BOOK RECOMMENDATION SYSTEM

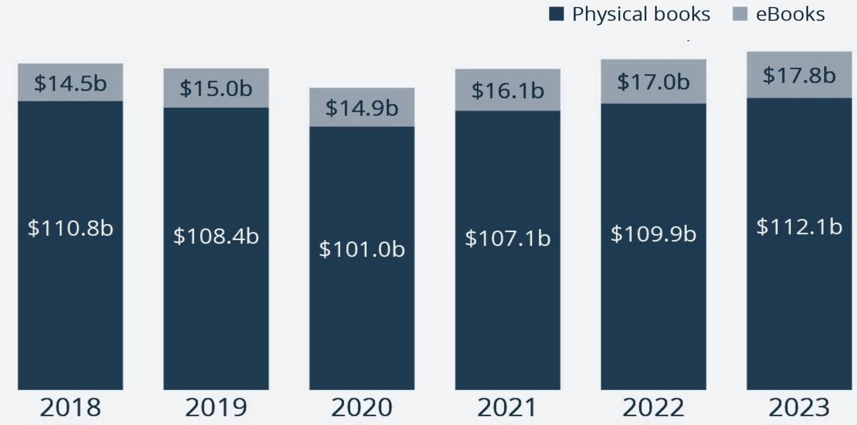
Team members

Anis Fathima
Bhargav Simha
Sravan
Sandhya
Dileep

With the guidance of
KARTHICK MUSKULA

BACKGROUND

Worldwide estimated revenue with eBooks and physical books



Source: Statista Advertising & Media Market Outlook

According to a survey conducted by Pew Research in 2019, the popularity of digital books is anticipated to rise over the coming years. The sales of e-book readers is increasing along with the popularity of e-books.

amazonkindle
goodreads

Amazon want to develop a high-quality recommendation system

Enhancing conversion rate for Amazon Kindle

Readers can find books that suit their taste

OBJECTIVE

GOAL

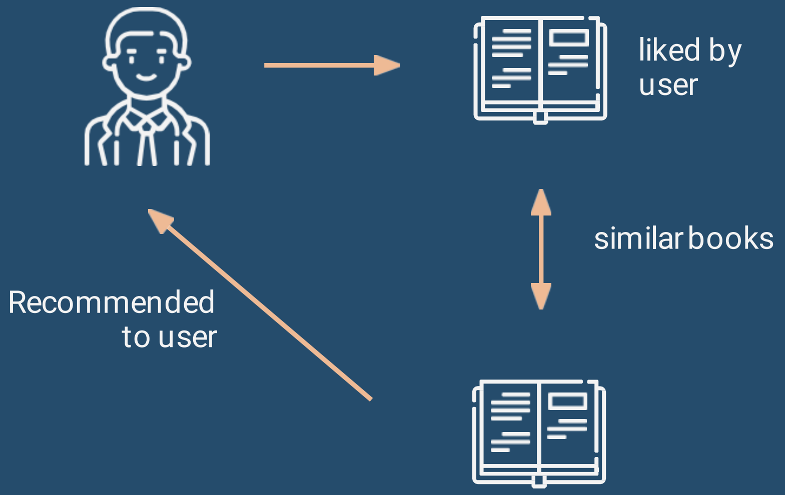
Design book recommendation system by several methods and explore the strengths and weaknesses of each method

RESEARCH QUESTION

1. How can we calculate the similarity between books based on their content? How can we apply it to a recommendation system?
2. How can we predict the rating that a user will give to books that they haven't read? How can we apply it to a recommendation system?
3. What are the strengths and weakness of each method? How to overcome that weakness?

TYPES OF RECOMMENDATION SYSTEM

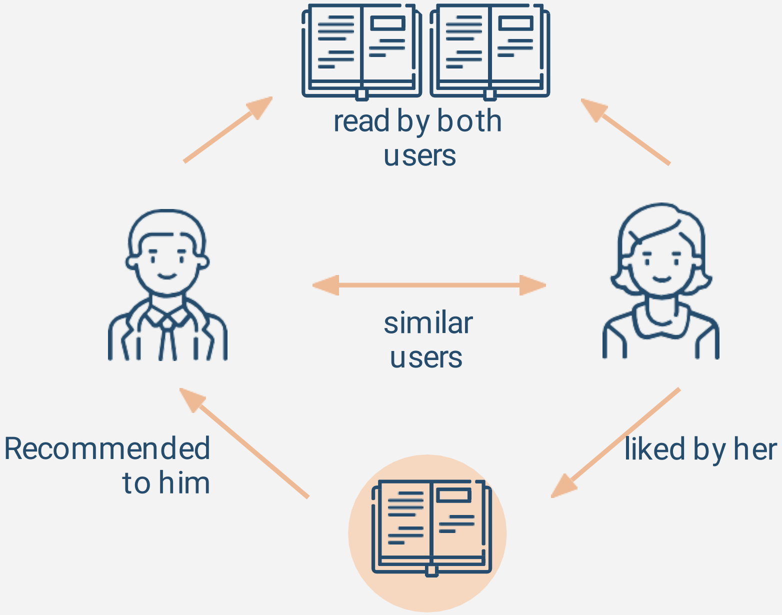
Content Based Filtering



Measure similarity between books Example: cosine similarity

Process: Text → Vector using Tf idfVectorizer

Collaborative Filtering



Memory-based:

Predict ratings by learning user's pattern of giving ratings. Example: KNN

Model-based:

Predict ratings by learning user latent factor and item latent factor. Example: SVD, SVD++

DATA UNDERSTANDING

This dataset was originally scraped from the **Goodreads API** in September 2017 by Zygmunt Zając and updated by Olivier Simard-Hanley. You can download the data from this Github repository.

