CSE488: Term Project

Term Project Title: Book Recommender System

Task Completed by: Sofia Noor Rafa (2020-1-60-226)

Submitted to : Dr. Mohammad Rezwanul Huq

Fall 2023

Colab link: CSE488 by Sofia.ipynb

Websites deploy pop-ups like “Recommended for You” but what is the secret behind it? We want our systems nowadays to be as intelligent as possible to bring us something we will instantly like. But what enables the websites/systems to mine our interests? Simple, they map us to other profiles that share common interests. Collaborative filtering is a technique applied in recommender systems to leverage and map collective data about user interactions with items present in a system and their preferences regarding those items. Collaborative filtering has two types: user-user and item-item. The difference between the user-user collaborative filtering with the latter resides in mapping different users’ shared interactions to predict preferences. The latter maps items that are similar to a user’s previously preferred items, for example, websites often show the “You may also Like” pop-up where an array of items are presented based on the item of interest to a user. Research shows that UBCF(User based collaborative filtering) is more applicable than IBCF(Item based collaborative filtering) to recommend items to a user, but the benefits of IBCF extend to creating a more personalized user experience.

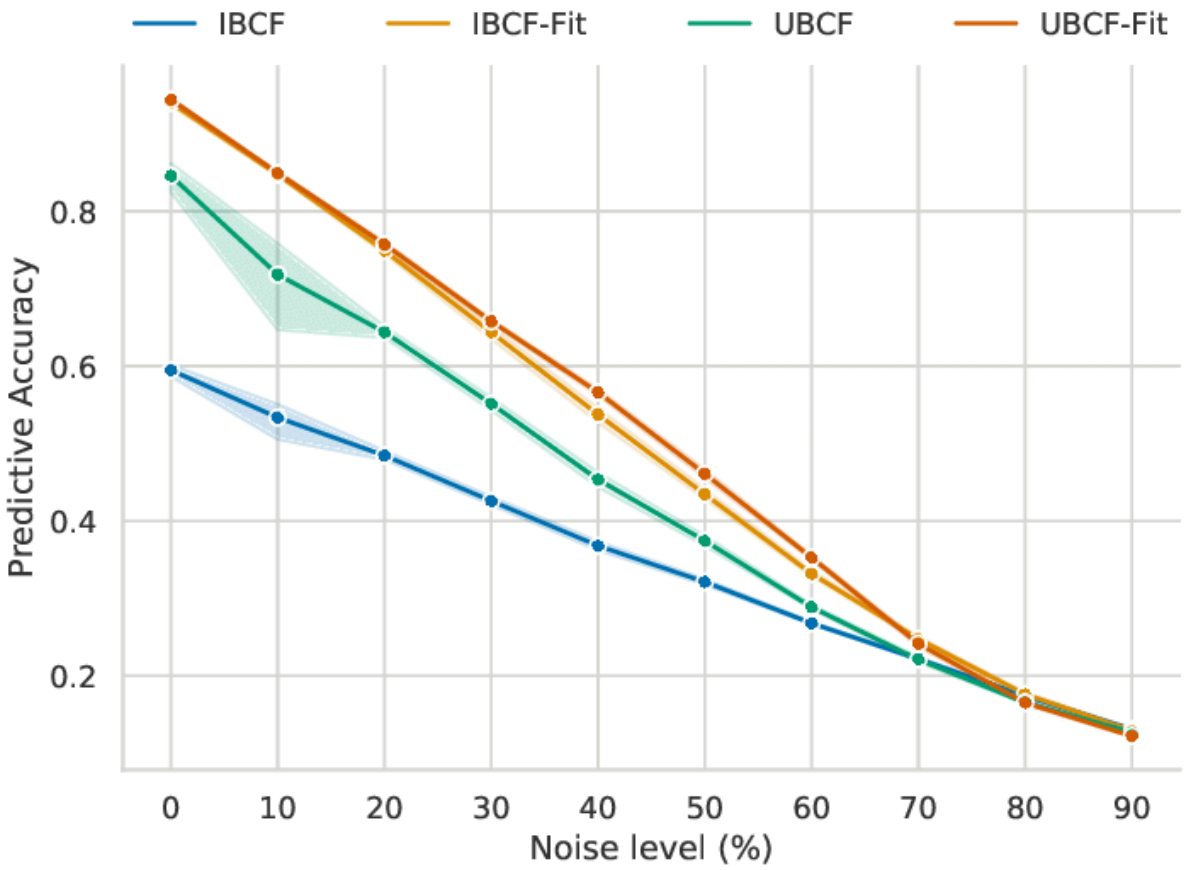
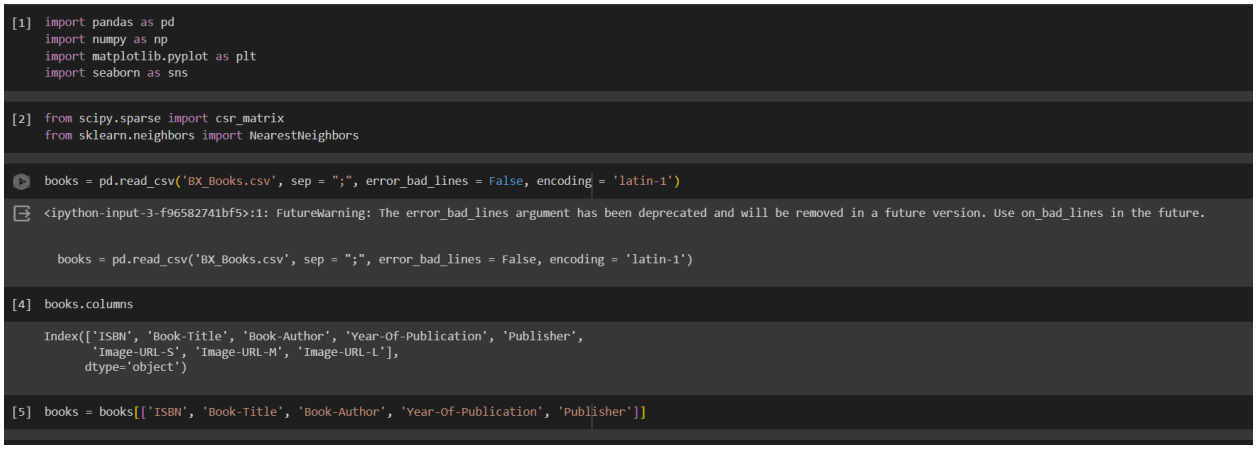


