Movie Recommendation Report

Billy Tomaszewski

2020-05-16

Contents

1 Overview

This project is part of a series on Data Science in the HarvardX edX series. The purpose of this project is to practice exploratory data analysis, model generation and training, and report making.

1.1 Introduction

The purpose of this project is to use a publicly available large data set, to create a recommendation system. For this particular project we will be using movie databases and ratings given from individual users. By understanding how a user has previously rated certain movies, one can build a model to predict how the user would rate other movies. By determining which movies a user might rate highly, recommendations can be made.

In this project we will train a machine learning algorithm to predict user ratings from a training data set (called the edx data set) and then test the algorithm on a validation data set. The goal will be to predict a users rating in the validation set that closely matches the users actual rating for a given movie. Successful prediction will be evaluated by Root Mean Square Error (`RMSE`). This metric serves to measure the difference between the predicted and actual values. The RMSE will be evaluated for several different models, with a lower RMSE indicating a more accurate prediction algorithm. The equation for RMSE prediction is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

1.2 Data Overview

The Movielens data was provided by the course instructors and can be downloaded.

The data is in tidy format, and is split into `edx` (training) and `validation` sets. The validation set is 10% of the total data set to maximize the amount of data available for training the algorithm. The model development will utilize only the `edx` data set, and the `validation` set will only be used at the end to assess the performance of the algorithm.

We will begin to familiarize ourselves with the edx data set.

Users	Movies
69878	10677

These data contain nearly 70k unique users, who have rated 11k unique movies.

Searching for Missing Data

	x
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

We also see here that there is no missing data, which is advantageous since addressing missing data is often a challenge in machine learning.

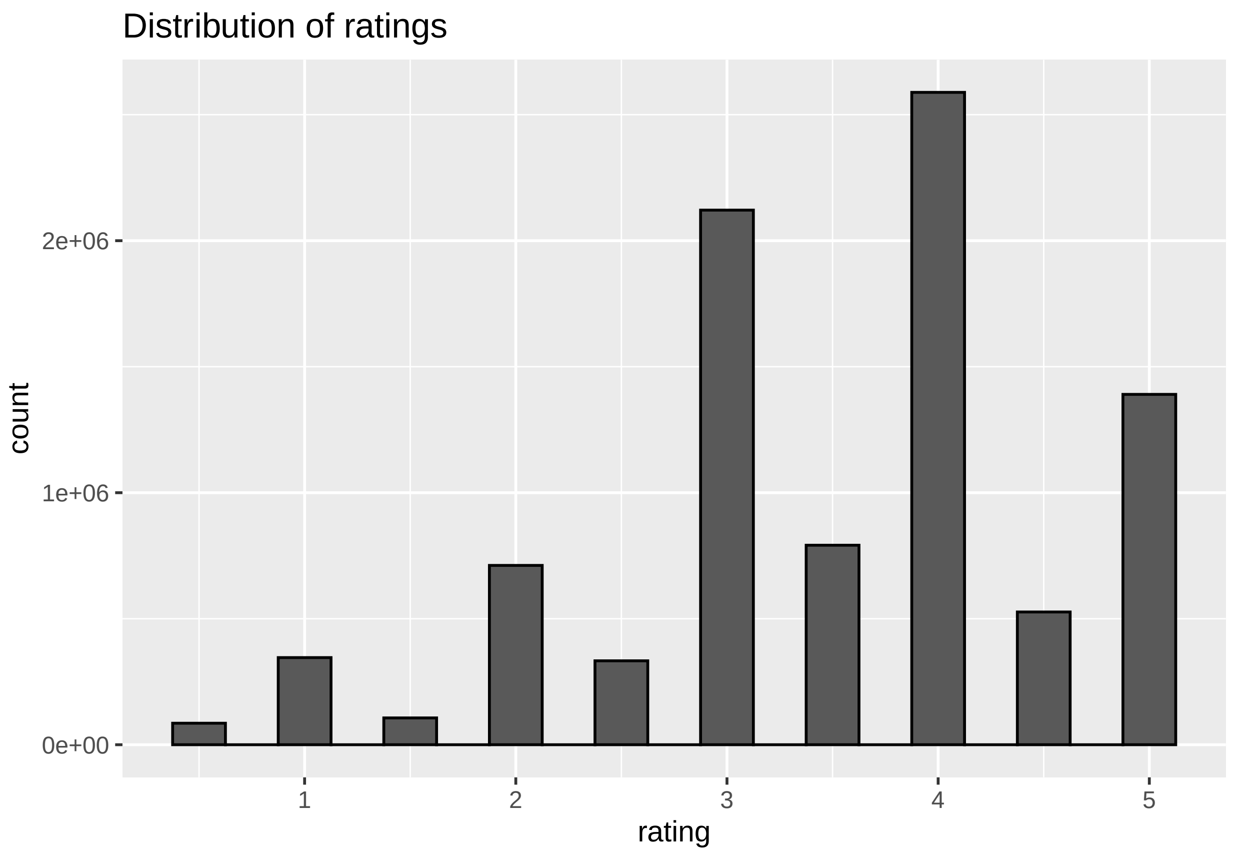
These are the features in each column:

- **userId** `<integer>` contains unique ID for each user
- **movieId** `<numeric>` contains unique ID for each movie
- **rating** `<numeric>` contains the rating of a movie by a user. Ratings are .5-5 with .5 increments
- **timestamp** `<integer>` contains the timestamp for when the user rated a movie
- **title** `<character>` contains the title of the movie that corresponds to the movieID as well as the year of release
- **genres** `<character>` describes the genre or genres that the movie belongs to

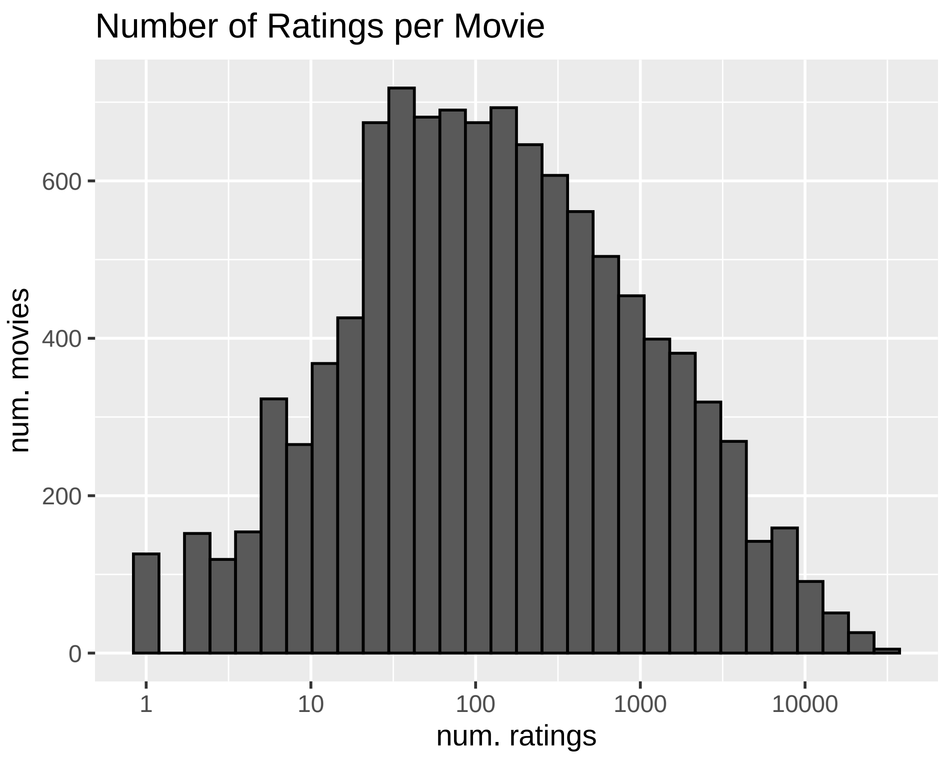
Portion of edx data

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

This gives you visual sense of how the data is laid out.



From this plot we see that ratings are generally right skewed, meaning that higher ratings are more common. We can also see from this plot that half start ratings are generally less common.

