MOVIELENS PROJECT

HARVARDX CAPSTONE

Nadim Yatim

Table of Contents

Introduction

Recommendation systems use the ratings that the users have already given in order to give recommendations for those users. The Netflix challenge was concerned with improving the recommendation algorithm, which predicts how many rating stars the user is going to give to a certain movie, by 10% for a prize of $1 million. This MovieLens Capstone Project, motivated by the Netflix challenge, requires creating a movie recommendation system by using the MovieLens dataset, especially the 10M version of the MovieLens dataset. The recommendation system will be achieved by training the algorithms considered and using the inputs in edx set in order to predict the movie ratings in the validation set. The root mean square error(RMSE) is going to be used to evaluate the closeness of the predictions to the true values in the validation set.

Methods/Analysis

Loss Function

After constructing different models, we need to compare them in order to see if the improvement that we might have to the baseline model. For this to be quantified, we need to have a loss function. In our case, the loss function will be the residual mean squared error or the RMSE. It is defined as follows where N is the number of user-movie combination and $y_{u,i}$ is considered to be the rating that user u gives for movie i , and $\hat{y}_{u,i}$ is the prediction that we obtain after running our models.

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

The RMSE is somehow similar to the concept standard deviation by being the error that we might make when we predict the rating of a certain movie.

To compute the RMSE for a vector of ratings and predictions then we need to use the following function

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
```

Data Exploration and Visualization

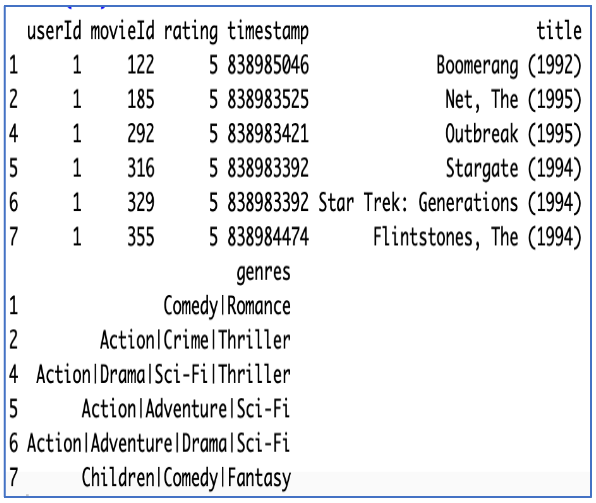
To start considering the datasets that are going to be used later on the in project, a code was provided in the Capstone project in order to create these datasets(edx and validation sets). A chunk of the code is found below.

```r
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
 edx <- movielens[-test_index,]
 temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
 edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

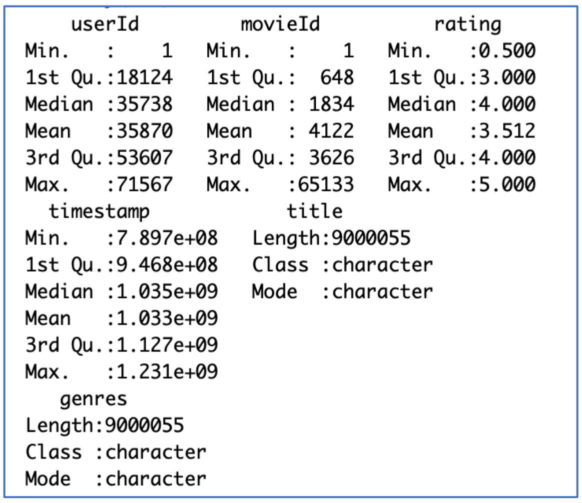
The edx dataset that is obtained is made up of 9,000,055 observations and has 6 features. Each row in this dataset represents a rating given by one user to one movie. The features, their classes and descriptions are as follows:

Feature	Class	Description
UserID	Integer	ID given for each user
MovieID	Numeric	ID given for each movie
Rating	Numeric	Number ranging between 0 and 5 inclusive with increments of 0.5 given to rate the movie
Timestamp	Integer	The timing, in seconds, from which the rating was given
Title	Character	Title of each movie
Genres	Character	Genre to which each movie belongs

A sample of the edx data as well as a summary of each feature is as follows:



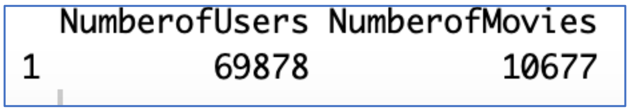
```
  userId movieId rating timestamp                         title
1      1     122      5 838985046              Boomerang (1992)
2      1     185      5 838983525               Net, The (1995)
4      1     292      5 838983421               Outbreak (1995)
5      1     316      5 838983392       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



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

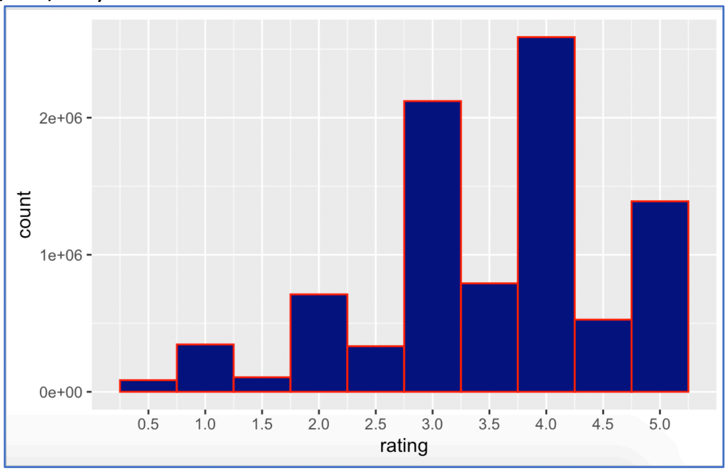
As for the validation set, using the above provided code, we make sure that all the userID and movieID are all present in the edx set. The validation set constitutes of 999,999 observations and 6 features(which are the same as those in the edx set).

Some userIDs as well as movieIDs are repeated more than once. So it's important to know how many people were involved in rating the movies and how many movies were rated. So in total we have 69,878 users that rated the movies and an overall number of 10,677 movies as shown below:

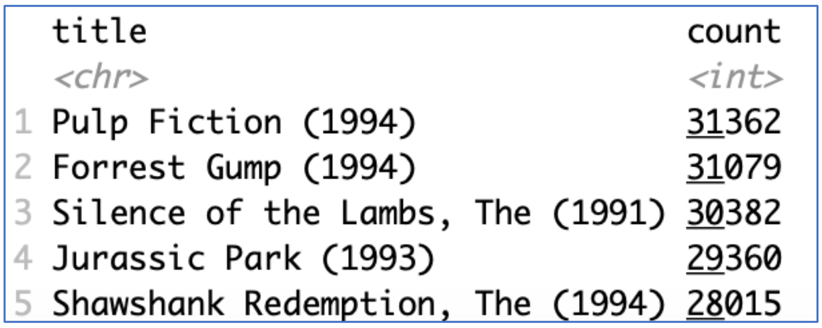


