Joseph Petersen - October 2018 Data Science Career Track Cohort

Capstone 2 - First Milestone Report

Library Book Recommendation Engine

Overview:

John Albert's Library in Fictional Town, USA has recently added to their website the ability for library card holders to rate the books they check out, and in accordance with their mission to spread the joy of reading to their local community, they're interested in also providing recommendations for other titles based on previous ratings data. This project aims to create that recommendation system based off the Book-Crossing dataset.

Dataset and Methods:

Dataset: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

The Book-Crossing dataset is comprised of three tables: BX-Users contains the user's ID as well as the location and age if available (null if not), BX-Books contains the ISBN, book title, author, year of publication, and publisher, and BX-Ratings contains the book ratings information on a scale of 1-10 with some implicit values marked as 0. The dataset contains 278,858 users, 1,149,780 ratings, and 271,379 books.

The dataset will need to be cleaned up as there are some missing, null values, and questionable outliers. The user IDs and ratings will be formed into a matrix for user based collaborative filtering. This allows recommendations for users based off other users with similar reading habits and likes.

Data Wrangling Steps:

The first steps were to examine each separate table to look for errors and issues. The BX-Books table required the following wranglig steps:

1. Manual adjustments were made to the base file due to a number of extra separator characters (';') present in book titles. This was done by reading the errors flagged when setting the 'error_bad_lines' parameter in the Pandas read_csv function to 'False'. Additionally, '&' were coded in the table as '&' which were changed.
2. The book image URL's were removed as they are not necessary for the recommendation engine. The columns were also renamed for ease of use.
3. One null value in the 'author' column and two in the 'publisher' column were easily added with online information.

4. The ‘year’ column contained both integers and strings as well as two erroneous publisher names. The two names were easily examined and fixed with the proper entries as they just needed a shift over. The column was then converted to numeric. Some entries didn’t make logical sense (0 and years later than 2004, the year the dataframe was compiled), and they were converted to NaN.

The BX-Book-Ratings table required to following wrangling steps:

1. The columns were renamed for ease of use.
2. A large number of books (70,386) rated had ISBN numbers that were not present in the BX-Books table. These were removed as the title is necessary for recommendation.
3. The ‘ratings’ column contained 0’s which indicate an implicit rating meaning a type of rating that can’t be quantified on the scale of 1-10. These were separated from those with explicit ratings as the latter are what is required for the recommendation engine.

The BX-Users table required the following wrangling steps:

1. The columns were renamed for ease of use.
2. The ‘age’ column had entries that didn’t make sense. It was decided that ages less than 5 or greater than 100 were likely erroneous entries and replaced with NaN.
3. The ‘location’ column was split into city, state, and country.

Data Insights:

One important takeaway from looking into the books dataset was that some books appear under different ISBN. This is because of different editions. The next steps will need to include creating a second level of identification that groups together books with the same title but different ISBN.

Selected Poems	27
Little Women	24
Wuthering Heights	21
The Secret Garden	20
Dracula	20
Adventures of Huckleberry Finn	20
Jane Eyre	19
The Night Before Christmas	18
Pride and Prejudice	18
Great Expectations	17
Masquerade	16
Frankenstein	16
Black Beauty	16
The Gift	15
Emma	15
Beloved	15
Nemesis	14
Robinson Crusoe	13
The Wedding	13
The Secret	13

Table 1: The above shows the top 20 book titles by count. It indicates that many books have separate editions and therefore separate ISBN.

The books dataset also mostly contained titles published within the past 100 years. There is a distinct left skew. If publication year ends up playing a part in the recommendation system, this will need to be taken into account.