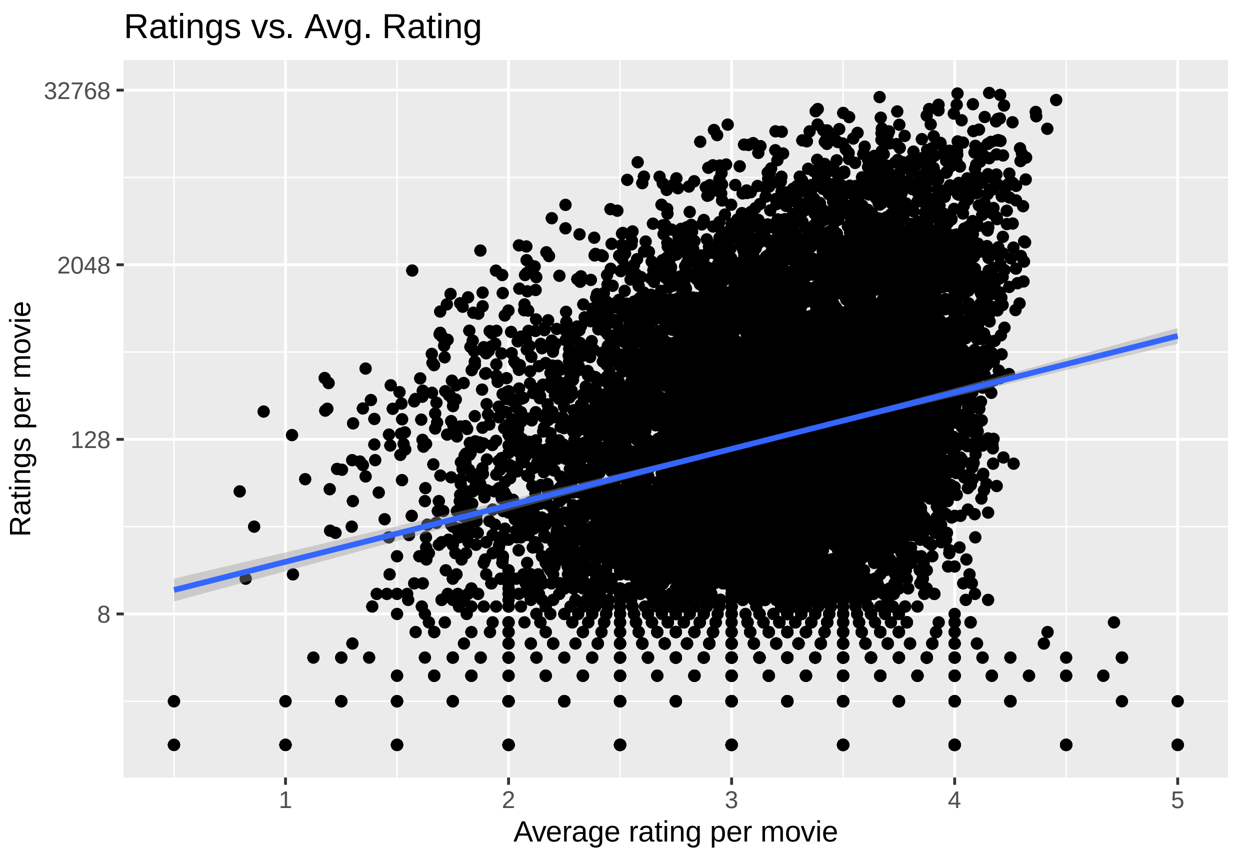


Here we see that not all movies are rated receive the same number of ratings. The movies that receive few ratings may be problematic for the algorithm because the amount of data for the individual movie is small.

