accuracy, scalability, robustness, reliability, performance, security, usability, integration, and compliance, ensuring it handles various scenarios and meets industry standards while maintaining user-friendly functionality.

7.2 System Testing Strategy:

The system testing strategy involves thorough evaluation to verify the system's accuracy, efficiency, and reliability in real-world scenarios, including diverse input data. Continuous testing and monitoring are crucial for ongoing performance.

7.3 Testing Approach (Black-box, White-box)

7.3.1. White box :

White-box testing for our project assesses the system's internal components, code, and algorithms to ensure accurate and efficient sentiment analysis. It uncovers potential issues, ensuring reliability.

7.3.2. Black box :

Black-box testing for our project evaluates system functionality based on input and output without examining internal details. It focuses on user requirements and external behavior to ensure expected performance and usability.

7.4. Test Techniques and Methods

In a book recommender project using collaborative filtering, essential testing techniques and methods include unit testing, integration testing, performance testing, user acceptance testing, and more. These methods collectively ensure the system's quality and reliability.

7.5. Test Cases and Scenarios

In a book recommender project using collaborative filtering, test cases and scenarios include proper picks and wrong picks by the model, performance, real-world data, extreme cases like sparse data in the matrix, user interactions with the particular item etc. These tests ensure the system's accuracy and reliability while meeting project requirements.

7.6. Test Execution

Test execution in a book recommender project using collaborative filtering involves running the defined test cases and scenarios. This process assesses the system's accuracy, efficiency, and performance. It includes executing the proper picks and wrong picks by the model, performance, real-world data, extreme cases like sparse data in the matrix, user interactions with the particular item to validate that the system functions as intended and meets project requirements. The results are recorded and analyzed to identify issues or areas for improvement.

7.7. Test Summary and Results

In a book recommender project using collaborative filtering, the test summary and results involve compiling findings from test execution. This includes data on test case outcomes, performance metrics, and any issues or bugs identified during testing. The summary provides a comprehensive view of the system's performance, accuracy, and any areas needing improvement. It guides decisions on whether the recommender system is ready for deployment or requires further refinement.

Test cases:

#1

```
def recommend__book (book_name) :
book_id = np.where(book_pivot. index == book_name) [0] [0]
distance, suggestion = model.
kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1),n_neighbors=6)
for i in range(len(suggestion) ):
books = book_pivot. index [ s u g g e s t i o n [ i l l
for j in books:
print (j )

book_name = 'Ground Zero and Beyond'
recommend_book (book_name)
```

Ground Zero and Beyond
A Secret Affair
The Blooding
About Face
Ssn
Man From St Petersburg

#Positive outcome from the model

#2

book_name = ‘Harry Potter and the Prisoner of Azkaban (Book 3)’
recommend_book(book_name)

Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Goblet of Fire (Book 4)
Harry Potter and the Chamber of Secrets (Book 2)
All I Need Is You
Eternity
Blood and Gold (Rice, Anne, Vampire Chronicles.)

#Positive outcome from the model

7.8 Issues encountered

```
def recommend_book(book_name):
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors = 6)
    for i in range(len(suggestion)):
        books = book_pivot.index[suggestion[i]]
        for j in books:
            print(j)
book_name = 'Atomic Habits'
recommend_book(book_name)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 9
      7             print(j)
      8 book_name = 'Atomic Habits'
----> 9 recommend_book(book_name)

Cell In[1], line 2, in recommend_book(book_name)
      1 def recommend_book(book_name):
----> 2     book_id = np.where(book_pivot.index == book_name)[0][0]
      3     distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors = 6)
      4     for i in range(len(suggestion)):

NameError: name 'np' is not defined
```

Table 7.1 Test Cases

#NameError whenever the browsed book is not in the list.

