MovieLens Report

Dave Maharaj

Executive Summary

In this project we will create a movie recommender system using the MovieLens dataset. The system will be used to recommend movies to users based on ratings provided by other users. We will use a dataset containing an estimated 10 million records. The full dataset contains over 20 million records but to allow the analysis to run more efficiently on a laptop, we have taken a subset of the data. Training will be done using 9 million records and the remaining 1 million records will be used to validate the training algorithm.

Reduced Mean Squared Error (RMSE) will be used to test the model for accuracy. Our goal is to achieve an RMSE of less than 0.86490 using appropriate modeling techniques. We will test different approaches to predicting the rating for movies throughout the project, with the objective of optimizing the RMSE value.

The report is divided into two (2) main sections. The first is the method/analysis where we pre-process the data and gain insights into the data using visualization, the second section is where we summarize our findings and provide an overall conclusion of our findings.

Method/Analysis

Exploratory data analysis

The initial dataset contains six(6) potential predictors and the outcome value, which is the rating. The predictors are - userId, movieId, timestamp, title, genres.

The table below shows the number of users and movies in the training dataset.

```
##   UserCount MovieCount
## 1     69878      10677
```

The first six (6) rows of the dataset are below. From this you can see that the title contains the year of release for the move and genres are formatted with pipes separating categories. This will require some pre-processing.

userId	movieId	rating	title	year	genres	rating_year	rating_month	rating_week
1	122	5	Boomerang	1992	Comedy	1996	8	31
1	185	5	Net, The	1995	Thriller	1996	8	31
1	292	5	Outbreak	1995	Sci-Fi	1996	8	31
1	316	5	Stargate	1994	Sci-Fi	1996	8	31
1	329	5	Star Trek: Generations	1994	Drama	1996	8	31
1	355	5	Flintstones, The	1994	Fantasy	1996	8	31

Data pre-processing

The table above shows that the data needs to be cleansed and formatted to allow us to better analyse the dataset. Below are the main items addressed in pre-processing of the dataset:

1. Split the title into 2 columns. The title and the year of release.
2. Timestamp needs to be converted to a date time and split into year and month fields for better visual
3. Simplify the genre field for better visual.

After pre-processing the first 6 rows of the dataset looks like the below. Note that while we have pre-processed the data to separate out columns etc, we may not use all the predictors for our algorithm.

userId	movieId	rating	title	year	genres	rating_year	rating_month	rating_week
1	122	5	Boomerang	1992	Comedy	1996	8	31
1	185	5	Net, The	1995	Thriller	1996	8	31
1	292	5	Outbreak	1995	Sci-Fi	1996	8	31
1	316	5	Stargate	1994	Sci-Fi	1996	8	31
1	329	5	Star Trek: Generations	1994	Drama	1996	8	31
1	355	5	Flintstones, The	1994	Fantasy	1996	8	31

