Capstone 2 – Book Recommendation System

I've come across a list of 278k users who have implicitly or explicitly reviewed 271k books. I'd like to build a recommendation system to recommend the right books to the right users. I've attempted a collaborative approach along with a random forests regressor to predict whether a user will read a book.

The Customer

My client for this project could be any retailer who sells books or a platform like Good Reads that allows users to follow each other and get recommendations on what book to read next. After this analysis, retailers should be able to offer better products to their customers and increase lifetime value or sales per customer metrics while Good Reads can demand higher advertising revenue by proving they can put publisher's books in front of the right audience under their sponsored content section.

The Data

I'll be using the "Book-Crossing" data set win which more info can be found at http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

> *Improving Recommendation Lists Through Topic Diversification*,
>
> *Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, Georg Lausen; Proceedings of the 14th International World Wide Web Conference (WWW '05), May 10-14, 2005, Chiba, Japan.*

Overall there are 1.1M reviews by 278k users across 271k books, most are implicit (not rated or in the case of this data, 0) while some of explicit ratings of 1 – 10. The data also contains other dimensions, such as the user 's physical location, book's publisher, year of publication, and thumbnail to the image.

Cleaning the Data

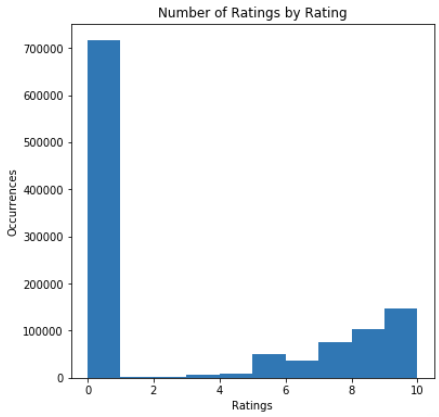
There were some initial issues opening the data. It was stored in a csv with a semicolon separator while book titles contained semicolons in their names. Also, ampersands were converted to "&" which also added to the complexity of opening the file. This was easily overcome by opening the file with the python open function, replacing the "&" occurrences with "&", and using regex to find semi colons located in between the open and close quotes of the file and replacing it with "||". This made the data much easier to split and load into pandas.

Once, in pandas, I found there were about 4600 books without a year associated with it (about 1.8%) and are annotated with a 0. If we end up using year as a feature to predict a top n book, we may have to drop them from the analysis or use the median year of the set.

Many titles and users are from different countries as well. For my purposes of interest, I'll focus on users from the united states only.

Looking at some of the feature data from the books, there are many areas to clean up. Authors are often misspelled and may be listed first name/last name or last name/first name. The same goes for publishers of those books.

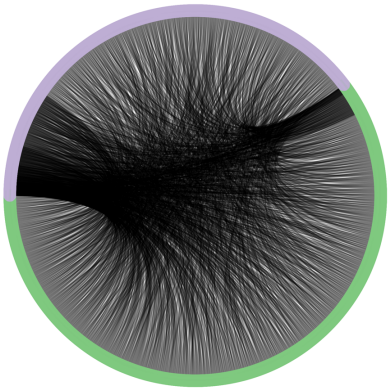
If we are to try and predict based on ratings, we need to take into account the fact that more than 70% of ratings are 0, or implicit. This is an issue with most rating recommendation systems as many users will not rate their product after purchase.



There is something to be said for users who were moved enough (or active in the community) enough to rate their books. We'll see if there is some ratio of explicit rating to implicit ratings that may help increase our model.

To help increase the accuracy of my model, I may want to grab additional information about the books being rated such as genre and a possible summary to add more features about books that users buy/read.