```
  NumberofUsers NumberofMovies
1         69878          10677
```

Furthermore, moving on to the ratings we can see through the summary that is ranges between 0.5 and 5 and the histogram of the rating below shows that ratings that were mostly used were 4, 3 and 5 in descending order. Another conclusion that could be made from the histogram is that whole ratings(1,2,3,4,5) are more commonly used than half ratings(0.5,1.5,2.5,3.5,4.5)

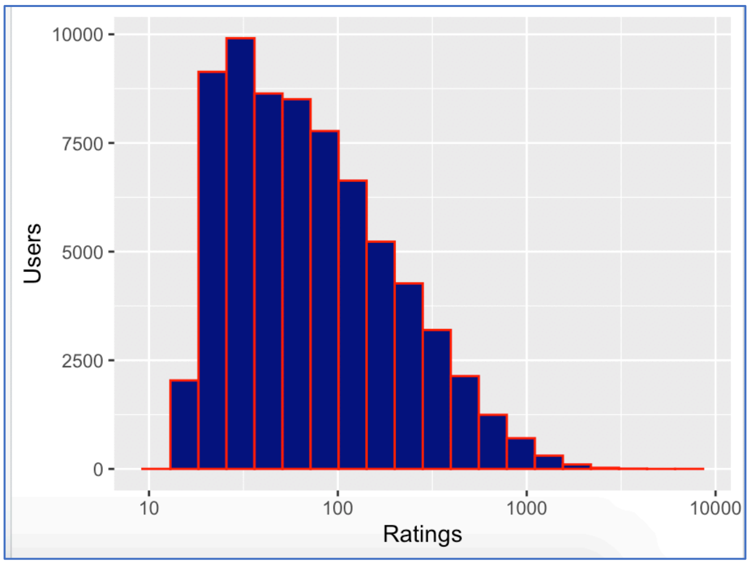


Moving on to the movie titles, we see that the top 5 movies per movie ratings as shown below:

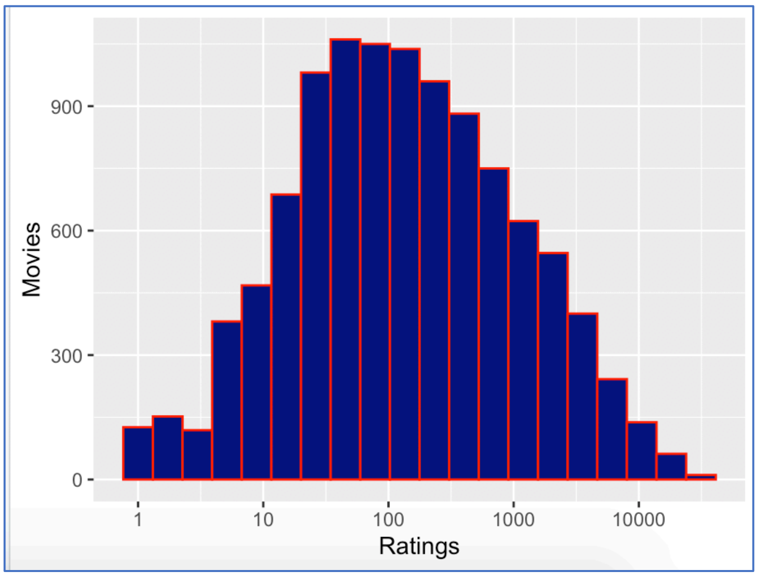


