# MovieLens Report

## Introduction

I set out to make a movie recommendation system based on the MovieLens dataset, which provides data on many users and how they rated many different movies. For my analysis I am using a smaller subset of the dataset as my computer does not have the memory or processing power to handle the larger dataset. You can find both datasets at

http://files.grouplens.org/datasets/movielens/ml-latest-small.zip

and

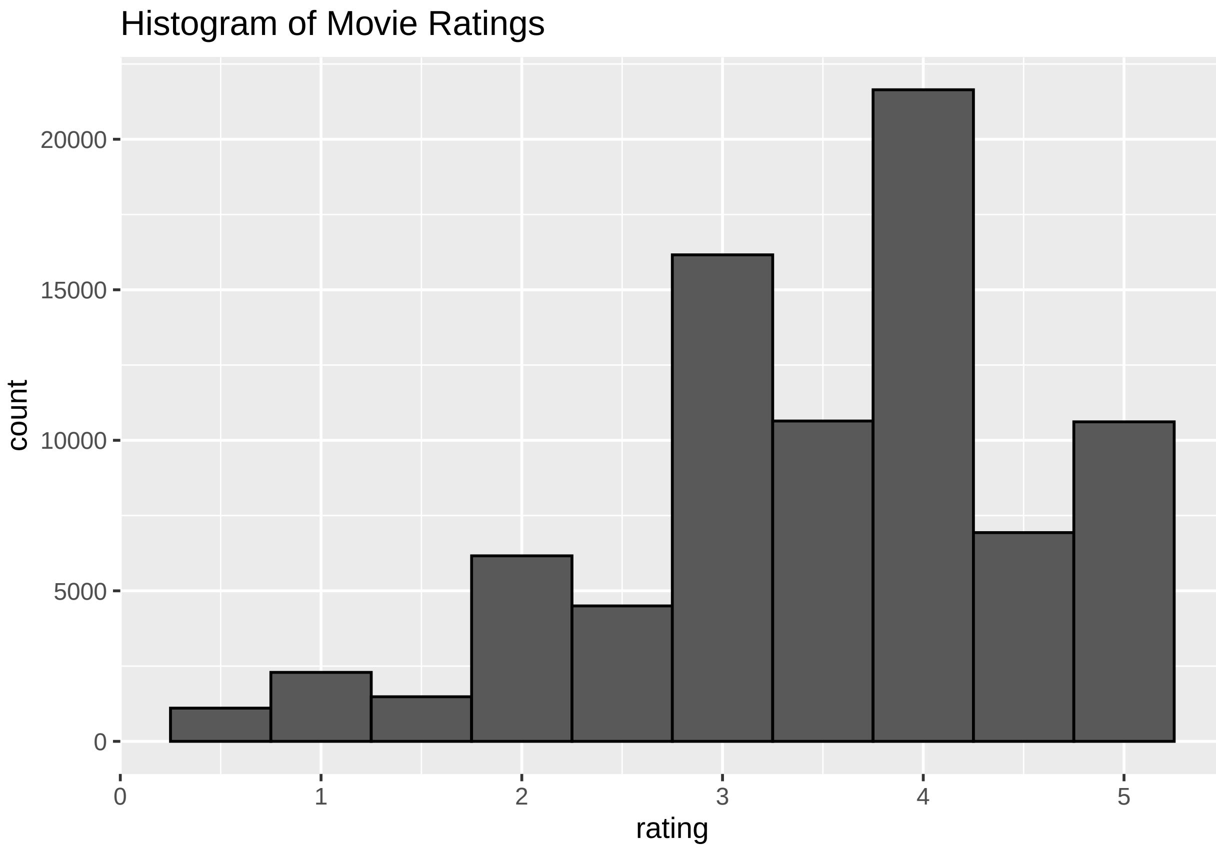
http://files.grouplens.org/datasets/movielens/ml-latest.zip

