Book Recommendation System

Project team

Team leader	20p1542	Omar Ashraf Mostafa Samy	20p1542@eng.asu.edu.eg
Team member	1901075	Micheal joseph Adeeb	1901075@eng.asu.edu.eg
Team member	17p6030	Omar Ashraf Abd - elkhalik	17p6030@eng.asu.edu.eg

Introduction and Background

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys.
In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As a proof of the importance of recommender systems, we can mention that, a few years ago, Netflix organised a challenges (the “Netflix prize”) where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.

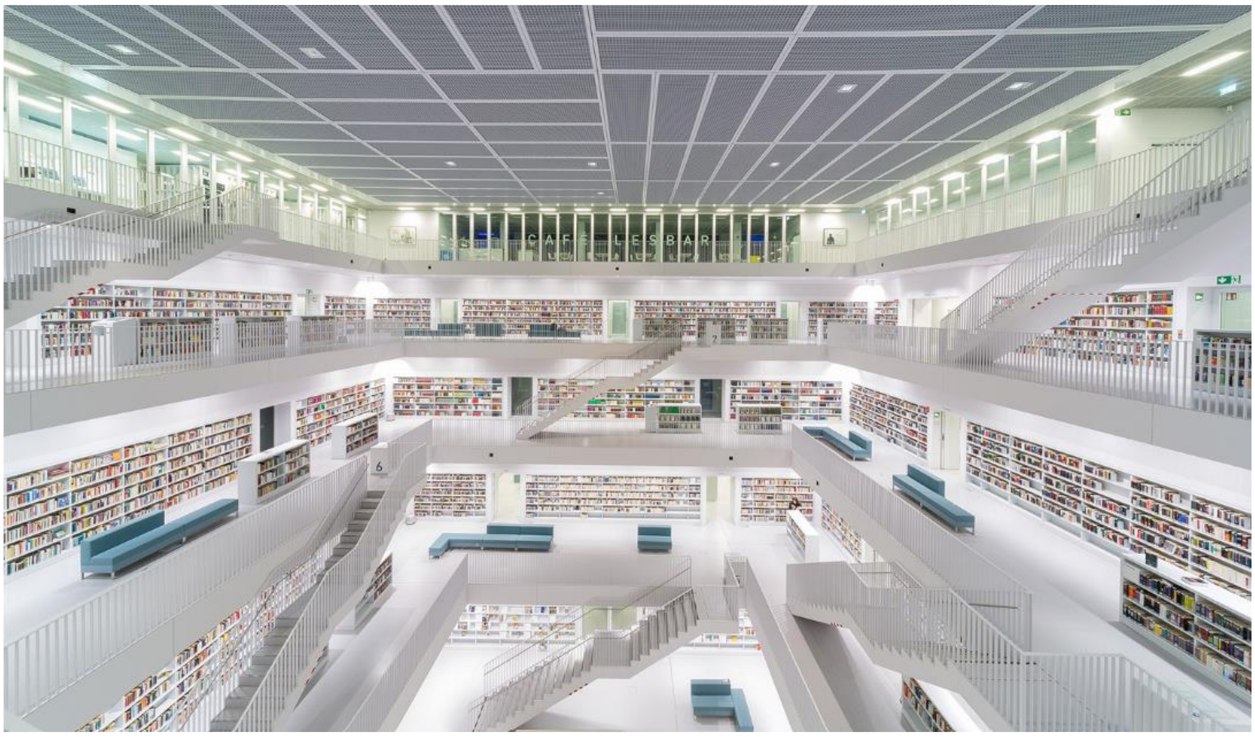


Figure 1 Stuttgart City Library | Stuttgart, Germany

Project Overview

Recommendation systems are information filtering systems that deal with the problem of information overload by filtering vital information fragments out of large amounts of dynamically generated information according to a user’s preferences, interest, or observed behavior about a certain item. A recommendation system has the ability to predict whether a particular user would prefer an item or not based on the user’s profile.

The aim of this project is to build a book recommendation system that can provide interesting book recommendations to the user based on a book's popularity or the user's interests/preferences.

The eventual result of the book recommendation system is not to make accurate predictions but instead to provide difficult to quantify insightful book recommendations.

Objectives

For our popularity-based approach:

- We want to be able to recommend books based on their ratings.

- We want to be able to recommend books according to the provided country and the popularity of the books in that country according to their rating.

- We want to be able to recommend books according to the provided Author name using the weighted average for each book’s average rating.

For our collaborative-filtering approach:

- We have a memory-based approach using KNN algorithm (Euclidean distance based), the recommended books are based on similarity between each other according to the provided book’s features.