- Ratings
- Book Metadata
- To-read
- Tag
- Book-tag

53,424 users

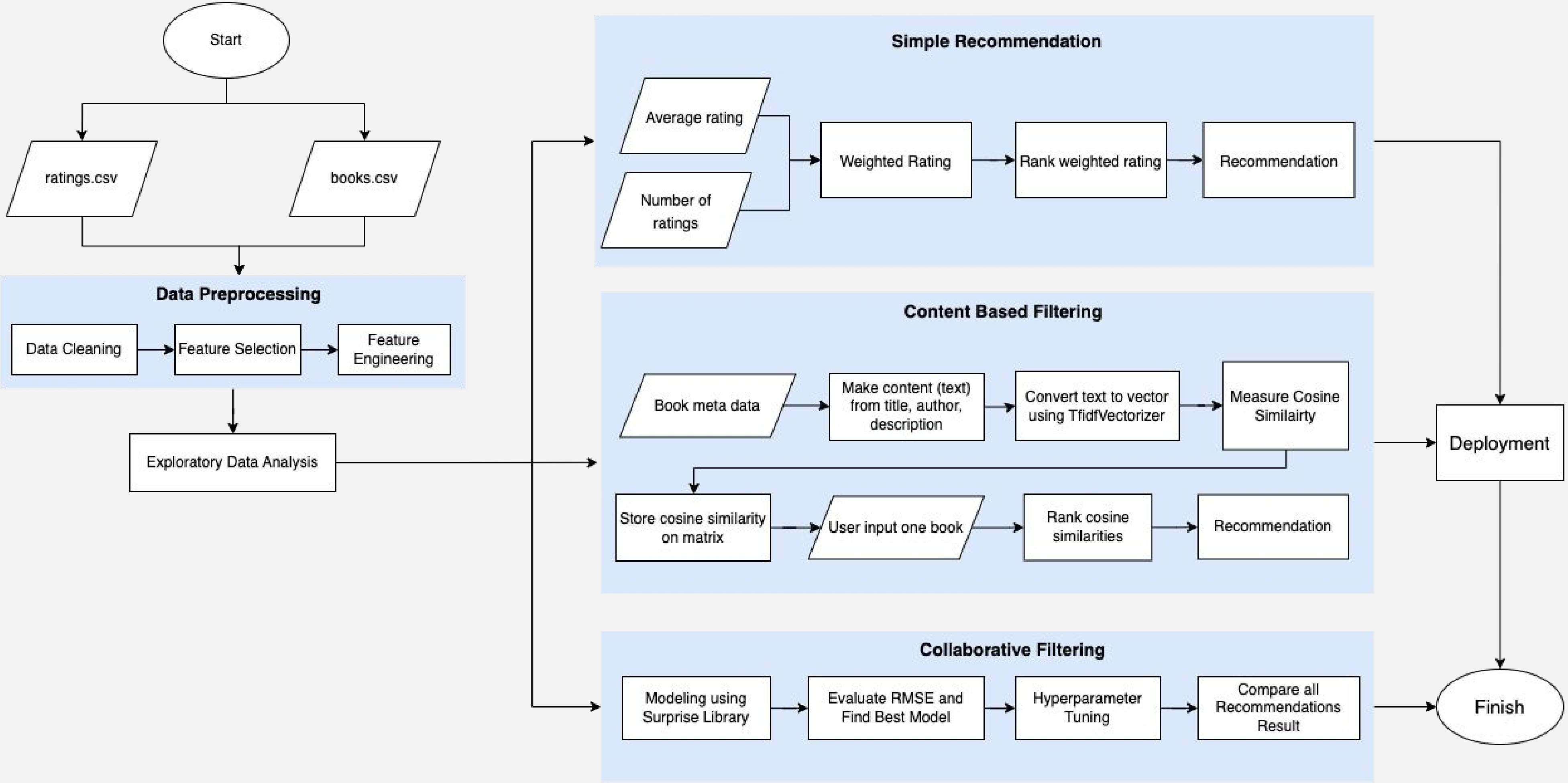
5,9M+ ratings

10,000 books

Metadata

- Identification number (6 cols)
- Title (2 cols)
- Authors (2 cols)
- Publication year related (2 cols)
- Rating related (9 cols)
- Image url (2 cols)
- Book count
- Language code
- Genres
- Description
- Pages
- Others (2 cols)

SYSTEM FLOWCHART



DATA PREPROCESSING

MISSING VALUE HANDLING

Column Name	percent_missing
isbn	7.00
original_title	5.85
isbn13	5.85
pages	0.73
description	0.57
original_publication_year	0.21
publishDate	0.08

- Impute original_publication_year using publishDate, then drop publishDate
- Impute pages with median
- Impute description with book's title
- Drop the rest

FEATURE SELECTION

30 features → 10 features

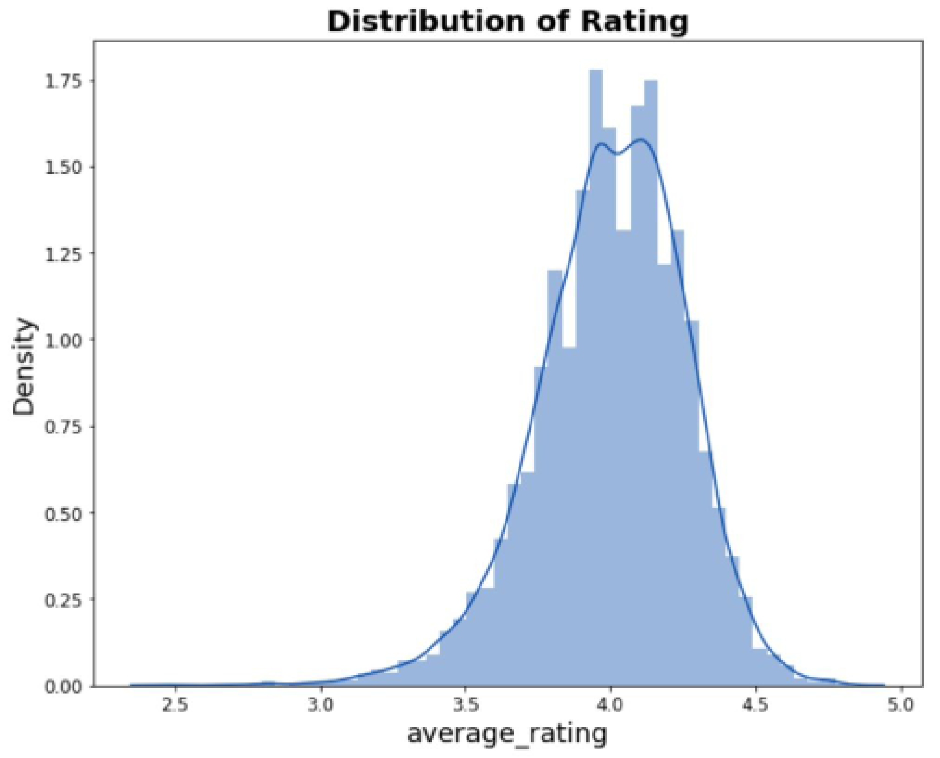
ratings.csv
book_id
user_id
rating

book_cleaned.csv
book_id
title
authors
year
pages
description
genres
average_rating
ratings_count
books_count