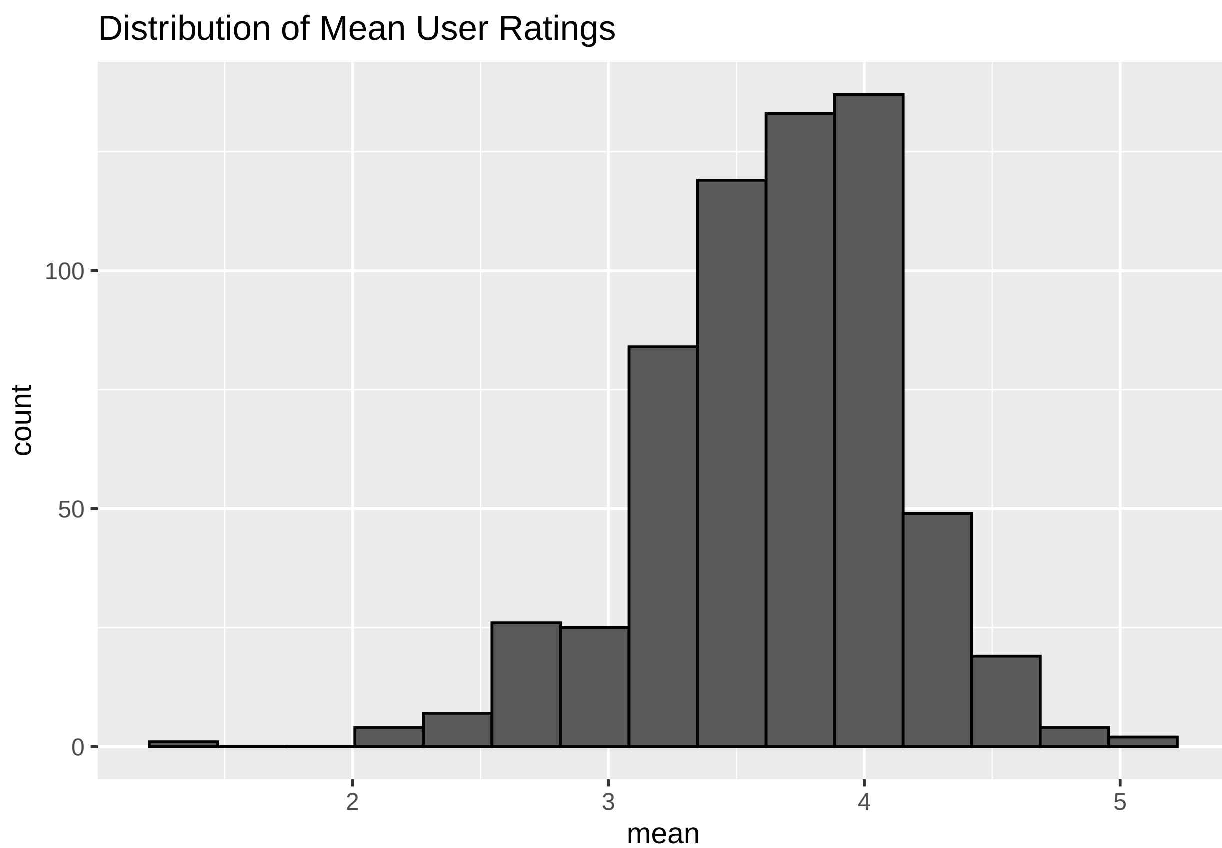
You may also recreate my results by the model2.R script (if it is not working, make sure you have downloaded the other 2 scripts download-data and train-validation-split. I have included these scripts with this report. I have also uploaded the datasets with this report in case the website ever stops working.

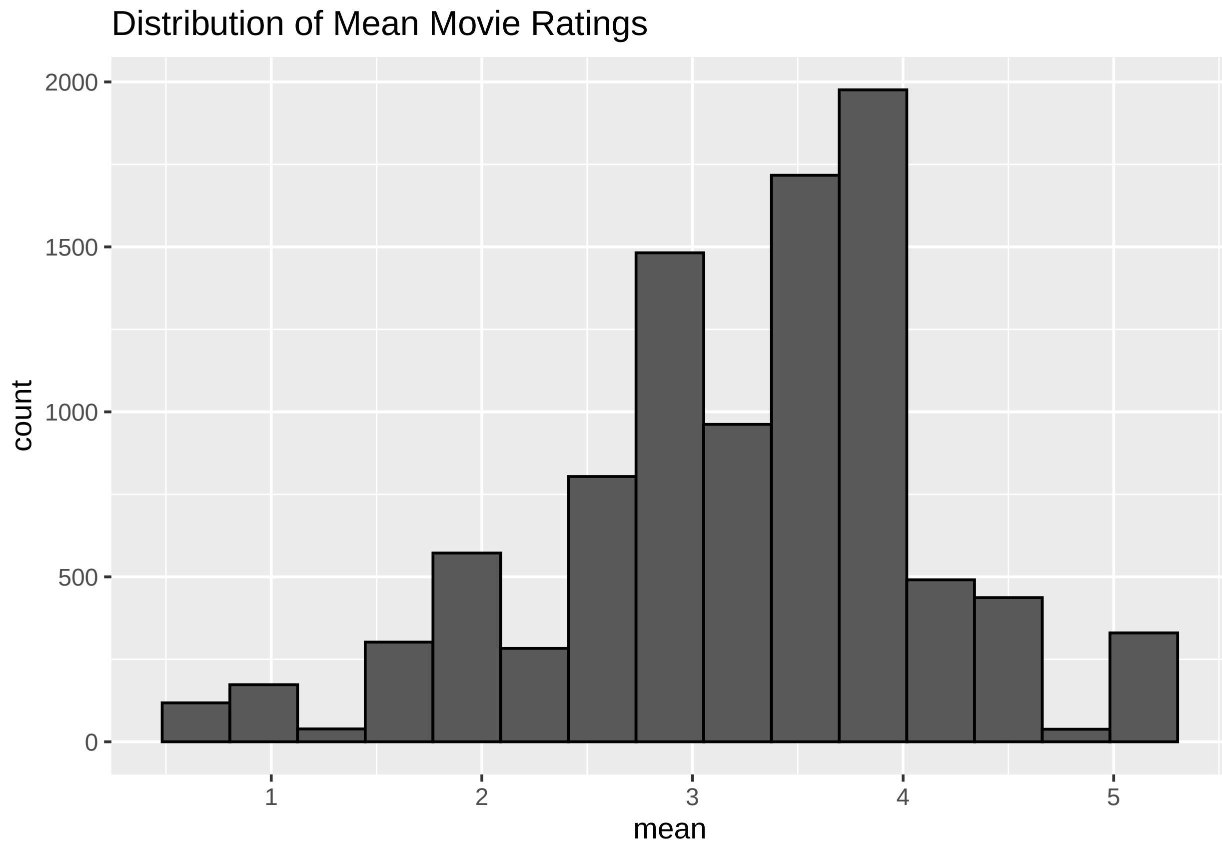
The last thing I did before beginning work with the data set was to set aside 20% of the data set at random to use for validating my final model to prevent overfitting.