```
  title                             count
  <chr>                             <int>
1 Pulp Fiction (1994)               31362
2 Forrest Gump (1994)               31079
3 Silence of the Lambs, The (1991)  30382
4 Jurassic Park (1993)              29360
5 Shawshank Redemption, The (1994)  28015
```

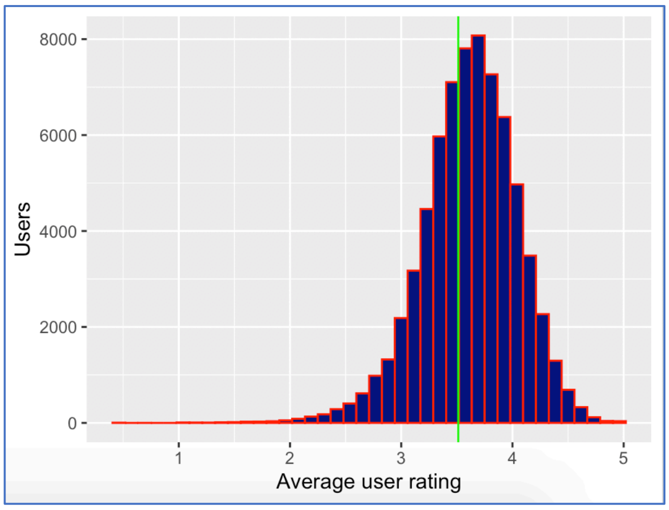
Moreover, we now need to consider what are the effects of some of the features, especially movieId and userId on the ratings. Starting with UserId, as we can see in the below histogram, some users are more active than others in rating movies where some users have rated more than 1000 movies while others only rated about 20 movies. The majority of users rated between 35 and 150 movies.

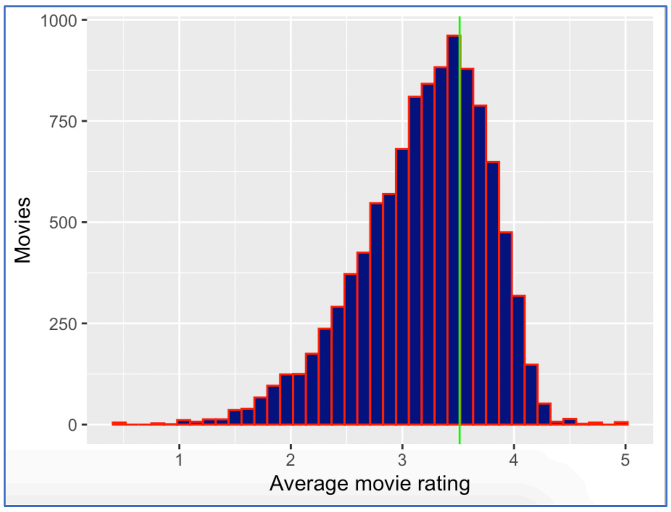


Moving on to the effect of movieId on ratings, as seen in the histogram below, we notice that some movies have a higher number of total ratings than others. This may be due to the popularity of the movie, like some being blockbusters while others aren't that popular.



Now, let's consider the average rating per userId as well as that per movieId as seen in the below constructed histograms. This indicates that there can be bias based on the user and also based on the movie where we can have very low as well as very high ratings.





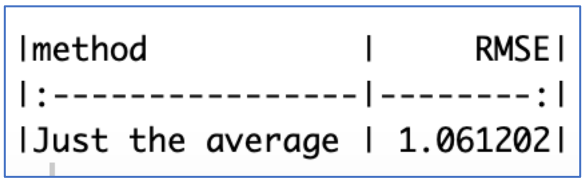
Models

The Average Model

The first model that we are going to use is going to predict ratings based on the sample mean. This model assumes that the ratings for all the movies and users are the same. The model is as follows where μ represents the true rating of all the movies and the users.

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

As we expect the RMSE is considered to be quite high and has a value of **1.061202**



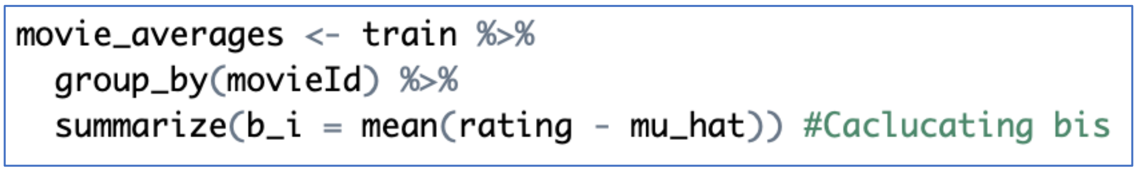
```
|method            |      RMSE|
|:-----------------|---------:|
|Just the average  |  1.061202|
```

Movie Effect Model

As we have seen previously in the histogram constructed for the average rating for each movie, generally some movies are rated higher than others. So we can add to the previous model the term b_i which will represent the average rating for movie i. So the resulting model is as follows

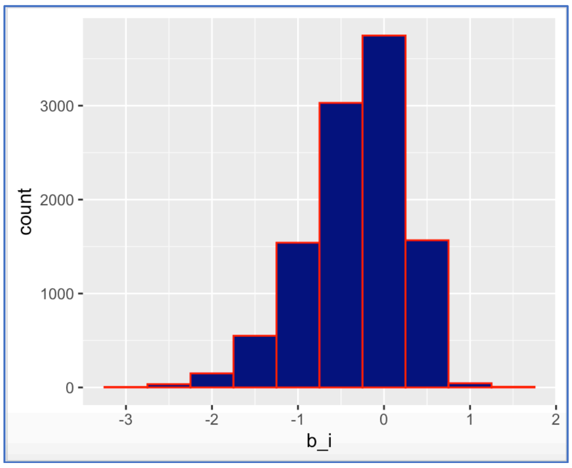
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

Instead of running the least squares to estimate the b's which would be very slow, we calculate the bi's using the following code:

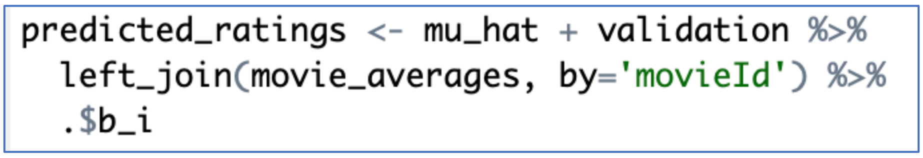