Image source: [Comparison of item-based collaborative filtering (IBCF) and user-based... | Download Scientific Diagram (researchgate.net)]

The goal of this project, therefore, is to predict user ratings and show top -k books to the users.

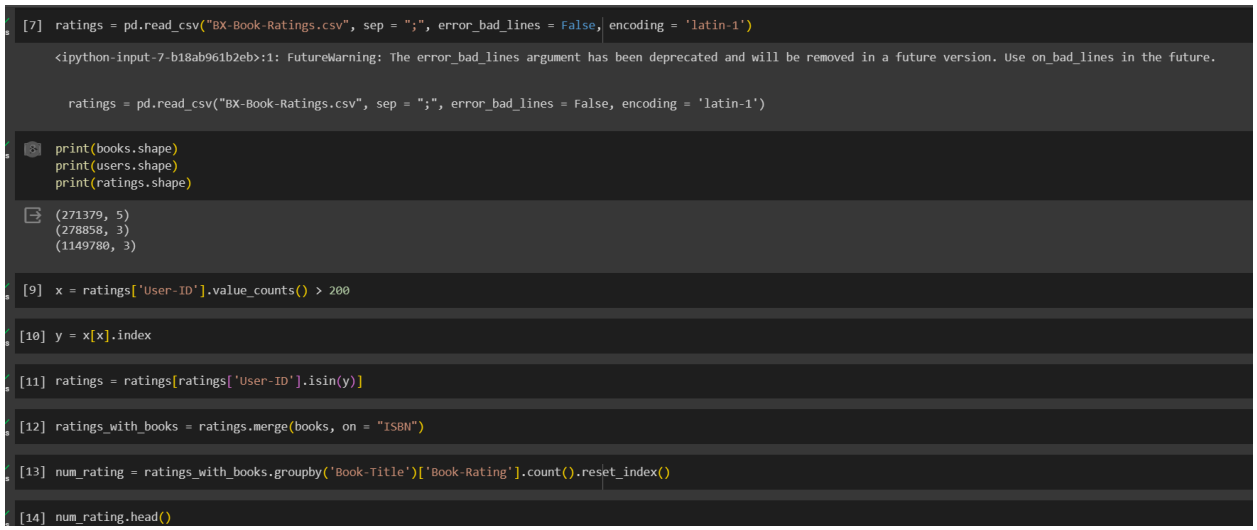
Steps of creating user-item matrix and recommending top-10 books for a user:

1. Three files, BX_Books, BX-Users, BX-Book-Ratings, are loaded, only the users who have rated more than 200 books are selected, a new dataframe is built by choosing only the rows where the users have rated more than 200 books.



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

[2] from scipy.sparse import csr_matrix
    from sklearn.neighbors import NearestNeighbors

    books = pd.read_csv('BX_Books.csv', sep = ";", error_bad_lines = False, encoding = 'latin-1')

    <ipython-input-3-f96582741bf5>:1: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_lines in the future.


      books = pd.read_csv('BX_Books.csv', sep = ";", error_bad_lines = False, encoding = 'latin-1')

[4] books.columns

    Index(['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
           'Image-URL-S', 'Image-URL-M', 'Image-URL-L'],
          dtype='object')

[5] books = books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher']]
```

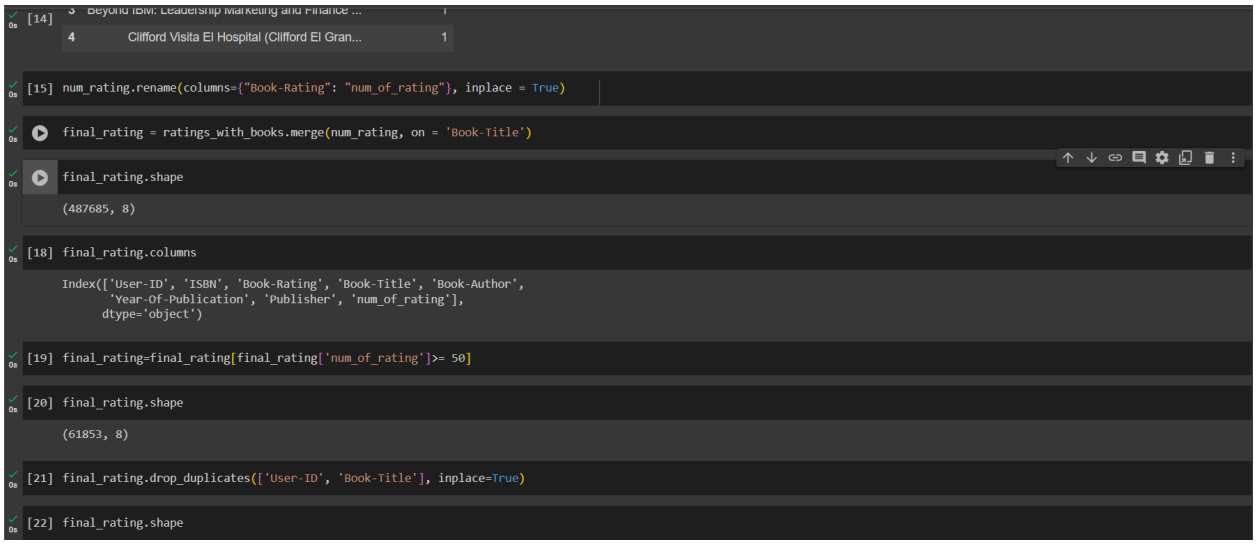
2. On the column ISBN, Books and Ratings data frames are merged.
3. Based on the Book_Title column, the newly created merged dataset is grouped, All the non-null values of each group are counted in the Book_Rating column. Total ratings received for each group is counted here.