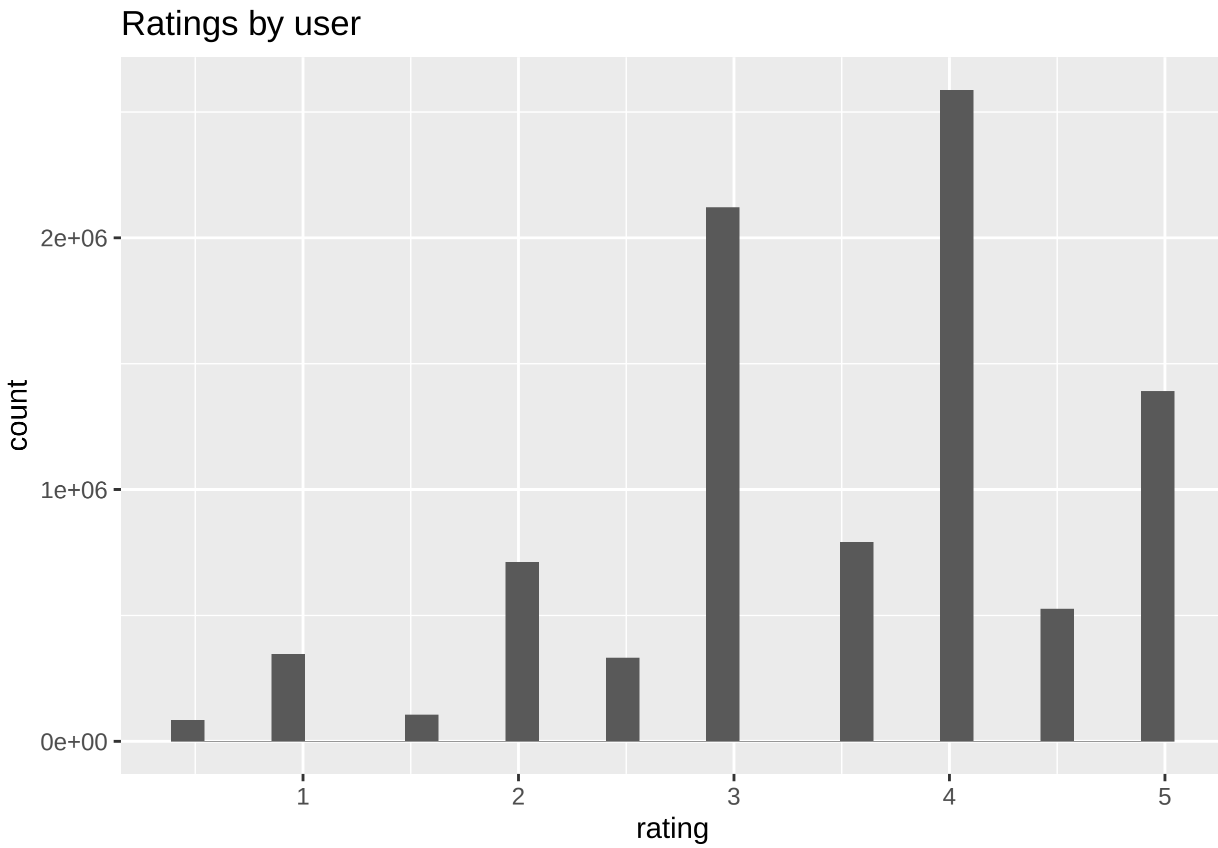
Visual exploration

Below we explore the data visually to gain insights into the distribution of specific predictors in the dataset.

Distribution of ratings by user:

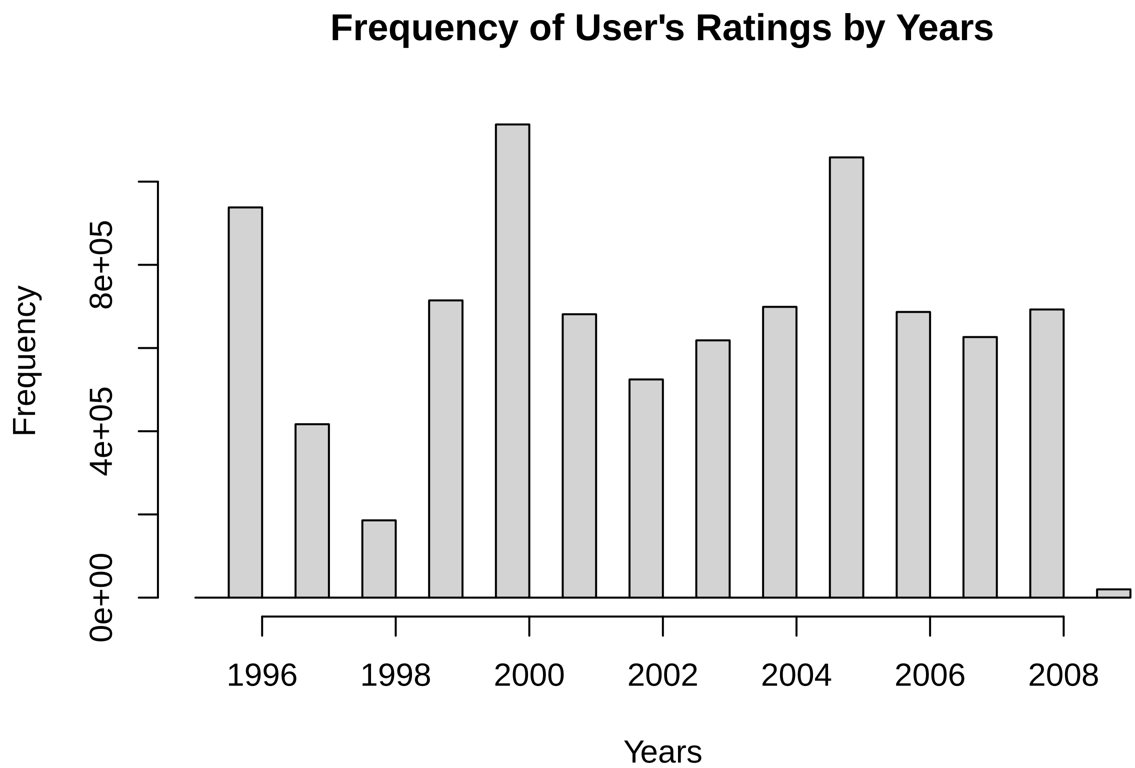
The plot shows a smaller quantity of users with half star values. 3,4 and 5 star values are most common in the dataset. 1 and 1.5 are least common.