FEATURE ENGINEERING

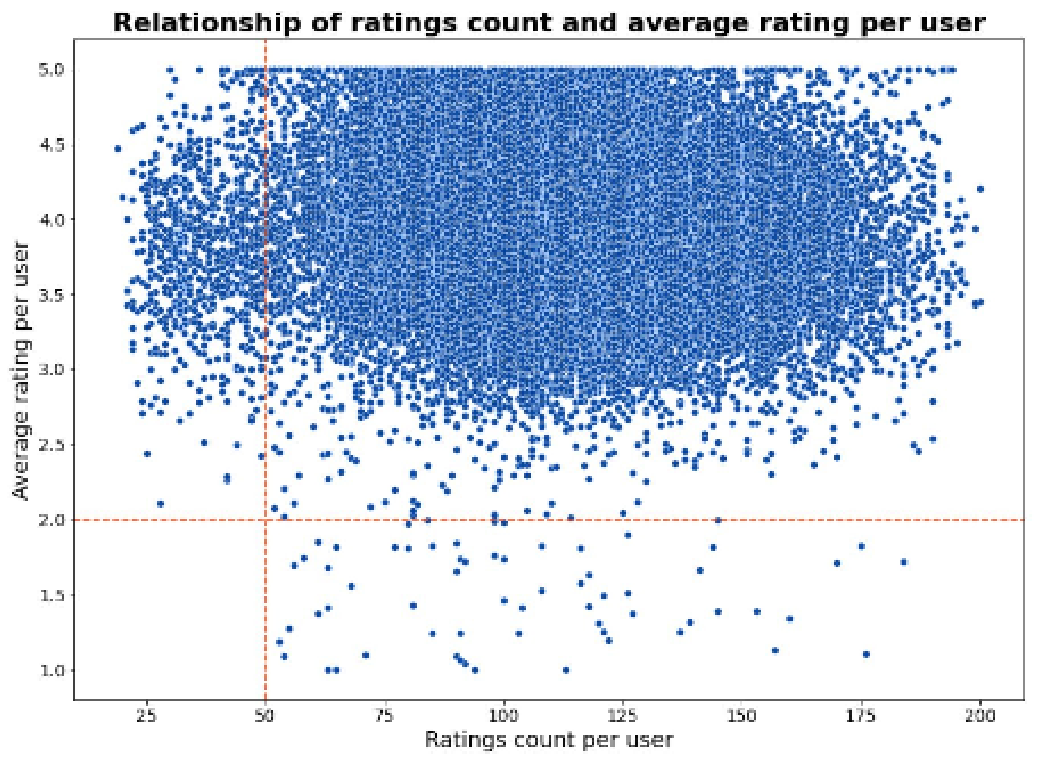
Exploratory Data Analysis

How is the distribution of ratings?



Since this is the list of 10,000 popular books, the majority of the books have **an average value of 4.02.**

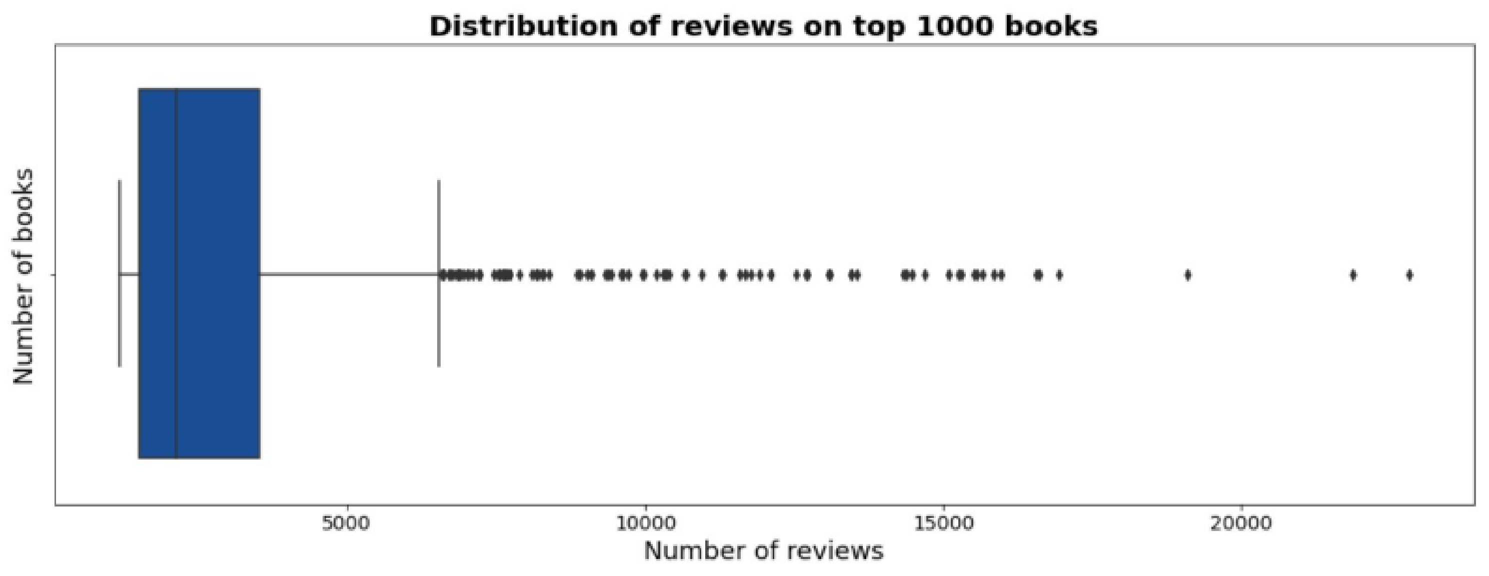
Does the number of reviews affect ratings?



People who rate < 50 books tend to give higher ratings. People start to give lower rating if they read more books.

This could be a result of an inappropriate book recommendation system, so that people end up reading books they don't like.

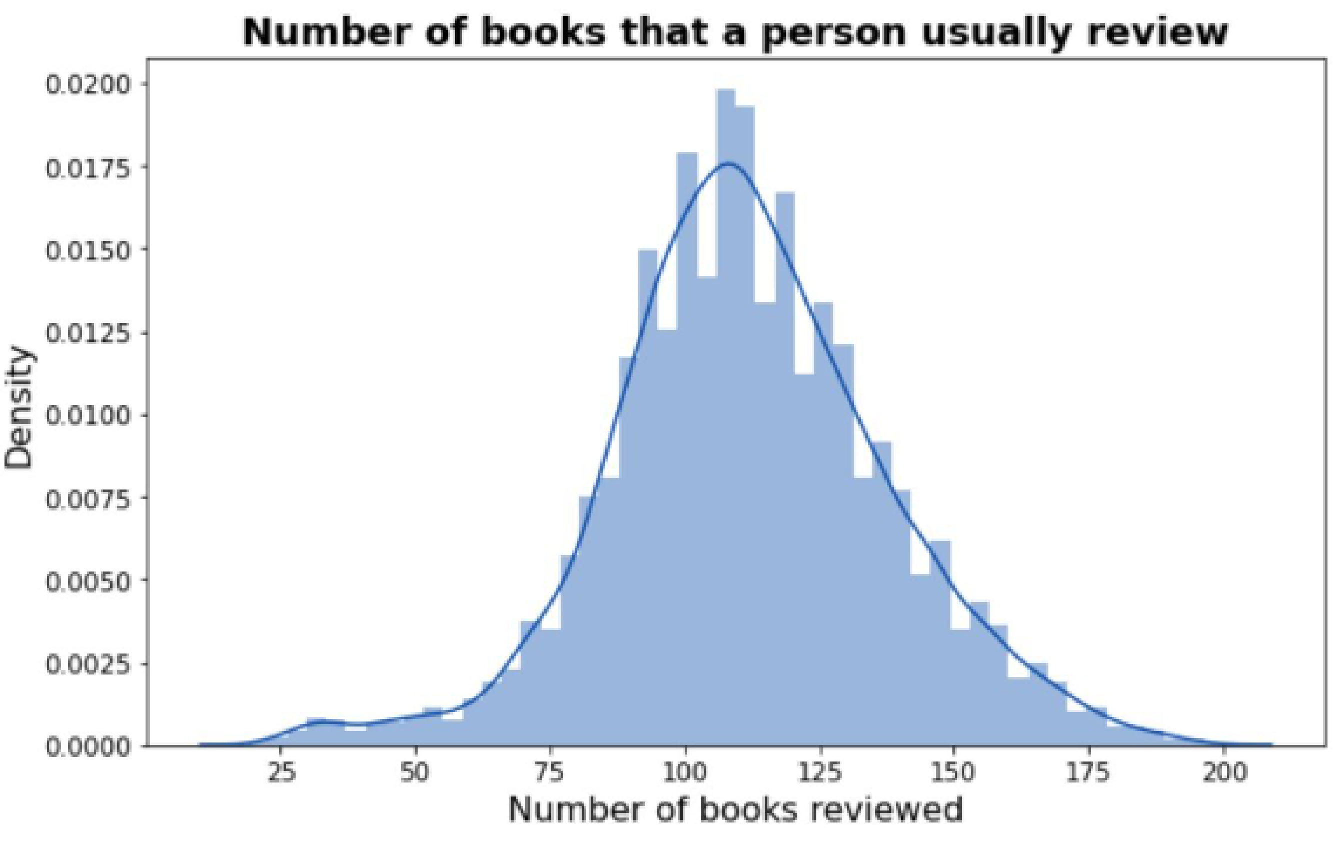
How many ratings does a book usually get?