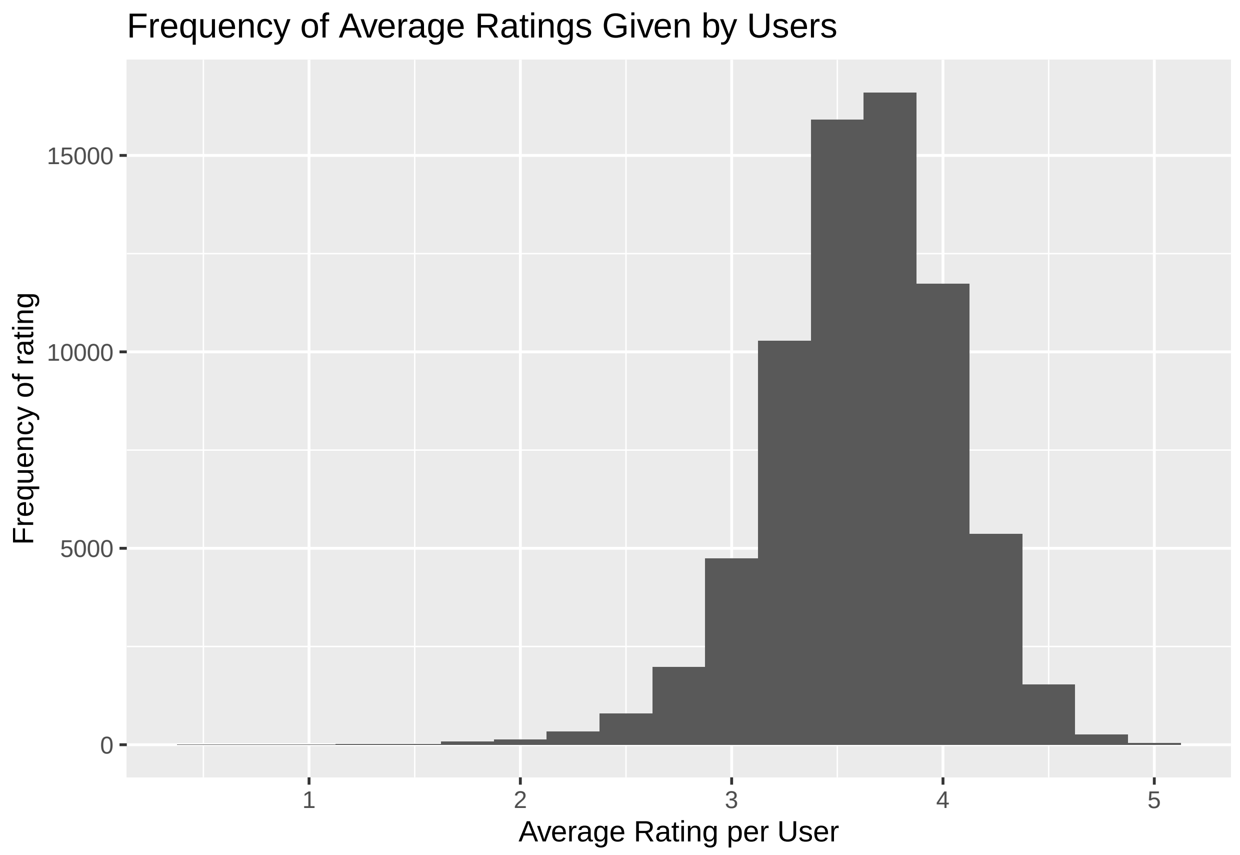
title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

Here we see that the movies that are infrequently rated, are often obscure.

In order to compensate for the effect these movies might have on the accuracy of our algorithm, we will apply a technique called regularization. This is done by applying a penalty to the error variable which minimizes the effect that movies that are sparsely reviewed have. Essentially, this adds extra uncertainty to movies that are rarely reviewed because there is not a big enough data set to confidently make a prediction.



This plot shows us that the number of times a movie is rated positively correlates with the average rating it receives. This suggests that movie itself, and perhaps its popularity, influence the rating it receives.



	Avg_Rating
	Min. :0.500
	1st Qu.:3.357
	Median :3.635
	Mean :3.614
	3rd Qu.:3.903
	Max. :5.000

This plot tells us that the average user gives a rating of ~3.5, but there are subsets of users that are very critical and on average have given all the movies they have rated a .5. The opposite is also true and some users give every movie they watch a 5. This suggests that we will have to account for user effects in our model.

2 Method

2.1 Modelling Approach

Here we describe the various models which attempt to maximize the accuracy of their prediction capability, and minimize their `RMSE` an error term which is calculated by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

In this equation y_hat represents our predicted rating that a user will give a movie, and y represents the actual rating that user gave. A `RMSE` of 1 indicates that our predictions are within are accurate to +/- 1. For this project we are expected to train an algorithm that produces a RMSE < **.86490**.

