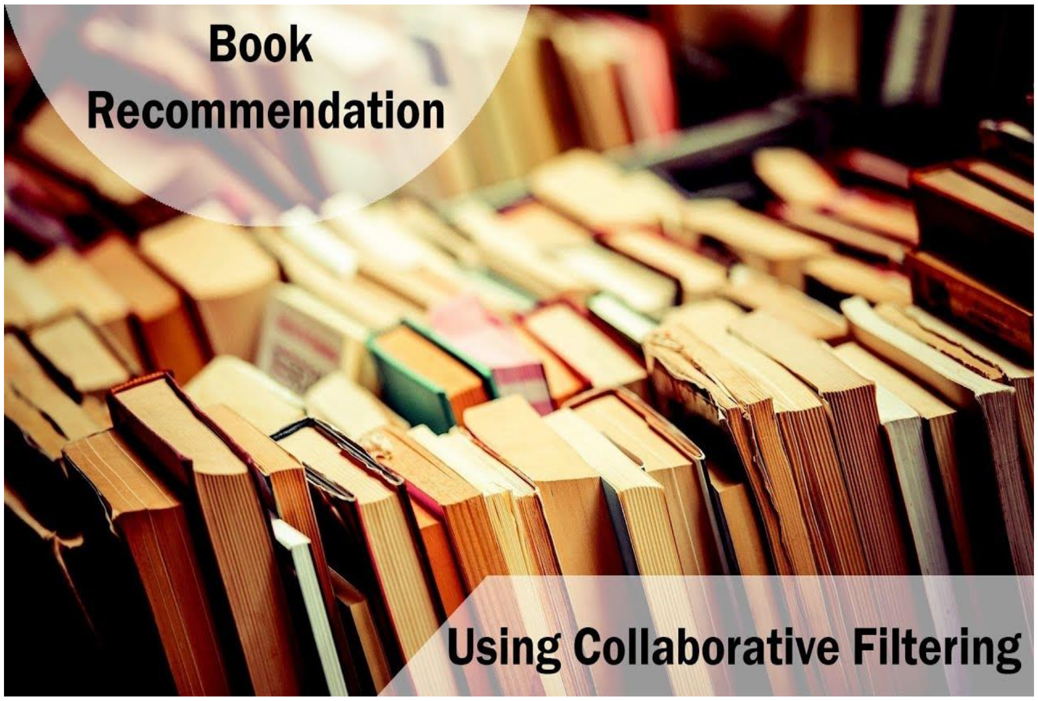
Feynn Labs Internship

Book Recommender system using collaborative Filtering.

By -Harshitha Immaneni

Date: 01/07/2023



Abstract:

This report presents a collaborative filtering-based book recommendation system designed to assist users in discovering books tailored to their preferences. The system leverages user-item interactions and similarities among users to generate personalized recommendations. The scope of the project includes considering user demographics, past reading history, genre preferences, and book ratings to provide accurate suggestions. The objective is to enhance the user experience by streamlining book discovery, increasing user satisfaction, and promoting diverse reading choices. The system aims to optimize recommendation accuracy by continuously learning from user feedback and behavior. The findings highlight the effectiveness of collaborative filtering in delivering personalized book recommendations, contributing to the field of information retrieval, and improving the reading experience for users.

In this Project, a book recommendation is building system utilizing collaborative filtering with a model-based approach. The system employs the K-Nearest Neighbors (KNN) model to analyze user-item interactions and generate personalized recommendations. By leveraging the KNN algorithm, the system identifies similar users based on their preferences and recommends books accordingly. The project's objective is to enhance the user experience by providing accurate and relevant book recommendations. The findings show the KNN model's effectiveness in delivering personalized suggestions and contributing to collaborative filtering-based book recommendation systems.

Introduction :

Recommendation system uses Algorithms to recommend items to user's based on the similarity between Customers. They're used by various large name companies like Google, Instagram, Spotify, Amazon, Reddit, Netflix etc. often to increase engagement with users and the platform. For example, Spotify would recommend songs similar to the ones you've repeatedly listened to or liked so that you can continue using their platform to listen to music. Amazon uses recommendations to suggest products to various users based on the data they have collected for that user.

There are many ways to build recommender systems, some use algorithmic and formulaic approaches like Page Rank while others use more modelling centric approaches like collaborative filtering, content based, link prediction, etc. All these approaches can vary in complexity, but complexity does not translate to "good" performance. Often simple solutions and implementations yield the strongest results. For example, large companies like Reddit, Hacker News and Google have used simple formulaic implementations of recommendation engines to promote content on their platform.

Book recommendation systems simplify the discovery of personalized books, saving time and effort. They introduce new authors, genres, and perspectives, promoting diversity in reading choices. These systems foster a sense of community, allowing users to share recommendations and engage in literary discussions.

Types of Book Recommendation System:

1. Collaborative Filtering Systems:

Collaborative filtering is the process of predicting the interests of a user by identifying preferences and information from many users. This is done by filtering data for information or patterns using techniques involving collaboration among multiple agents, data sources, etc. The underlying intuition behind collaborative filtering is that if users A and B have similar taste in a product, then A and B are likely to have similar taste in other products as well.