At most: 22,806 ratings
At least: 8 ratings
Average: 248 ratings

The distribution of reviews on books is positively skewed. There are more books that has less ratings.

How many ratings does a user usually give?



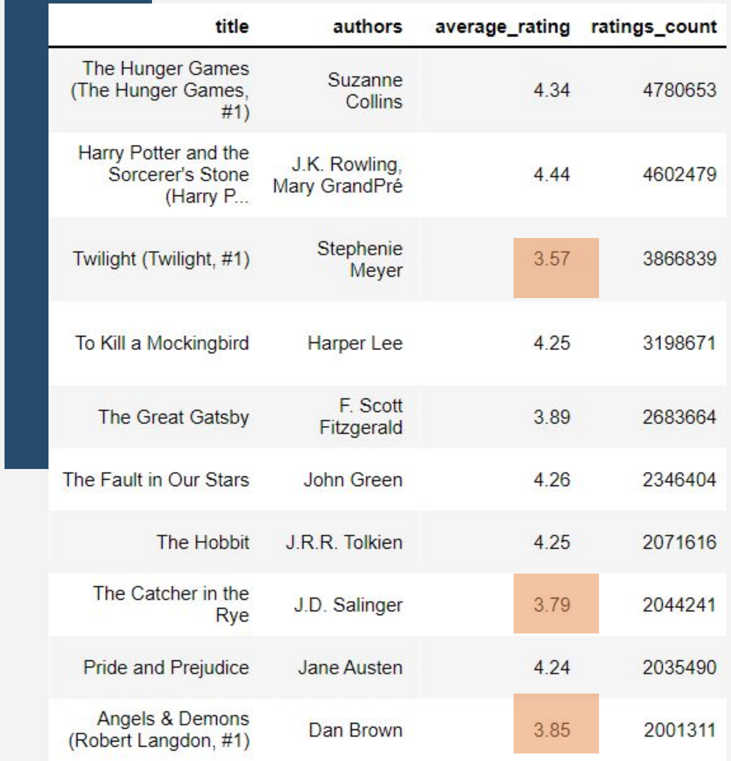
At most: Rated 200 books
At least: Rated 19 books.
Average: Rated 111 books.

From 10,000 books in our dataset, even the user with the highest number of reviews managed to give a rating to only 2%of all of the books.

Data in user is very sparse, so it will be better to use **item-based collaborative filtering.**

RANK THE BOOKS

Books (ratings_count)



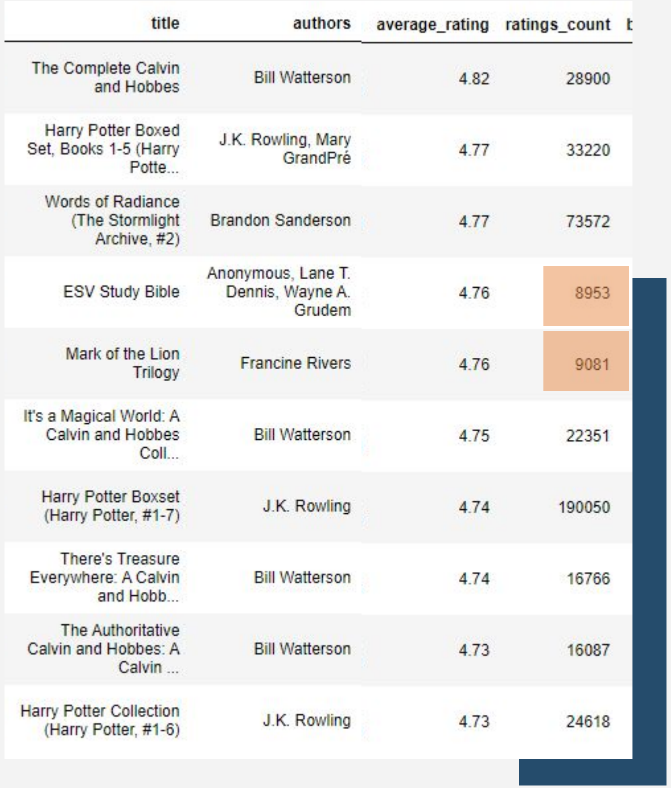
title	authors	average_rating	ratings_count
The Hunger Games (The Hunger Games, #1)	Suzanne Collins	4.34	4780653
Harry Potter and the Sorcerer's Stone (Harry P...	J.K. Rowling, Mary GrandPré	4.44	4602479
Twilight (Twilight, #1)	Stephenie Meyer	3.57	3866839
To Kill a Mockingbird	Harper Lee	4.25	3198671
The Great Gatsby	F. Scott Fitzgerald	3.89	2683664
The Fault in Our Stars	John Green	4.26	2346404
The Hobbit	J.R.R. Tolkien	4.25	2071616
The Catcher in the Rye	J.D. Salinger	3.79	2044241
Pride and Prejudice	Jane Austen	4.24	2035490
Angels & Demons (Robert Langdon, #1)	Dan Brown	3.85	2001311

There are several popular books that have a lower rating

higher rank

lower rank

Books (average_rating)



title	authors	average_rating	ratings_count
The Complete Calvin and Hobbes	Bill Watterson	4.82	28900
Harry Potter Boxed Set, Books 1-5 (Harry Potte...	J.K. Rowling, Mary GrandPré	4.77	33220
Words of Radiance (The Stormlight Archive, #2)	Brandon Sanderson	4.77	73572
ESV Study Bible	Anonymous, Lane T. Dennis, Wayne A. Grudem	4.76	8953
Mark of the Lion Trilogy	Francine Rivers	4.76	9081
It's a Magical World: A Calvin and Hobbes Coll...	Bill Watterson	4.75	22351
Harry Potter Boxset (Harry Potter, #1-7)	J.K. Rowling	4.74	190050
There's Treasure Everywhere: A Calvin and Hobb...	Bill Watterson	4.74	16766
The Authoritative Calvin and Hobbes: A Calvin ...	Bill Watterson	4.73	16087
Harry Potter Collection (Harry Potter, #1-6)	J.K. Rowling	4.73	24618

Books with a high rating sometimes have a lower number of reviews

We need to make **a weighted score** that combines average_rating and rating_count

Modelling

1. Simple Recommender
2. Content Based Recommendation
3. Collaborative Filtering Recommendation

1. SIMPLE RECOMMENDER

This is a **non-personalized recommender**.
One of the easiest way to give recommendation is to rank the book based on rating (average_rating) or ratings_count.
However, as we mentioned in EDA, we need to make a weighted rating of average_ratingand rating_count.