8. OUTPUT SCREEN

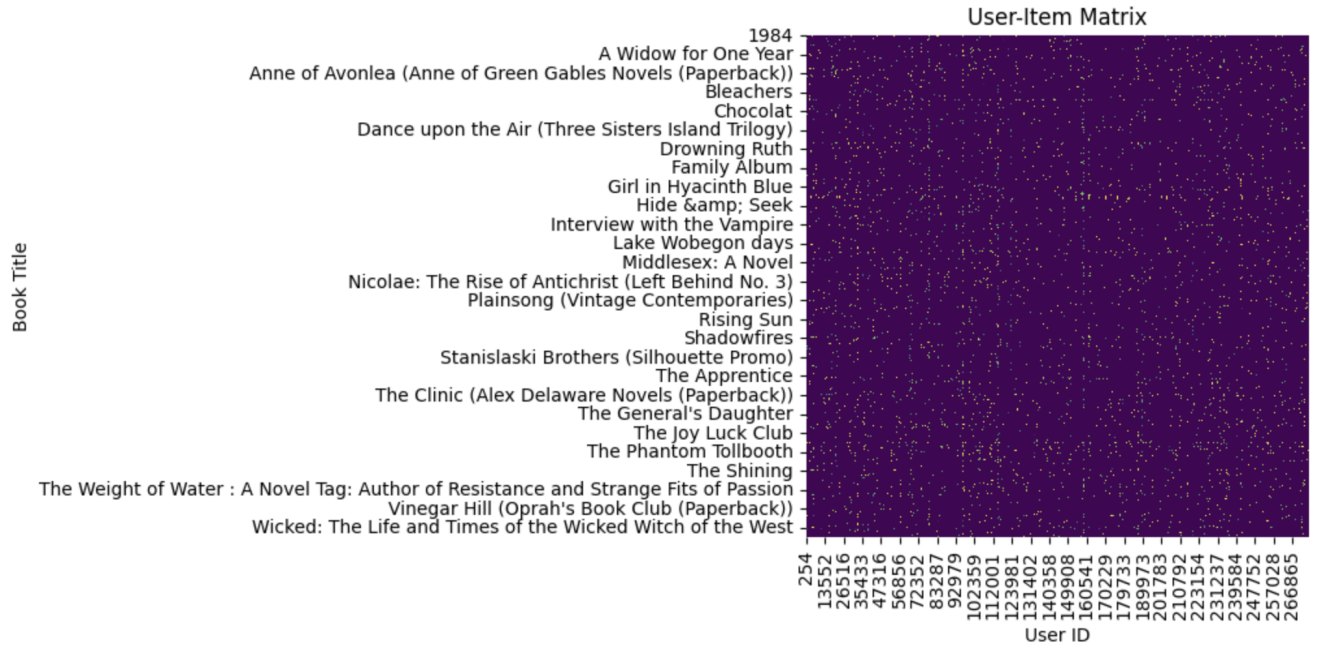


FIG 8.1 User Item Interaction Matrix

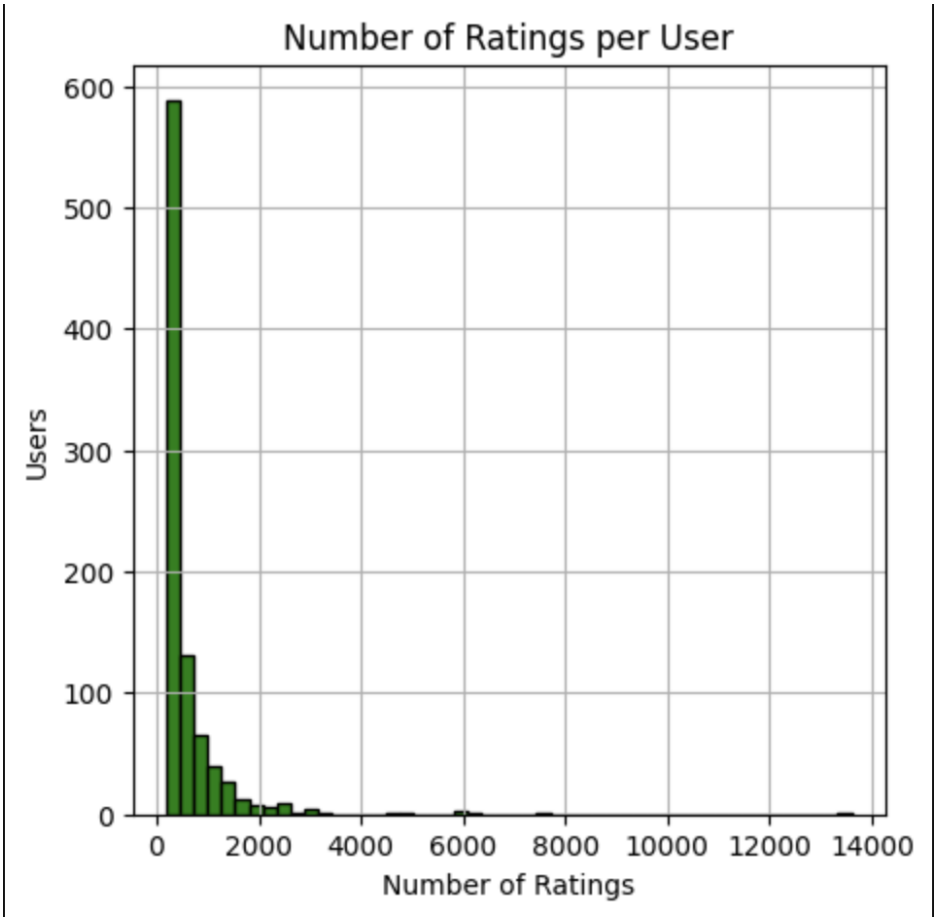


FIG 8.2 Number of ratings per user

```
[38]: def recommend_book(book_name):
          book_id = np.where(book_pivot.index == book_name)[0][0]
          distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors = 6)
          for i in range(len(suggestion)):
              books = book_pivot.index[suggestion[i]]
              for j in books:
                  print(j)
      book_name = 'Harry Potter and the Chamber of Secrets (Book 2)'
      recommend_book(book_name)

      Harry Potter and the Chamber of Secrets (Book 2)
      Harry Potter and the Prisoner of Azkaban (Book 3)
      Harry Potter and the Goblet of Fire (Book 4)
      Harry Potter and the Sorcerer's Stone (Book 1)
      Exclusive
      The Cradle Will Fall
```

FIG 8.3 Output 1

```
[40]: def recommend_book(book_name):
          book_id = np.where(book_pivot.index == book_name)[0][0]
          distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.reshape(1,-1), n_neighbors = 6)
          for i in range(len(suggestion)):
              books = book_pivot.index[suggestion[i]]
              for j in books:
                  print(j)
      book_name = '1984'
      recommend_book(book_name)

      1984
      No Safe Place
      A Civil Action
      Foucault's Pendulum
      Long After Midnight
      Abduction
```