Plot of users (purple) and books (green) and how they are connected



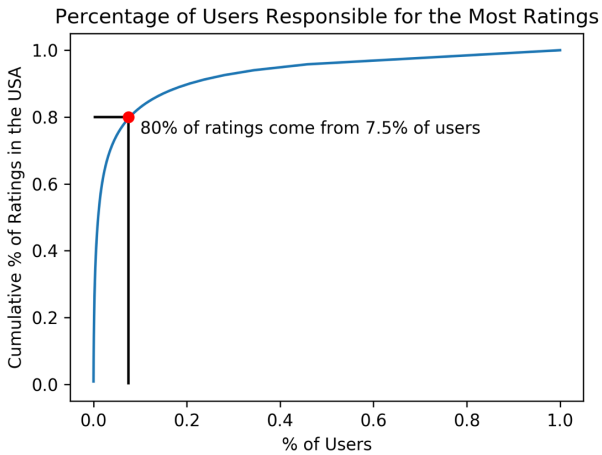
Early Findings

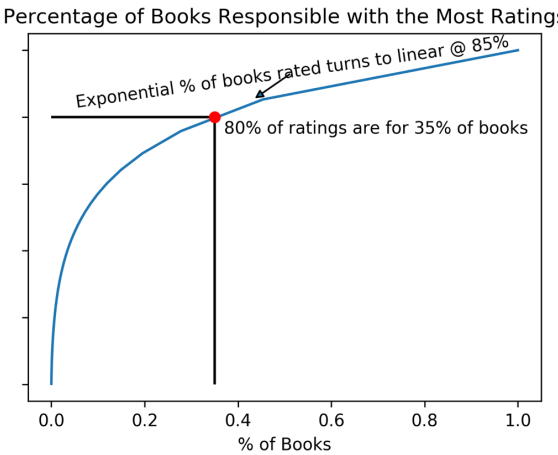
I decided to use a graph data structure to model this data. The data is partitioned (bipartite will be used as the partitioning term) by books and users.

The visualizations that follow are a sampling of the data as there is too much data to show all at once.

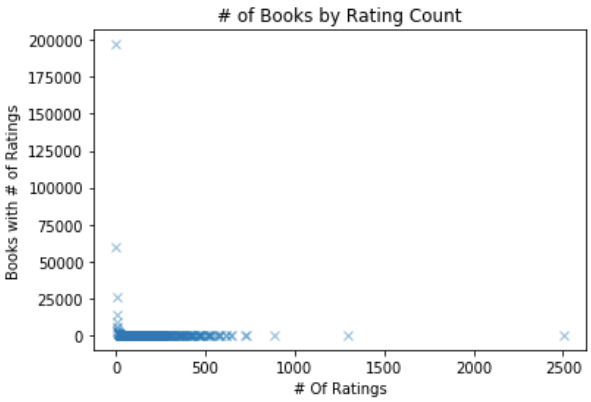
In the Circos plot on the left, we are able to identify users by the color purple and books by the color green. Although you can't see it, all user nodes (Nodes: each individual occurrence of customers or books) flow into the book nodes. Each line represents a rating in our data.

What we see is that there are a large amount of ratings coming from a fairly small number of customers with most of the reviews flowing into a small amount of books. In fact, about 80% of reviews come from 7.5% of users and 80% of reviews flow into 35% of books. This may indicate that most activity is generated by a few key users and we may want to focus our models on those users who are active on this service.

