HarvardX PH125.9x Data Science Capstone-Movielens Project

Alvin Subakti

January 4, 2021

1. Introduction

Background

A recommendation system is a method of information filtering system that has the capability to predict the preference of a user toward a certain item. Recommendation systems have been applied in a variety of areas including music, news, books, research articles, search queries, even in more advanced aspects like financial services, life insurances, and many more. One of them which is being applied in this project is regaring movie ratings. Movie rating recommendation system will be able to predict the rating a user would give toward a movie giving their traits.

Netflix realized the importance of this recommendation system in its movie rating system and held a open competition in 2006. In the competition, Netflix offered a million dollar prize to anyone that is able to imrpove the effectiveness of their recommendation system by 10%. This project is inspired by the competition. Here we are trying to solve a similiar problem but with a much simpler approach and a different dataset. Since the approach done by the participants in the competition are far more advanced and we may not have the capabilities or hardware requirements for it, so a simpler yet effective approach is used as a way to show understanding of this course. Also since the dataset used in the competition is not publicly available, This project will use another publicly available dataset related to movie ratings provided in the 'MovieLens'.

DataSet

The Dataset used in this problem is the 'MovieLens' dataset, this dataset can be found and downloaded through these following links:

https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

Generation of 'Movielens' Dataset that will be used in this project will be loaded using the instructions given in the course.

Goal

The goal of this project is to be able to analyze and gain insights from the 'Movielens' dataset. Then also able to construct machine learning models or algorithms that will be trained by using the training dataset, and finally has the ability to predict ratings given a movie in the validation dataset.

The parameter that will be used in this project is the RMSE or Rooted Mean Square Error. A model has a better performance if it has a smaller value of RMSES given the same validation dataset.

2. Exploratory Data Analysis

First, the prevew of the dataset can be seen in the following table, where the dataset consisted of user ID, Movie ID, movie ratings, timestamp, title of the movies, and genre of the movies.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                 Comedy|Romance
## 2          Action|Crime|Thriller
## 3  Action|Drama|Sci-Fi|Thriller
## 4        Action|Adventure|Sci-Fi
## 5 Action|Adventure|Drama|Sci-Fi
## 6       Children|Comedy|Fantasy
```

The descriptive statistics for the dataset can be seen in the following table.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Now, by using the following code we can see that the edx dataset generated contains 69878 unique users and 10677 unique movies.

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

However, the amount of observation is not exactly 69878×10677 which indicated that not every users rate every movies in the dataset. This can be proven in the following table that show a user rate some of the movies shown by an existing rating but not all of the movies shown by the missing value NA.