There are two common types of approaches in collaborative filtering, memory based and model based approach.

1. Memory based approaches — also often referred to as neighborhood collaborative filtering. Essentially, ratings of user-item combinations are predicted on the basis of their neighborhoods. This can be further split into user based collaborative filtering and item based collaborative filtering. User based essentially means that like minded users are going to yield strong and similar recommendations. Item based collaborative filtering recommends items based on the similarity between items calculated using user ratings of those items.
2. Model based approaches — are predictive models using machine learning. Features associated to the dataset are parameterized as inputs of the model to try to solve an optimization related problem. Model based approaches include using things like decision trees, rule based approaches, latent factor models etc.

2. <u>**Content Based Systems:**</u>

Content based systems generate recommendations based on the users preferences and profile. They try to match users to items which they've liked previously. The level of similarity between items is generally established based on attributes of items liked by the user. Unlike most collaborative filtering models which leverage ratings between target user and other users, content based models focus on the ratings provided by the target user themselves. In essence, the content based approach leverages different sources of data to generate recommendations.

3. Item level data source — you need a strong source of data associated to the attributes of the item. For our scenario, we have things like book price, num_pages, published_year, etc. The more information you know regarding the item, the more beneficial it will be for your system.
4. User level data source — you need some sort of user feedback based on the item you're providing recommendations for. This level of feedback can be either implicit or explicit. In our sample data, we're working with user ratings of books they've read. The more user feedback you can track, the more beneficial it will be for your system.

Problem statement:

The problem addressed in this project is the lack of an accurate and efficient book recommendation system using a model-based approach of collaborative filtering. Existing systems often provide generic or inaccurate recommendations, resulting in a suboptimal user experience. Furthermore, the challenge lies in effectively preprocessing and merging the books, users, and ratings datasets to create a unified dataset for modeling. The problem statement focuses on developing a recommendation system that utilizes the K-Nearest Neighbors (KNN) algorithm with k-neighbors and distance-based calculations. The goal is to improve the accuracy and relevance of book suggestions, enhancing the book discovery process for users.

Objective:

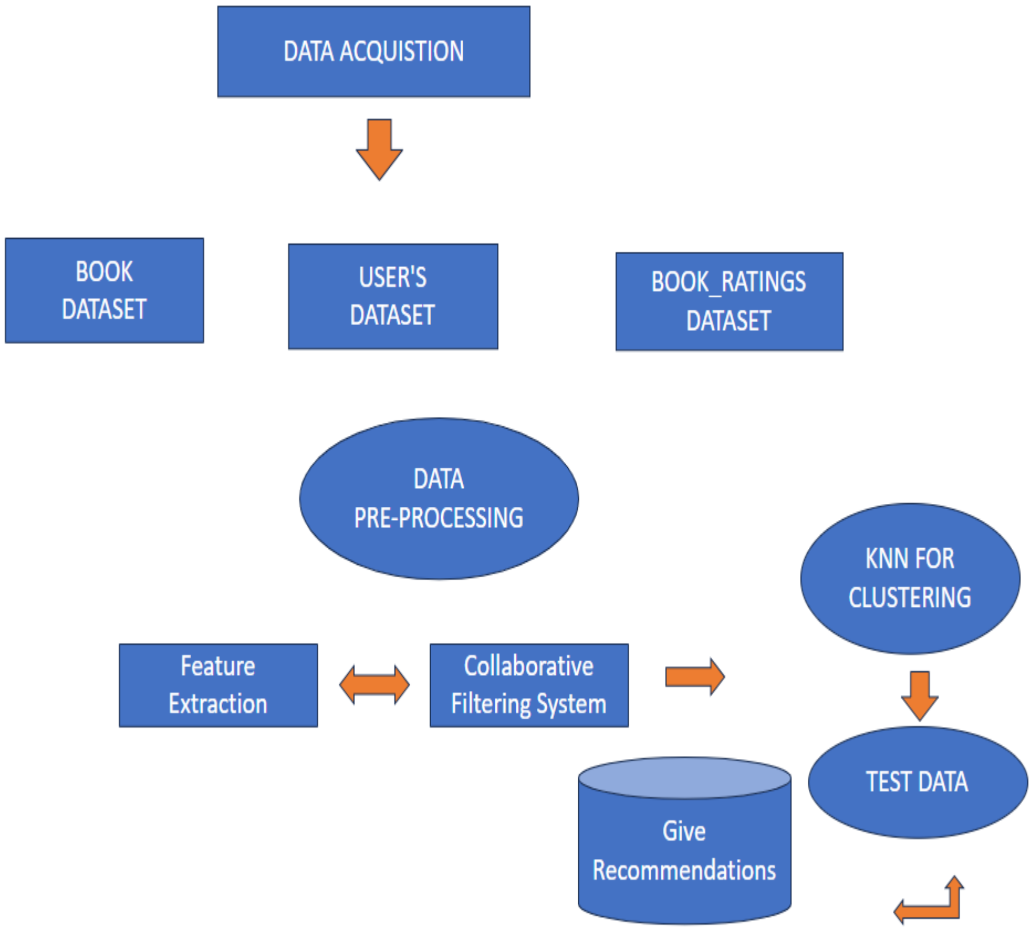
The project's objective is to develop and evaluate a book recommendation system using a model-based collaborative filtering approach. The system aims to provide accurate and personalized book recommendations to users based on their preferences and interests. The project seeks to leverage the K-Nearest Neighbors (KNN) algorithm k- neighbors and distance-based calculations to enhance the accuracy and relevance of the recommendations. By achieving this objective, the project aims to improve the user experience by simplifying the book discovery process and increasing user satisfaction.

