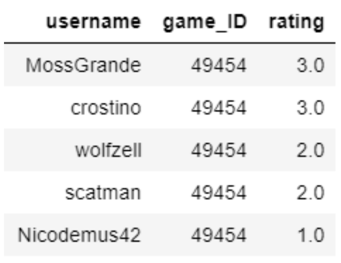
Implementing A Recommender System for BoardGameGeek

In-Depth Analysis

The technique used to build this recommender system is collaborative filtering. There are two approaches used with this technique, memory-based and model-based. In collaborative filtering, the machine learning algorithm predicts the preference of user A, for example, to an unrated item by comparing user A's historical preferences to those of many other users. The algorithm will then assume that user A's rating for the unrated item will be similar to the ratings given to that item by all the other users that have similar preferences to user A. For the analysis of BoardGameGeek's data, the data set consist of every username, every board game's ID number, and the rating given to that game by that user. Table 3 below is five rows of the data set.



username	game_ID	rating
MossGrande	49454	3.0
crostino	49454	3.0
wolfzell	49454	2.0
scatman	49454	2.0
Nicodemus42	49454	1.0

Table 3. Five rows of the data set to be converted to matrix format.

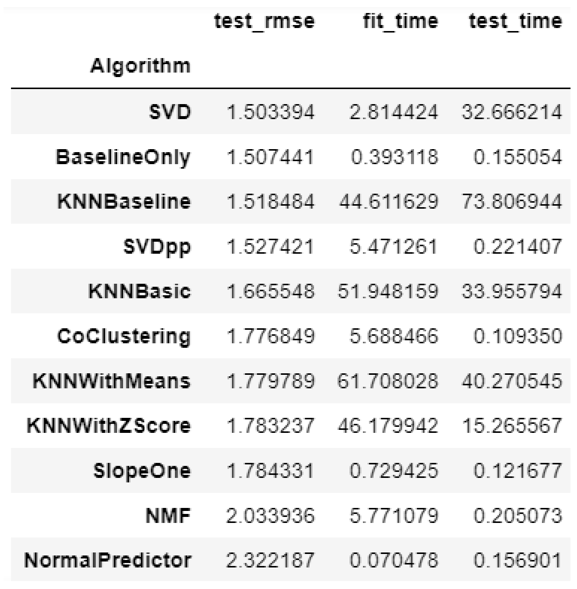
The data is then transposed into a matrix format where each row is a user, each column is a board game, and the fields are filled with the user's rating for that board game. Table 4. below is an example of the format. Given that not every user rates every game, the data produces a sparse matrix.

	game 1	game 2	game3	game 4	game 5
user 1	8			4	
user 2		7			
user 3	6		5		
user 4				6	
user 5		2			3

Table 4. Example of the matrix format.

The data set consisting of username, board game ID, and ratings had 6,669,077 rows. This large a data set was too much for the hardware available. Therefore, a random sampling of 50,000 rows was taken from the full data set then transposed to the matrix format mentioned above.

A benchmark analysis of the recommender system library "Surprise" was ran using RMSE as the test statistic. The lower the RMSE, the better the results are said to be. The results below are provided after a 3-fold cross validation.



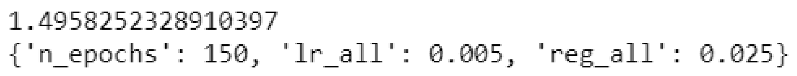
	test_rmse	fit_time	test_time
Algorithm			
SVD	1.503394	2.814424	32.666214
BaselineOnly	1.507441	0.393118	0.155054
KNNBaseline	1.518484	44.611629	73.806944
SVDpp	1.527421	5.471261	0.221407
KNNBasic	1.665548	51.948159	33.955794
CoClustering	1.776849	5.688466	0.109350
KNNWithMeans	1.779789	61.708028	40.270545
KNNWithZScore	1.783237	46.179942	15.265567
SlopeOne	1.784331	0.729425	0.121677
NMF	2.033936	5.771079	0.205073
NormalPredictor	2.322187	0.070478	0.156901

Table 5. RMSE results of different algorithms with time shown in seconds.

The BaselineOnly algorithm gives a random rating to the empty fields based on the distribution of the data set (assumed to be normal). The RMSE of this algorithm will be the standard to beat as it just predicts ratings based on normal distribution. The results from the benchmark analysis show that only the SVD algorithm provided the better, albeit marginally, RMSE.

Model-Based Approach

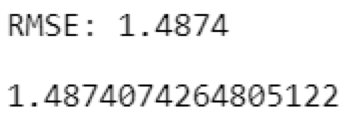
Since SVD did best, a train-test-split analysis was conducted after a grid search was performed to find the best parameters for SVD. The grid search provided an RMSE of 1.4958 with a 3-fold cross validation. The best parameters tested are in the figure below.



```
1.4958252328910397
{'n_epochs': 150, 'lr_all': 0.005, 'reg_all': 0.025}
```

Fig 9. SVD GridSearchCV result.

The parameters above were then used on a train-test-split analysis. This resulted in an RMSE of 1.4874 which beats the RMSE of the BaselineOnly algorithm RMSE of 1.507441.



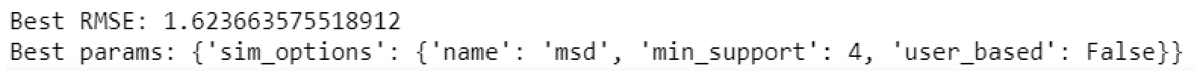
```
RMSE: 1.4874

1.4874074264805122
```

Fig 10. RMSE on the SVD test set.

Memory-Based Approach

A grid search was performed to find the best parameters of KNNWithMeans algorithm. However, the RMSE result was higher than both the SVD and BaselineOnly algorithms.



```
Best RMSE: 1.623663575518912
Best params: {'sim_options': {'name': 'msd', 'min_support': 4, 'user_based': False}}
```

Fig 11. KNNWithMeans GridSearchCV result.

The best parameters were then used to run an estimator that produces the estimated rating any chosen user gave to any chosen game.

Conclusion

The collaborative filtering method that provided the best results for the recommender system was the SVD algorithm. Its RMSE was better than that of the BaselineOnly algorithm which was used as the base model. KNNWithMeans did not provide a better result than either of the other two. The results of the algorithms, however, can be improved greatly by using more data, which is available. The full data set was over 6.6 million rows of data for board games that had 30 ratings or more. This amount of data unfortunately is too large to be feasible for the hardware available. Only 50,000 rows were able to be used in the analysis.