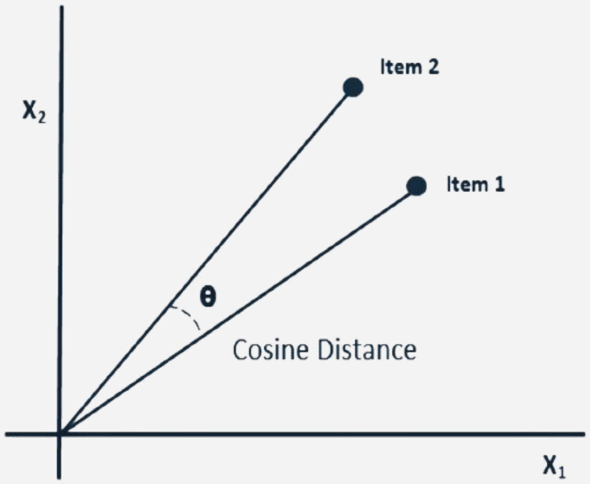
$$New\ Rating\ Score\ = \left(\frac{v}{v+m} \ .\ R\right) + \left(\frac{m}{v+m} \ .\ C\right)$$

New Rating Score formula used in Internet Movie Database (IMDb)

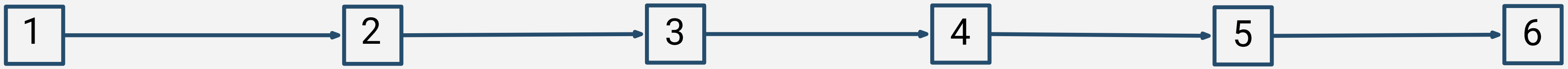
where:
v = number of ratings (ratings_count)
m = minimum ratings_count required to be recommended R =
average of ratings (average_rating)
C = the mean ratings for all books

2. CONTENT BASED FILTERING

This approach makes recommendations to users based on the features or characteristics of the books. Using item metadata, the computer will assess how similar the books are to one another and then recommend the books that are most like the one the user loved. One of the way is using cosine similarity.



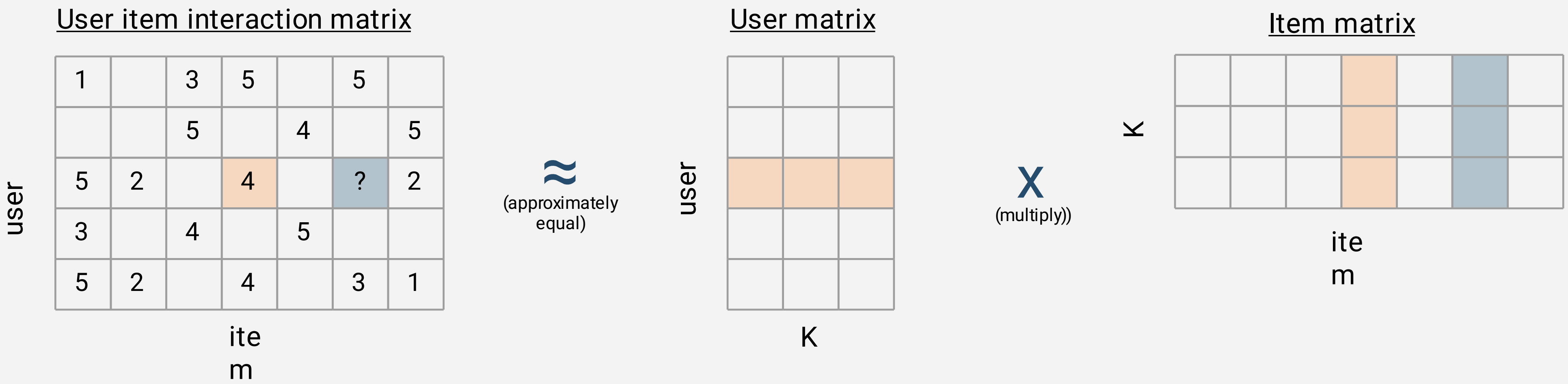
Flow Content Based Filtering



1. Collect feature (title, author, description and genre) as a content
2. Convert content using Tf idfVectorizer
3. Measure cosine similarity for all books
4. Input favorite book
5. Rank the cosine similarity of the books based on input
6. Recommend books

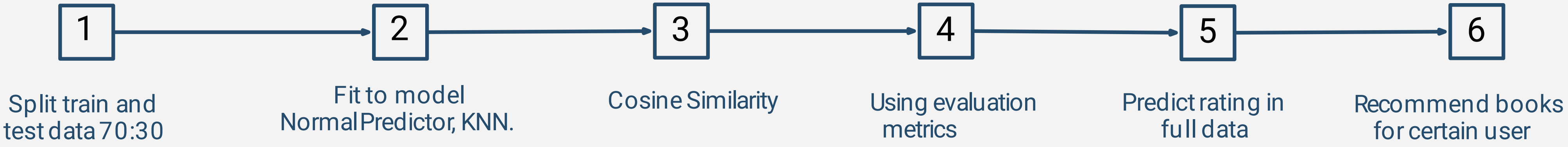
3. COLLABORATIVE FILTERING

This approach recommends books to users based on their prior reading habits and the preferences of other users.



1		3	5		5	
		5		4		5
5	2		4		?	2
3		4		5		
5	2		4		3	1

Flow Collaborative Filtering



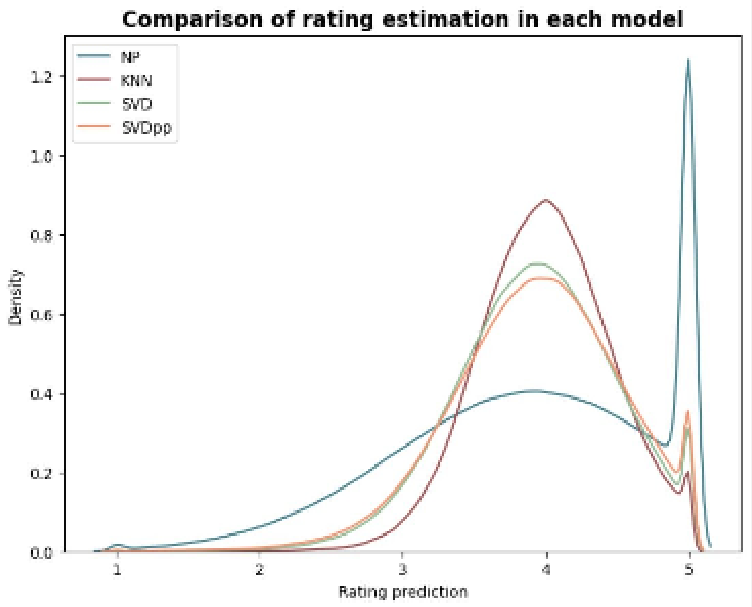
1. Split train and test data 70:30
2. Fit to model NormalPredictor, KNN.
3. Cosine Similarity
4. Using evaluation metrics
5. Predict rating in full data
6. Recommend books for certain user