Customer needs of books recommender system:

A book recommendation system is a type of recommendation system where we have to recommend similar books to the reader based on his interest. The books recommendation system is used by online websites which provide ebooks like google play books, open library, good Read’s, etc.

Customers have diverse needs when it comes to a book recommender system. They expect personalized recommendations based on their preferences, reading habits, and interests. Accuracy and relevance are crucial, as customers want recommendations that match their tastes and increase the likelihood of enjoyment. Exploration and diversity are valued, as customers appreciate discovering books from various genres, authors, and perspectives. A user-friendly interface that allows easy navigation, searching, and access to personalized suggestions is essential. Seamless integration with their preferred platforms or devices enhances convenience. Trust and privacy are paramount, with customers seeking assurance that their data is handled securely. Timeliness and freshness are valued, as customers want recommendations that consider recent releases and trends. By meeting these needs, a book recommender system can enhance the user experience, promote discovery, and foster trust and loyalty.

METHODOLOGY



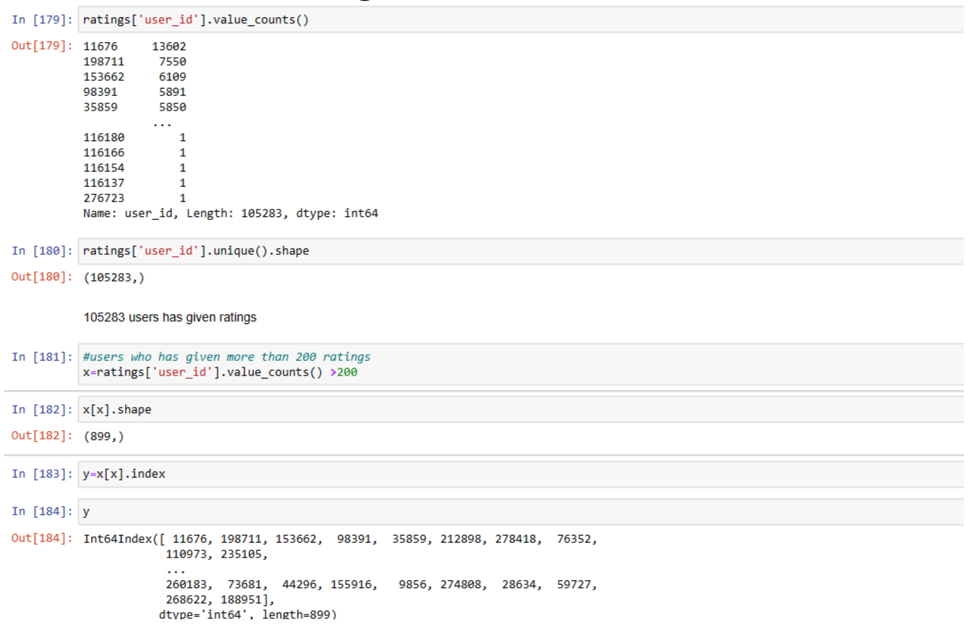
Exploratory Data Analysis:

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process, including building recommendation systems. While EDA primarily focuses on understanding the data, identifying patterns, and gaining insights, it can indirectly inform feature extraction and data preparation for recommendation systems.

During EDA, I have explored various aspects of the data such as the distribution of ratings, user-item interactions, item popularity, user preferences, and other relevant information. This exploration helped me to make informed decisions on feature selection and engineering for your recommendation system.

Insights:

1. Extract users and ratings of more than 200

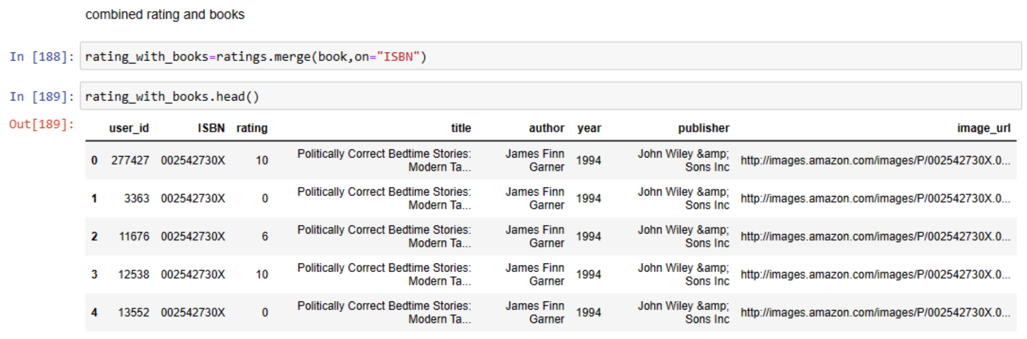


```
In [179]: ratings['user_id'].value_counts()

Out[179]: 11676     13602
          198711     7550
          153662     6109
          98391      5891
          35859      5850
                    ...
          116180        1
          116166        1
          116154        1
          116137        1
          276723        1
          Name: user_id, Length: 105283, dtype: int64

In [180]: ratings['user_id'].unique().shape

Out[180]: (105283,)
```