```
[7] ratings = pd.read_csv("BX-Book-Ratings.csv", sep = ";", error_bad_lines = False, encoding = 'latin-1')

    <ipython-input-7-b18ab961b2eb>:1: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version. Use on_bad_lines in the future.


      ratings = pd.read_csv("BX-Book-Ratings.csv", sep = ";", error_bad_lines = False, encoding = 'latin-1')

    print(books.shape)
    print(users.shape)
    print(ratings.shape)

    (271379, 5)
    (278858, 3)
    (1149780, 3)

[9] x = ratings['User-ID'].value_counts() > 200

[10] y = x[x].index

[11] ratings = ratings[ratings['User-ID'].isin(y)]

[12] ratings_with_books = ratings.merge(books, on = "ISBN")

[13] num_rating = ratings_with_books.groupby('Book-Title')['Book-Rating'].count().reset_index()

[14] num_rating.head()
```

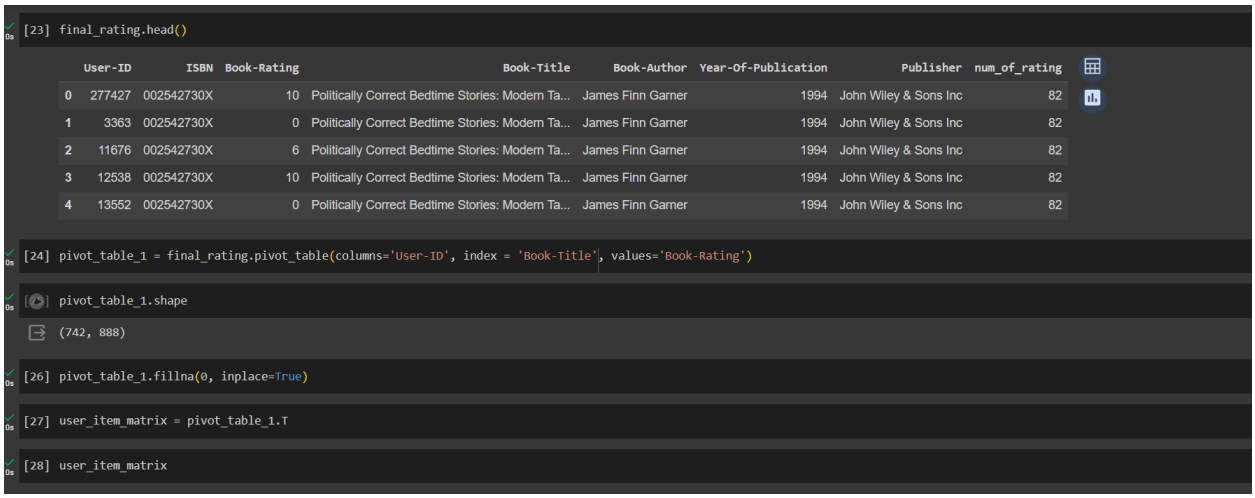
4. After counting all the values, the columns are put back onto the regular dataframe.

5. The num_rating data frame contains all the book_titles and their total rating counts.
6. The book_rating column converted onto the number_of_ratings column shows the total rating counts for all the books.



```
[14]  3  Beyond IBM: Leadership Marketing and Finance ...   1
      4  Clifford Visita El Hospital (Clifford El Gran...   1

[15] num_rating.rename(columns={"Book-Rating": "num_of_rating"}, inplace = True)

     final_rating = ratings_with_books.merge(num_rating, on = 'Book-Title')

     final_rating.shape

     (487685, 8)

[18] final_rating.columns

     Index(['User-ID', 'ISBN', 'Book-Rating', 'Book-Title', 'Book-Author',
            'Year-Of-Publication', 'Publisher', 'num_of_rating'],
           dtype='object')

[19] final_rating=final_rating[final_rating['num_of_rating']>= 50]

[20] final_rating.shape

     (61853, 8)

[21] final_rating.drop_duplicates(['User-ID', 'Book-Title'], inplace=True)

[22] final_rating.shape
```

7. Only the books with 50 ratings or more are kept with their unique combinations with the USER_ID column to keep only the unique user-book combinations.
8. Finally a user-item matrix is formed by transposing the existing one.



