Michael Nguyen

Book Recommendation System

Introduction

Recommendation systems are one of the most integral parts of many online systems. A sound recommendation system not only increases a company's short-term revenue but also supports customer satisfaction, which is vital to the company's long-term benefit. In 2016, Netflix estimated that its recommendation is worth $1 billion per year in revenue. Netflix even held a competition in designing recommendation systems with a grand prize of $1 million. With people now spending more time online due to Covid, a sound recommendation system is essential to keep people engaged. This project will build a recommendation system on books using collaborative filtering based on users. The codes performed in this project can be found at: mnguyen494/Capstone-3: Book Recommendation (github.com)

Dataset

In this project, we will use three datasets provided by Ruchi Bhatia on Kaggle. The datasets are BX-Book-Ratings, BX-Users, and BX-Books. In the BX-Book-Ratings, each row corresponds to a book that a user reads. The columns then contain information on the books, the users, and ratings. In particular, the dataset's columns are:

- User-ID: the unique id of the reader
- ISBN: the unique if of the book
- Book-Rating: the rating of a book which range from 0 to 10.

The following dataset, BX-Users, contains information about the readers. Each row corresponds to a reader with the following columns:

- User-ID: the unique id of the reader
- Location: the city, state, country where the reader lives

- Age: the age of the reader

The last dataset, BX-Books, contains information about the books in each row. It has the following columns:

- ISBN: the unique id of the book
- Book-Title: the title of the book
- Book-Author: Author of the book
- Year-of-Publication: the year when the book is published
- Publisher: the publisher of the book
- Image-URL-S: a link leading to a small image of the book
- Image-URL-M: a link leading to a medium image of the book
- Image-URL-L: a link leading to a large image of the book

We will merge these three datasets into an official one for easier manipulation. Before merging, however, we will look at each dataset first to inspect for any problem.

Data Cleaning

- BX-Books Dataset

  First, we look at the BX-Books dataset, which contains the book's information. At first glance, the dataset contains only one null value in Book-Author. We then fill in the author's name by searching for the book online. However, upon closer inspection, we discover that many books with the same title and author are re-published in the dataset. We decide to merge these books into only one version. This way, the model we create will not be distracted by how and when the books were published but by their content only.

- BX-Book-Ratings

Next, we look at the BX-Book-Ratings dataset, which contains information about users' ratings on books. After checking for null values, we change the ISBNs of the books that we merge in the BX-Books dataset into the ISBN of the version we keep.

- BX-Users

The last dataset we need to clean is the BX-Users. Upon first inspection, the “Age” column of the dataset is missing 40% of its entries. In addition, there are many entries with ages as small as 0 and as high as 244. To address this issue, we first make all age entries smaller than ten and bigger than ninety nulls. We then fill the null values using the remained age distribution.

After cleaning the data, we then merge the three datasets. We first combine the BX-Books Dataset to the BX-Book-Ratings using the ISBN. Next, we merge the BX-Users dataset into the combined one with User-ID. Unfortunately, there are many books that the readers in the BX-Book-Ratings rate that the BX-Books dataset does not have information on. This results in 20% null entries in the final dataset. We decide to drop these rows because there is no easy way to fill them. At the end of the cleaning, our dataset has over 900,000 rows and nine columns.