105283 users has given ratings

```
In [181]: #users who has given more than 200 ratings
          x=ratings['user_id'].value_counts() >200

In [182]: x[x].shape

Out[182]: (899,)

In [183]: y=x[x].index

In [184]: y

Out[184]: Int64Index([ 11676, 198711, 153662,  98391,  35859, 212898, 278418,  76352,
                      110973, 235105,
                      ...
                      260183,  73681,  44296, 155916,   9856, 274808,  28634,  59727,
                      268622, 188951],
                     dtype='int64', length=899)
```

2. Merge ratings with books

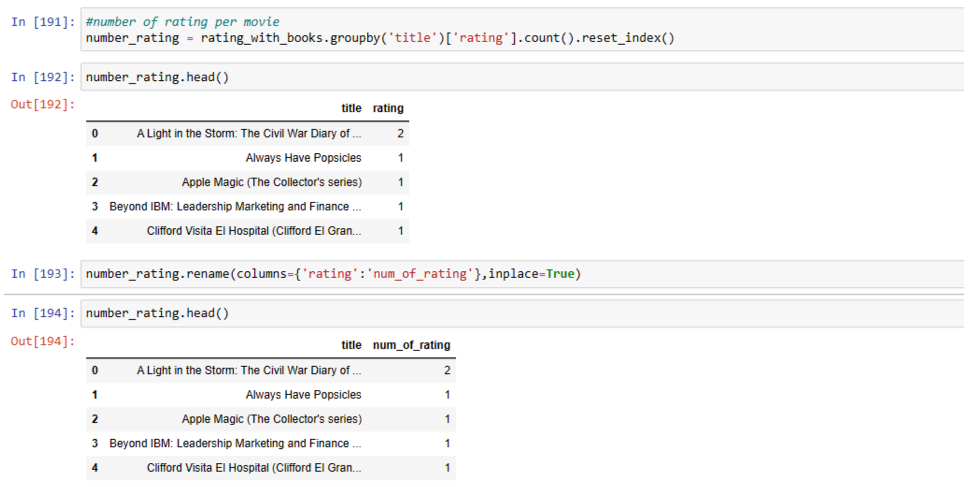


combined rating and books

```
In [188]: rating_with_books=ratings.merge(book,on="ISBN")

In [189]: rating_with_books.head()

Out[189]:
```

	user_id	ISBN	rating	title	author	year	publisher	image_url
0	277427	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley &amp; Sons Inc	http://images.amazon.com/images/P/002542730X.0...
1	3363	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley &amp; Sons Inc	http://images.amazon.com/images/P/002542730X.0...
2	11676	002542730X	6	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley &amp; Sons Inc	http://images.amazon.com/images/P/002542730X.0...
3	12538	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley &amp; Sons Inc	http://images.amazon.com/images/P/002542730X.0...
4	13552	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley &amp; Sons Inc	http://images.amazon.com/images/P/002542730X.0...

3. Number of ratings per movie



```
In [191]: #number of rating per movie
          number_rating = rating_with_books.groupby('title')['rating'].count().reset_index()
```

```
In [192]: number_rating.head()
Out[192]:
```

	title	rating
0	A Light in the Storm: The Civil War Diary of ...	2
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

```
In [193]: number_rating.rename(columns={'rating':'num_of_rating'},inplace=True)
```

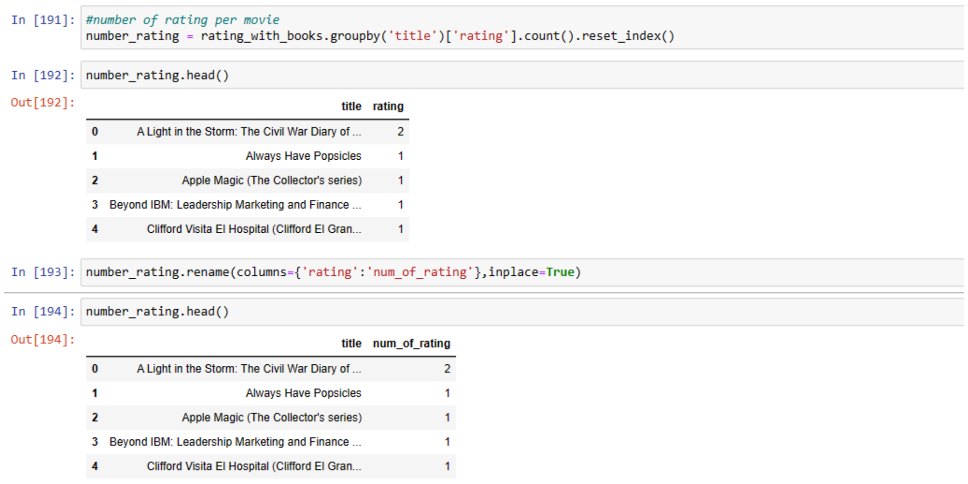
```
In [194]: number_rating.head()
Out[194]:
```

	title	num_of_rating
0	A Light in the Storm: The Civil War Diary of ...	2
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

The number of ratings per Books provides valuable insights into movie popularity, user preferences, recommendation diversity, and data sparsity, which can be utilized in various recommendation strategies and algorithms to improve the effectiveness and quality of recommendations.