Methodology

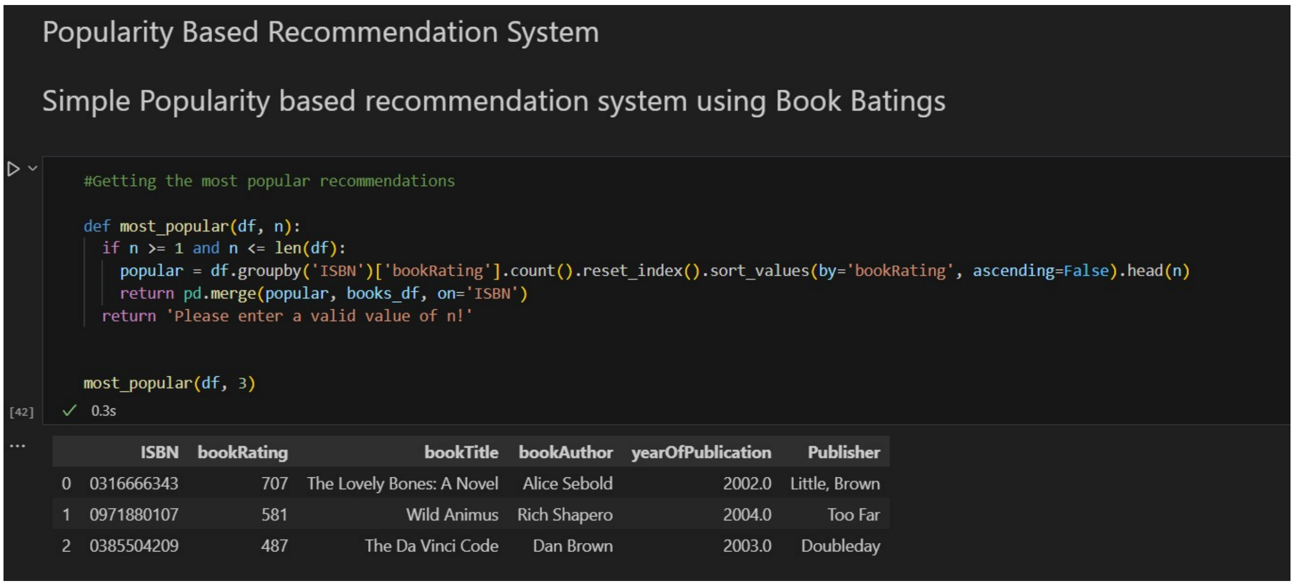
To build this recommendation system, We have taken two approaches:

1. Popularity Based Recommendation System.

With its simplicity, this is the most basic recommendation system which offers generalized recommendation to every user based on their popularity. In a bookstore, if a certain book is popular among its customers & is also critically acclaimed, in the scenario that a new customer walks in & asks for the best, they would be suggested to try that book too. The same is true for movies, shows, music, etc. Whatever is more popular among the general public, is more likely to be recommended to new customers too.

This type of recommendation system makes generalized recommendation not personalized, meaning that this system will not take into account the personal preferences or choices, rather it would tell that this particular thing is liked by most of the users.

To build one, the count of user ratings were taken for different books & the top 10 rated books are displayed below:



Popularity Based Recommendation System

Simple Popularity based recommendation system using Book Batings

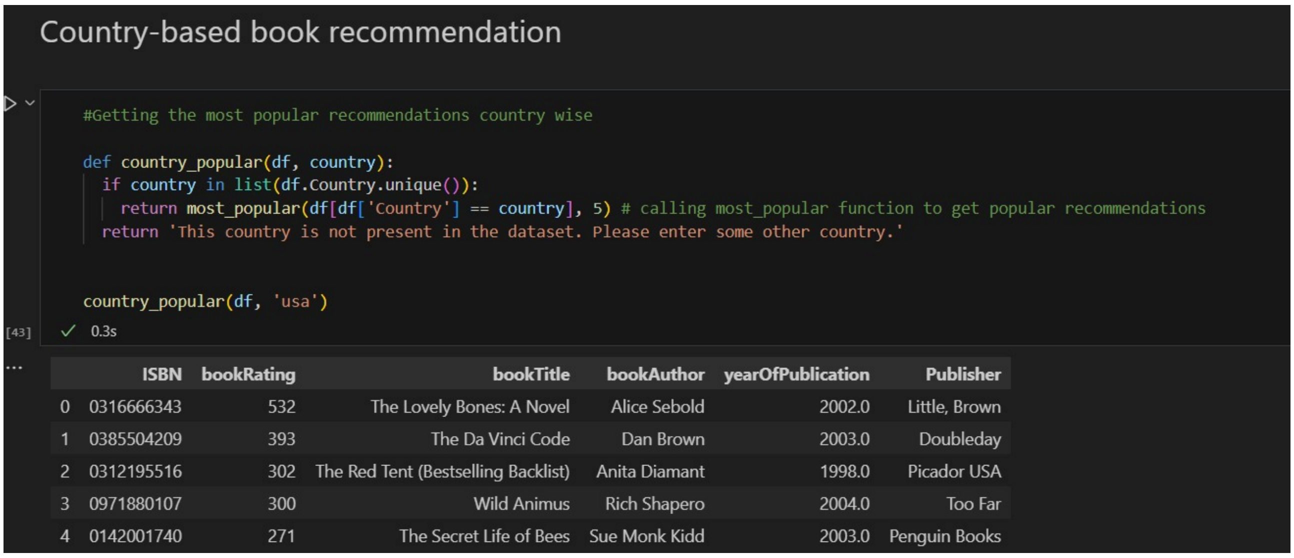
```
#Getting the most popular recommendations

def most_popular(df, n):
  if n >= 1 and n <= len(df):
    popular = df.groupby('ISBN')['bookRating'].count().reset_index().sort_values(by='bookRating', ascending=False).head(n)
    return pd.merge(popular, books_df, on='ISBN')
  return 'Please enter a valid value of n!'


most_popular(df, 3)
```