Michael Nguyen

Exploratory Data Analysis

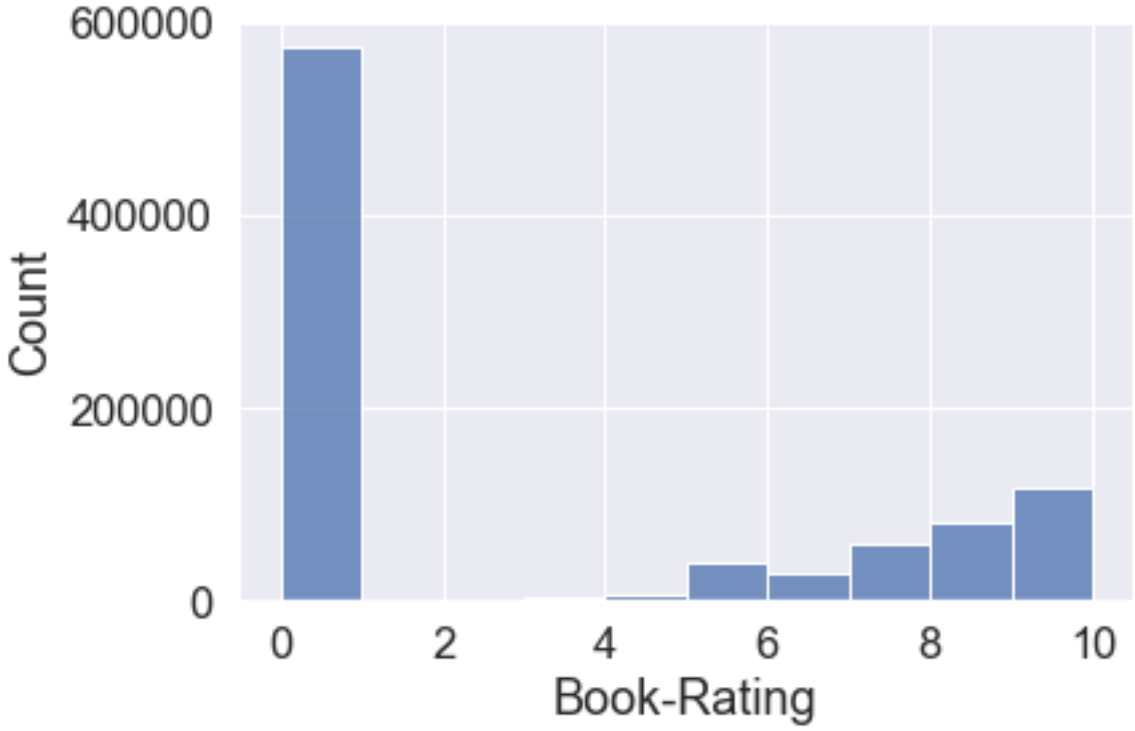

Fig. 1: Histogram of the ratings from the readers. According to the figure, most ratings are either 0 or 10.

First, we examine the histogram of ratings in all the books. According to Fig. 1, most ratings are either 0 or 10. Since the ratings are not very well spread, it's probably better to phrase the recommendation problem into a classifying one instead of a regression one. In other words, we will create a new column called "recommended_book" with only two values to indicate whether a reader likes a book or not instead of using the rating system. In this column, books with "Book-Raring" bigger than six will be assigned the value two while the rest will be given one. The distribution of the "recommended_book" column is shown in Fig. 2.

Next, we look at the distribution of the number of ratings by a user across the dataset. This is useful because we need to drop users without many ratings for our collaborative filtering to do well. Ideally, we would like users with more than 100 ratings in our filtering; however, this might remove too many users for our analysis depending on the