In collaborative filtering-based recommendation systems, where user-item interactions are used to make recommendations, the number of ratings per Books can help address data sparsity issues. Books with a higher number of ratings tend to have denser user-item interaction matrices, making it easier to find similar items and improve the accuracy of collaborative filtering algorithms.

4. Extract books that have received more than 50 ratings



```
In [191]: #number of rating per movie
          number_rating = rating_with_books.groupby('title')['rating'].count().reset_index()
```

```
In [192]: number_rating.head()
Out[192]:
```

	title	rating
0	A Light in the Storm: The Civil War Diary of ...	2
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

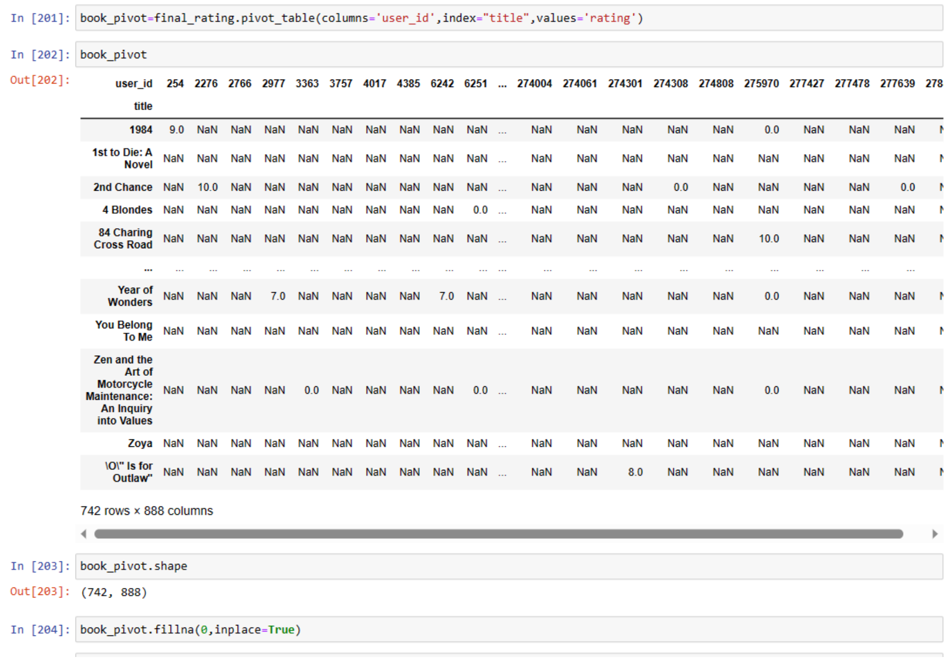
```
In [193]: number_rating.rename(columns={'rating':'num_of_rating'},inplace=True)
```

```
In [194]: number_rating.head()
Out[194]:
```

	title	num_of_rating
0	A Light in the Storm: The Civil War Diary of ...	2
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Beyond IBM: Leadership Marketing and Finance ...	1
4	Clifford Visita El Hospital (Clifford El Gran...	1

5. Create Pivot Table

created a pivot table where columns will be user ids, the index will be book title and the values are ratings. And the user id who has not rated any book will have value as NAN so impute it with zero.



```
In [201]: book_pivot=final_rating.pivot_table(columns='user_id',index="title",values='rating')

In [202]: book_pivot
Out[202]:
```

user_id	254	2276	2766	2977	3363	3757	4017	4385	6242	6251	...	274004	274061	274301	274308	274808	275970	277427	277478	277639	278
title																					
1984	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	N
1st to Die: A Novel	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2nd Chance	NaN	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	0.0	N
4 Blondes	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
84 Charing Cross Road	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	10.0	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Year of Wonders	NaN	NaN	NaN	7.0	NaN	NaN	NaN	NaN	7.0	NaN	...	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	N
You Belong To Me	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	N
Zoya	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
\O\" Is for Outlaw"	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	8.0	NaN	NaN	NaN	NaN	NaN	NaN	N

742 rows × 888 columns

```
In [203]: book_pivot.shape
Out[203]: (742, 888)

In [204]: book_pivot.fillna(0,inplace=True)
```

Modeling

We have prepared our dataset for modeling. We used the nearest neighbor's algorithm, the same as K nearest used for clustering based on Euclidian distance.

But here in the pivot table, we have lots of zero values and on clustering, this computing power will increase to calculate the distance of zero values so we will convert the pivot table to the sparse matrix and then feed it to the model.

Sparse Matrix

A sparse matrix is a special case of a matrix in which the number of zero elements is much higher than the number of non-zero elements. As a rule of thumb, if 2/3 of the total elements in a matrix are zeros, it can be called a sparse matrix. Using a sparse matrix representation — where only the non-zero values are stored — the space used for representing data and the time for scanning the matrix are reduced significantly.