## Exploratory Data Analysis

I made sure that all ratings were between the value of 0 and 5, as any ratings outside of this range would indicate faulty data. Then I visualized the distribution of all ratings, mean user ratings, and mean movie ratings to get a feel for the data. I also calculated the summary statistics for our movie ratings.


Histogram of Movie Ratings
count
20000
15000
10000
5000
0
0
1
2
3
4
5
rating


Distribution of Mean User Ratings
count
100
50
0
2
3
4
5
mean




```
##       mean median       sd
## 1 3.498215    3.5 1.043641
```

## Modeling

The first thing I did split out another test set of 20% of the data to test various different models on.

We can see that the mean value of the ratings is 3.49. I used this as a default prediction to see how accurate a model is that simply guesses the mean value for any and all ratings. I evaluated the models based on root mean squared error (RMSE) which is a measurement of how far my predictions are from the actual ratings. Simply guessing the mean rating for all ratings yields a RMSE of 1.036, which is my benchmark to beat.

The next step in the modeling process was to examine different user and movie averages. As visualized above, we can see that certain users and movies are biased towards higher and lower scores: we expected this, as different movies should receive different ratings.

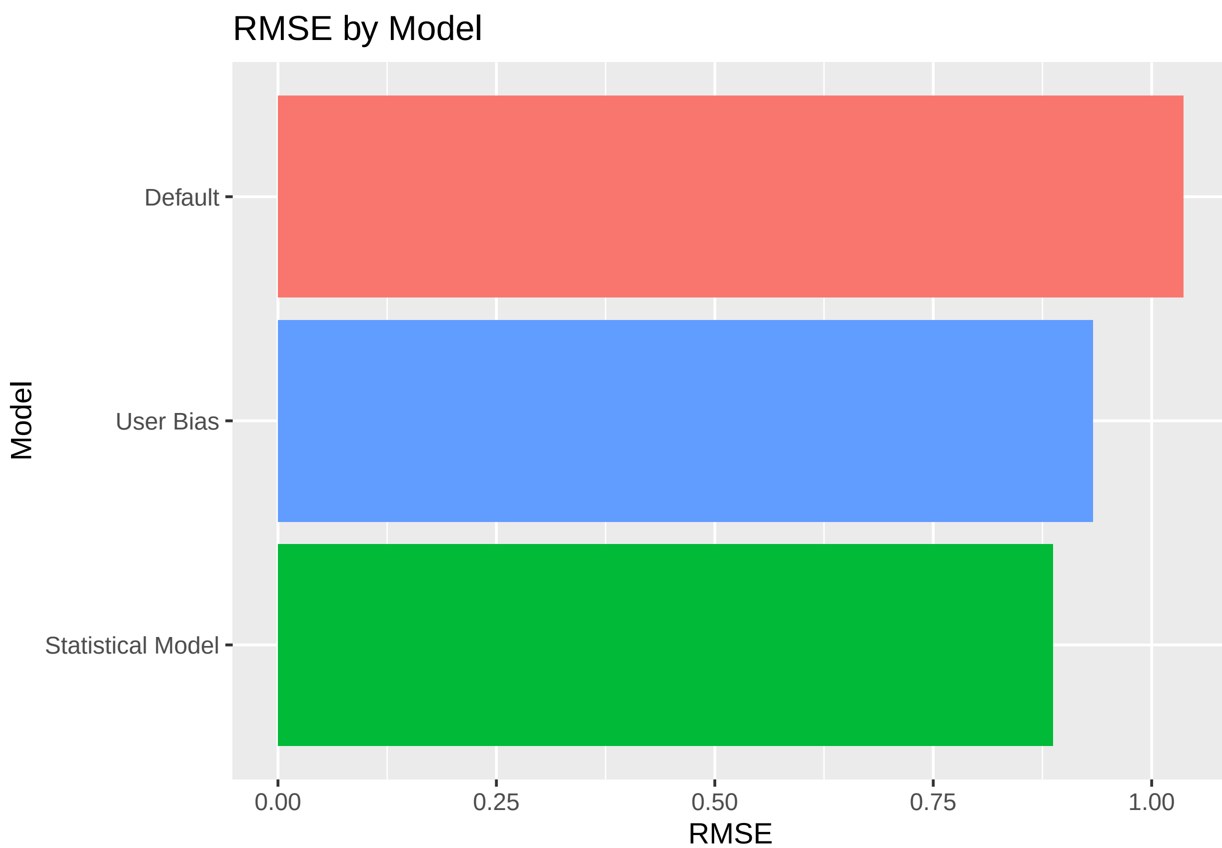
My next step is to factor this variance into my model to hopefully improve my predictions. I chose to first include the difference in users, which I termed “user bias”. User bias was calculated by subtracting the total mean rating of 3.49 from all ratings and then taking the average rating for each user, which shows the users average difference from the mean rating.

```
## # A tibble: 610 x 2
##    userId user_bias
##     <int>     <dbl>
##  1      1    0.854
##  2      2    0.384
##  3      3   -0.671
##  4      4    0.175
##  5      5    0.395
##  6      6   -0.0441
##  7      7   -0.165
##  8      8   -0.114
##  9      9   -0.383
## 10     10   -0.248
## # ... with 600 more rows
```