```
## Selecting by n
```

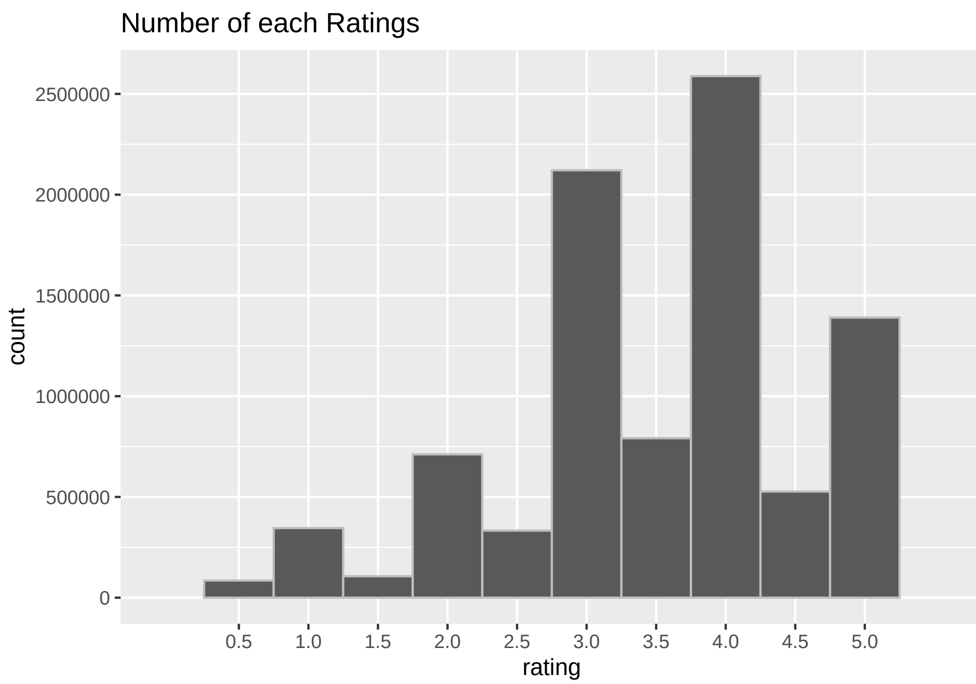
userId	Forrest Gump (1994)	Jurassic Park (1993)	Pulp Fiction (1994)	Shawshank Redemption, The (1994)	Silence of the Lambs, The (1991)
13	NA	NA	4	NA	NA
16	NA	3	NA	NA	NA
17	NA	NA	NA	NA	5

userId	Forrest Gump (1994)	Jurassic Park (1993)	Pulp Fiction (1994)	Shawshank Redemption, The (1994)	Silence of the Lambs, The (1991)
18	NA	3	5	4.5	5
19	4	1	NA	4.0	NA

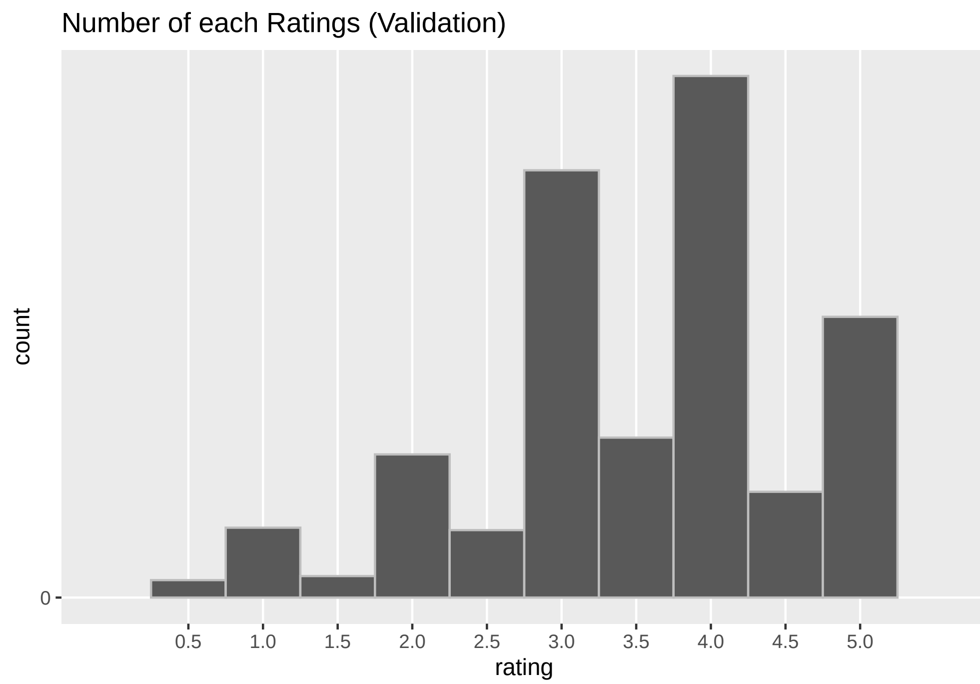
Distributions

Now the exploratory data analysis will continue by analyzing distributions in several aspects.

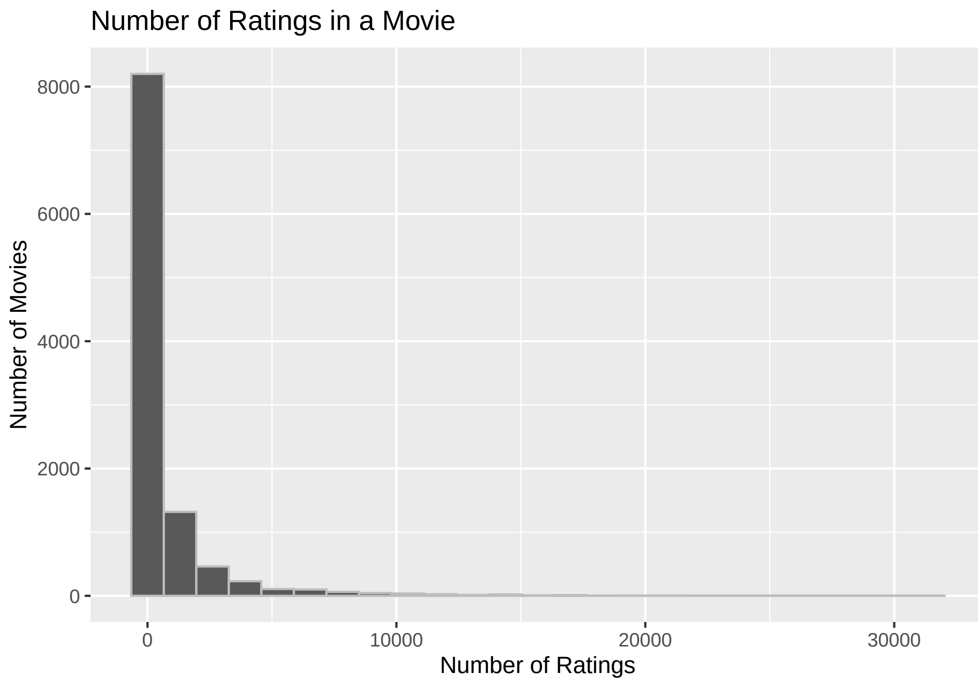
The distribution for the number of ratings given by all of the users in the edx dataset can be seen in the following plot. It is clear that most users give a rate of between 3 to 4 for a movie shown by the significant peak in this interval.



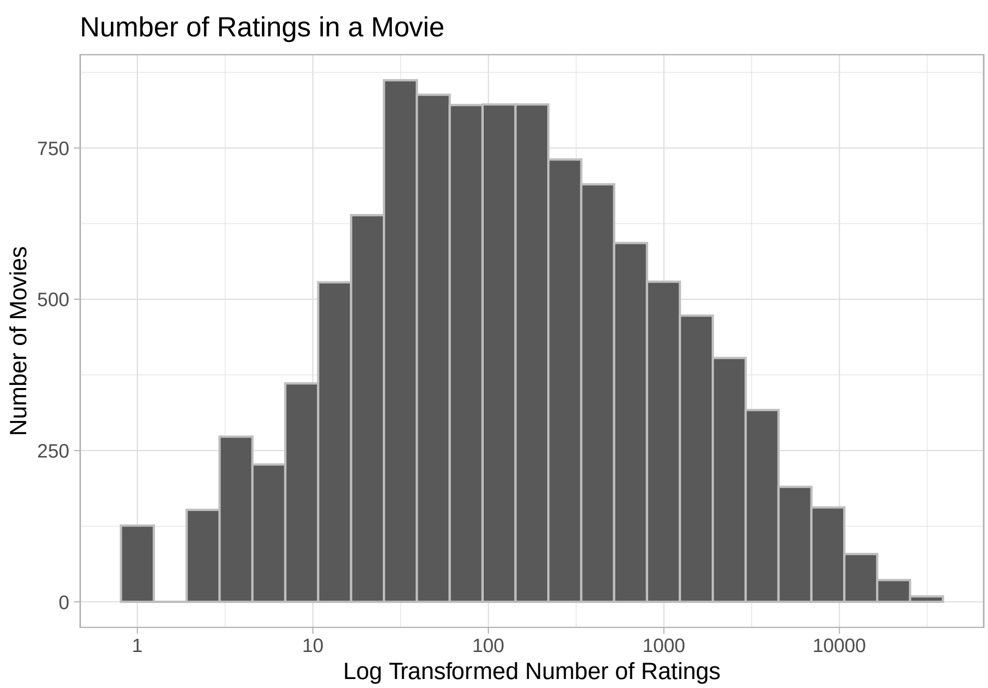
Now for the validation dataset the rating distribution also can be seen in the plot below. By comparing the result of this two plot, we can see that edx and validation dataset has similar rating distribution.


Number of each Ratings (Validation)
count
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
rating

Next, by plotting the distribution of the number of ratings given for a movie can be seen in the following plot