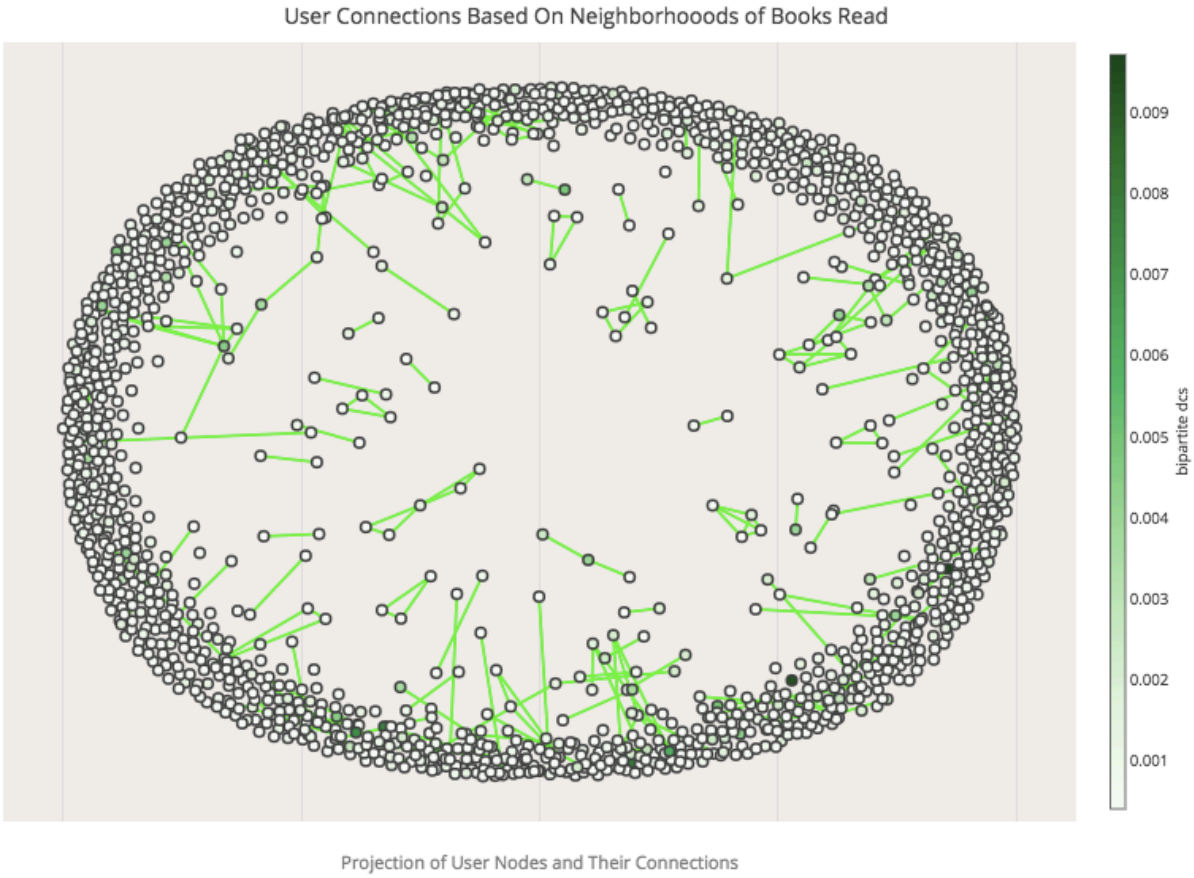




Looking at the number of books by ratings is important as well. Over 57% of books have one rating while over 75% of books have less than 3 ratings. As we're looking at using a collaborative recommendation system, these books do not add a lot of value, especially those with 1 rating. These books may never come up in a top nth predication and could slow the algorithm down. We may want to think about filtering these out.

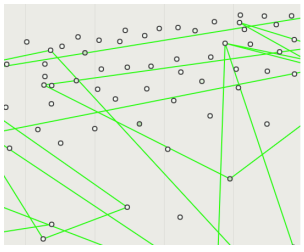


Next, I took a projected graph by bipartite for both users and books.



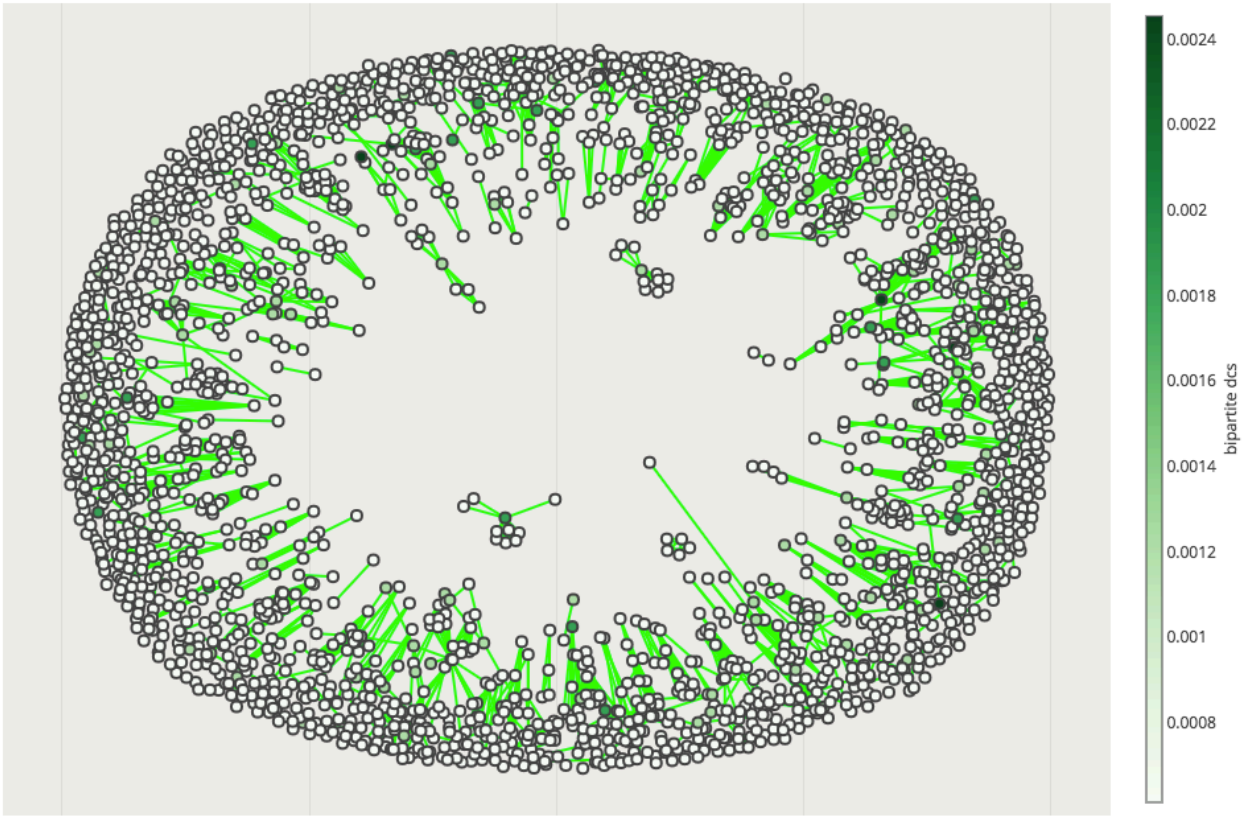
Here we see user nodes represented by the circles and colored based on their bipartite degree centrality, or by how many books the user rated over the total number of books. The nodes are connected by lines based on whether or not they have a rating for the same book whether it's implicit or explicit.