If I add the user bias to the default prediction of 3.49, my predictions should more accurately reflect the true ratings. Incorporating user bias into the model decreases the RMSE to .93, which is a significant improvement. I now look to add the movie bias into the model, but the movie bias must account for the fact that I have already added the user bias. I will calculate the movie bias by subtracting both the overall mean rating and the user bias from each rating, and the mean of the resulting ratings will show whether certain movies are above or below average.

```
## # A tibble: 9,724 x 2
##    movieId movie_bias
##      <dbl>      <dbl>
##  1       1     0.368
##  2       2    -0.0235
##  3       3    -0.193
##  4       4    -1.20
##  5       5    -0.540
##  6       6     0.370
##  7       7    -0.316
##  8       8    -0.716
##  9       9    -0.348
## 10      10     0.0246
## # ... with 9,714 more rows
```

We can see that movie bias also plays a role as some movies are above average and some are below average. As such, a new model given by the overall mean + the user bias + the movie bias should improve our predictions. This newest model gives an RMSE of .89, another improvement.




If we examine our model, we can see something slightly odd. Let us see the top 10 movies as predicted by our model.

```
## # A tibble: 10 x 2
##    title                              mean
##    <chr>                             <dbl>
##  1 'Salem's Lot (2004)                   5
##  2 12 Angry Men (1997)                   5
##  3 12 Chairs (1976)                      5
##  4 20 Million Miles to Earth (1957)      5
##  5 3-Iron (Bin-jip) (2004)               5
##  6 42nd Street (1933)                    5
##  7 61* (2001)                            5
##  8 7 Faces of Dr. Lao (1964)             5
##  9 9/11 (2002)                           5
## 10 A Detective Story (2003)              5
```

We get back some blockbusters we would expect like 12 Angry Men, alongside movies that we would not expect to be the top 10 movies such as 3-Iron (Bin-jip). This is because the model is not accounting for the number of ratings each movie has received, and cannot tell the difference between movies that have received 1000 ratings and movies that have received 1 rating. It becomes clear we need to add a regularization term that makes our model less confident in predictions with low sample sizes, causing these predictions to be closer to the mean value.