Result and Evaluation

Evaluation in Collaborative Filtering Recommender

Base Model Comparison

Model	RMSE* Train	RMSE* Test	Time	Estimated Memory Use in AWS
Normal Predictor (NP)	1.3233	**1.3236**	3m 40s	8.7 GB
KNN	0.8001	**0.8851**	24m 45s	16+ GB

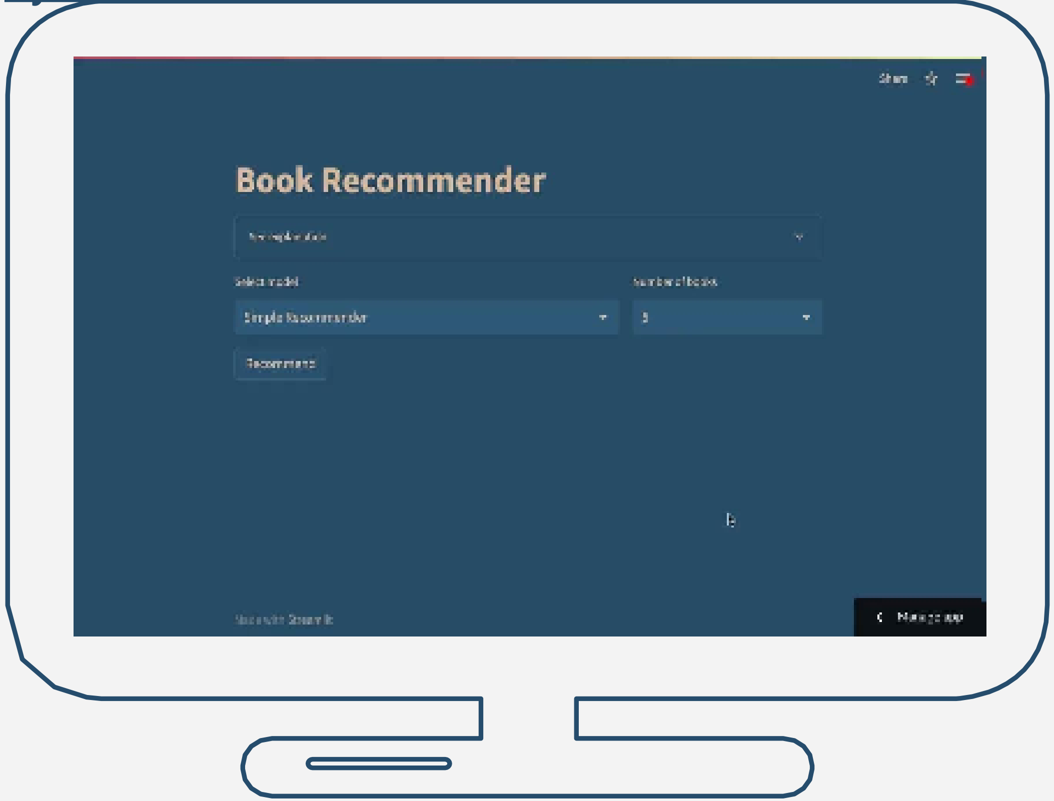


- NP predicts higher ratings more often
- KNN predictions are concentrated around the mean

DEPLOYMENT

Book Recommender can be accessed through this link:

[Flow when using the recommendation system]



Share

Book Recommender

[illegible]