[42] ✓ 0.3s

	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	Publisher
0	0316666343	707	The Lovely Bones: A Novel	Alice Sebold	2002.0	Little, Brown
1	0971880107	581	Wild Animus	Rich Shapero	2004.0	Too Far
2	0385504209	487	The Da Vinci Code	Dan Brown	2003.0	Doubleday

Figure 2 popularity based recommendation system

Country based book recommendation (USA):



Country-based book recommendation

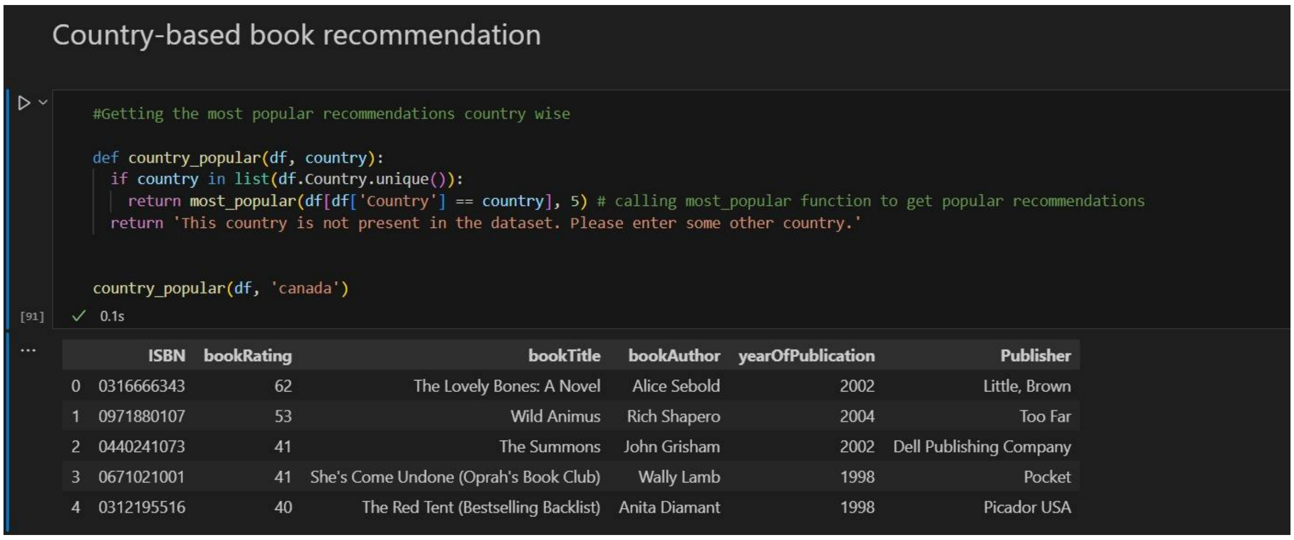
```
#Getting the most popular recommendations country wise

def country_popular(df, country):
    if country in list(df.Country.unique()):
        return most_popular(df[df['Country'] == country], 5) # calling most_popular function to get popular recommendations
    return 'This country is not present in the dataset. Please enter some other country.'


country_popular(df, 'usa')
```

[43] ✓ 0.3s

	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	Publisher
0	0316666343	532	The Lovely Bones: A Novel	Alice Sebold	2002.0	Little, Brown
1	0385504209	393	The Da Vinci Code	Dan Brown	2003.0	Doubleday
2	0312195516	302	The Red Tent (Bestselling Backlist)	Anita Diamant	1998.0	Picador USA
3	0971880107	300	Wild Animus	Rich Shapero	2004.0	Too Far
4	0142001740	271	The Secret Life of Bees	Sue Monk Kidd	2003.0	Penguin Books

Figure 3 USA-based book recommendation

Country based book recommendation (Canada):



Country-based book recommendation

```
#Getting the most popular recommendations country wise

def country_popular(df, country):
    if country in list(df.Country.unique()):
        return most_popular(df[df['Country'] == country], 5) # calling most_popular function to get popular recommendations
    return 'This country is not present in the dataset. Please enter some other country.'


country_popular(df, 'canada')
```