Here we see that adding a regularization term lambda has a positive effect on the RMSE up until approximately lambda = 3, at which point we have reached the optimal amount of regularization and any increase

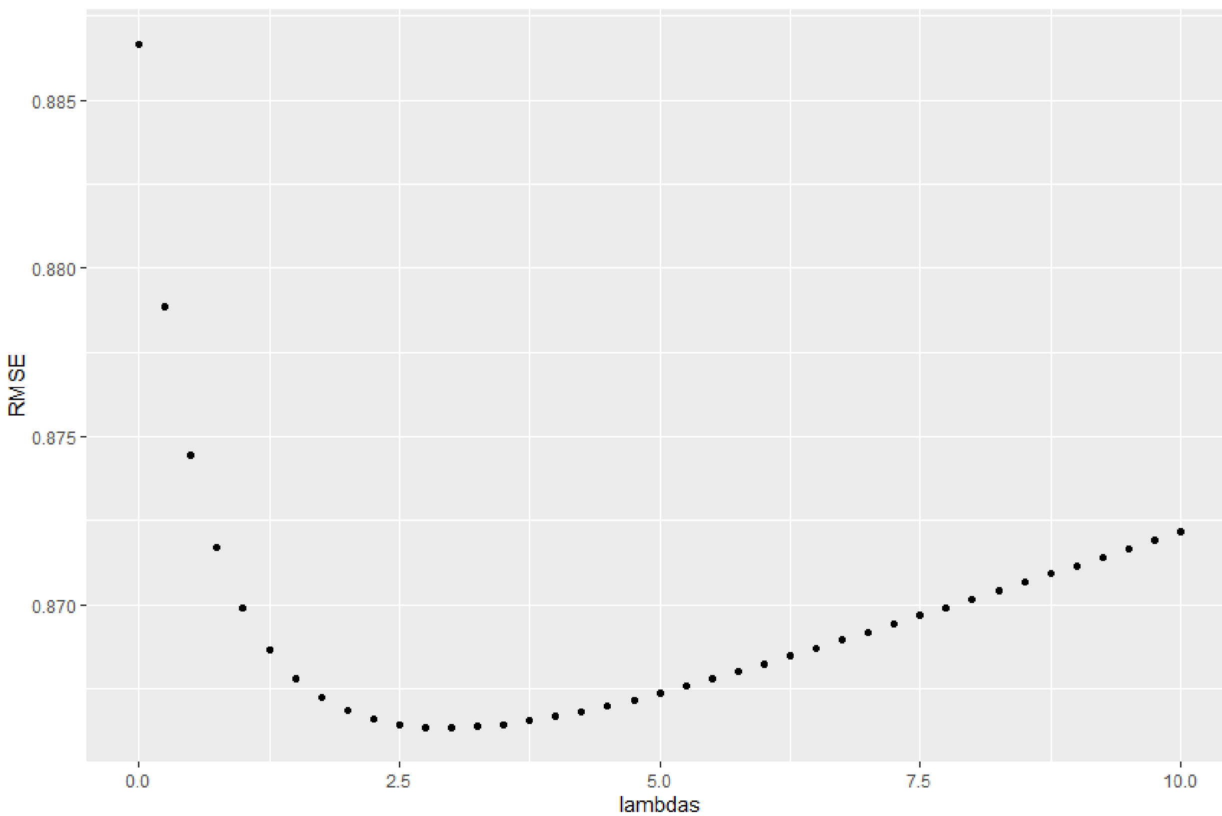



Figure 1: Optimization of the regularization parameter

also increases the RMSE of our model. So for our model, we will add the regularization term of lambda = 3 which decreases our RMSE down to .87, another improvement.

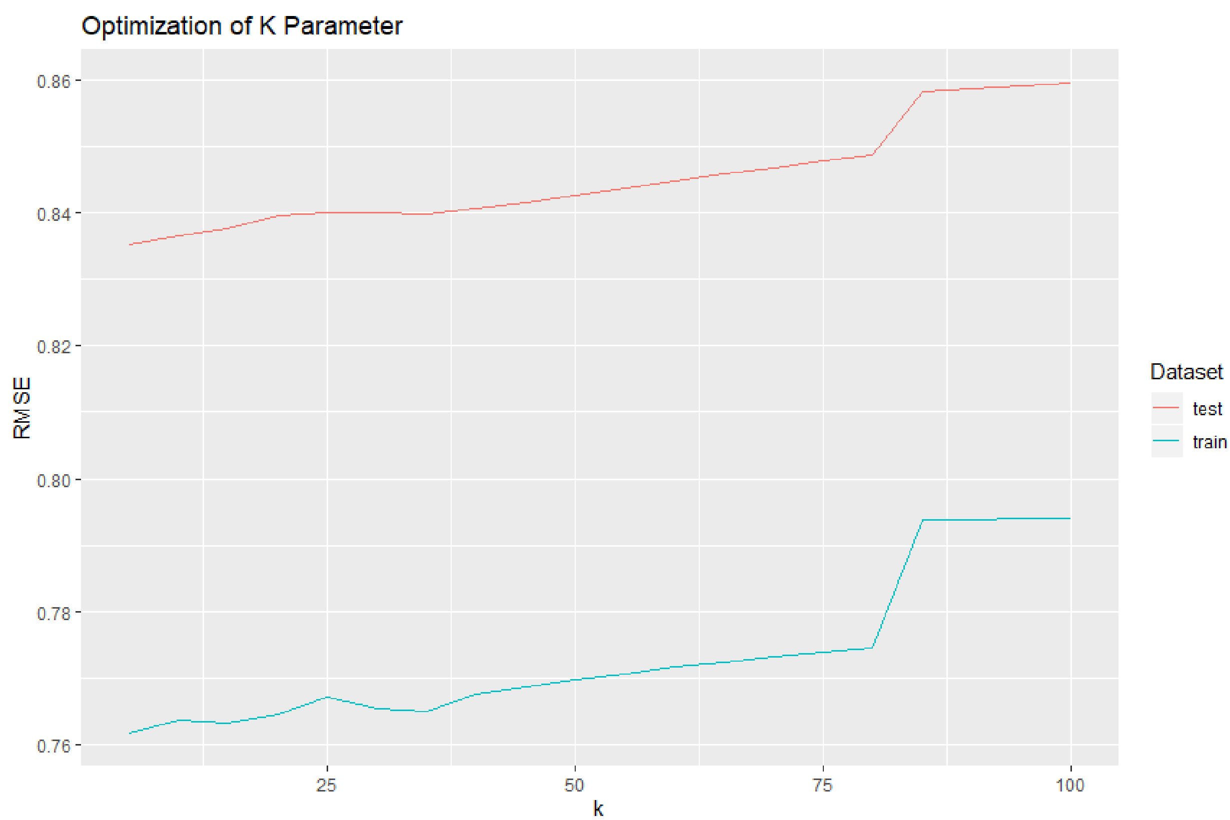
## Machine Learning with Funk Singular Value Decomposition

