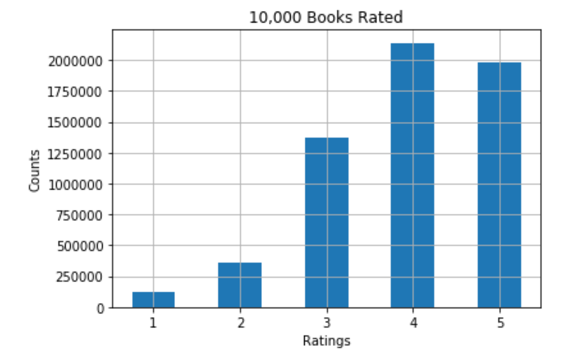
Book Recommender

The original problem statement that I came up with is: How might we predict the market demands for each book genre based on the sample data we have obtained from Goodreads website. This problem is important because the market demand for a specific book genre might be low but the market supply is high, or vice versa where the market demand for a book genre might be high yet the market supply is low. In another word, my capstone project was trying to reduce the scenario where many readers out there like a specific book genre yet not many writers are interested in writing about them or publishers refuse to publish those books.

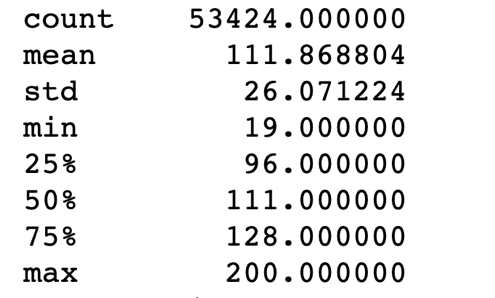
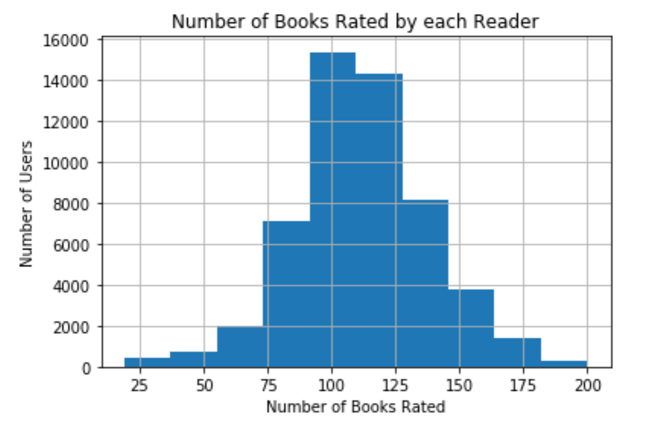
I was going to use the book review information and author review information provided by Goodreads site to close down the gap between the market demand and market supply for each book genre. My clients for this project would be the book writers and book publishers. Publishers would use my report to make more informed decisions on whether to say yes or no to a book genre, to a book or to a book author thus increase the publisher’s reputation as well as its profit. Writers would use my report as a reference to decide which topic or genre to write about for their next book and have a clear idea on the odds that their books would make a profit or become a best seller.

Later, I realized that there are too many factors that influence how well a book is selling. Thus I revised my problem statement. My problem statement now is to build a system that can automatically recommend books to users based on the preferences of other readers. Instead of using the datasets that I previously analyzed, I would be using different ones but similar. The new datasets contain 53,000 users’ reviews on 10,000 books with over 34,000 unique tags describing these books.

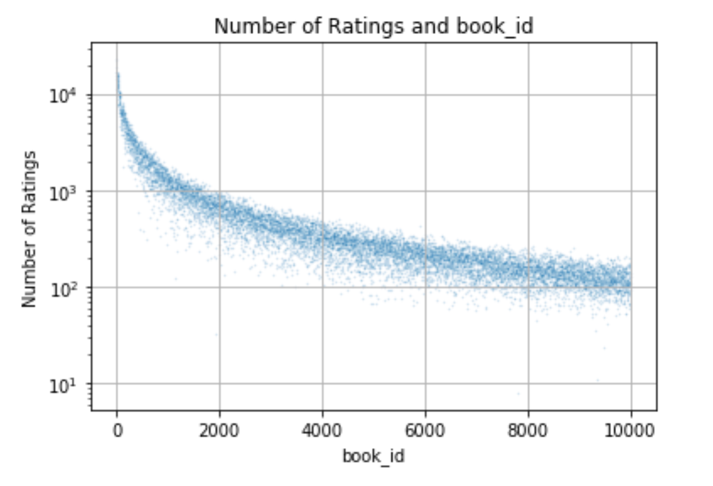
I acquired the Goodreads dataset from Kaggle. The dataset was fairly clean. All I had to do was to drop some unnecessary columns. After I did my analysis, I found some interesting insights.