In a book recommendation system, a sparse matrix can be constructed where each row represents a user, each column represents a book, and the cells contain information about the interaction between users and books. For example, a cell value of 1 might indicate that a user has purchased a book, while a value of 0 might indicate no interaction.

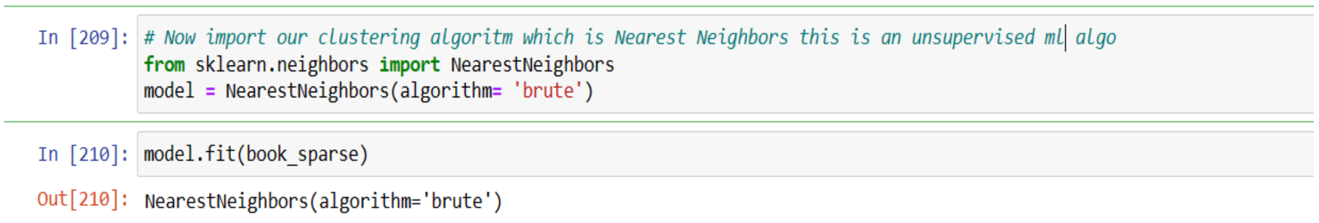
Here's a simplified example of a sparse matrix for book recommendations:

	Book1	**Book2**	**Book3**	**Book4**
User1	**1**	**0**	**1**	**0**
User2	**0**	**1**	**0**	**1**
User3	**1**	**1**	**1**	**0**

In this matrix, User1 has purchased Book1 and Book3, User2 has purchased Book2 and Book4, and User3 has purchased all three books.

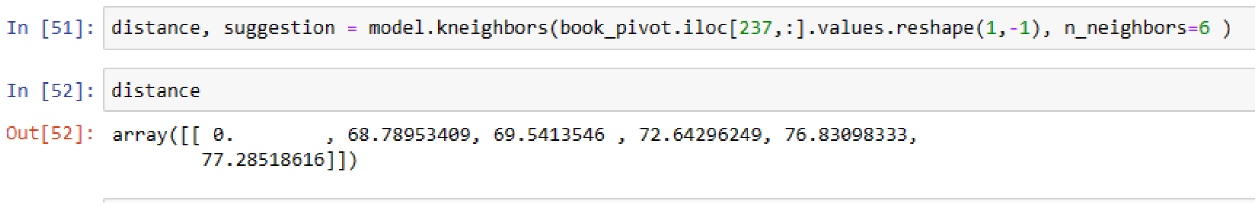
Sparse matrices allow recommendation systems to efficiently store and process user-item interaction data. Various recommendation algorithms, such as collaborative filtering or matrix factorization, can then analyze this sparse matrix to identify patterns, similarities, or relationships between users and books. These algorithms can use the information in the matrix to make personalized book recommendations based on the preferences and behavior of similar users or items.

Implementation: We need to specify an algorithm which is brute means find the distance of every point to every other point.



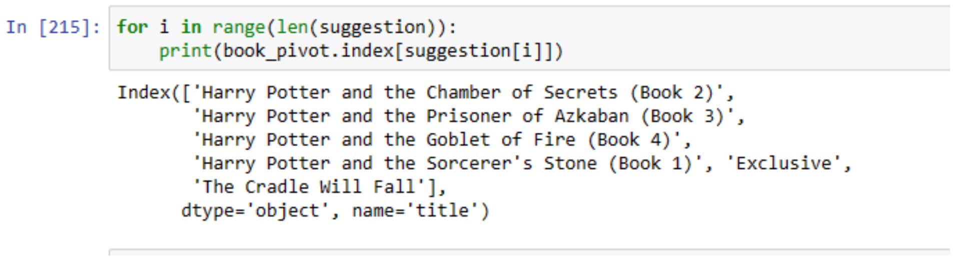
```
In [209]: # Now import our clustering algoritm which is Nearest Neighbors this is an unsupervised ml algo
          from sklearn.neighbors import NearestNeighbors
          model = NearestNeighbors(algorithm= 'brute')

In [210]: model.fit(book_sparse)

Out[210]: NearestNeighbors(algorithm='brute')
```

Let’s make a prediction and see whether it is suggesting books or not. we will find the nearest neighbors to the input book id and after that, we will print the top 5 books which are closer to those books. It will provide us distance and book id at that distance. let us pass harry potter which is at 237 indexes.



```
In [51]: distance, suggestion = model.kneighbors(book_pivot.iloc[237,:].values.reshape(1,-1), n_neighbors=6 )

In [52]: distance

Out[52]: array([[ 0.        , 68.78953409, 69.5413546 , 72.64296249, 76.83098333,
                 77.28518616]])
```