From the plot above it can be seen that every movies has different amount of rating given, and the number of movie rated has a tendency to decrease exponentially as the frequency of ratings increase. To observe a much better relationship, we will now plot the distribution of the number of ratings given for a movie but by also applying log transformation on the number of ratings (the x axis)

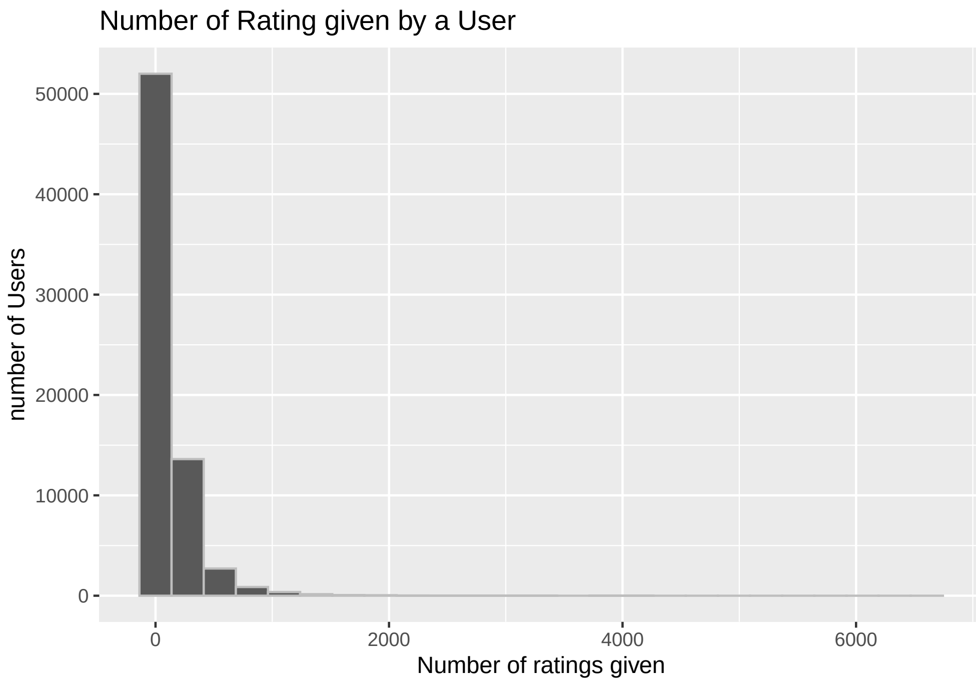


The following tables shows the 20 least and most rated movies as well as the number of ratings given. We can see that there is a significant difference in the rating since there are movies that is only rated once and there are also blockbusters that has thousand of ratings.

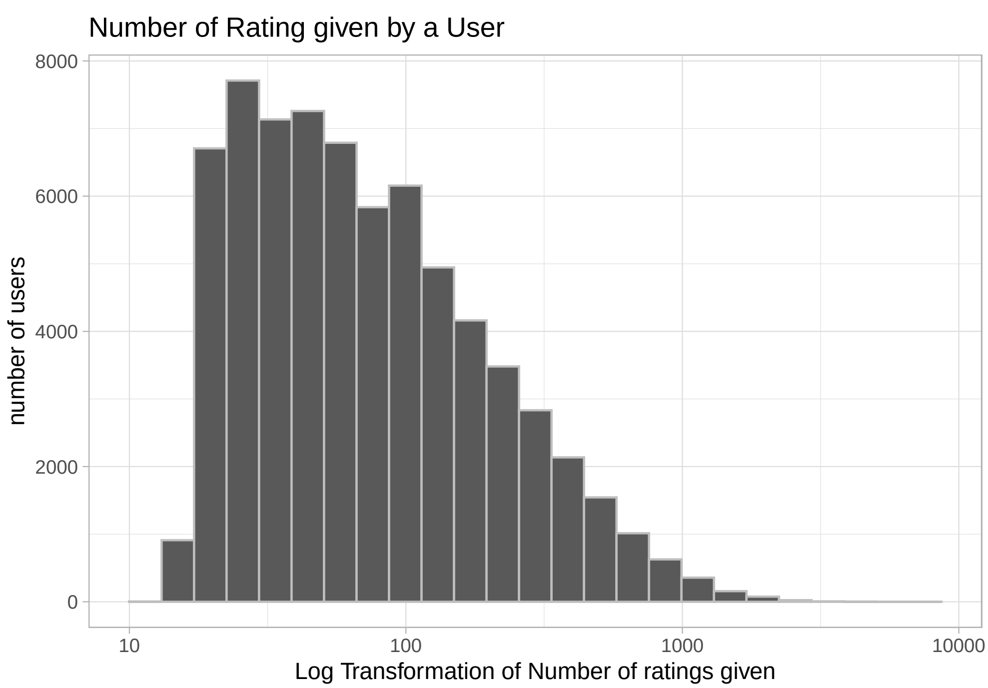
movieId	title	count	rating
3191	Quarry, The (1998)	1	3.5
3226	Hellhounds on My Trail (1999)	1	5.0
3234	Train Ride to Hollywood (1978)	1	3.0
3356	Condo Painting (2000)	1	3.0
3383	Big Fella (1937)	1	3.0
3561	Stacy's Knights (1982)	1	1.0
3583	Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	1	3.0
4071	Dog Run (1996)	1	1.0
4075	Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)	1	1.0
4820	Won't Anybody Listen? (2000)	1	2.0
5257	In the Winter Dark (1998)	1	3.5
5565	Dogwalker, The (2002)	1	2.0
5616	Mesmerist, The (2002)	1	3.5
5676	Young Unknowns, The (2000)	1	2.5