This visualization was a bit surprising to me as I expected to see the higher degree centrality nodes to be more connected to other nodes. In fact, there is practically no correlation with a Pearson coefficient of .15. When you zoom in to the outer regions of the plot, you can see mostly no users being connected.

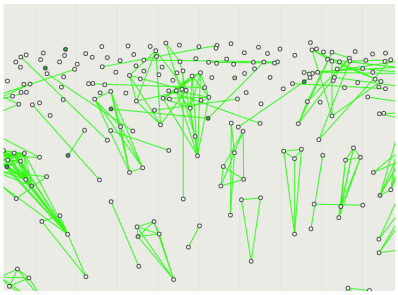


Looking at book nodes that are connected, we see a slightly different story. The connections are based on two books are read by the same user.

Book Connections Based On Neighborhoods of Readers



Projection of Book Nodes and Their Connections



There are more connections here because a connection between books is made when any user reads two books. This does not rely on users having books in common to make connections like the prior visualization.

When zoomed in, we still see more connections albeit still a high count of nodes without connections.

Although there are more connections, there is still no correlation between degree centrality and number of connections as there are still mostly 0 connection values. We find higher correlations (although still low) with the betweenness centrality score of the nodes and the number of connections being .25 and a correlation of .14 between the number of connections between books and their betweenness centrality.

I'm going to start with a collaborative filter recommendation system and move into a content based recommendation system as well. My goal will be to maximize the % of books that show up in a top 10 recommendations filter.

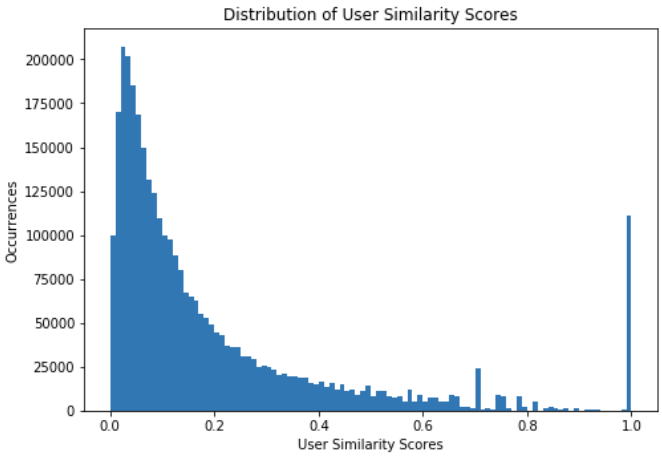
The Model

Collaborative Filter

My first step was to find the cosine similarity scores between all users. Using the networkX bipartite.biadjacency_matrix, I returned a sparse matrix of users by books with ratings as the intersection values. Next, by using the cosine_similarity from the sklearn.metrics.pairwise object, I returned a user by user matrix with the cosine similarity score between any two users. I used this as my similarity score.