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



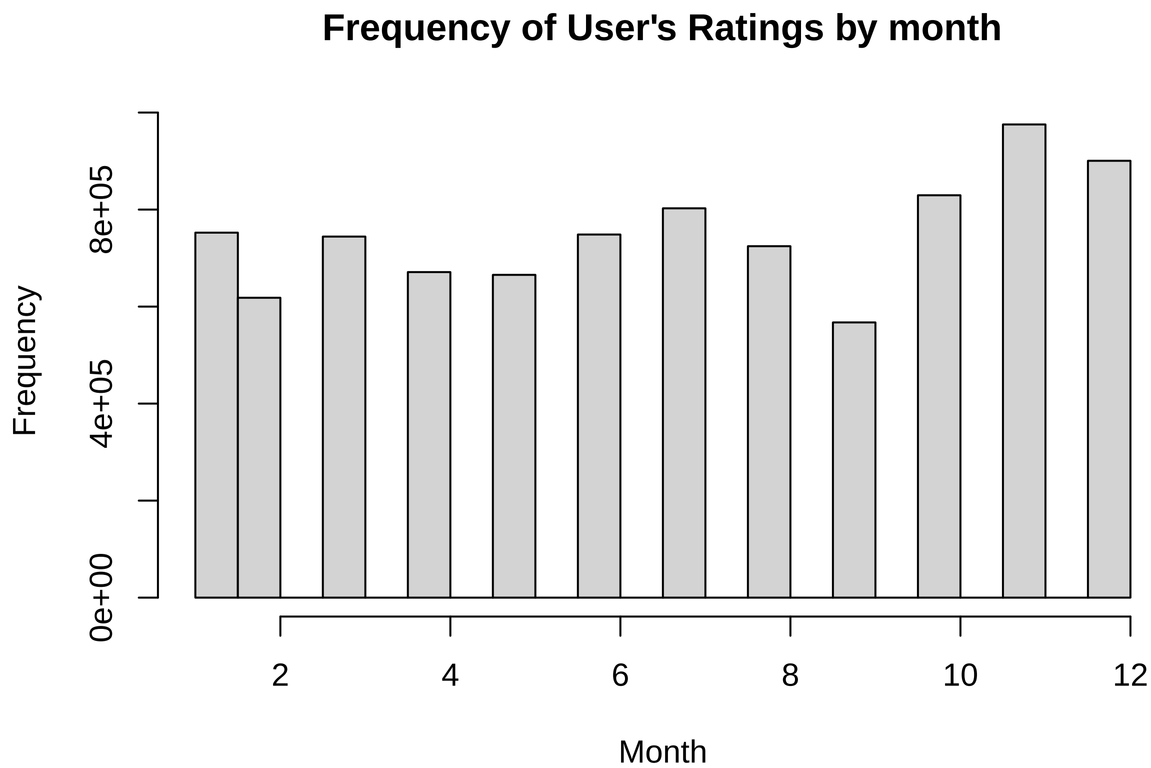
Ratings by year of rating:

This shows the distribution of ratings by year.