[91] ✓ 0.1s

	ISBN	bookRating	bookTitle	bookAuthor	yearOfPublication	Publisher
0	0316666343	62	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown
1	0971880107	53	Wild Animus	Rich Shapero	2004	Too Far
2	0440241073	41	The Summons	John Grisham	2002	Dell Publishing Company
3	0671021001	41	She's Come Undone (Oprah's Book Club)	Wally Lamb	1998	Pocket
4	0312195516	40	The Red Tent (Bestselling Backlist)	Anita Diamant	1998	Picador USA

Figure 4 Canada-based book recommendation

2. Collaborative Filtering Based Recommendation System.

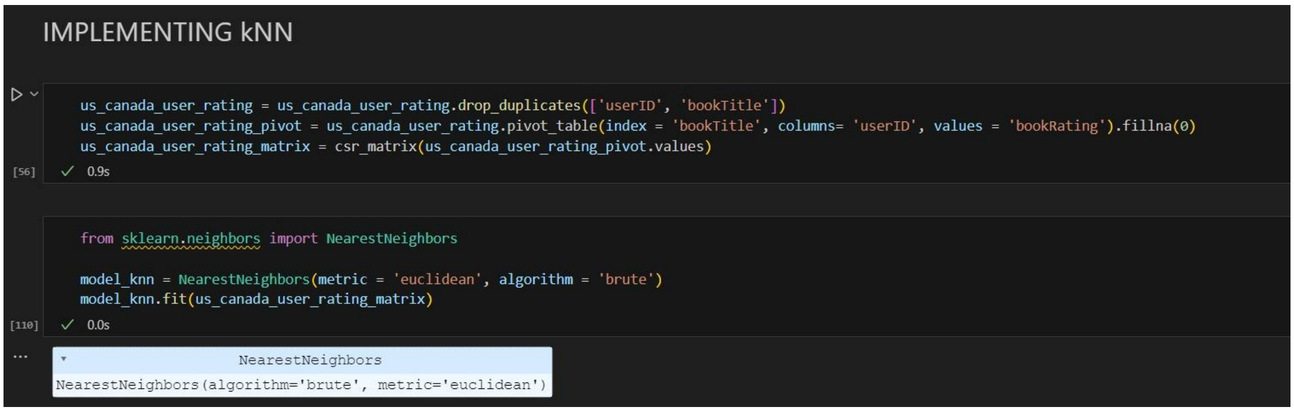
In Collaborative Filtering, we tend to find similar users and recommend what similar users like. In this type of recommendation system, we don't use the features of the item to recommend it, rather we classify the users into the clusters of similar types, and recommend each user according to the preference of its cluster.

a.) Using K-Nearest Neighbors:

K-Nearest Neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. KNN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.

Using this algorithm, clusters of similar users based on common book ratings can be found and predictions can be made using the average rating of the top-k nearest neighbors.

To look for popular books, the 'books' data was combined with the 'ratings' data.



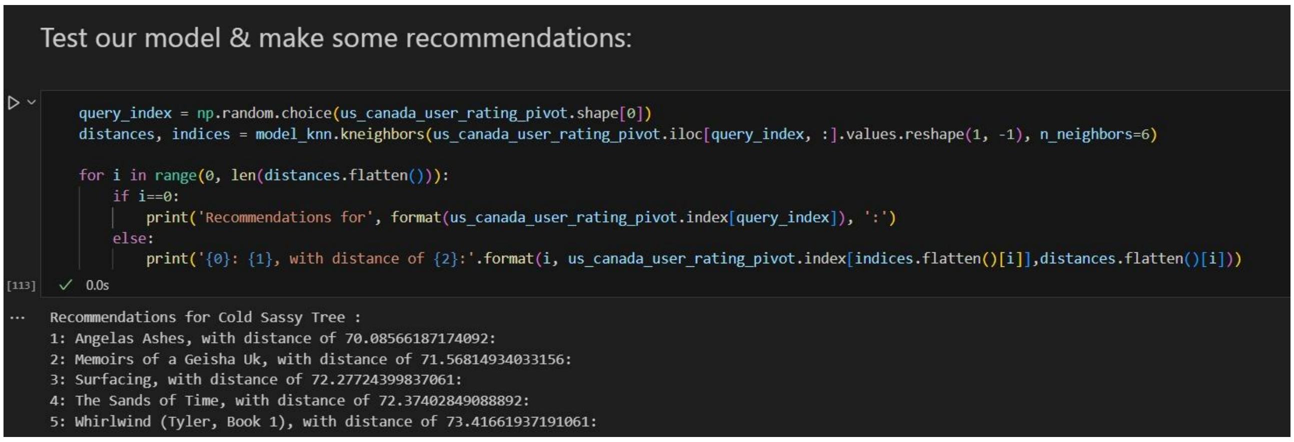
IMPLEMENTING kNN

```
us_canada_user_rating = us_canada_user_rating.drop_duplicates(['userID', 'bookTitle'])
us_canada_user_rating_pivot = us_canada_user_rating.pivot_table(index = 'bookTitle', columns= 'userID', values = 'bookRating').fillna(0)
us_canada_user_rating_matrix = csr_matrix(us_canada_user_rating_pivot.values)
```
[56] ✓ 0.9s

```
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'euclidean', algorithm = 'brute')
model_knn.fit(us_canada_user_rating_matrix)
```
[110] ✓ 0.0s

```
NearestNeighbors
NearestNeighbors(algorithm='brute', metric='euclidean')
```