The actual filter returns books read by the top n users (based on the cosine similarity) that have not been read by the user I'm attempting to recommend books for. Initially, the recommendation list was ranked by the number of times a book appeared within the top n similarity users.

My test statistic at this point was % of books in top 10 recommendations that were actually read by the user. This changes to a precision score later. To test the accuracy of the model, I randomly withheld 20% of book ratings (hold out set) and built the model on the other 80% (training set). I can measure the model's accuracy based on the model's performance on the test set.



Using the naïve recommendation system above, my model had 0 accuracy. As you can see on the graph to the right, most users share a similarity score of below .20, which is fairly low.

To overcome this, I engineered features that I thought would predict if books that were actually read rose to the top 10 of books recommended and tested both a random forest classifier and a logistic regression classifier for which books would actually be bought/read.

Feature Engineering

I engineered 12 features that I thought would be of importance to recommending books for users to read. They include:

Average rating: Mean rating of the book from the top n users. This feature is lower if users don't take the time to explicitly rate the book and could skew the results.

Ratings: Sum of all ratings from top n users that bough the recommended book

Average explicit rating: Mean ratings of the books for users that actually rated the books (value of 1-10). If more users took the time to explicitly rate this book, it may be more predictive of if a book should be recommended.

Explicit/Implicit Rating Ratio: Sum of all ratings from top n users that bough the recommended book

Implicit ratings: Number of implicit ratings of a book. Does a high number of implicit ratings mean the book is read less?

Average cosine: The mean similarity score of users that read this book. The higher the score, the more similar you are to the other users that read this book.

Cosines: Sum of all user similarity scores. Same theory applies as the average cosine score.

User count: Number of top n similar users that read this book. Main component of the naïve recommender.

Ratings: Total ratings for a book from the top n similar users. Does a higher rating predict if the book will be read?

Platform popularity: The count of users on the platform that bought/read the book. My question here was is a book that's more popular overall have more weight than in a user's network?

Platform Average Rating: The mean rating of a book on the platform. Same question applies as in platform popularity.

Platform Popularity Percent: Count of users who bought a book divided by the count of users who bought the most popular book. This may be a bit redundant to platform popularity, but I wanted to see if the distance from the most popular book on a scale of 1 mattered.

Classifier Model

Using the above 12 features, I wanted to build a classifier model to more accurately predict which books would be read by a user. I went through 5 steps.

1.) Split the training set into a 75% training set / 25% cross validation set
2.) Set the top N user similarities to 5
3.) Set the number of books to return from the naïve recommender to 300
4.) Train my models (logistic regression & random forests classifier)
5.) Cross validate the model on the test set
6.) Final test of the model on the holdout data

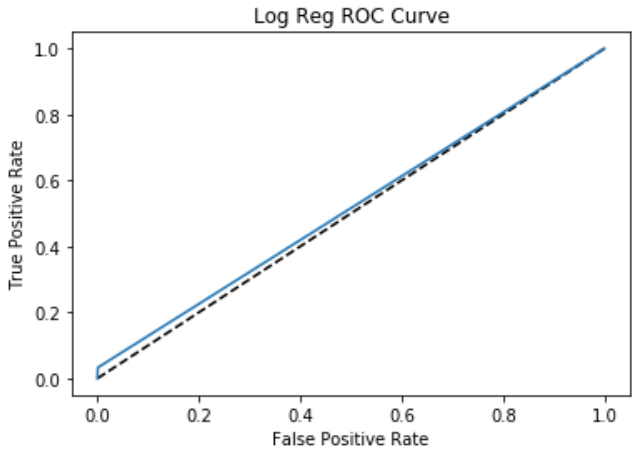
At this point, I decided to switch my test metric to the precision score from a confusion matrix. My goal was to have the books that were recommended from the model have been actually read by the user from the test set. The precision score is perfect for this. The higher the precision score means more books predicted as read we're actually read. The inverse being predicted books we're not actually read.

Using logistic regression, I was able to obtain a precision score on training data of 0.55 with cross validated scores between .25 and .625.

```
Logistic Regression Cross Val Score on Precision Metric
Overall model score: 0.550
Cross Validated Model Scores: [0.25       0.625      0.57142857 0.5        0.4        ]
```

Finally, the score on the hold out data was .50.