FIG 8.4 Output 2

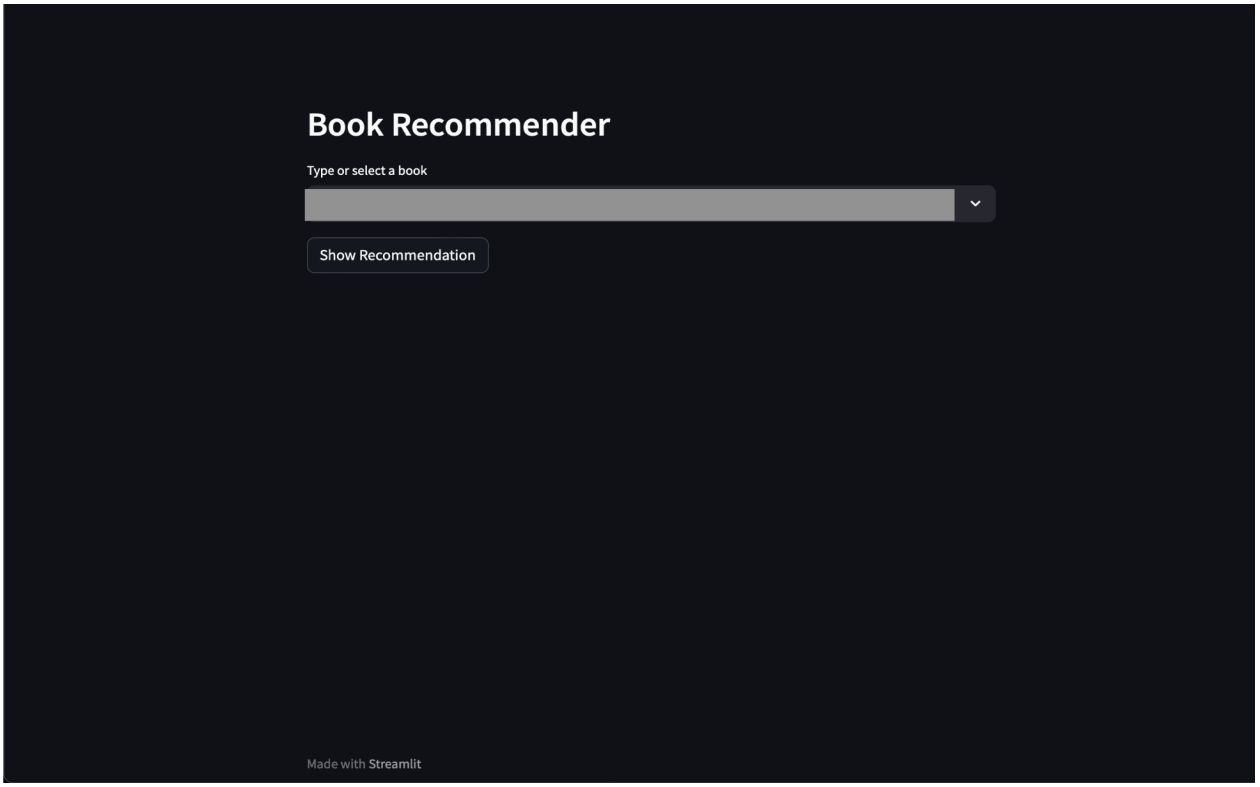


FIG 8.5 output 3

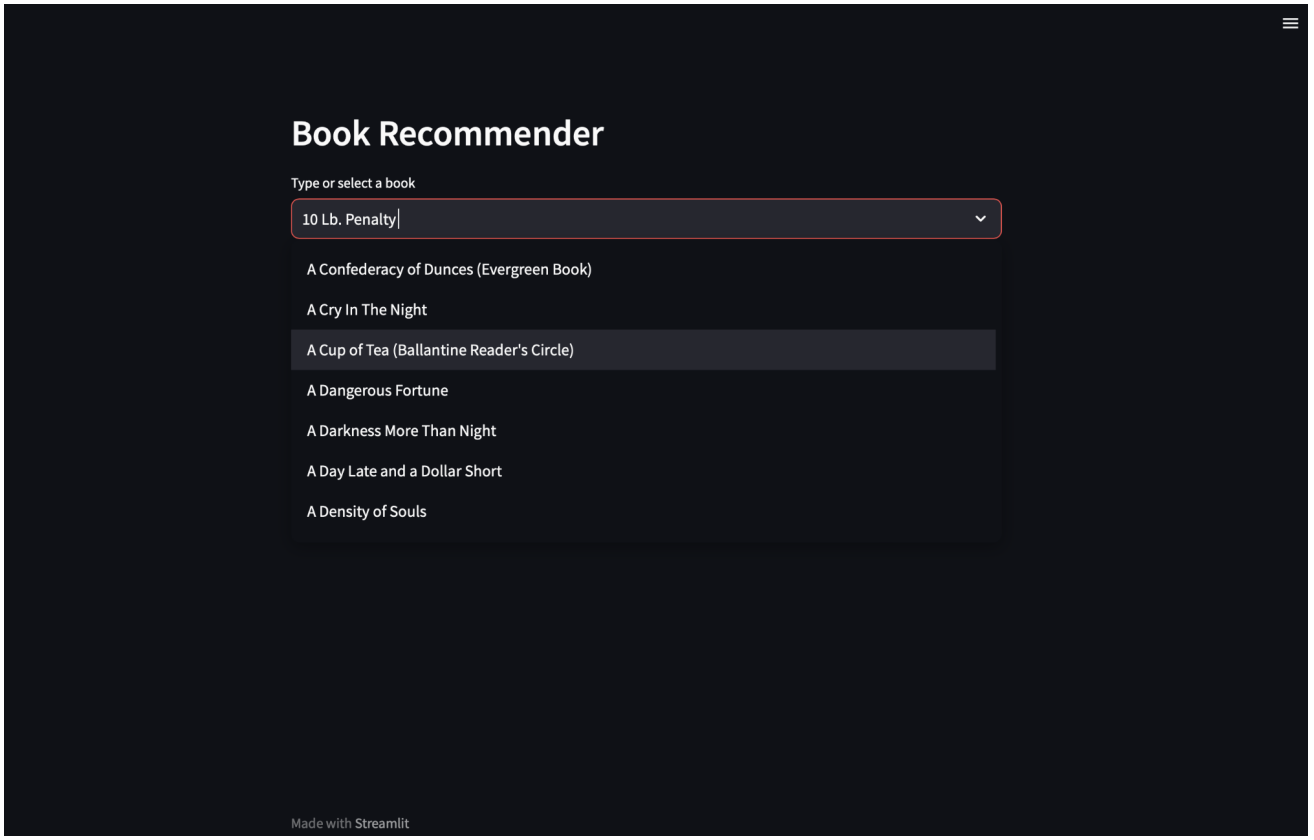


FIG 8.6 Output 4

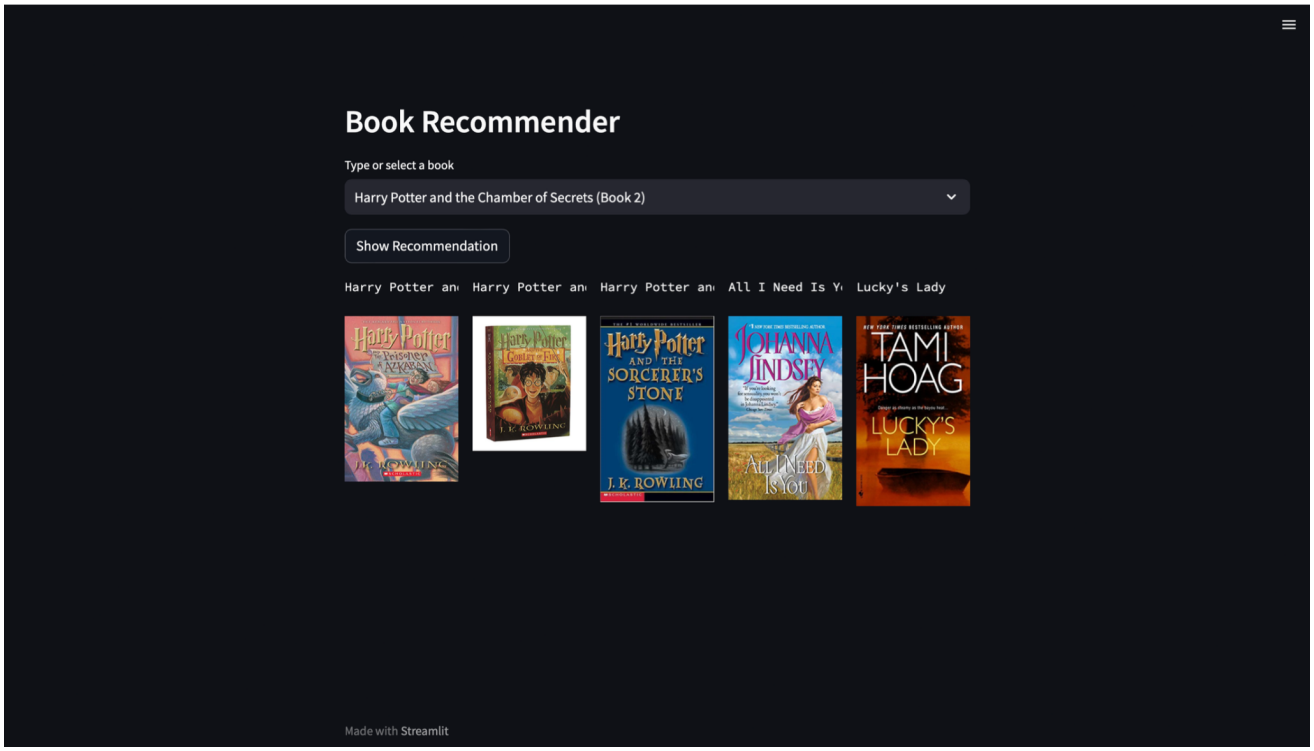


FIG 8.7 Output 5

9. CONCLUSION

Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they desire. Conversely, they help the businesses by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the web.

In this project, we recommended the books for a user using collaborative filtering approach using K- nearest neighbors algorithm. We also compared different models built using methods and identified the best model and justified that collaborative filtering is advanced and recommended for building recommender systems. We have used datasets from kaggle. Books, ratings and users datasets are used for this particular project which helped to leverage the whole system into an efficiently performing model.