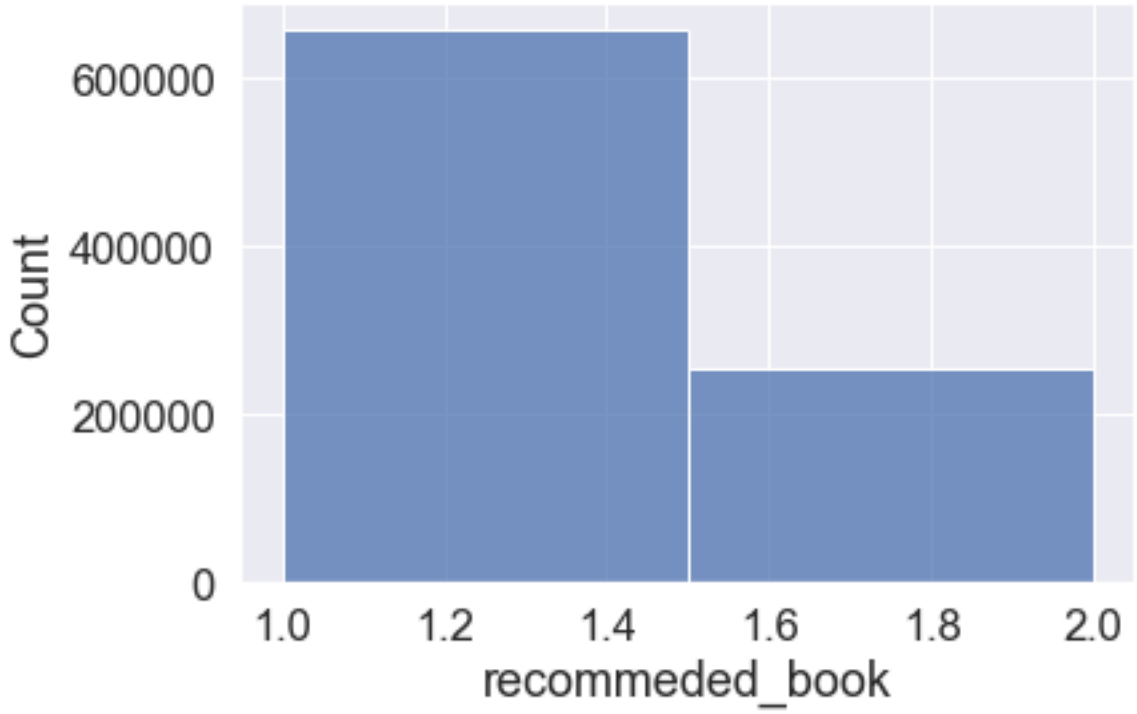

Fig. 2: Histogram of the recommended_book. Books that are liked will have value 2 while the rest will have value 1.

dataset. We thus limit ourselves to only removing a maximum of 20% of users from the dataset. From Fig. 3, users with less than 100 ratings account for about 40% of the total users. Thus, we can only remove users with less than ten reviews, accounting for about 20% of the total users.

Similar to the users with small ratings, books with too little rating also did not help our collaborative filtering. We saw from Fig. 4 that books with just one rating contribute to more than 20% of the total books. Thus, we can only remove books with less than two ratings. After all this cleaning, our dataset goes from over 900,000 rows to 600,000 rows.