The challenge with this data set is most books are overwhelming not read by any given user. This lowers our recall score of books that are true purchases because there is much more data to be trained on unread books. You wind up with a ROC curve like the one on the right. You have a small amount of true positives rate which come from the precision score, but it misclassifies the majority of books that are read as not read.



Looking at the logistic regression coefficients of our features, it seems that total user similarity (cosines) raises to the top, far beyond any other feature. This makes sense to me as it combines both the total count of books read by users like you and it measures total similarity to those users. On the opposite end is user count. User count is contained in cosines and is probably placed so low due to high correlation to cosines.

	Feature Coefficients
cosines	1.789557
explicit_implicit_ratio	0.296904
ratings	0.223231
avg_cosine	0.220017
implicit_ratings	0.167845
platform_avg_rating	0.108834
avg_explicit_rating	0.079091
platform_popularity	0.003051
platform_popularity_popularity_percent	0.000003
avg_rating	-0.128231
popularitymetric	-0.240650
user_count	-0.673909

Next is the explicit ratings count to implicit ratings count ratio. This also makes sense as if a book is good enough for more users to explicitly rate it, more users will end up buying it due to the higher count of ratings.

Finally, looking at the other two negative coefficients, we could probably place those on correlation with other metrics such as average rating with other ratings metrics and popularity metric with the other two popularity features.

My next model I attempted was a random forests classifier. First, I performed a randomized grid search on the parameters n_estimators, min_samples_split, criterion, and min_samples_leaf with the results of 19, 3, entropy, and 7 respectively.

The model has a better precision score on the training data, but it does not generalize well as seen by our cross validation. Even though the score on the holdout data is .50, I would still tend to go with the logistic regression model.

```
Random Forests Cross Val Score on Precision Metric
Overall model score: 0.817
Cross Validated Model Scores: [0.         0.4        0.18       0.66666667 0.625      ]
```

Looking at the feature importance's, we see cosines rise to the top as the most important feature. This is similar to the logistic regression model. Interestingly enough, the explicit/implicit ratio falls to the bottom of feature importance.

```
Feature Importancs

cosines                                  0.154081
popularitymetric                         0.148728
avg_cosine                               0.137946
platform_avg_rating                      0.123616
platform_popularity_popularity_percent   0.122939
platform_popularity                      0.117514
ratings                                  0.075619
avg_explicit_rating                      0.049482
avg_rating                               0.032618
user_count                               0.017601
implicit_ratings                         0.017207
explicit_implicit_ratio                  0.002650
```

Other Thoughts

Limitations

There are a few limitations to this model. The largest being the poor recall score on true read books. Other limitations include only being built on a user collaborative filter. We don't have a lot of information on the user and the info we do have is very messy.

Some options to help better my model can include:

- Finding appropriate age buckets to better recommend books that certain ages are more likely to read (young adult novels are probably read by those 15-22 more so than those who are 45-55).
- Filter the data to perform only for the most active users
- Clean up the geography data by using fuzzy matching. Are different books read by those in the north east vs those who reside in the south west?
- Build a segmentation of users based on books they've read and only pull recommendations from their segment
- Other data the platform might be able to aggregate that would help could include:
    - Gender
    - Household Income
    - Profession
    - Race
    - Etc.
- Building a book similarity score that recommends books based on how similar they are

Summary

In this project, I performed network analytics on a user and book review data set to identify similar users based on books they've read with the goal of predicting books which books the user would like to buy and or read. We found that most users don't engage much but most of the engagement is from a few users, which is common in most platforms.

Using a naïve filter based on a user's neighborhood and combining with a logistic regression classifier, I was able to generate a precision score of .5 that is generalized fairly well. While a score of .5 is not ideal, it gives a starting point to iterate from. The current model points to user similarity and explicit ratings as a starting point to predict what a user will end up reading.

The biggest issue to solve will be increasing our recall score. Because most books are not read often by very few users, they will not be predicted as read when in reality, they have been read in our test sample.

For the next round of analysis, I suggest,

- After running the naïve filter, we may duplicate the books that have actually been read before running it through a classification model. This will give the model more signals of when a book should be predicted as read and may increase our recall score.
- Building an item based collaborative filter and combining the prediction outcomes with the user based collaborative filter.