5702	When Time Ran Out... (a.k.a. The Day the World Ended) (1980)	1	1.0
6085	Neil Young: Human Highway (1982)	1	1.5
6189	Dischord (2001)	1	1.0
6501	Strange Planet (1999)	1	2.0
6758	Emerald Cowboy (2002)	1	3.0
6838	Once in the Life (2000)	1	3.0

movieId	title	count	avg_rating
296	Pulp Fiction (1994)	31362	4.154789
356	Forrest Gump (1994)	31079	4.012822
593	Silence of the Lambs, The (1991)	30382	4.204101
480	Jurassic Park (1993)	29360	3.663522
318	Shawshank Redemption, The (1994)	28015	4.455131
110	Braveheart (1995)	26212	4.081852
457	Fugitive, The (1993)	25998	4.009155
589	Terminator 2: Judgment Day (1991)	25984	3.927859
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672	4.221311
150	Apollo 13 (1995)	24284	3.885789
592	Batman (1989)	24277	3.386292
1	Toy Story (1995)	23790	3.927638
780	Independence Day (a.k.a. ID4) (1996)	23449	3.376903
590	Dances with Wolves (1990)	23367	3.742628
527	Schindler's List (1993)	23193	4.363493
380	True Lies (1994)	22823	3.500285
1210	Star Wars: Episode VI - Return of the Jedi (1983)	22584	3.996436
32	12 Monkeys (Twelve Monkeys) (1995)	21891	3.874743
50	Usual Suspects, The (1995)	21648	4.365854
608	Fargo (1996)	21395	4.134821

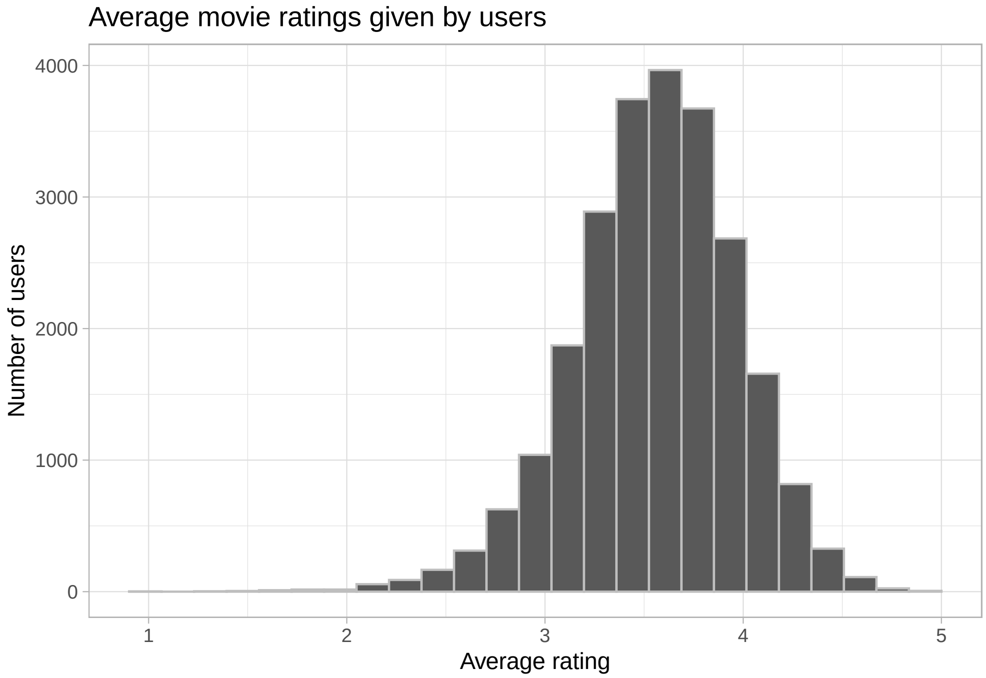
Next, we will observe the distribution of the number of ratings given by a user. The following plot shows the relationship



The previous founding regarding that not every user rate the same amount of movies is supported by the plot above. it can also be seen that the number of users has the tendency to decrease exponentially as the number of rates given increase. To see a better relationship, we use log transformation on the number of ratings(x axis).



Next, we will going to plot the distribution of mean movie ratings given by users, to avoid outliers and high bias, only users with more than 100 movies rated are used. It can be seen from the plot below that most user give an average rating of between 3 and 4



3. Modelling with Least Square Error and Regularization

Naive Models

We will first construct a naive model by giving the same predictions for all of the movies in the validation dataset which is the average of all the movie ratings. First we calculate the mean of the ratings given using the following code, it can be seen that the mean is 3.512465 which means that all of the future movie will be predicted to have a rating of 3.512465.

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

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

This average of the movie rating will also be used in further models as a baseline model. The following RMSE is obtained for the validation dataset