Figure 5 KNN implementation:

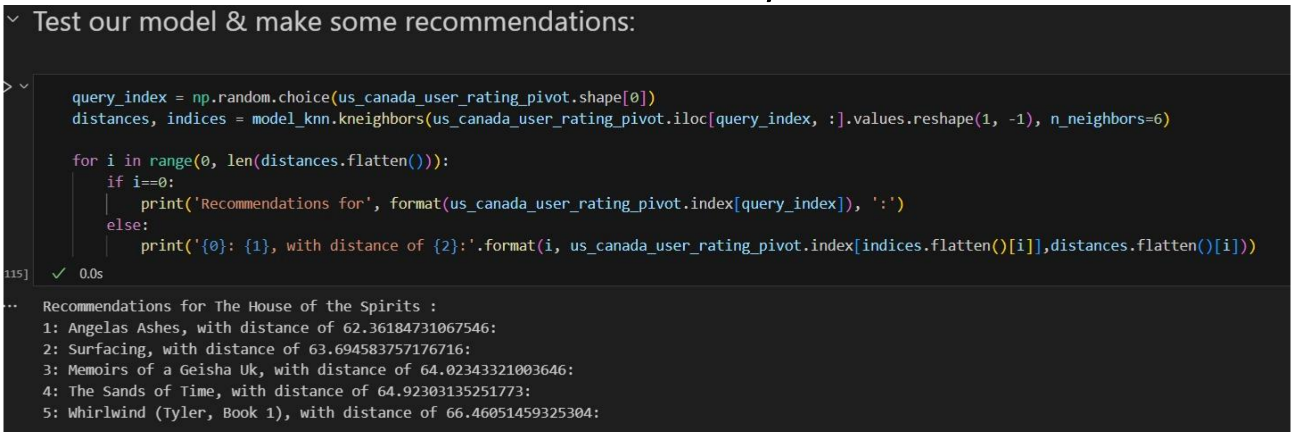


Test our model & make some recommendations:

```
query_index = np.random.choice(us_canada_user_rating_pivot.shape[0])
distances, indices = model_knn.kneighbors(us_canada_user_rating_pivot.iloc[query_index, :].values.reshape(1, -1), n_neighbors=6)

for i in range(0, len(distances.flatten())):
    if i==0:
        print('Recommendations for', format(us_canada_user_rating_pivot.index[query_index]), ':')
    else:
        print('{0}: {1}, with distance of {2}:'.format(i, us_canada_user_rating_pivot.index[indices.flatten()[i]],distances.flatten()[i]))
```
[113] ✓ 0.0s

```
Recommendations for Cold Sassy Tree :
1: Angelas Ashes, with distance of 70.08566187174092:
2: Memoirs of a Geisha Uk, with distance of 71.56814934033156:
3: Surfacing, with distance of 72.27724399837061:
4: The Sands of Time, with distance of 72.37402849088892:
5: Whirlwind (Tyler, Book 1), with distance of 73.41661937191061:
```