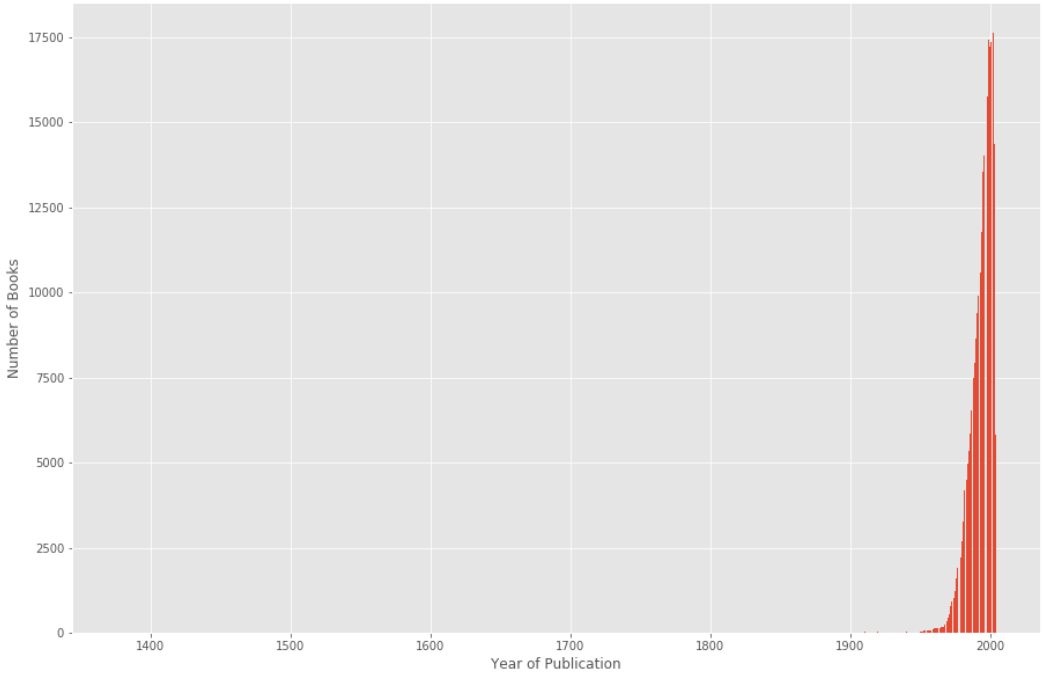

Figure 1: The above shows the heavy left skew of the distribution of publication years.

There's another skewed distribution when looking at the number of ratings per user. The vast majority of users only rate a few titles while a few distinct outliers rate more titles than most would even dream of. One user has rated over 14,000 titles. This indicates that the user rating matrix will be sparse (contains a lot of entries with no rating information). This was corroborated by checking the size of the user rating matrix, 121,052,287,740, versus the number of ratings, 1,149,780. Such a disparity indicates sparsity.

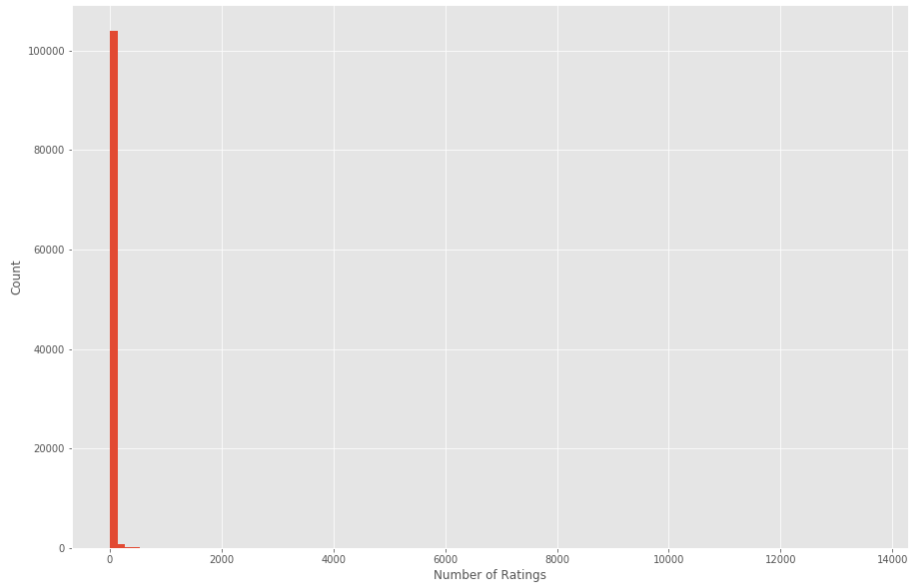

Figure 2: The above shows the incredibly heavy right skew in the distribution of number of ratings.