```
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

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

Then by using the following code, we will save the result of the RMSE obtained for this Naive Average Model so that later it can be used as a comparison for the other models.

```
rmse_results <- data_frame(method = "Naive Average movie rating model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
```

```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
rmse_results %>% knitr::kable()
```

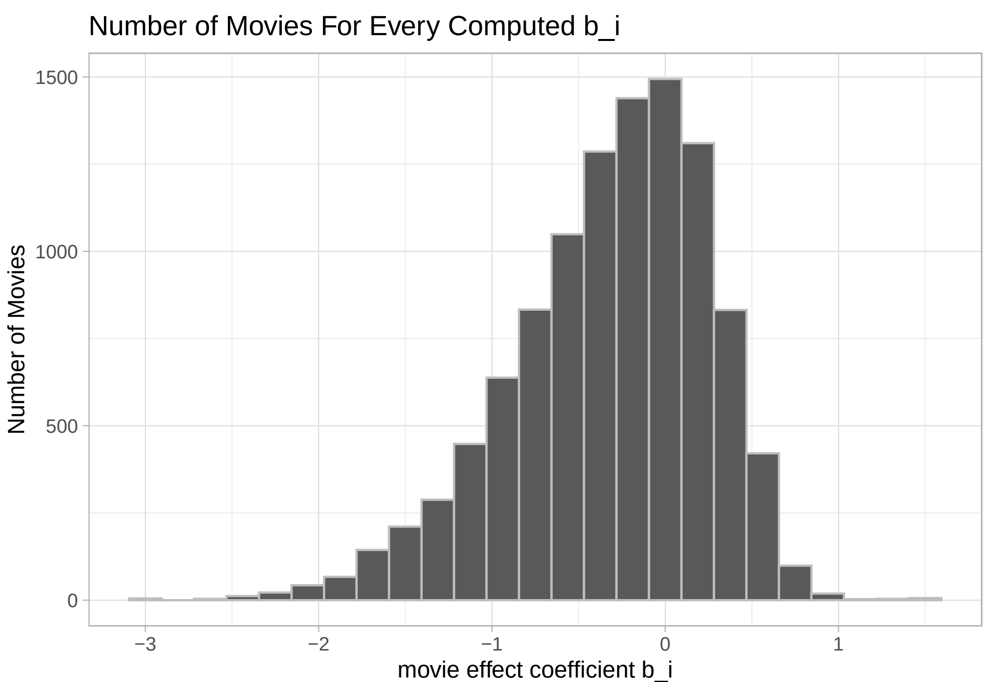
method	RMSE
Naive Average movie rating model	1.061202

Movie Effect Models

Now we will try to take into account the effect of movie in the model. since the it is logical that every movie will have its own unique rating, we will now give a movie effect parameter denoted by b_i or b_i in the code. b_i can be obtained by getting the average of subtracting every rating received in a movie by the mean which is the 3.15 in the previous naive model.

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

The plot that shows the distribution of b_i generated can be seen below. It can be seen that most movies will have a computer b_i with the value near to 0.



Now, Prediction are done by adding the mean with the b_i corresponding to the movieID in validation dataset. Then RMSE will also be calculated to indicate the performance of the model.

```
predicted_ratings_1 <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_1_rmse <- RMSE(predicted_ratings_1, validation$rating)
```

The following code is used to add the current model's RMSE in the table previously created. This same code will also be applied for the future model. From the result obtained, we can see that RMSE obtained decreased from the initial value of larger than 1 to less than 1 which is 0.944.