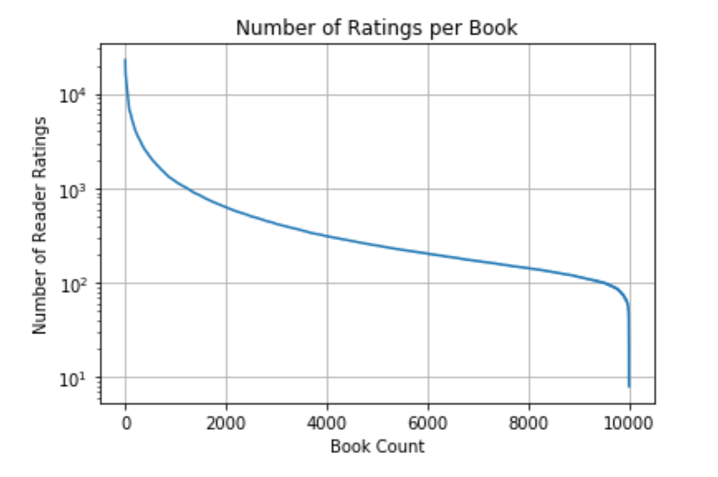
The percentage of 4 or 5 ratings accounts for nearly 69% of the total ratings.



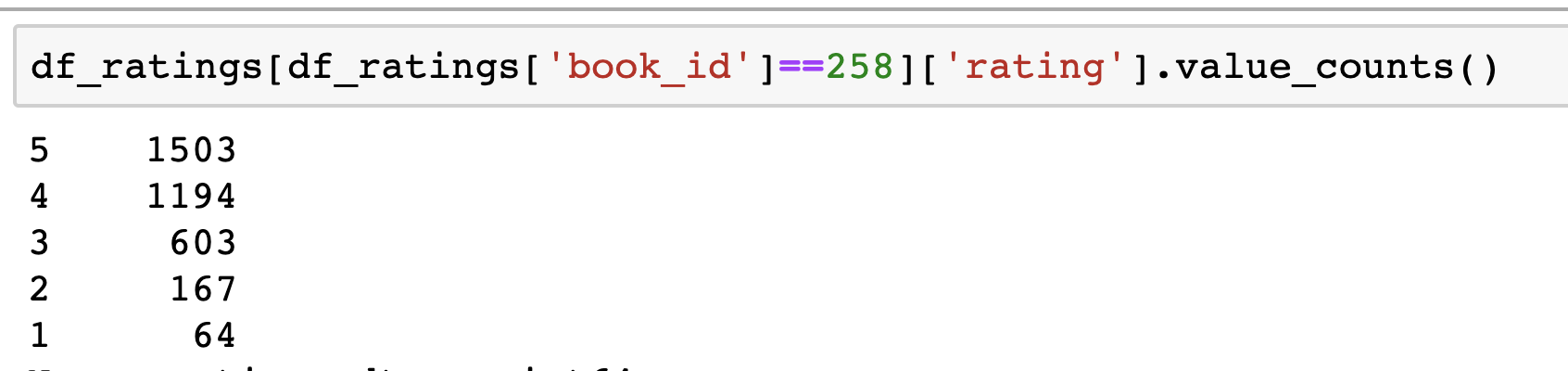
All the readers have rated at least 19 books, with a maximum of 200 and median of 111.