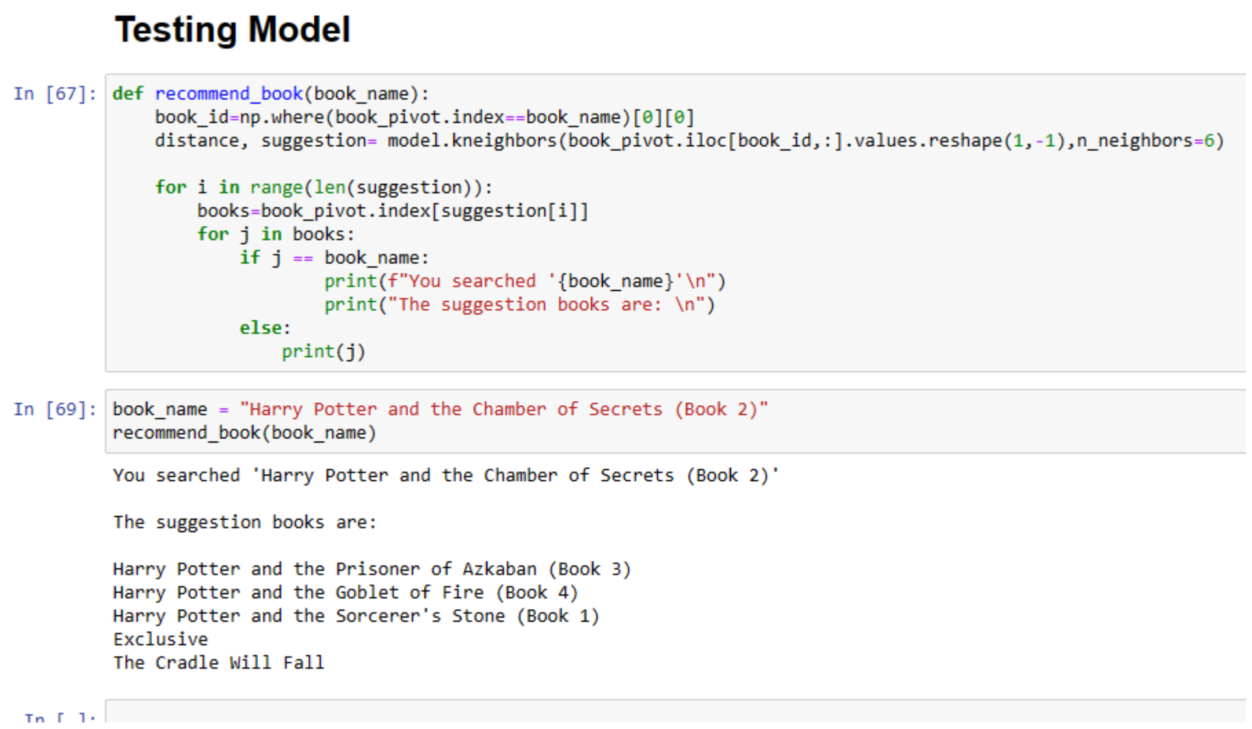
let us print all the suggested books,



```
In [215]: for i in range(len(suggestion)):
              print(book_pivot.index[suggestion[i]])

          Index(['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'],
                dtype='object', name='title')
```

hence, we have successfully built a book recommendation system.

The purpose of this loop is to print the book names for each suggestion, providing recommendations to the user. However, it is important to note that the code does not distinguish between the input book and the recommended books. All book names in the suggestions are treated equally and printed without any distinction.



Testing Model

```
In [67]: 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:
                     if j == book_name:
                             print(f"You searched '{book_name}'\n")
                             print("The suggestion books are: \n")
                     else:
                         print(j)
```

```
In [69]: book_name = "Harry Potter and the Chamber of Secrets (Book 2)"
         recommend_book(book_name)
```

```
You searched 'Harry Potter and the Chamber of Secrets (Book 2)'

The suggestion books are:

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
```

```
In [ ]:
```

The code snippet presented is a function called **recommend_book()** that takes a book name as input and provides recommendations of similar books based on a collaborative filtering model.

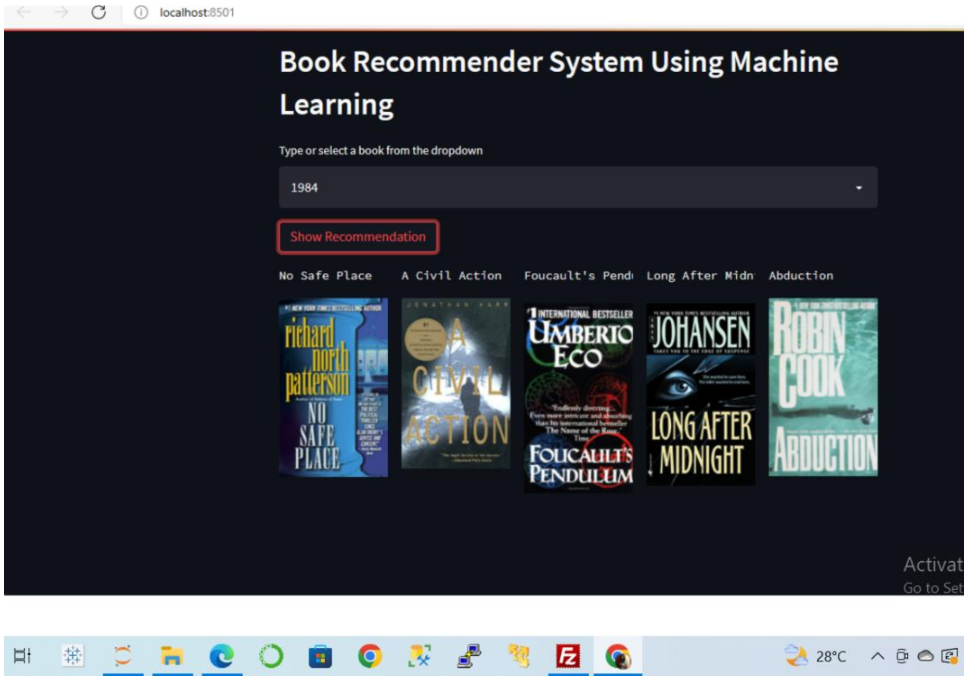
First, the code utilizes the NumPy library to identify the index of the input book within the book pivot table. This table, referred to as **`book_pivot`**, likely contains a sparse matrix representation of user-book interactions.