Rating by Month:

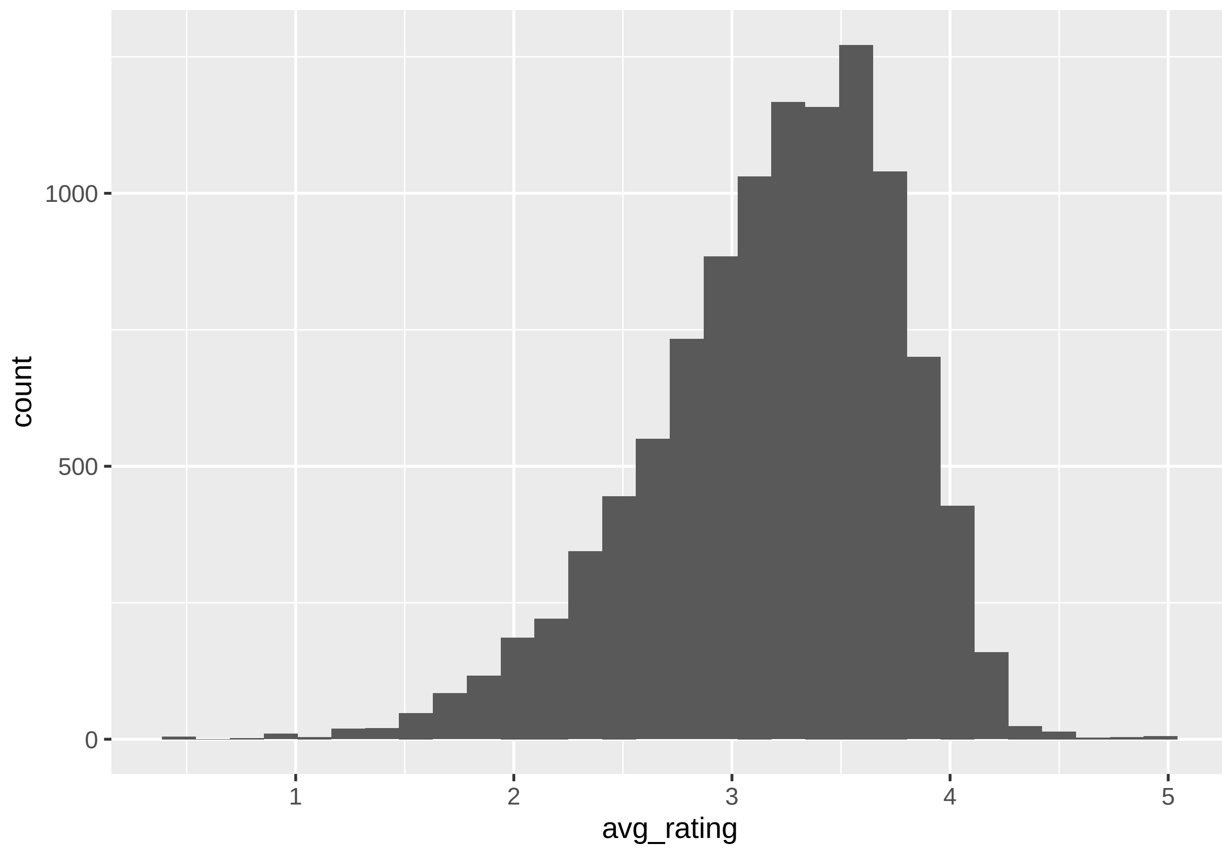
The distribution shows that the months of November and December has the highest number of ratings.



Average rating by movie:

The distribution below shows the average rating by movie. This shows that the centre of the distribution is around 3.5

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The table below summarize the mean, sd and median of the rating in the dataset.

avg_rating	sd_rating	median_rating
3.512465	1.060331	4

Modeling approach

Because of the size of the dataset it will be slow and inefficient to run complex algorithms on the dataset using a laptop. While these options may work, it is preferred to use cloud resources to execute models like this. Instead we will employee similar less complex methodologies to reach our goal of an RMSE of less than 0.86.

