Introduction/Overview

This project's goal is to build a movie recommendation system that beats Netflix's. It will predict the ratings for movies that thousands of users have rated already and then check the predicted ratings with the actual ones in order to see how we did. We will beat Netflix if we can report a residual mean squared error (RMSE) of less than 0.86490.

We will use the MovieLens data set for the basis of our predictions. This data set contains more than 10 million pairing of different users with different movies. Each row of the data set pairs a user with a movie and contains the rating that this person gave to this particular movie.

Key steps that were performed in this project were honing in on four attributes of the state that could be used to make accurate recommendations. After finding those four attributes, each was regularized so as to allow for best possible accuracy.

Methods/Analysis

Before starting out with generating new code, the code provided to start this project was carefully followed. This submitters machine is running R version 4.1, so the commands provided for R 4.0 or higher were ran. The code needed for earlier versions of R were left in for graders that may be running an earlier version.

Following this supplied code carefully, the MovieLens data set was organized into a large data set with 6 columns (userId, movieId, rating, timestamp, title, genre) and about 11 million rows with each row containing a unique pairing of a userId with a movieId. This data set was split into two sets. 90% of the data was put into a data set named "edx" and the other 10% was put into a data set called the "validation" set. The instructions were careful to supply the code to ensure that no userId or movieId that was in the validation set was not in edx as well. Without a userId and movieId to guide our algorithm in making a recommendation, NAs would result, thus this step was vital.

After following these directions carefully, the edx data set was further sub- divided into a test set and training set. 90% of edx was put into the training set and 10% into a test set. Of course, all userIds and movieIds contained in the test set must also be contained in the train set to avoid NAs.

The train set would be used for building an algorithm while the test set was used to find out the residual mean squared error (RMSE). The stated goal is to generate an algorithm with the RMSE of less than 0.86490.

Critically, before starting out with generating an algorithm, a baseline was set. The average of all the ratings in the train set was found to be around 3.5, and using the average to predict all ratings was found to result in a RMSE of close to 1.06. This means every addition to the algorithm this submitter is presenting here must make that RMSE smaller than 1.06.

Using the rule of thumb that only the best survives the test of time, the first attribute that was tried as a basis of improved predictions was the year. Unfortunately, the year data was contained in a parenthesis inside the title. This means that they data had to be split out. Thankfully the stringr library had the necessary tools to split the year data found in the parenthesis. This data was split off into a character value and then joined with the train set and test set.

After adding this seventh column to both the test and train set, this submitter took the average of ratings for each year in the train set and then subtracted the average of ratings from it. This amount was added to the average rating and then used to predict all ratings in the test set. The resulting RMSE from using this predictor was only a few decimal places better than the baseline, but still helpful none-the-less. This year predictor was called the "Classic Effect".

Following the above process, each of the following predictors were added to the algorithm: genre (the average of the ratings for each genre or "Genre Effect"), user bias (average of the ratings that each user submitted or "User Bias"), and movie rating (average of the ratings that each movie has or "Rating Factor").

Of these four predictors, the one have the most effect on RMSE was User Bias and Rating factor, which when each was applied resulted in the RMSE dropping by 0.07 on both occasions. However the result of all four predictors left the algorithm with an RMSE of 0.8688785, this is better but of course not the goal.

At this point, each predictor was then regularized to remove the effect that small sample sizes had in our algorithm, or to give proper rate to each pred- ictor based on the amount of data underpinning that prediction.

A function was created to find the best tuning parameter, which we call lambda. This function included each of the four predictors and then cross-validates with the train set to find which lambda will give us the smallest RMSE. The best lambda resulted in an RSME below the goal!

Results

Our winning lambda was then added to each of predictors, and then run on the edx set, which also needed the year data split into another column.

Finally, the validation set also was adapted to add a year data column, and the algorithm that was run on the edx set with the four predictors that had been regularized with lambda resulted in predicting ratings on the validation set with a RMSE of 0.8648142! This of course, exceeds our goal!

Conclusion

In summary, it was found that the four predictors of year, movieId, userId, and genre all could be regularized to accomplish the goal of building an algo- rithm that predicted movie ratings with a residual mean squared error less than 0.86490. However, there is much more work that could be done. The genres column often contained many genres in each row. These could be split off and further honed to result in an even lower RMSE. Also, there is probably a way to use the ratings that each user has given to determine each user's age, family size, perhaps even underlying beliefs. If we were to look at this data set with the eyes of trying to use it to determine who the person is that gave each rating, that would be another exciting project.