Business Problem

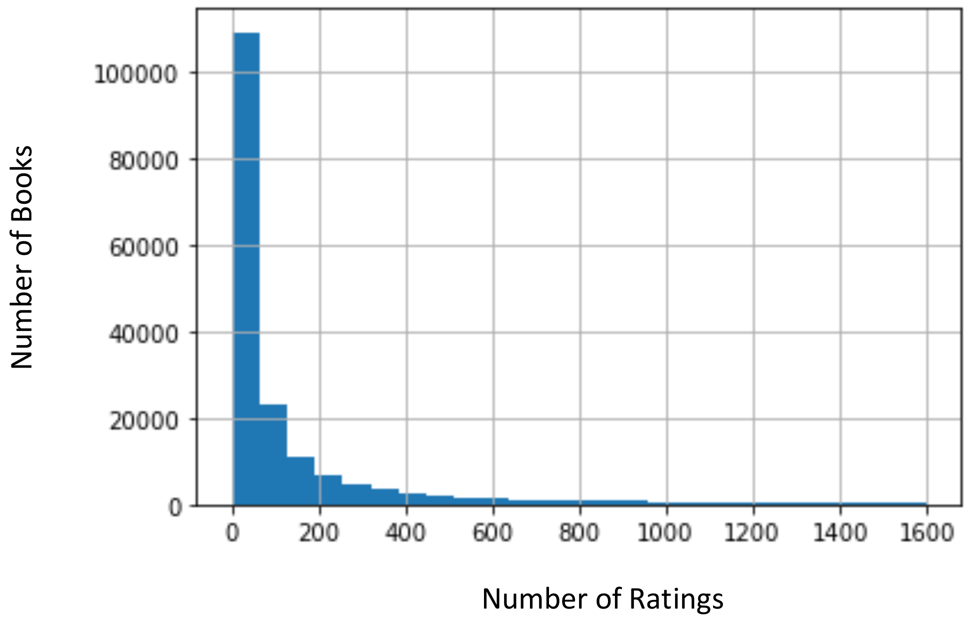
A recent increase in shipping prices is jeopardizing BigOnlineBookStore's trademark $1 shipping costs. They are looking to increase sales to keep shipping at just $1 as their analysis shows the shipping cost is a huge driver for their loyal customers and is responsible for them overtaking the market share of physical books sold online. They are looking to implement a recommendation engine to try and engage users to consider more books. They already have a repository of user information as they track ratings users give, along with a data on the books themselves. BigOnlineBookStore is also extremely concerned about operating cost, so they prefer a simple, fast algorithm with decent results as opposed to a super accurate one which would cost a lot in compute power.

Data

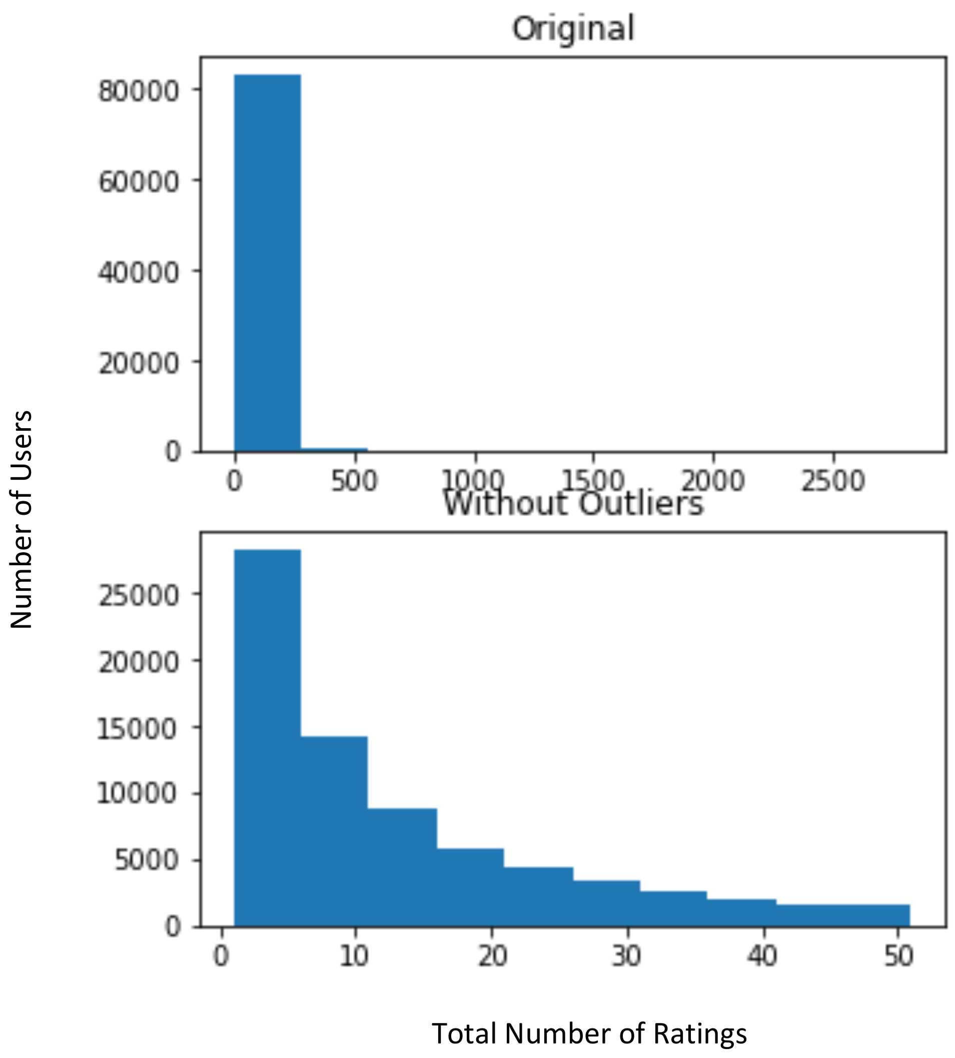
The data used is broken into two sections: book data which includes book_id, author information, series information, format, and others along with user data which includes books read, the rating given, text of review (if present), and similar data.