In order to test the accuracy of our training set without using our validation set we will partition the edx data set.Where our `test_set` will be used to validate the models that are built on the `train_set`. The `test_set` will be 20% of the edx data set.

2.1.1 Average Movie Rating Model

A simple model is one where we predict the same rating for all users. This represents a model where we assume

The formula for this model:

$$Y = \mu + \epsilon$$

This model makes the assumption that all variation from the mean rating is random. Here μ is the mean of the data set, and ϵ is the error term which describes the random variability. The average for the train set is:

```
## [1] 3.512536
```

When we predict our rating with μ, we obtain the first RMSE:

```
## [1] 1.060821
```

Here, we represent results table with the first RMSE:

method	RMSE
Average movie rating model	1.060821

This is our baseline RMSE, and as we previously mentioned, it is > 1 so our algorithm has not yet met the performance criteria.

In order to improve this we will incorporate insights from our exploratory data analysis and include more features in our model.

2.1.2 Movie Effect Model

To make our model more accurate, we will consider that the movie itself has an effect on the rating it receives. As we saw in our exploratory data analysis, movies which receive more ratings are often rated more highly. In order to account for this we will introduce a movie bias term, where `b` is the deviation of the average rating for an individual movie `i` from the average rating for all movies. The equation for our new model is:

$$Y_i = \mu + b_i + \epsilon_i$$

method	RMSE
Average movie rating model	1.0608210
Movie effect model	0.9439473

This reduces our `RMSE` to ~**.94**, and while it is great that we are now below 1, we still have a way to go in order to improve our accuracy to the necessary performance.

2.1.3 Movie and User Effect Model

This model aims to consider the parameters of both user and movie in making our prediction. As previously mentioned, there are users that give high ratings on average and users that give low ratings on average. We will include the bias of a given user in our model by introducing a bias term "b" for a given user "u". This gives us this new formula:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

In other words, this models a distribution centered at μ that deviates from that average because some movies are inherently more or less popular, some users are inherently more or less generous with their ratings, and a random sampling error ϵ.

method	RMSE
Average movie rating model	1.0608210
Movie effect model	0.9439473
Movie and user effect model	0.8665919

The RMSE is now **.865** which is just above where we need it before we can test on our `validation` data set.

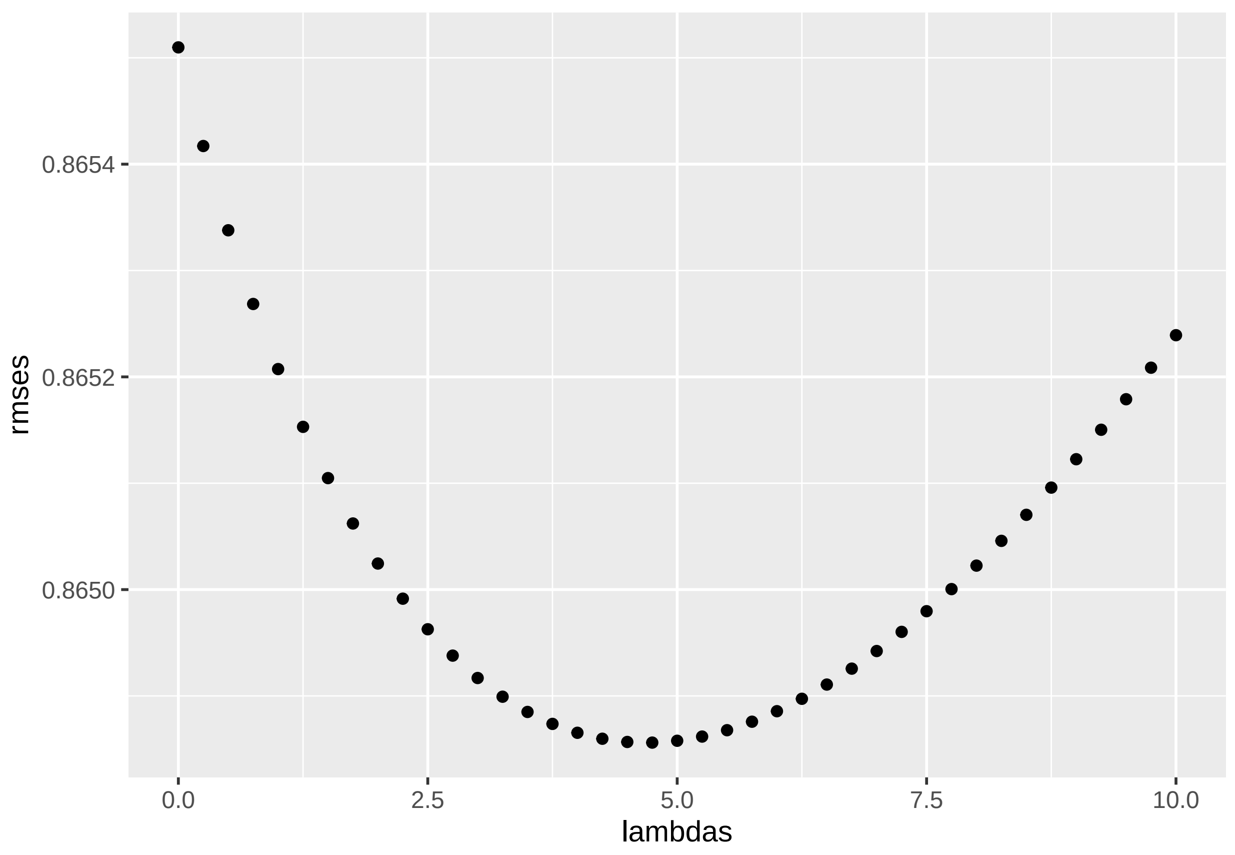
2.1.4 Regularization of Movie and User Effect Model