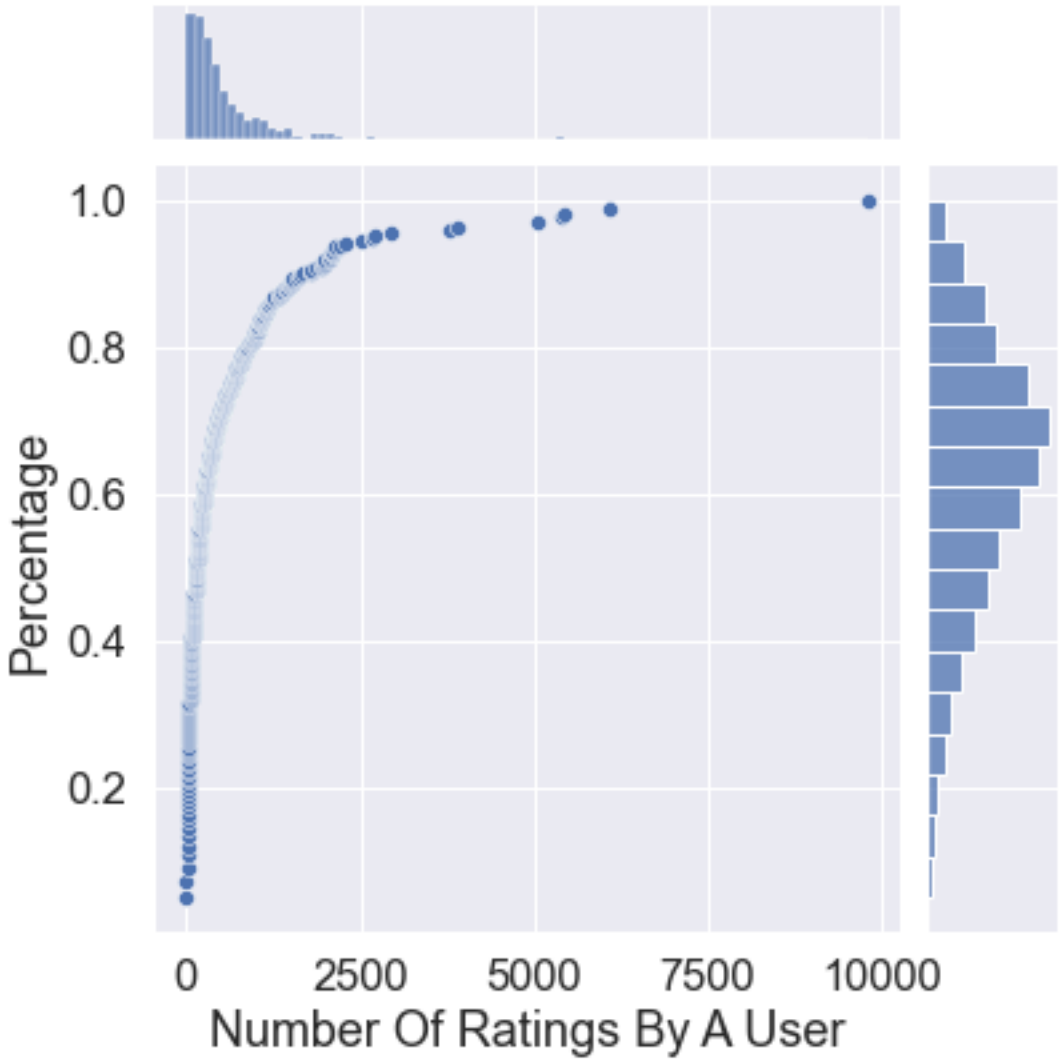

Fig. 3: Percentage Vs. the number of ratings given by a user. Around 20% of users in our dataset are those who gave less than 11 reviews.

Collaborative Filtering

The dataset is now ready for us to build a recommendation system. This project will focus on using collaborative filtering based on user. Collaborative filters are one of the most powerful models for recommendation used in the industry, and many big companies such as Amazon have found great success with them.

Fig. 4: Percentage Vs. the number of ratings of a book. Around 20% of books in our dataset are those with just one rating.

In essence, the user-based collaborative filter is built on the idea that users with the same interests in the past also tend to buy and like similar items in the future. For example, let's assume we have identified that both user A and user B like and dislike the same books. We then can recommend a book to user B from the list of books that user A likes, given that user B hasn't read the book.

Our first step in building user-based collaborative filtering is creating a rating matrix. As shown in Fig. 5, each row represents a user in a rating matrix while each column

represents a book. The matrix entries are the recommendations of a book by a user. In our case, the entry will have a value of one if a reader dislikes a book and a value of two if a reader likes it. Fig. 5 also shows that most of our entries are null, indicating the matrix is very sparse.

Book-ID	1	2	7	10	13	22	23	28	34	36	...	248779	248782	248784	248786	248788	248791
User-ID																	
86123	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
86134	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
86145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
86159	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
86170	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

Fig. 5: The rating matrix: Each row represents a user, and each column represents a book. The entries in the matrix are recommendations of a user for a book. As shown here, the matrix is very parse with a lot of null entries.

In this project, we will input the user and the book into the models under consideration. The models then will tell us if the book should be recommended to the user or not. With this, the most straightforward collaborative filtering we can build simply outputs the average recommendation across users. That is, it averages out all the values in a column of the rating matrix. In this case, all the users are treated equally. However, as we mentioned earlier, the power of user-based collaborative filtering is based on finding users similar to each other. Thus, a more similar user to the input user should have more weight in determining whether a book should be recommended. One method to determine the similarity between users is using the similarity cosine score between the users. Thus, a better method than the simple mean one involves taking the recommendation of each user and multiplying it by the similarity score before summing up to find the final recommendation.

We will also consider more advanced methods of collaborative filtering in this project. These methods rely on machine learning to identify user clusters in the rating matrix. The users in the same cluster will then be used to determine the recommendation of the books. For this, we will use the surprise toolkit to implement the machine learning model to our recommendation system.

Michael Nguyen

Models Comparison

Before further analysis, we have created a dummy classifier in which a book is recommended by stratification only. We then compare the models we build with this dummy classifier using three metrics: AUC, accuracy, and precision. We particularly care about precision because we do not want to recommend a book that users do not like. On the other hand, it wouldn't matter as much if a good book is not recommended to a user.

In Fig. 6, we plot the models' performances based on the three metrics. The blue bar represents AUC, the red bar represents accuracy, and the green bar represents precision. As we go from left to right, the algorithms used are dummy classifier, mean recommendation, weighted mean recommendation, k-NN (k-nearest neighbor), k-NN