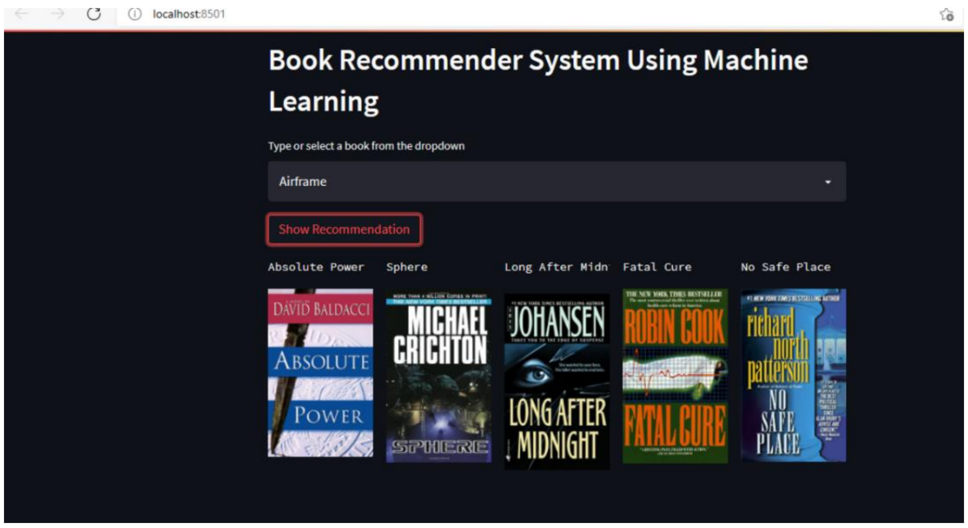
Next, the **`model.kneighbors()`** function is called with the input book's data to find the nearest neighbors based on the collaborative filtering model. The **`n_neighbors`** parameter is set to 6, indicating that six similar books will be returned as recommendations.

The function then proceeds to iterate through the suggestions and extract the book names from the book pivot index. If the current book being iterated matches the input book name, a message is printed to indicate that the user has searched for that specific book. Otherwise, the book name is printed as a recommendation.

UI design

This is a stream lit web application that can recommend various kinds of similar books based on an user interest. here is a demo,





CONCLUSION:

In this project, we have recommended books for a user using the model trained using KNN which is a Collaborative Filtering Technique using Model Based approach. The Dataset is available on Kaggle.

Utilizing a sparse matrix is a practical approach when dealing with recommendation systems since the interaction data between users and books often results in a sparse representation. Sparse matrices can efficiently handle large datasets and reduce memory usage.

Employing KNN for clustering can be useful to identify similar items or users based on their characteristics or preferences. This clustering can help in generating relevant book suggestions based on the preferences of similar users.

Future Scope

- Incorporate more advanced recommendation algorithms: Explore advanced collaborative filtering algorithms such as matrix factorization techniques (e.g., singular value decomposition or factorization machines) or deep learning-based approaches (e.g., neural networks) to enhance the accuracy and performance of the recommendations.

- Integrate content-based filtering: Augment collaborative filtering with content-based filtering techniques, where book attributes such as genre, author, or plot summaries are considered alongside user preferences. This hybrid approach can provide more diverse and accurate recommendations.

- Incorporate implicit feedback: Take into account implicit user feedback, such as browsing history, time spent on each book page, or click-through rates. This additional information can help capture user preferences and improve the accuracy of the recommendations.

- Utilize contextual information: Consider incorporating contextual information like user location, time of day, or reading patterns to provide personalized recommendations that align with the user's current context.

- Implement a hybrid recommendation system: Combine multiple recommendation techniques, including collaborative filtering, content-based filtering, and popularity-based recommendations, to leverage the strengths of each approach and provide a diverse set of recommendations.

- Apply sentiment analysis: Integrate sentiment analysis techniques to analyze book reviews or social media sentiment about books. This can provide insights into user sentiment and help tailor recommendations based on positive or negative sentiment.

Reference:

https://www.analyticsvidhya.com/blog/2021/06/build-book-recommendation-system-unsupervised-learning-project/#:~:text=A%20book%20recommendation%20system%20is,library%2C%20good%20Read's%2C%20etc.

https://www.researchgate.net/publication/321753015_Model-based_approach_for_Collaborative_Filtering

https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed

Git-hub Link

https://github.com/Harshithaimmaneni/Feynn_Labs.git