When looking at the counts of the ratings of 1-10, it showed that raters were generally favorable to the books they've read with the majority sitting over 5.

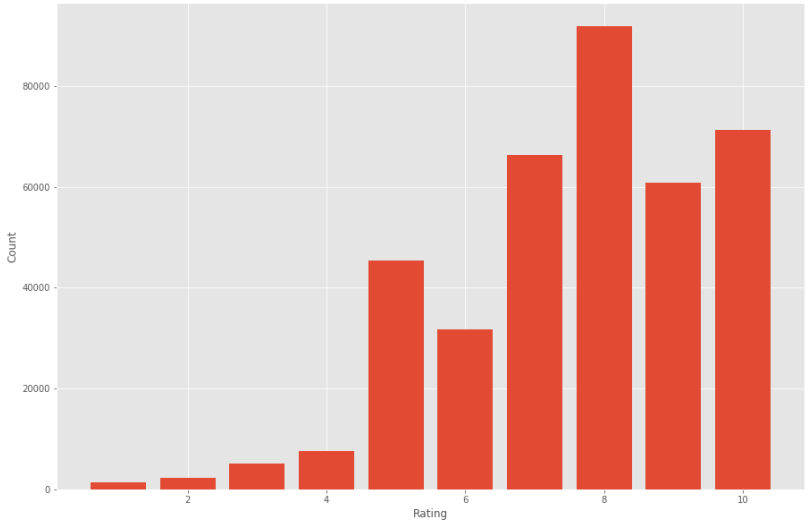

Figure 3: The above shows the counts of the ratings 1-10. It shows a general favorability by raters towards the books they've read.

The ages of raters slightly skewed right with the majority sitting around ages 20-40. This, unfortunately, does not take into account the large number of NaN values. Should age become a factor in the recommendation engine then a solution to those NaN values will need to be found. Likely, they would be imputed to the mean.

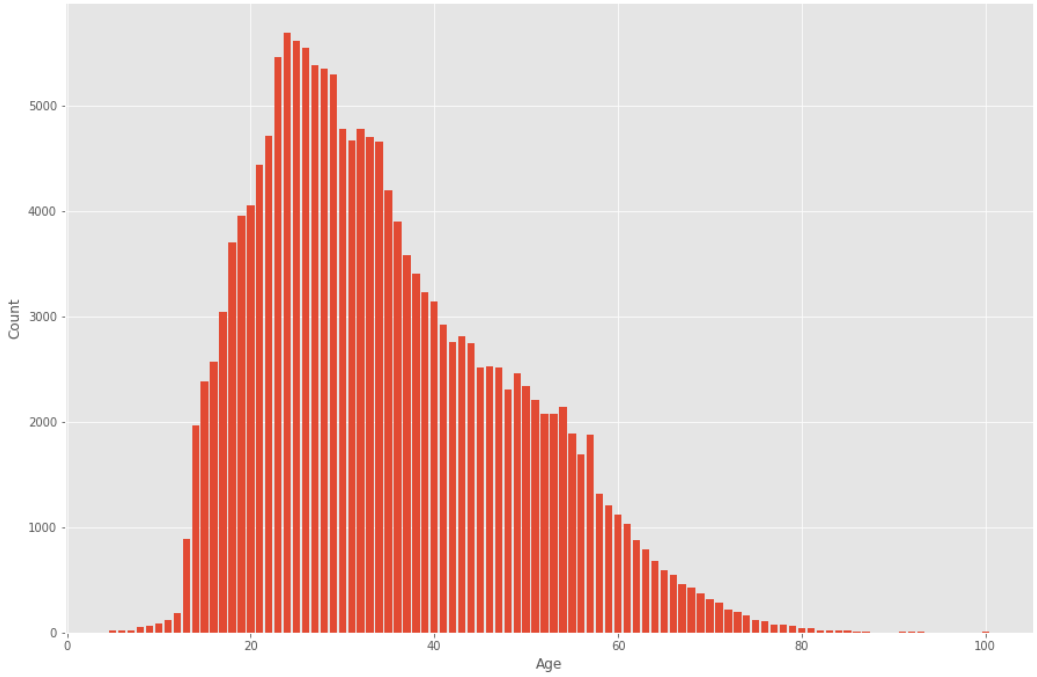

Figure 4: The above shows the distribution of the ages of users. The majority of the users lie between 20 and 40 years old.