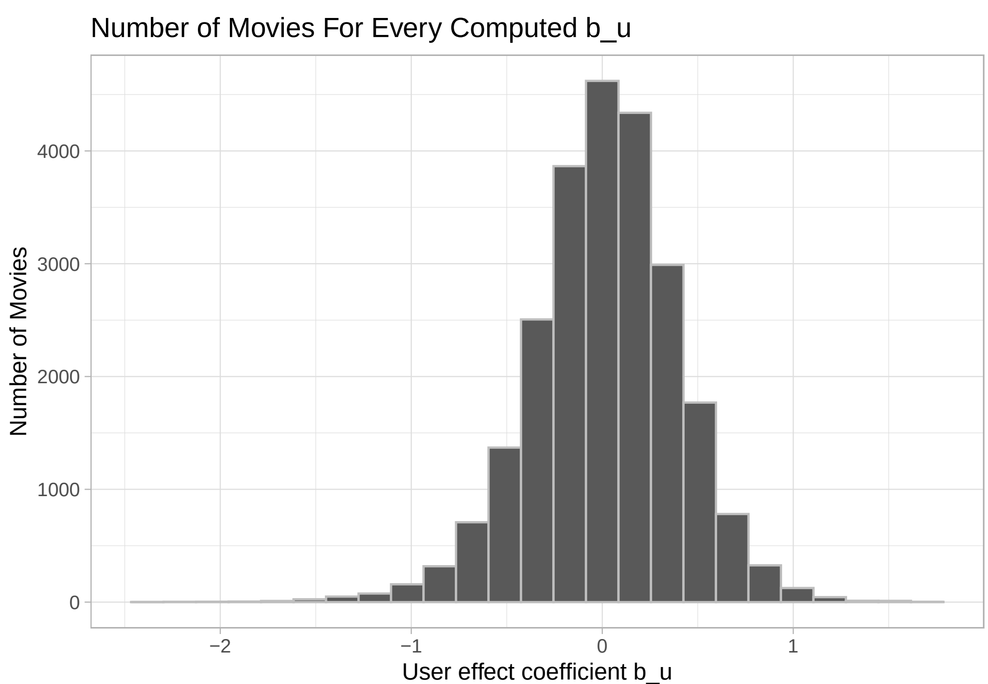
```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive Average movie rating model	1.0612018
Movie effect model	0.9439087

Movie and user effect model

Continuing with the previous idea, now we can see that besides movies, every users also have different tendencies on giving movie ratings. Some users like to give high ratings to any movies while other may be very objective and nitpicky in giving ratings. Due to this, in this model we are going to add an additional user effect denoted by b_u or b_u to the previously movie effect model. First, we can see the distribution of the computed b_u. The plot constructed are based on users that has already rated more than 100 movies to prevent any outliers or bias in the plot. It can be seen below that the distribution of computed b_u is similar to normal distribution

```
`summarise()` ungrouping output (override with `.groups` argument)
```



Then we compute the value b_u for all users

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

Prediction are done by adding the mean with the b_i corresponding to the movieID and b_u corresponding to the userID in validation dataset

```r
predicted_ratings_2 <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

The following table shows that there is an improvement from the previous model since the RMSE obtained become better with the value 0.865

method	RMSE
Naive Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

Regularized movie effect model

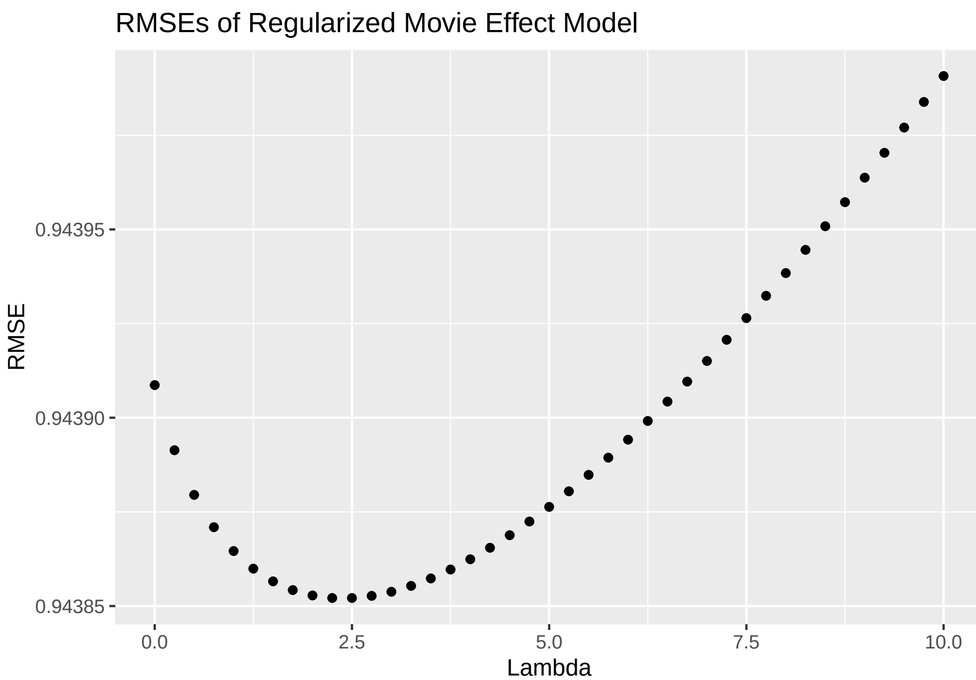
Now we are going to see that there are bias caused by small amount of observation in a group, in this case small amount of ratings given to a movie and small amount of ratings done by a user would cause a bias and hence affecting the performance of the model. Due to this, we will add a regularization parameter that gives a big change (commonly known as penalty) if the amount of observation is smaller and will give small to no change as the number of observations increase indefinitely.

To observe this change, we will now add a regularization to the previously created movie effect model. The initial step is to create a set of values which will be the candidates for the regularization parameter more commonly known as lambda or λ.