10. FUTURE ENHANCEMENTS

As the future endeavors of the project and the system, it has adequate scope of modification; if it is necessary. Development and launching of a mobile app, refining existing services and adding more services. Also, system security, data security and reliability are the key features which can further be included in future work.

In addition to this, the API for shopping and payment gateway can be added so that the user can purchase a book at the moment. Furthermore, in the existing system, there are only few selected categories, so as an extension to the catalog, we can include few more categories based on user interests.

Adding to this, we can add admin side with some functionalities like books catalog management, user management and server management.

11. REFERENCES

• [1] Billsus, D., and Pazzani, M. J. (1998). Learning Collaborative Information Filters. In Proceedings of ICML ’98. pp. 46-53. x

• [2] Borchers, A., Leppik, D., Konstan, J., and Riedl, J. (1998). Partitioning in Recommender Systems. Technical Report CS-TR-98-023,
Computer Sci- ence Dept., University of Minnesota.

• [3] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In
Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43-52.

• [4] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry.
Communications of the ACM. December.

• [5] Sarwar, B. M., Konstan, J. A., Borchers, A., Her- locker, J., Miller, B., and Riedl, J. (1998). Using Filtering Agents to Improve
Prediction Quality in the GroupLens Research Collaborative Filtering System. In Proceedings of CSCW ’98, Seattle, WA.