Select model: Simple Recommender

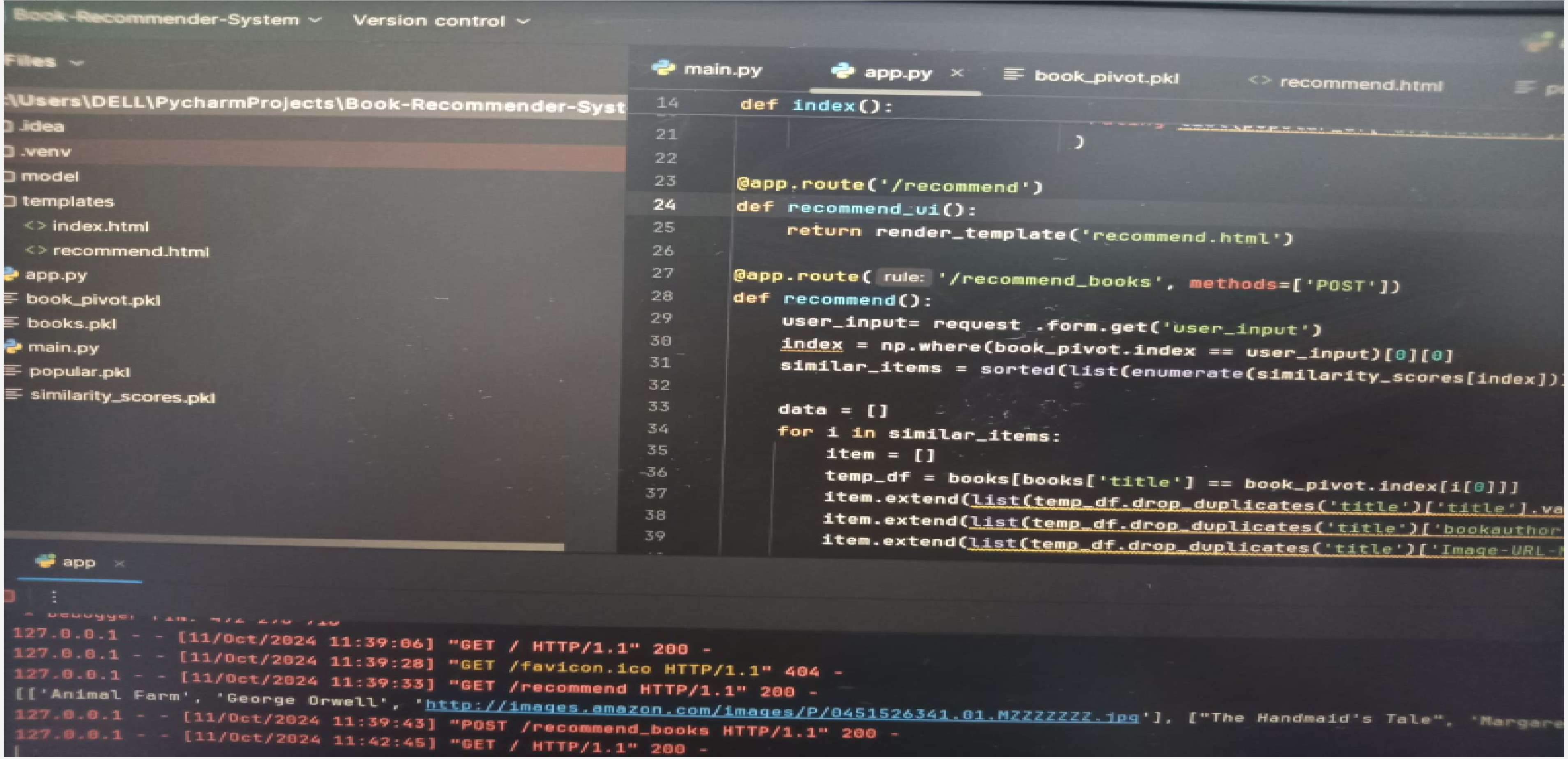
Number of books: 5

Recommend

Made with Streamlit

Manage app

Pycharm code

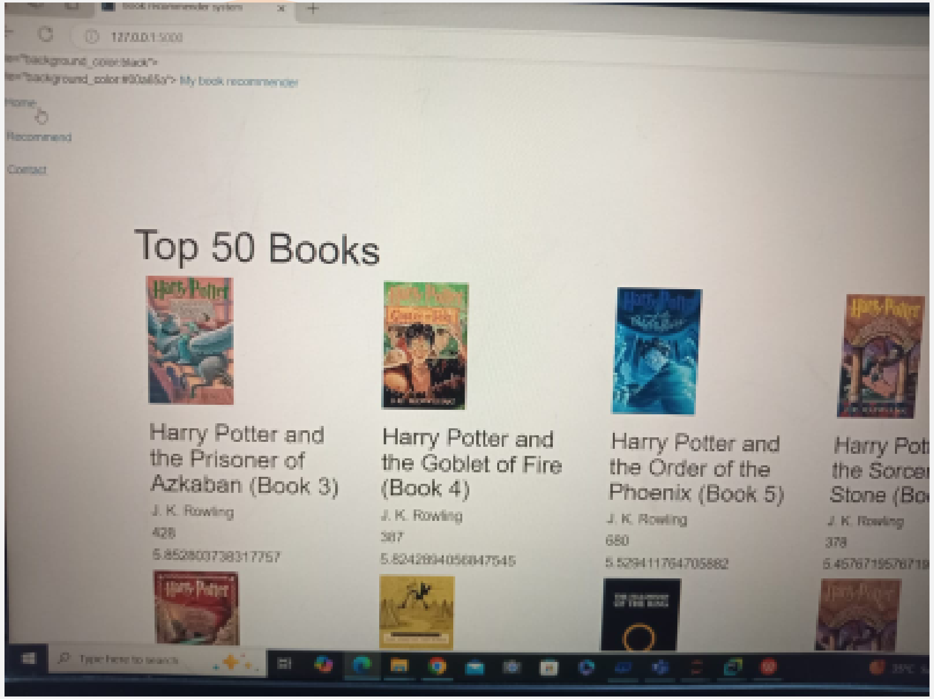


```python
def index():
            )

@app.route('/recommend')
def recommend_ui():
    return render_template('recommend.html')

@app.route( rule: '/recommend_books', methods=['POST'])
def recommend():
    user_input= request .form.get('user_input')
    index = np.where(book_pivot.index == user_input)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index]))

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['title'] == book_pivot.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('title')['title'].va
        item.extend(list(temp_df.drop_duplicates('title')['bookauthor
        item.extend(list(temp_df.drop_duplicates('title')['Image-URL-
```

```
127.0.0.1 - - [11/Oct/2024 11:39:06] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Oct/2024 11:39:28] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Oct/2024 11:39:33] "GET /recommend HTTP/1.1" 200 -
[['Animal Farm', 'George Orwell', 'http://images.amazon.com/images/P/0451526341.01.MZZZZZZZ.jpg'], ["The Handmaid's Tale", 'Margare
127.0.0.1 - - [11/Oct/2024 11:39:43] "POST /recommend_books HTTP/1.1" 200 -
127.0.0.1 - - [11/Oct/2024 11:42:45] "GET / HTTP/1.1" 200 -
```

Output

Top 50 Books



Book recommender system

127.0.0.1:5000

le="background_color:black">

le="background_color:#00a65a"> My book recommender

Home

Recommend

Contact

Top 50 Books

Harry Potter and the Prisoner of Azkaban (Book 3)
J. K. Rowling
428
5.852803738317757

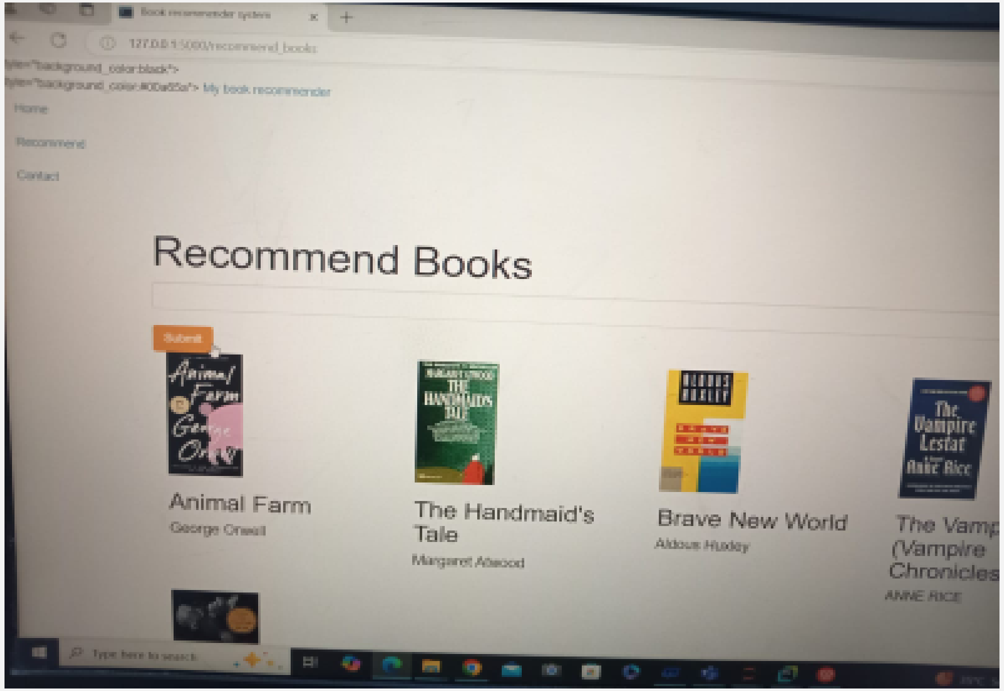
Harry Potter and the Goblet of Fire (Book 4)
J. K. Rowling
387
5.824289405684754

Harry Potter and the Order of the Phoenix (Book 5)
J. K. Rowling
680
5.529411764705882