```
[23] final_rating.head()
```

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	num_of_rating
0	277427	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	82
1	3363	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	82
2	11676	002542730X	6	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	82
3	12538	002542730X	10	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	82
4	13552	002542730X	0	Politically Correct Bedtime Stories: Modern Ta...	James Finn Garner	1994	John Wiley & Sons Inc	82

```
[24] pivot_table_1 = final_rating.pivot_table(columns='User-ID', index = 'Book-Title', values='Book-Rating')

     pivot_table_1.shape

     (742, 888)

[26] pivot_table_1.fillna(0, inplace=True)

[27] user_item_matrix = pivot_table_1.T

[28] user_item_matrix
```

```
[29] book_sparse = csr_matrix(pivot_table_1)

[30] myModel = NearestNeighbors(algorithm='brute')
     myModel.fit(book_sparse)

     NearestNeighbors(algorithm='brute')
```

9. Pairwise cosine similarity is calculated using the user-item matrix. The similarity function returns top-10 books based on the similarity of other users.

```
        return top_ten_books


user_id = 277478
recommended_books = recommend_books_for_user(user_id)
recommended_books
```

```
['Intensity',
 'The Fellowship of the Ring (The Lord of the Rings, Part 1)',
 'Presumed Innocent',
 'Moonlight Becomes You',
 'Dawn (Cutler)',
 'The Chamber',
 'Two for the Dough',
 'Cradle and All',
 'Rebecca',
 "The Pilot's Wife : A Novel"]
```

10. A KNN model using the brute force approach takes a user-ID as input and shows top-10 books.

```
# Example: Get recommendations for User with ID 275970
user_id = 277478
recommended_books = recommend_books_for_user(user_id)
recommended_books
```

```
["The Pilot's Wife : A Novel",
 'The Prince of Tides',
 'The Red Tent (Bestselling Backlist)',
 'Welcome to the World, Baby Girl!',
 'Divine Secrets of the Ya-Ya Sisterhood: A Novel',
 'The Cider House Rules',
 "The Bonesetter's Daughter",
 'Heartbreaker',
 'The Alchemist: A Fable About Following Your Dream',
 '1st to Die: A Novel']
```

For predicting user's rating via ISBN:

1. To predict the rating, a new pivot table is formed using the previously existing dataframe that contains books with a minimum rating of 50.
2. Missing values are filled, this time by turning the ISBN into rows/indices, the user_id into columns, the matrix is transformed into an item_item_matrix.

```
[41] pivot_table_2 = final_rating.pivot_table(columns='User-ID', index='ISBN', values='Book-Rating')

[42] pivot_table_2.fillna(0, inplace=True)

pivot_table_2
```

User-ID	254	2276	2766	2977	3363	3757	4017	4385	6242	6251	...	274004	274061	274301	274308	274808	275970	277427	277478	277639	278418
ISBN																					
0001047973	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0006177379	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0020697406	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
002542730X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0
002542730x	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
B00009EF82	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3. The prediction function takes the user_id, isbn as input, finds out their location in the pivot_table_2, checks existing ratings.
4. The function finds out the nearest neighbors for a user, and predicts ratings based on the nearest neighbors. For similar users, if ratings exist, they are aggregated and their mean is calculated as the predicted rating.

```
# Example: Predict rating for user with ID 254 and ISBN '002542730X'
user_id = 277427
isbn = '002542730X'
predicted_rating = predict_book_rating(user_id, isbn)
print(predicted_rating)
```

```
User 277427 has already rated the book with ISBN 002542730X. Rating: 10.0
```