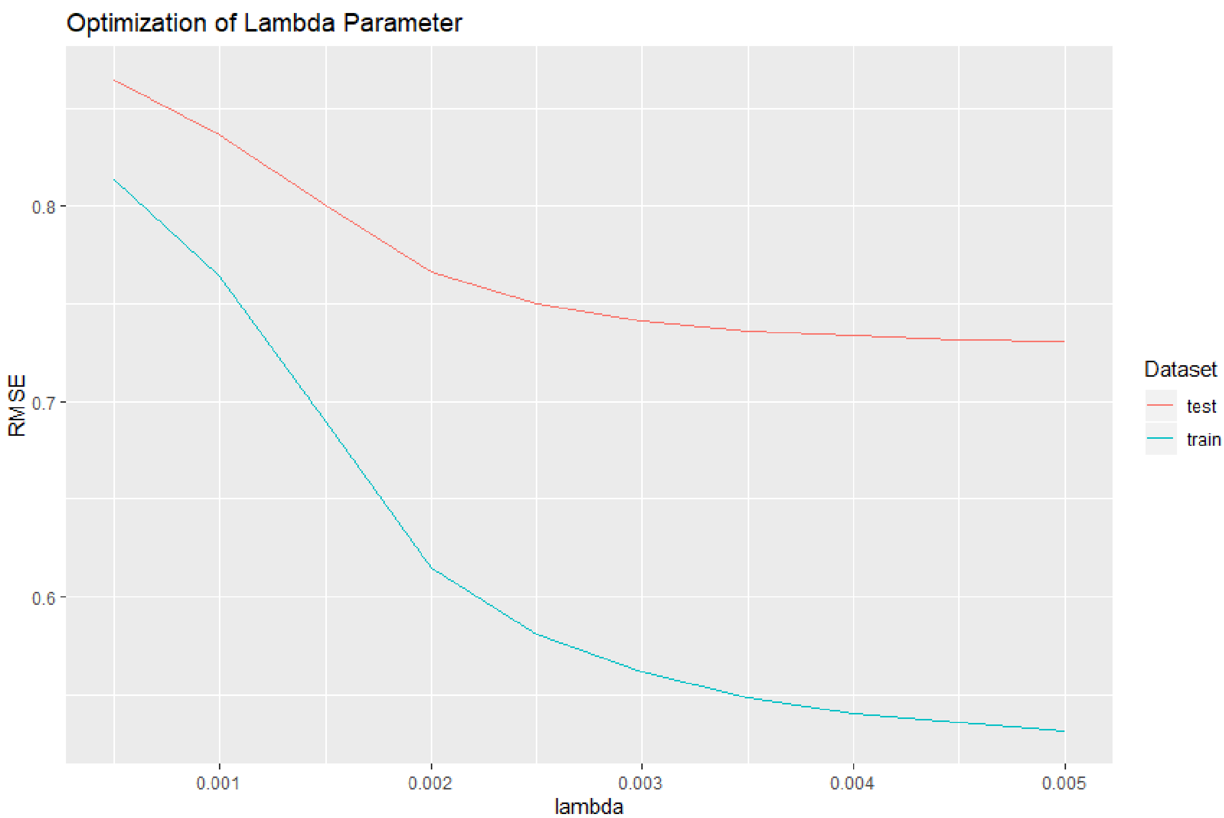
I felt that in order to see further improvement in the model I would need to use machine learning. I decided to attempt using matrix factorization from the recommenderlab package, as this was the method used by the winning team in the Netflix competition. However, it seemed to generate extremely overfit results, so I switched to the closely related Singular Value Decomposition from the same recommenderlab package. This approach is similar to matrix factorization in that it requires a matrix, but instead of finding its principal components, it will break down the matrix into two vectors whose crossproduct returns as close to the original matrix as possible. This required me to rework the data into a matrix format of users and movies.