The tables were joined on the BX-Book-Ratings table after the ISBN in it that did not correspond with ISBN in the BX-Books table were removed. It was a left join with BX-Book-Ratings as the caller due to those ratings being the target.

Recommendation System:

Before building the recommendation system, the issue of books with different ISBN numbers for separate editions needed to be addressed. If it weren't, books that are ostensibly the same would be counted as separate and therefore dilute their rating pool. This was done by standardizing the authors and titles into all uppercase then grouping by title and author and assigning a unique 'book_id' that represents all titles of any edition.

The data was then minimized to the three columns needed for collaborative filtering: 'user_id', 'book_id', and 'rating'. Initially, memory-based collaborative filtering through matrix factorization was attempted. This involves creating a user-item ratings matrix and calculating

cosine similarity or Pearson correlation coefficients. However, hardware limitations became too much to overcome, and a new approach was found.

Model-based matrix factorization proved to be easier for the hardware to handle. The Surprise package handled this. Surprise comes equipped with machine learning models that can be fit and trained such as Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF). Hardware limitations were still a factor, and the dataset was shortened to include only users who have contributed 100 or more ratings.

After running a five-fold cross validation using both models, SVD had a lower mean root mean squared error (RMSE) of 1.6355 compared to the NMF's 2.4725. After a train/test split, the SVD model scored a RMSE of 1.6416. So, the parameters of SVD were tuned. After finding the optimal parameters, the SVD model achieved a slightly better RMSE score of 1.6320.

With the tuned model, an algorithm was created that took the prediction object returned by the model and extracted the 'user_id', 'book_id', and predicted rating into a list of tuples of top rated predictions. Further, it then returned to the full dataset and translated the 'book_id' back into the book's title and produced a list of titles and ratings.

```
In [52]: top_n[277427]

Out[52]: [(121848, 9.629515536078355),
          (118782, 9.60014398415964),
          (99496, 9.57619222677314),
          (32350, 9.571841484725704),
          (44757, 9.537935247692188),
          (3420, 9.526558068686974),
          (72132, 9.501878239410088),
          (46059, 9.473703203746004),
          (933, 9.464327798325476),
          (36801, 9.459354565936092)]
```

Figure 5: The algorithm initially creates a list of tuples with 'user_id', 'book_id', and predicted rating. The above figure shows the output from the list for user 277427.

```
In [57]: example = get_reading_rec(277427, top_n)
         for book, rating in example.items():
             print(f'{book}: {rating}')

         THE TWO TOWERS (THE LORD OF THE RINGS, PART 2): 9.629515536078355
         THE SECRET GARDEN: 9.60014398415964
         THE AMBER SPYGLASS (HIS DARK MATERIALS, BOOK 3): 9.57619222677314
         DUNE (REMEMBERING TOMORROW): 9.571841484725704
         GRIFFIN & SABINE: AN EXTRAORDINARY CORRESPONDENCE: 9.537935247692188
         A PRAYER FOR OWEN MEANY: 9.526558068686974
         MY SISTER'S KEEPER : A NOVEL (PICOULT, JODI): 9.501878239410088
         HARRY POTTER AND THE PRISONER OF AZKABAN (BOOK 3): 9.473703203746004
         84 CHARING CROSS ROAD: 9.464327798325476
         FAHRENHEIT 451: 9.459354565936092
```

Figure 6: The algorithm continues and translates each 'book_id' back into the original title. The above figure again shows the output for user 277427.

Client Recommendations:

An underlying issue of this type of recommendation system is that it suffers when a user hasn't rated many titles. As such, here are the recommendations of how to move forward for John Albert's Library in Fictional Town, USA.

1. Before a user has rated 25 books, use a simply recommend the most popular (i.e. highest weighted ratings average) books to the user.
2. After a user has rated 25 books, use the tuned SVD model and algorithm to provide a list of recommended titles.
3. In that vein, encourage the rating of books through easily available and understandable online tools to do so.