Then we will do a cross validation for each value of λ. first b_i will be computed by using the now modified formula of least square error with the additional λ, then the predicted ratings for the validation dataset is also calculated with similar approach in previous models. The λ value that will be chosen is the value that minimizes RMSE.

```r
rmses_bi <- sapply(lambdas, function(l){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  predicted_ratings<-validation%>%
    left_join(b_i, by='movieId') %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, validation$rating))
})
```

using the plot below it can be seen that the lambda that returns the smallest RMSE is around 2.5 and when calculated exactly it will also produce the value 2.5



Adding this results with the previous results shows that the RMSE obtained is not better then the movie and user effect model, however it has a small improvement from the initial movie effect model.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie effect model",
                                     RMSE = min(rmses_bi)))
```

method	RMSE
Naive Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie effect model	0.9438521

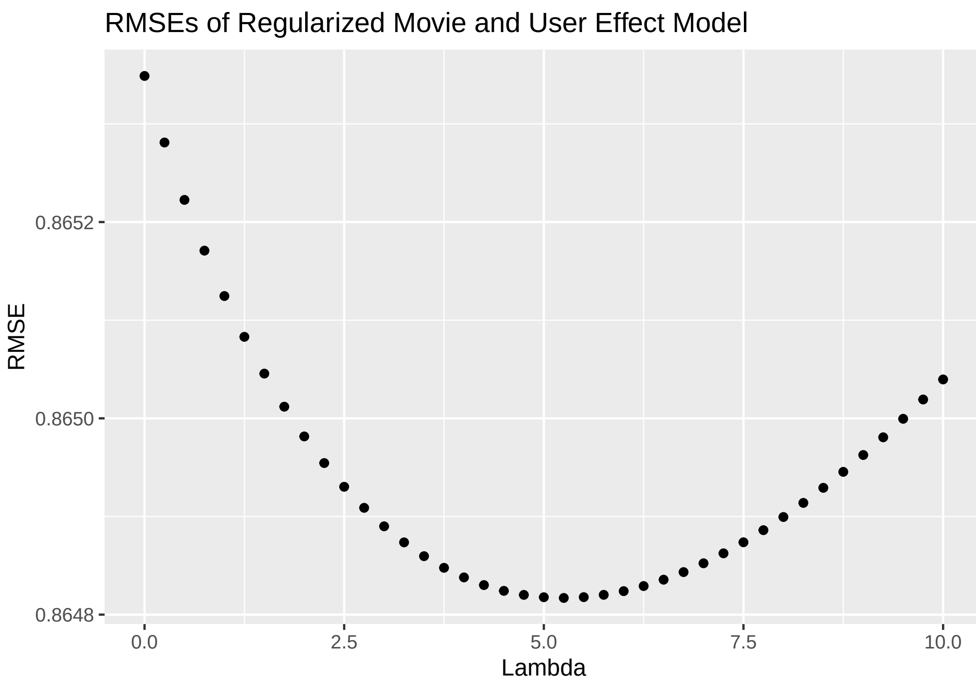
Regularized Movie and User Effect Model

Following the previous regularized model, next the movie and user effect model will also be regularized by first finding the best λ value from the same set of candidates as before.

```
rmses_bi_bu <- sapply(lambdas, function(l){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
```

```
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
})
```

The results in the plot below shows that the λ that returns the smallest RMSE is aroung 5, and it can be checked that the exact amount for λ is 5.25




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

In the result in table below we can see that there is a slight improvement compared to the last model and user effect model.

method	RMSE
Naive Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie effect model	0.9438521
Regularized movie and user effect model	0.8648170

Results

The table below shows the final result of RMSE from different kind of models used to predict movie ratings

method	RMSE
Naive Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

method	RMSE
Regularized movie effect model	0.9438521
Regularized movie and user effect model	0.8648170

From the table, it is clear that as we increase the amount of effect used to explain the model, the RMSE will also decrease indicating a better performance. This is because there might be less error and the model having more capabilities to express the real observations. Then, we can conclude that the best model that can be used to predict movie ratings is the regularized movie and user effect with RMSE 0.8648170.

However it is also advised to conduct a further observation whether the regularized movie and user effect do perform better than the normal movie and user effect. This is due to the relatively small difference between the RMSE of the two model (around 0.0005), which might not be that significant than 0. The significance of this difference needs to be checked by using statistical testing between two means.

Appendix

```
## [1] "Operating System:"
```

```
##                _
## platform       i386-w64-mingw32
## arch           i386
## os             mingw32
## system         i386, mingw32
## status
## major          4
## minor          0.1
## year           2020
## month          06
## day            06
## svn rev        78648
## language       R
## version.string R version 4.0.1 (2020-06-06)
## nickname       See Things Now
```