Naive approach

Modeling using a Naive approach is first tried on the data. This is done using the mean of the data. The equation for this is:

$$Y_{u,i} = \mu \tag{1}$$

In this equation $Y_{u,i}$ is the rating from user u for movie i. mu is the average across all ratings in the training dataset. Below is the calculation for this.

```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Using the mean as the prediction for a rating, we will now test against the validation data ratings to check the RMSE value.

```
RMSE(validation$rating, mu)
```

```
## [1] 1.061202
```

From the results from the RMSE, we are over 1, which is much higher than what we want.

Using a Bias term

In this model we include a bias term for predicting the rating. This bias term b_i for movie ratings difference. This term is considered to be independent of the data. The equation for predicting the rating with the bias term is below:

$$Y_{u,i} = \mu + b_i \tag{2}$$

The b_i is calculated using the training data. The code below is used to generate the b_i term.

```
ratings_term_bi <- edx %>% group_by(movieId) %>% summarise(bi = mean(rating-mu))
```

Predictions are then generated using the equation above. The code below is used to do this for the validation dataset.

```
predictions_bi <- validation %>%
  left_join(ratings_term_bi, by='movieId') %>%
  mutate(h_rating = mu + bi) %>%
  select(movieId, userId, title, year, bi, h_rating, rating)
```

Once we have the predictions, the RMSE is then calculated using the RMSE function.

```
RMSE(validation$rating, predictions_bi$h_rating)
```

```
## [1] 0.9439087
```

This gives us an RMSE of 0.9439, which is closer to what we want. It looks like we're on the right track from the results. Adding another bias term should get us closer to our objective. A bias term of b_u is now added to the model, this term is based on the user ratings rather than overall movie ratings. The equation for this model now looks like the below:

$$Y_{u,i} = \mu + b_i + b_u \tag{3}$$

The b_u term was created using the training data. The code for this is below:

```
ratings_term_bu <- edx %>% left_join(ratings_term_bi, by='movieId') %>% group_by(userId) %>% summarise(
```

Predictions were then generated using both the b_i and b_u terms. The code for this is below.

```
predictions_bi_bu <- validation %>% left_join(ratings_term_bi, by="movieId") %>% left_join(ratings_term_
  select(movieId, userId, title, year, bu, bu, h_rating, rating)
```

Once the predictions were generated, the RMSE value was calculated for this model.