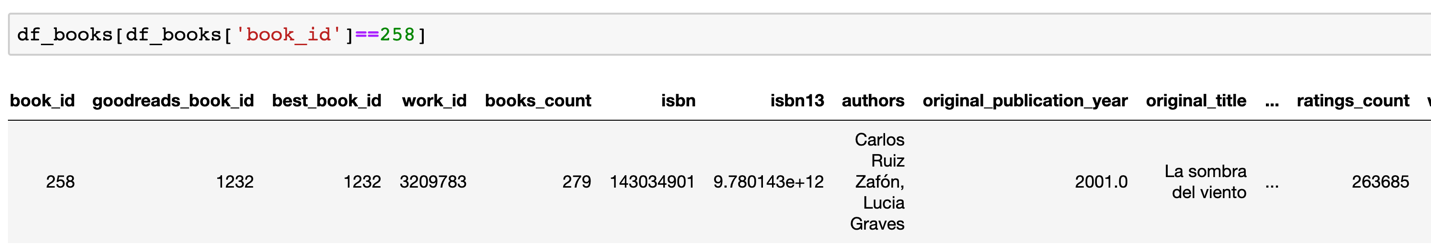
The book\_id is strongly correlated with the book’s total number of rating count. The larger the book\_id is, the higher its total number of rating count is



Most books have been rated by 100 to 1000 readers

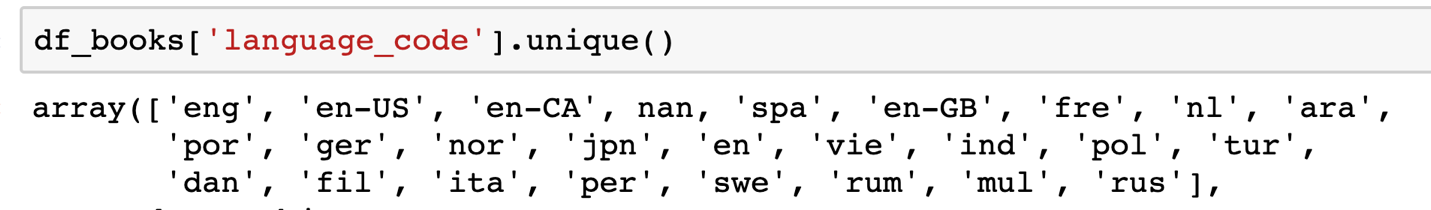


The rating count for book\_id 258 from ratings dataset

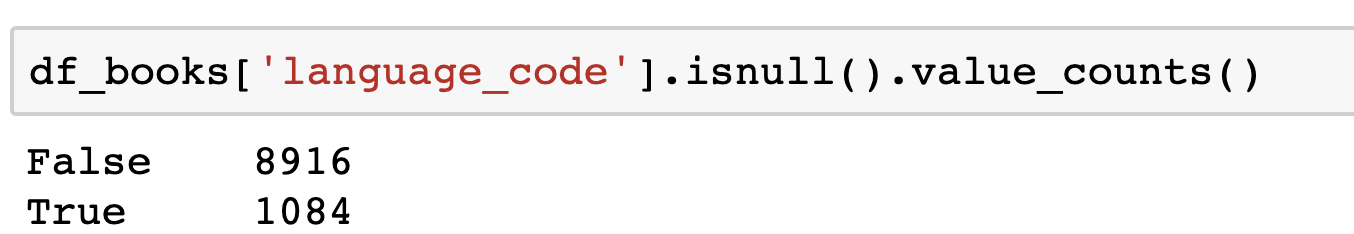


The total rating count for book\_id 258 from book dataset

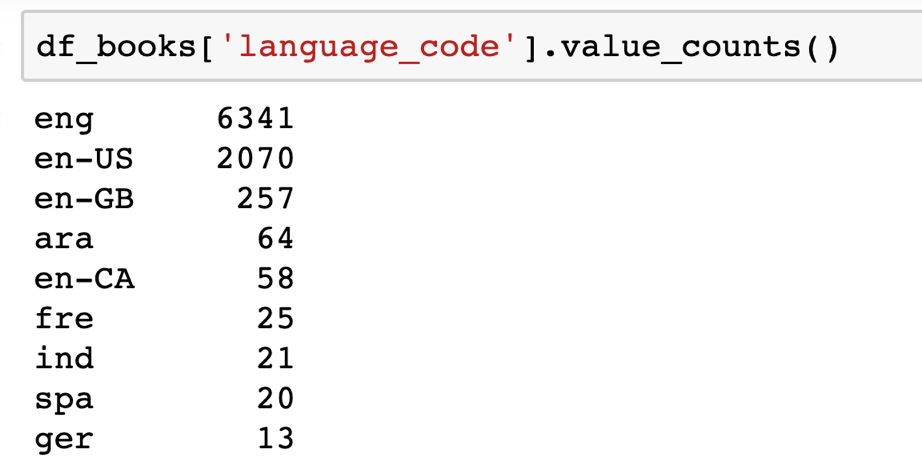
The rating count from ratings dataset and book datasets are different because rating dataset refer to the registered 53,000+ readers; whereas the ratings in book dataset include from non-registered users.