Harry Pot
the Sorcer
Stone (Bo
J. K. Rowling
378
5.45767195767

Type here to search

Recommended Books



Book recommender system

127.0.0.1:5000/recommend_books

yle="background_color:black">

yle="background_color:#00a65a"> My book recommender

Home

Recommend

Contact

Recommend Books

Submit

Animal Farm
George Orwell

The Handmaid's Tale
Margaret Atwood

Brave New World
Aldous Huxley

The Vamp
(Vampire
Chronicles
ANNE RICE

Type here to search

SAMPLE RESULT USING SIMPLE RECOMMENDER

Harry Potter and the Deathly Hallows (Harry Potter #7)

Book Title	average_rating	rating_count	weighted_rating
Harry Potter and the Deathly Hallows (Harry Potter #7)	4.61	1746574	4.56
Harry Potter and the Half-Blood Prince (Harry Potter #2)	4.54	1678823	4.49
Harry Potter and the Prisoner of Azkaban (Harry Potter #3)	4.53	1832823	4.49
Harry Potter and the Goblet of Fire (Harry Potter #4)	4.53	1753043	4.48
Harry Potter and the Sorcerer's Stone (Harry Potter #1)	4.44	4602479	4.43

SAMPLE RESULT USING CONTENT BASED FILTERING

for user that input books ‘1984’:

Title	Authors	Notes
Animal Farm / 1984	George Orwell	same author, same genre (fiction, science fiction, classics)
We	Yevgeny Zamyatin	same genre (fiction, science fiction, classics)
Homage to Catalonia	George Orwell	same author
1Q84 #1-2 (1Q84, #1-2)	Haruki Murakami	similar title, same genre(fiction, science fiction), has word 1984 in description
The Far Side Gallery	Gary Larson	same genre(fiction), has word 1984 in description

🟨 Has a high rating but not popular

🟦 Is popular but has a lower rating

🟧 Is popular and has a high rating

SAMPLE RESULT USING COLLABORATIVE FILTERING

for user 12874 (user that gave most ratings)

Normal Predictor	KNN	SVD	SVD++	SVD Tuned
🟧 The Hunger Games (The Hunger Games, #1)	🟧 The Help	🟧 The Help	🟧 The Help	🟧 The Help
🟧 The Fault in Our Stars	🟧 The Girl Who Played with Fire (Millennium, #2)	🟧 Ready Player One	🟧 World Without End (The Kingsbridge Series, #2)	🟧 The Nightingale
🟦 Lord of the Flies	🟧 Tuesdays with Morrie	🟧 Wonder	🟧 I Am Pilgrim (Pilgrim, #1)	🟧 Harry Potter Collection (Harry Potter, #1-6)
🟦 Life of Pi	🟦 The Five People You Meet in Heaven	🟦 Go Ask Alice	🟧 The Holy Bible: English Standard Version	🟧 The Indispensable Calvin and Hobbes
🟧 Water for Elephants	🟧 The Girl Who Kicked the Hornet's Nest (Millennium, #3)	🟧 Morning Star (Red Rising, #3)	🟧 [illegible]	🟨 The Divan

🟨 Has a high rating but not popular

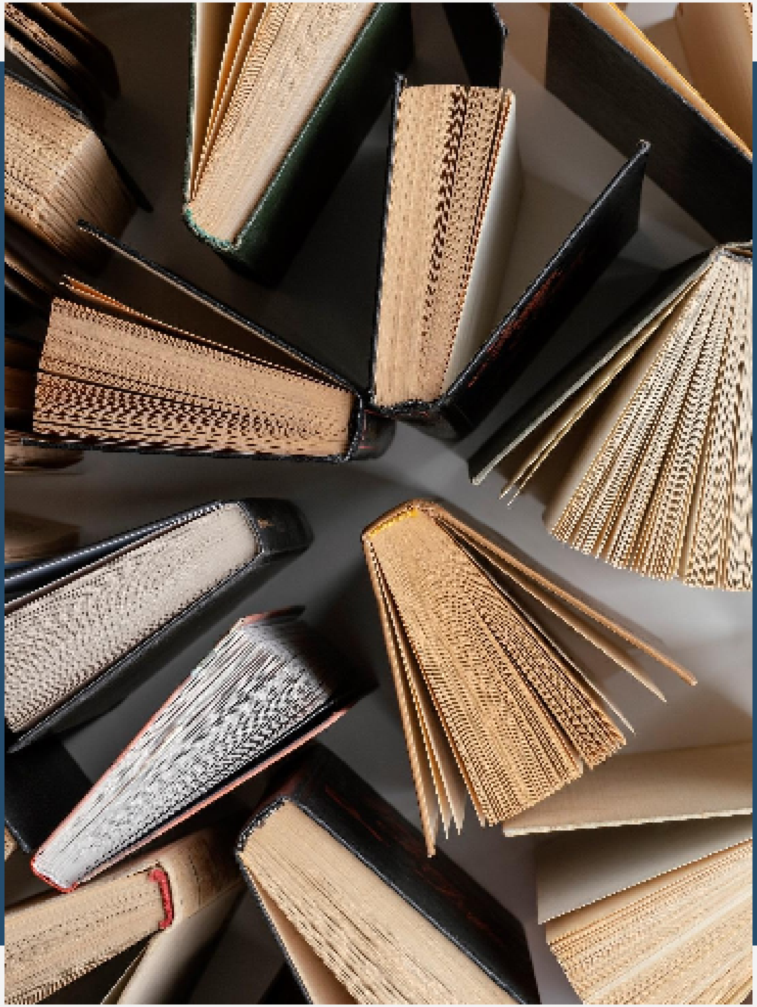
🟦 Is popular but has a lower rating

🟧 Is popular and has a high rating

Strength and Weakness of Each Model

METHOD	PROS	CONS
Simple Recommender	● suitable for new user ● relatively easy	● does not provide user-specific recommendations ● result is obvious
Content Based Filtering	● personalized, recommend based on user interest ● doesn’t need data about other users	● cannot provide recommendations across genres (not diverse) ● result rely on the what is written in the content
Collaborative Filtering	● more personalized ● more diverse	● cold-start problem when there is new user/new item ● need data about other users

Conclusion and Recommendation



Conclusion and Recommendation

1. We need to understand when to use each model.
   - ○ When we want to recommend a completly new user, we can use **Simple Recommender**.
   - ○ If the user want personalized recommendation, they can provide one book they liked. By applying a **Content Based Filtering**, instead of having to rate 30 books to start the recommendation engine, users can just pick only one book for Goodreads to provide good recommendations. We must pay attention to what content is chosen, whether only the author, the description etc.
   - ○ When they want even more diverse recommendation, we can use **Collaborative Filtering.** Here, we need other users data so that we can make the recommendation.

1. In this method, we utilize cosine similarity for our recommendation system on Goodreads. This approach is advantageous due to its ability to measure the similarity between user preferences and book attributes, which can lead to more personalized recommendations. Unlike RMSE, which focuses on prediction accuracy, cosine similarity provides insights into the relationships between items based on user interactions, making it suitable for a platform like Goodreads where user tastes vary widely.
2. While a cosine similarity value can indicate the strength of recommendations, it’s essential to complement this metric with user feedback to ensure that the suggestions align with individual preferences.
3. Additionally, integrating data from Amazon Kindle could enhance this system by tracking which books are purchased after being recommended through Goodreads, allowing for further refinement of the algorithm.

Thank you