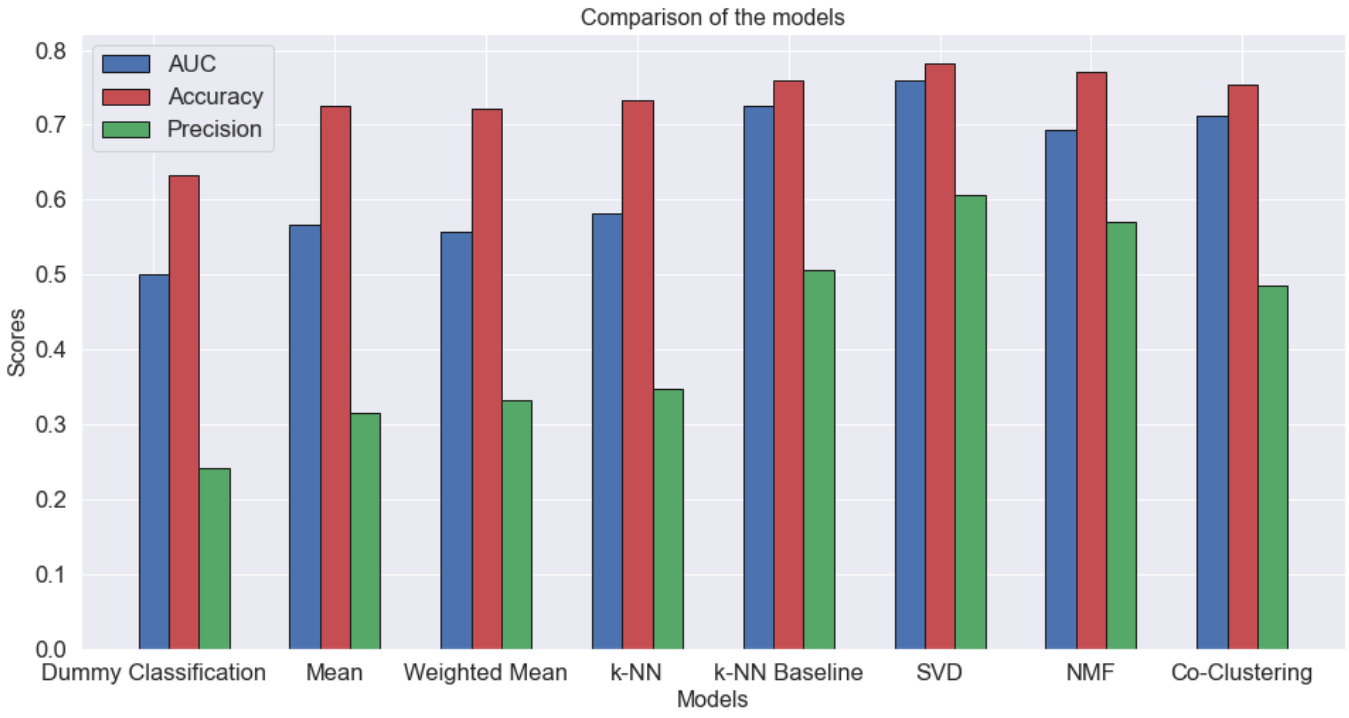

Fig. 6: Comparison of various models using three different metrics: AUC, Accuracy, and Precision. From left to right: dummy classifier, Simple Mean Collaborative Filtering, Weighted Mean Collaborative Filtering, k-NN, k-NN Baseline, SVD, NMF, and Co-Clustering. Blue bar: AUC. Red bar: Accuracy. Green bar: Precision. According to the figure, the model using SVD beats all other models in all three metrics.

baseline, SVD (Singular Value Decomposition), NMF (non-negative matrix factorization), and Co-clustering. Among these models, SVD is the best in all three metrics. SVD is a famous model that Simon Funk popularized during the Netflix Contest. We then optimize the SVD with Grid Search using precision as the primary metric. The optimized model gives an AUC of 0.74, an accuracy of 0.78, and a precision of 0.64. This is more than twice as good as the dummy classifier in terms of precision.

Conclusion And Future Directions

In this project, we have created recommendation systems on books using collaborative filtering based on users. The best model, using SVD, can predict whether a reader will like a book with precision more than two times the baseline model. However, the methods in this project and collaborative filtering in general suffer from a serious problem called the cold start problem. In other words, the system needs a large amount of data on the ratings of the products. This might not pose a problem to more established companies and products like Amazon or iPhone. However, newly formed companies cannot utilize the power of collaborative filtering. One way to resolve this problem is using a content-based recommendation system. This system provides recommendations utilizing the information of the products. It thus does not require a large data on customers' ratings. Unfortunately, the datasets in this project lack many features such as the content of the book, its genre, etc., to effectively use a content-based system. More data will have to be collected to implement a content-based recommendation and combine it with collaborative filtering to create a hybrid system.