```
RMSE(validation$rating, predictions_bi_bu$h_rating)
```

```
## [1] 0.8653488
```

The results showed an RMSE of 0.8653488. This is close to the goal of less than 0.86490.

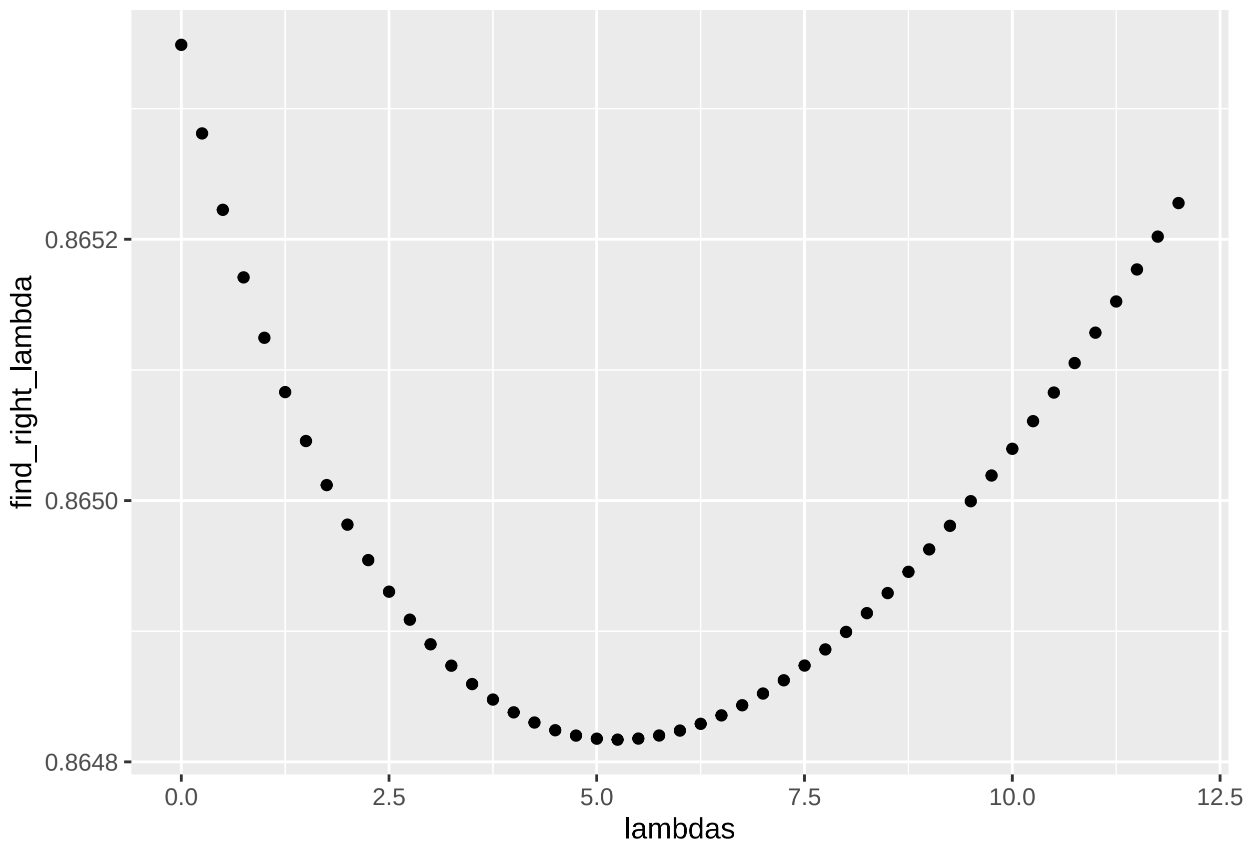
Regularization will be used to reduce the effect of large errors in our predictions. Regularization penalizes incorrect estimates on small sample sizes. The equation for the model with regularization is below:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2), \tag{4}$$

The equation shows regularization being applied to each effect (b_i, b_u).

The value of λ is a parameter we need to find. In order to do this we will try a series of values for lambda between 0 and 12 in increments of 0.5. The plot below shows the results of the testing for each value of lambda.

```
qplot(lambdas, find_right_lambda)
```



Based on the results produced, the lambda value that provided the RMSE that met our goal is 5.25.

```
lambdas[which.min(find_right_lambda)]
```

```
## [1] 5.25
```

The final model for predicting ratings for our recommendar system is below.

```
lambda = 5.25

ratings_term_bi_lambda <- edx %>% group_by(movieId) %>%
  summarise(bi = sum(rating-mu)/(n()+lambda))
ratings_term_bu_lambda <- edx %>% left_join(ratings_term_bi_lambda, by='movieId') %>%
  group_by(userId) %>% summarise(bu = sum(rating-mu-bi)/(n()+lambda))
predictions_bi_bu_l <- validation %>% left_join(ratings_term_bi_lambda, by="movieId") %>%
  left_join(ratings_term_bu_lambda, by="userId") %>% mutate(h_rating = mu + bi + bu) %>%
  select(movieId, userId, title, year, bu, bu, h_rating, rating)
```

The RMSE results of of the model:

```
RMSE(validation$rating, predictions_bi_bu_l$h_rating)
```

```
## [1] 0.864817
```

The results show an rmse value of 0.864817, which is within our goal.

Conclusion

Our findings show that ratings cannot be modeled using a naive approach with simply the mean of the ratings. This did not produce favorable results in our testing. Using bias terms for ratings based on specific groups drastically improved our results. We moved from 1.06 to under 0.87, an improvement of 0.19. Once the regularization term was added to the model we got a further improvement in the model. Adding additional effects and using additional data may yield further improvements in the model.

For reference we have included the final table of resilts from our testing with the 4 models used during this project.

```r
kable(rmse_tb_results, align='l', format= 'pipe')
```

Method	RMSE
Naive Means	1.0612018
Movie Effect	0.9439087
Movie and User Effect	0.8653488
Movie and User Effect with Regularization	0.8648170