Figure 6 Test our model 1st run

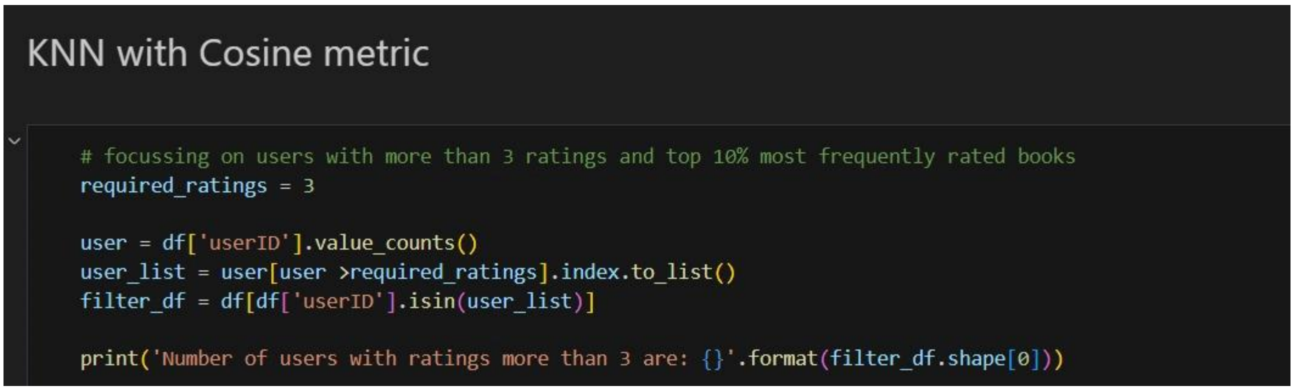


Test our model & make some recommendations:

```
query_index = np.random.choice(us_canada_user_rating_pivot.shape[0])
distances, indices = model_knn.kneighbors(us_canada_user_rating_pivot.iloc[query_index, :].values.reshape(1, -1), n_neighbors=6)

for i in range(0, len(distances.flatten())):
    if i==0:
        print('Recommendations for', format(us_canada_user_rating_pivot.index[query_index]), ':')
    else:
        print('{0}: {1}, with distance of {2}:'.format(i, us_canada_user_rating_pivot.index[indices.flatten()[i]],distances.flatten()[i]))
```

```
Recommendations for The House of the Spirits :
1: Angelas Ashes, with distance of 62.36184731067546:
2: Surfacing, with distance of 63.694583757176716:
3: Memoirs of a Geisha Uk, with distance of 64.02343321003646:
4: The Sands of Time, with distance of 64.92303135251773:
5: Whirlwind (Tyler, Book 1), with distance of 66.46051459325304:
```

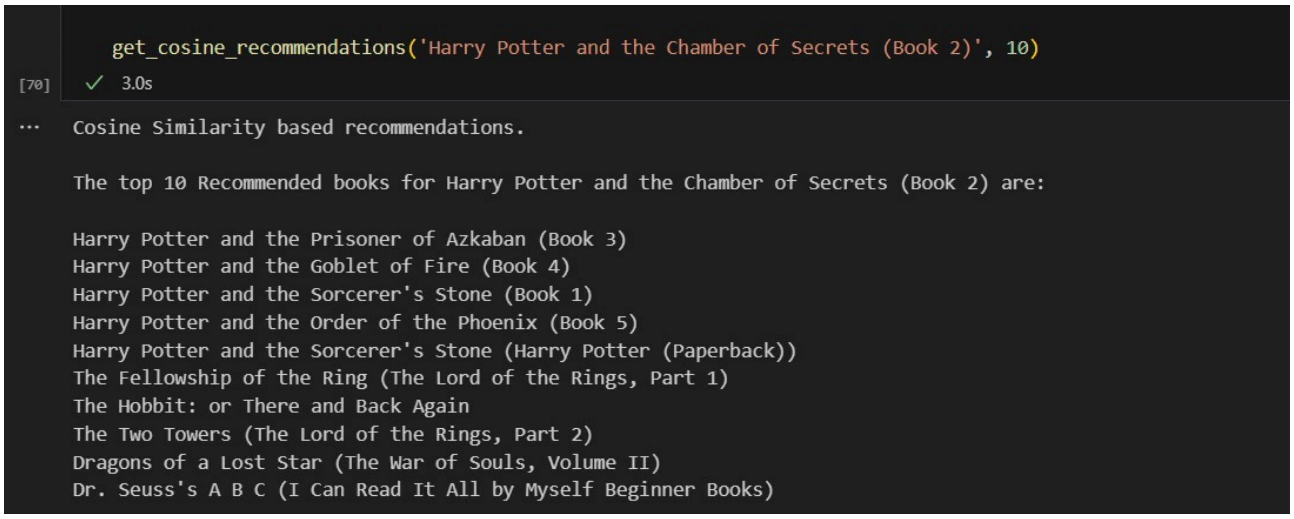
Figure 7 Test our model 2nd run



KNN with Cosine metric

```
# focussing on users with more than 3 ratings and top 10% most frequently rated books
required_ratings = 3

user = df['userID'].value_counts()
user_list = user[user >required_ratings].index.to_list()
filter_df = df[df['userID'].isin(user_list)]

print('Number of users with ratings more than 3 are: {}'.format(filter_df.shape[0]))
```

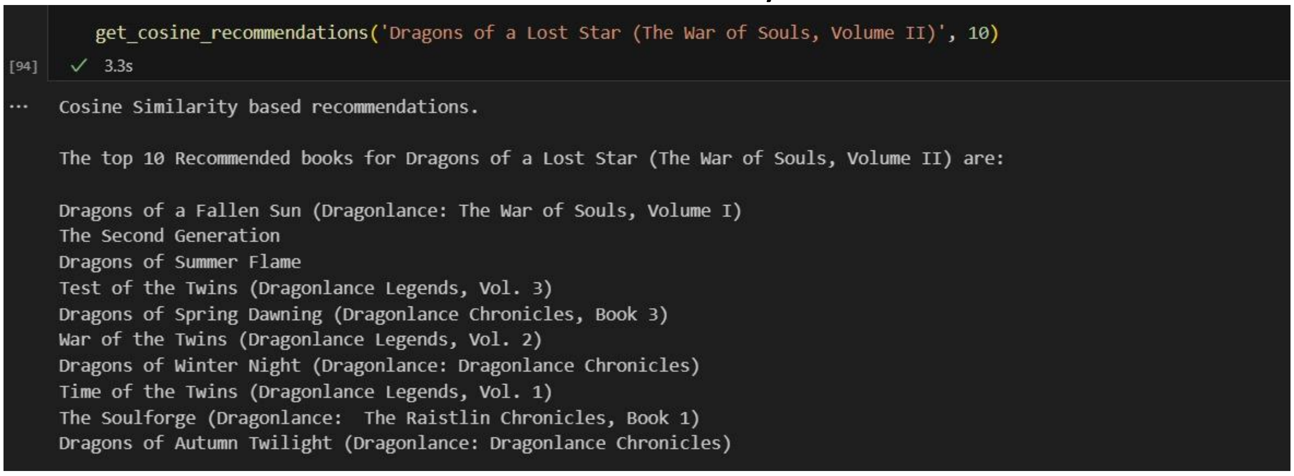
Figure 8 KNN with cosine metric



```
get_cosine_recommendations('Harry Potter and the Chamber of Secrets (Book 2)', 10)
```

```
Cosine Similarity based recommendations.

The top 10 Recommended books for Harry Potter and the Chamber of Secrets (Book 2) 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)
Harry Potter and the Order of the Phoenix (Book 5)
Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
The Fellowship of the Ring (The Lord of the Rings, Part 1)
The Hobbit: or There and Back Again
The Two Towers (The Lord of the Rings, Part 2)
Dragons of a Lost Star (The War of Souls, Volume II)
Dr. Seuss's A B C (I Can Read It All by Myself Beginner Books)
```