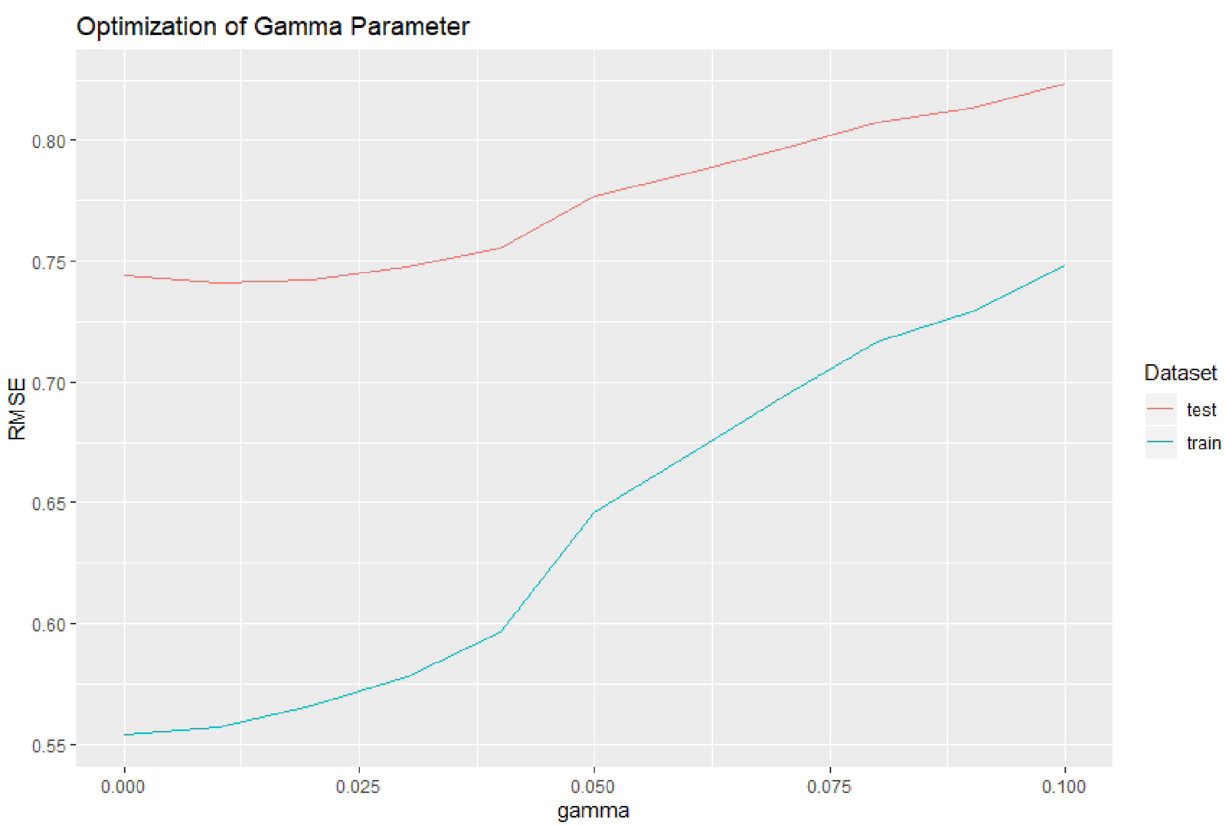
Running singular value decomposition on the normalized ratings gave a training RMSE of .77 and a test RMSE of .87. There was much less overfitting occurring in this model, but unfortunately it also didn't outperform our simple statistical model.

I then ran singular value decomposition on the variance not explained by the statistical model. the training RMSE was .76 and the test RMSE was .84, an improvement and our best RMSE to date. I decided to select this model to move forward with and optimize the parameters for. There are several parameters to optimize for this algorithm: K, the minimum #of ratings a movie needs to have to be considered; gamma, the regularization parameter; and lambda, the learning rate.

Here is the performance based on each optimization I ran. (The computation for these was extremely intensive as fitting a funkSVD model takes some time. As such, I present premade plots rather than redo the computations here.)