In order to further reduce the RMSE we will apply regularization, to constrain the total variability of the effect sizes, using the term λ (lambda). the new formula for our model is as follows:

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_i b_u^2)$$

Here b_i is influenced by movies with very few ratings b_u and is influenced by users who only rated a small number of movies. The use of the regularization allows us to penalize these effects. We can treat λ as a tuning parameter that allows us to minimize the RMSE.

The below plot helps visualize how we minimize RMSE with different values of λ.



Our value of λ which minimizes RMSE is:

```
## [1] 4.75
```

For the full model, the optimal lambda is: **4.75**

The new RMSE for our model is: **.8648**

method	RMSE
Average movie rating model	1.0608210
Movie effect model	0.9439473
Movie and user effect model	0.8665919
Regularized movie and user effect model	0.8648561

Now we can test if our trained model will perform the same on our `validation` data set.

method	RMSE
Average movie rating model	1.0608210
Movie effect model	0.9439473
Movie and user effect model	0.8665919
Regularized movie and user effect model	0.8648561
Final RMSE on Validation Data set	0.8648201

The final RMSE for the validation data set is **.8648**, which is **<.86490**, indicating we have successfully met the accuracy performance criteria.

3 Results and Discussion

The results from our models are below:

method	RMSE
Average movie rating model	1.0608210
Movie effect model	0.9439473
Movie and user effect model	0.8665919
Regularized movie and user effect model	0.8648561
Final RMSE on Validation Data set	0.8648201

Using our Regularized User and Movie model, we were able to accurately predict user ratings.

The final using this model on the validation data set is **.8648**, which is <**.86490**, indicating we have successfully met the accuracy performance requirement.

4 Conclusion

In conclusion, we have built a machine learning algorithm which accurately predicts ratings given by users. This was accomplished by utilizing the publicly available MovieLens data set, and sub-setting it into training and validation partitions. The training subset of the data was further partitioned so that tuning parameters could be optimized without using the validation data set, so as to avoid over-fitting. After assessing the accuracy of multiple different models using the training data set, it was determined that the Regularized User and Movie model would be the most accurate. I then applied this model to the validation data set to determine the final accuracy estimate. The error in this model was determined to be less than that was required indicating successful completion of the project.

This model could likely be further improved by including other parameters (time of day,time of year,year, genre), or by using more sophisticated machine learning algorithms (knn, neural network, etc.).

5 Appendix

5.1 Environment

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.0
## year           2020
## month          04
## day            24
## svn rev        78286
## language       R
## version.string R version 4.0.0 (2020-04-24)
## nickname       Arbor Day
```