Figure 9 run of cosine metric



```
get_cosine_recommendations('Dragons of a Lost Star (The War of Souls, Volume II)', 10)
[94]  ✓ 3.3s

Cosine Similarity based recommendations.

The top 10 Recommended books for Dragons of a Lost Star (The War of Souls, Volume II) are:

Dragons of a Fallen Sun (Dragonlance: The War of Souls, Volume I)
The Second Generation
Dragons of Summer Flame
Test of the Twins (Dragonlance Legends, Vol. 3)
Dragons of Spring Dawning (Dragonlance Chronicles, Book 3)
War of the Twins (Dragonlance Legends, Vol. 2)
Dragons of Winter Night (Dragonlance: Dragonlance Chronicles)
Time of the Twins (Dragonlance Legends, Vol. 1)
The Soulforge (Dragonlance:  The Raistlin Chronicles, Book 1)
Dragons of Autumn Twilight (Dragonlance: Dragonlance Chronicles)
```

Figure 10 2nd run of cosine metric using a different book

Note: To cope with the computing power of my machine, the users have been limited to those in the US & Canada. The user data is then combined with the rating data & the total rating count data.

The table is converted into a 2D matrix & the missing values are filled with zeroes since the distances between rating vectors will be calculated. The values of the matrix dataframe are then transformed into a scipy sparse matrix for more efficiency calculations

The algorithm used to compute the nearest neighbors is **'brute'** & the metric is **'cosine'** so that the algorithm will calculate the cosine similarity between rating vectors.

We Use K-NN as a Collaborative Filtering Based Recommendation System in our Project? Because its characteristics is:,

- **User-friendly Explanation**: We can explain that k-NN is a straightforward algorithm that identifies the k most similar users (or items) to you based on their past interactions. It's intuitive because it's similar to how we might ask for recommendations from friends who have similar tastes.
- **Personalized Recommendations**: Emphasize that k-NN helps in generating personalized recommendations tailored to your specific preferences. By identifying users with similar tastes, it can suggest items that you're likely to enjoy based on what those similar users have liked or interacted with.
- **Flexibility and Adaptability**: Highlight that k-NN is flexible and adaptable to different types of recommendation problems. Whether it's recommending books, movies, products, or anything else, k-NN can be applied effectively.
- **No Assumptions About Data Distribution**: Unlike some other recommendation algorithms that make assumptions about the underlying distribution of the data, k-NN is non-parametric and makes no such assumptions. This can be beneficial when dealing with datasets that may not adhere to specific statistical distributions.
- **Interpretability**: Mention that k-NN is interpretable because the recommendations are based on the actual behavior of similar users. This transparency can be reassuring, especially in contexts where understanding the rationale behind recommendations is important.
- **Scalability**: While k-NN can be computationally expensive for very large datasets, for moderate-sized datasets or in scenarios where real-time recommendations are not required, it can be quite feasible and effective.

b.) Using Matrix Factorization:

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrice.

Singular value decomposition (SVD) is used here.
The US-Canada users' rating table is converted into a utility matrix & the missing values are filled with zeros.