Data Cleaning & EDA

First, data was cleaned replacing NaN values with averages of the same books, pluggin in the minimum year of multiple books when an original year was unavailable, and basic statistics were taken. It is of note that most books do not have much for ratings:

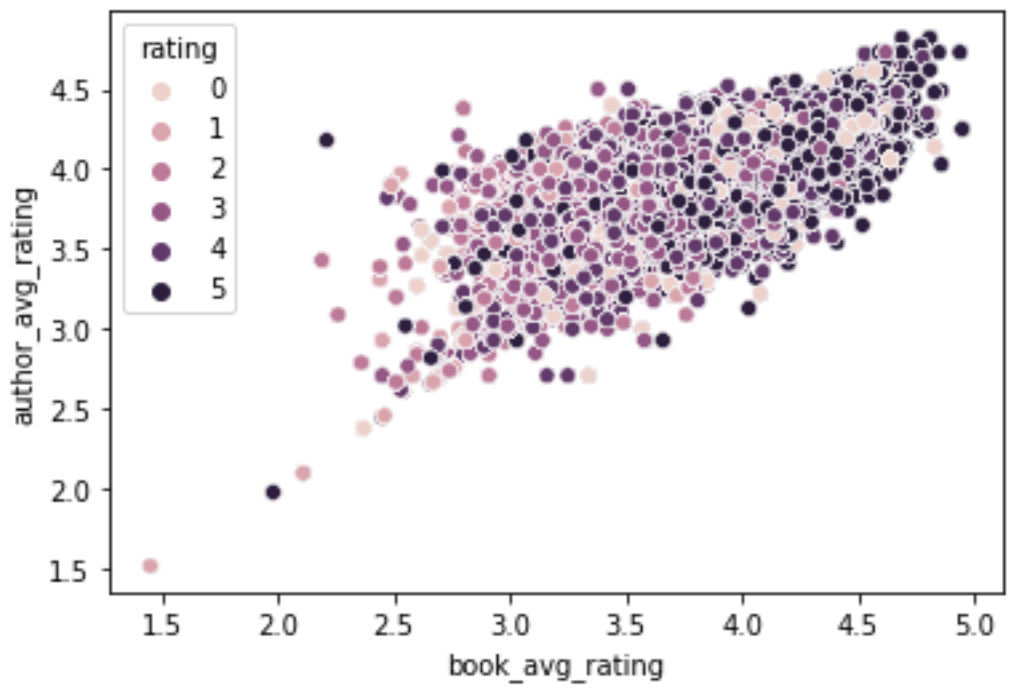


and most users generally have a lower number of ratings:

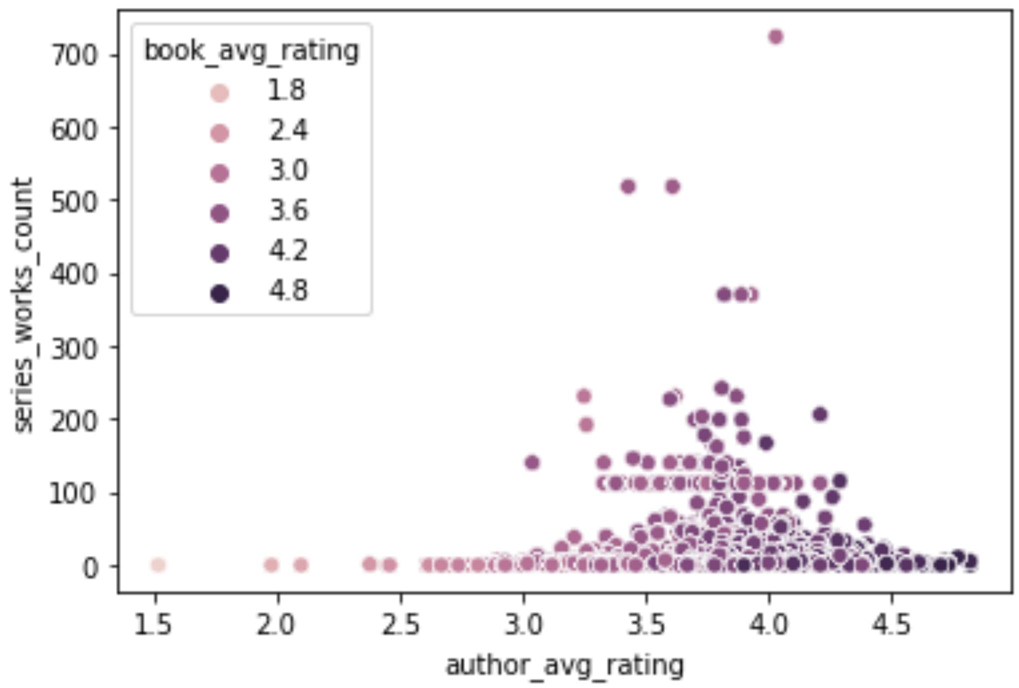


It was decided that the best book_id (the one with the most ratings), would be used for each work (as a book can be published in different formats by different publishers). We took into account the data only for the best_book_id. This id was then joined with the consumer interactions table using an inner_join so each entry had a user id, book id, and a wealth of other data (number pages, number reviews, ratings, distribution of ratings by count, etc.).

When looking at this data, it became clear that the user prescribed rating was really correlated with nothing else; if anything, it mostly relied on the book's author's total average rating, but even that was weak:



and another view, looking at average book rating along with series counts:



This view shows the average ratings which seem much more correlated than individuals. From this, we can speculate that group ratings are much easier to correlate compared with individuals.

Therefore, it was decided to use the Surprise package, which takes into account only user id, book id, and rating. The skinny dataset helps ensure the algorithms are simpler and hopefully will address both the needs of BigOnlineBookStore and keep their operating costs low.

Algorithms and ML

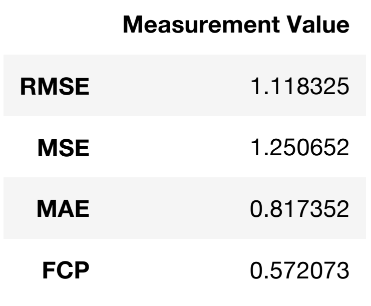
5 algorithms were initially tested: SVD, SlopeOne, Baseline AGS, Baseline SGD, CoClustering.

SVD and the 2 Baseline performed the best, so those 3 were further analyzed. The baseline SGD algorithm performed near equally to SVD at a fraction of the compute cost. See Metrics below:

	RMSE	MSE	MAE
NormalPredictor	1.723034528	2.968847986	1.338750646
BaselineOnly_als	1.121923214	1.258711699	0.815304353
BaselineOnly_sgd	1.119968207	1.254328784	0.818951347
SVD	1.119882651	1.254137152	0.801898207

Predictions

Final model metrics:



	Measurement Value
RMSE	1.118325
MSE	1.250652
MAE	0.817352
FCP	0.572073

Most predictions were within less than 1 star of the user rating. If a book is rated highly by a user, we do not mind if it is overrated; however, if the algorithm consistently rates a user's 0-2 star books as 4 (overrates, that is more of an issue.

Future Work

While user rating does not correlate much with book information, the book metrics do correlate with each other. Two things could happen in the future: creating a dummy rating for Surprise that encompasses more than just the user rating as well as taking the books themselves and clustering. The algorithms could work in tandem to find the best books and create a hierarchy (promoting the books most like that the user has already identified that they like).