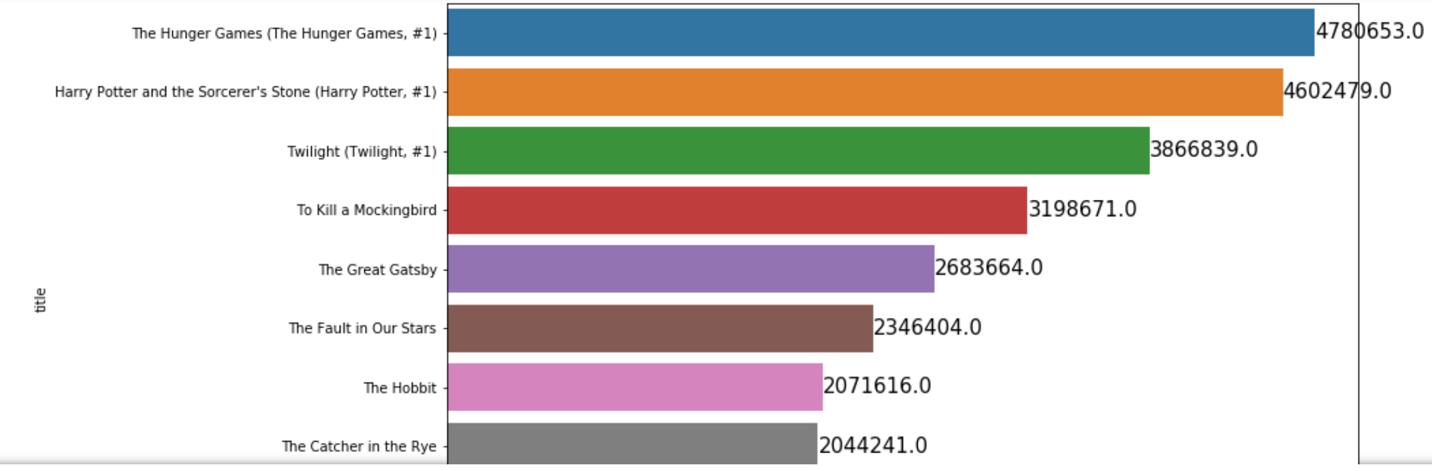
All the language exist in the book dataset