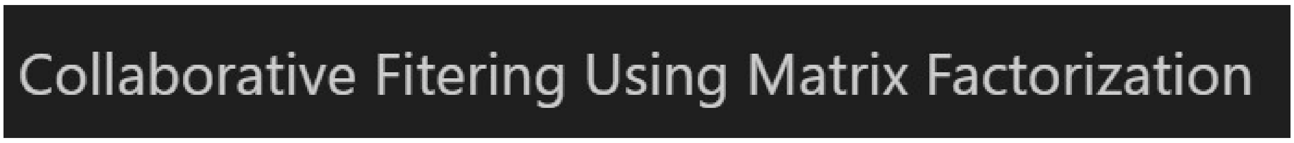
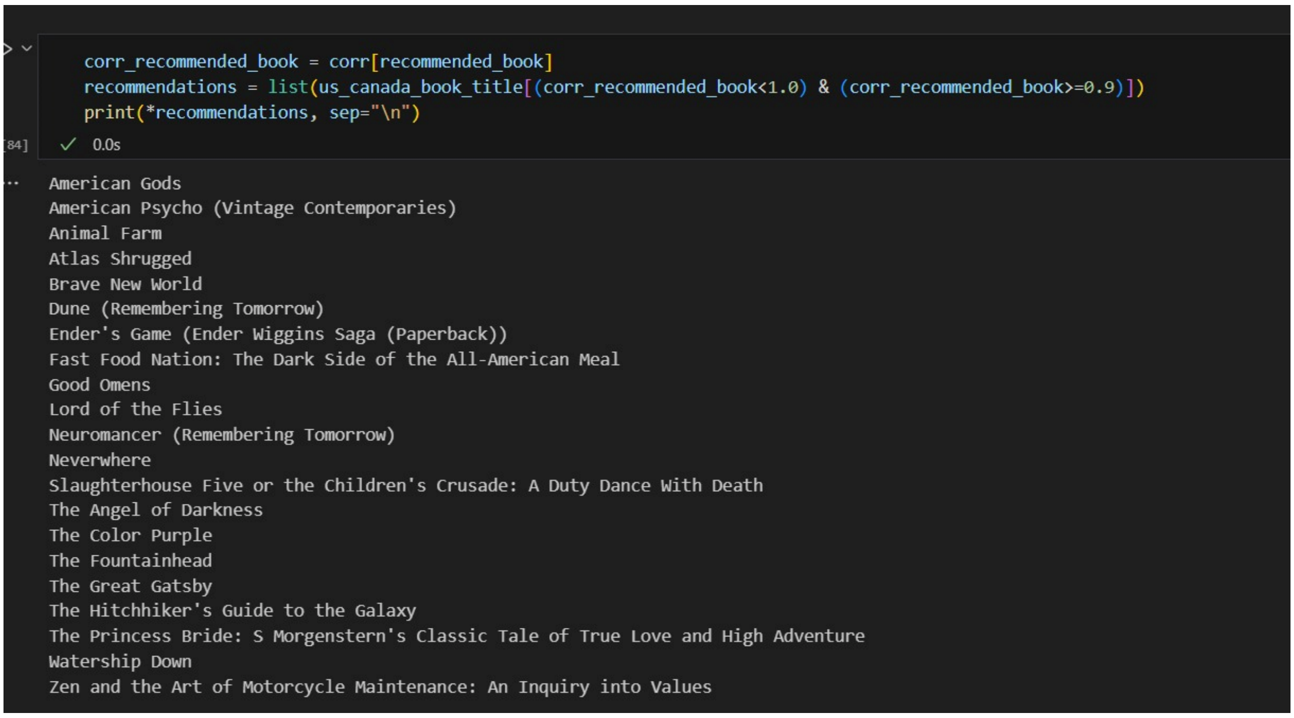

Collaborative Fitering Using Matrix Factorization

Figure 11 filtering using matrix factorization

.


```
corr_recommended_book = corr[recommended_book]
recommendations = list(us_canada_book_title[(corr_recommended_book<1.0) & (corr_recommended_book>=0.9)])
print(*recommendations, sep="\n")
```
[84] ✓ 0.0s

```
American Gods
American Psycho (Vintage Contemporaries)
Animal Farm
Atlas Shrugged
Brave New World
Dune (Remembering Tomorrow)
Ender's Game (Ender Wiggins Saga (Paperback))
Fast Food Nation: The Dark Side of the All-American Meal
Good Omens
Lord of the Flies
Neuromancer (Remembering Tomorrow)
Neverwhere
Slaughterhouse Five or the Children's Crusade: A Duty Dance With Death
The Angel of Darkness
The Color Purple
The Fountainhead
The Great Gatsby
The Hitchhiker's Guide to the Galaxy
The Princess Bride: S Morgenstern's Classic Tale of True Love and High Adventure
Watership Down
Zen and the Art of Motorcycle Maintenance: An Inquiry into Values
```

Results and Evaluation

All results match the data in the dataset as shown in the figures. So In evaluating our book recommendation system, we're making sure that the suggestions it gives and how they're arranged match up well with what's actually in the dataset. We're checking if the books it recommends are similar to what users have liked before. We're also using pictures to see if the recommendations line up nicely with the data in the dataset. This helps us make sure that the system is doing a good job of suggesting books that fit with what users have shown they like.

Conclusion

From the recommendations made, it is apparent that the system was able to recommend books to a user that has yet to read/rate based on the popularity of a book, the similarity of a previously read/rated book or their respective ratings. With these recommendations, a successful book recommendation system has been created that can make predictions & recommend books to its users. Lastly, based on the system's ability to make recommendations, it can be concluded that the system can be effective to help user get the book recommendation they want.