Optimization of Lambda Parameter
RMSE
0.6
0.7
0.8
0.001
0.002
0.003
0.004
0.005
lambda
Dataset
test
train

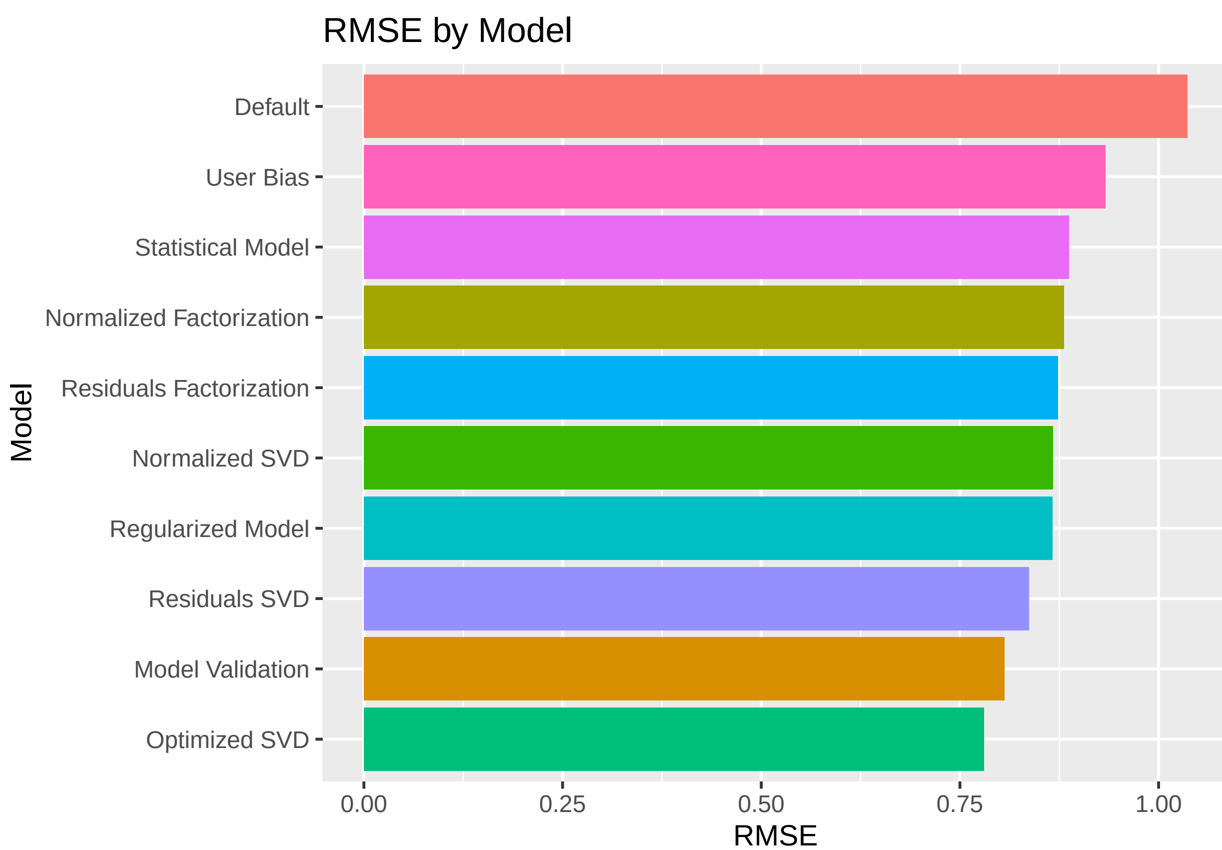

Optimization of Gamma Parameter
RMSE
0.55
0.60
0.65
0.70
0.75
0.80
0.000
0.025
0.050
0.075
0.100
gamma
Dataset
test
train

We can see that low values of K are optimal, so I chose to use the default value of 10. For lambda, I chose a value of .003 which is where the curve for the test set begins to flatten indicating no further improvement. However, this has introduced some overfit as there is significant distance between the training and test sets. To adjust for this, I chose a slightly high value of gamma of .05 which is not optimal on the test set, but partially closes the gap between training and test sets.

Here we have achieved an RMSE of .78, and the last step is to check this against the validation set to see if the model is overfit or if it will hold up on data it has never seen.

## Validation Data Set

The validation dataset has not been used at all in the analysis. I transformed the validation set into variance not predicted by the statistical model and ran the funkSVD model on the result. The end result was an RMSE of .806, only slightly worse than the performance on the test set and still better than our previous best model. This indicates that the model is not badly overfit and likely has predictive power in the future. Here is a summary of all the models and their performance.




Further improvement could be generated by incorporating the Genre information to the model or by using NLP on the tags provided as part of the dataset.