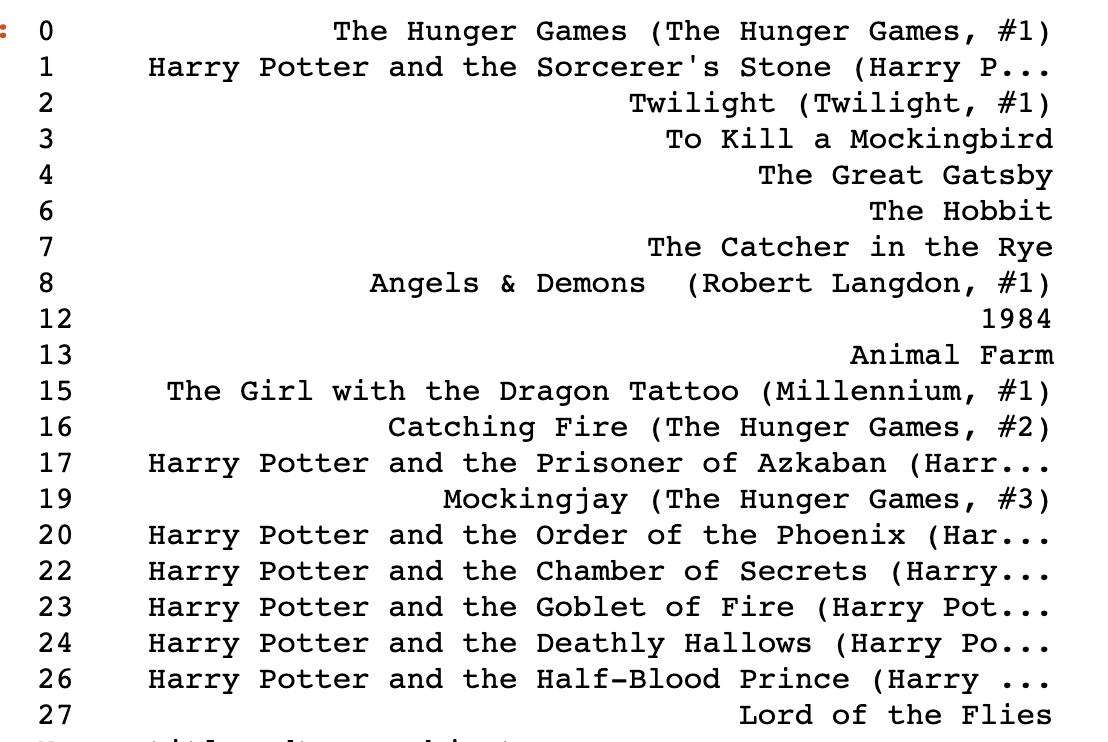


Out of these 10,000 books, approximately 1100 of them do not have language code information



Of the 10,000 books, there is info on the language code for 8916 of them. And majority of them are English.



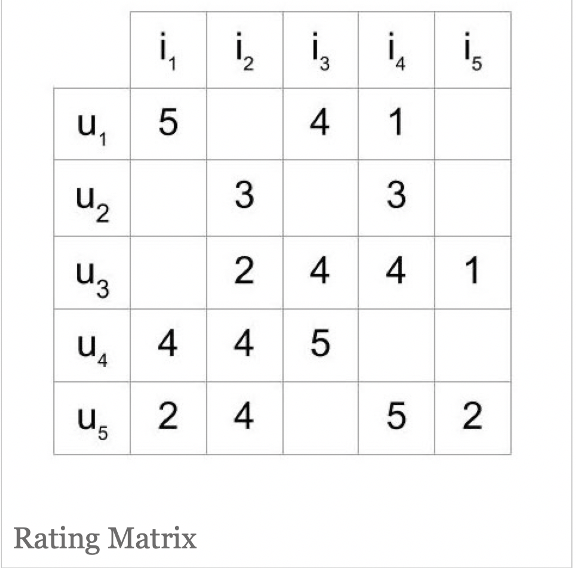


These are the most rated books

I applied the machine learning technique, user based collaborative filtering, on my datasets. Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. Companies like Amazon, Netflix and Youtube included the collaborative filtering technique in their sophisticated recommendation algorithm.

The first step of the recommendation algorithm is to identify the user’s preferences. It asks the user for a list of books that he or she has read and the number of rating were given on a scale of 1 to 5. In addition, the genre of books the person prefer. The algorithm would use the information user provided to create an user profile.

The second step, the algorithm would create a matrix that contains all the readers, all the books and the number of ratings each readers give to a specific book. Later, it would try to identify which readers are similar to the user based on the ratings the other readers give to those books that the user has rated. The algorithm then uses cosine similarity to determine whether two readers are alike. If the angle between the lines is increased, then the similarity decreases, and if the angle is zero, then the users are very similar.





The third step is to calculate the book ratings weighted mean average. If the reader has not rated the book, it is excluded from the mean. It gives higher weighting for readers who are more similar to the user and lower weighting for readers who are less similar to the users.

The fourth step is to identify the books to recommend. The recommended list will not include the books the user already read. It will sort the remaining books by their weighted mean average. The highest one will show at the top.

The last step, it is to calculate the accuracy of the book recommendation system. The accuracy is measure by the formulas: the number of correct recommendations divide by the number of reader’s highly rated books. For example, if a reader has rated 10 books in total for a rating of 5 and the engine is able to include two out of these ten books in the recommendation list, then the accuracy of the book recommendation system is two out of ten. If the highest ratings an user has given are 3’s, then it would count the total number of 3 the user has given, and try to predict how many of these books with rating of 3 are included in the recommendation list.

For this algorithm, there are still many room for improvement. Some of the improvement include:

A more user friendly user interface

The accuracy of the recommendation engine