```
movie_averages <- train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat)) #Caclucating bis
```

When plotting the b_i s we see that these estimates actually vary a lot where some movies are considered to be good while others are not. Since the average is about rating is about 3.5, then if we have a bi of 1.5 this means that the we have a rating of 5.0

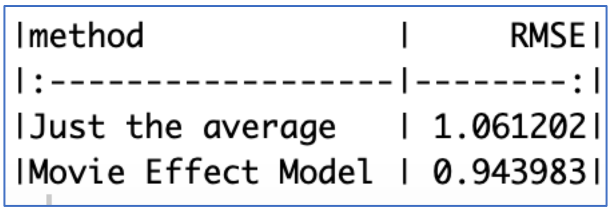


In order to see if our predictions improve or not, we execute the following piece of code.



```
predicted_ratings <- mu_hat + validation %>%
  left_join(movie_averages, by='movieId') %>%
  .$b_i
```

We then calculate the RMSE and as shown in the RMSE results, the second method that we used improved our results and reduced the RMSE to **0.943983**



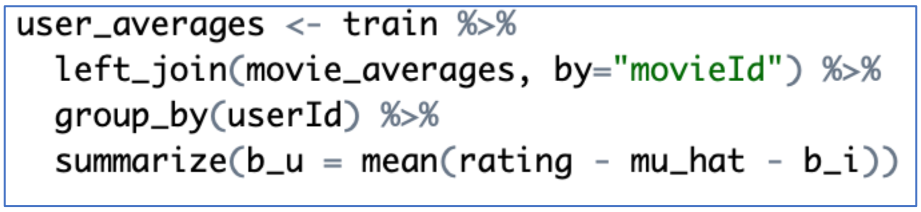
```
|method             |     RMSE|
|:------------------|--------:|
|Just the average   | 1.061202|
|Movie Effect Model | 0.943983|
```

Movie and User Effects Model

As seen previously in the histogram showing the average rating foe each user, we seen that there is high variability in the user ratings. Thus, we can further improve our model by adding a term to our established model, b_u, that is related the user-specific effect and how it can affect our ratings. The model obtained is as follows

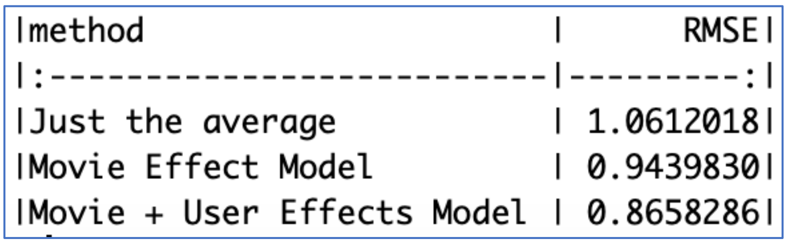
$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

This shows that if a user rates poorly a movie having a positive bi, the effects of both biases will cancel each other leading to a correct and improved prediction.
In order to fit this model, we are going to compute the values of the b_u as follows:



```
user_averages <- train %>%
  left_join(movie_averages, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))
```

If we predict the values based on our new model and recalculate the RMSE we observe an additional improvement than the previous model reaching a value of **0.8658286**



```
|method                        |      RMSE|
|:-----------------------------|---------:|
|Just the average              | 1.0612018|
|Movie Effect Model            | 0.9439830|
|Movie + User Effects Model    | 0.8658286|
```

Regularized Movie Effect Model

If we return to our second model, we see that the best and worst movies were rated by only very few users.

Regularization allows us to penalize large estimates that come from small sample sizes. It is similar to the Bayesian approaches in which it has several commonalities with it. Moreover, it works by adding a penalty for large values of b, the bias or effect, to the sum of squares equations that is being minimized.

Instead of minimizing the residual sum of squares as the least squares does, now we are minimizing the following equation in order to estimate the b_is:

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

This equation can be split into two terms where the first is as follows.

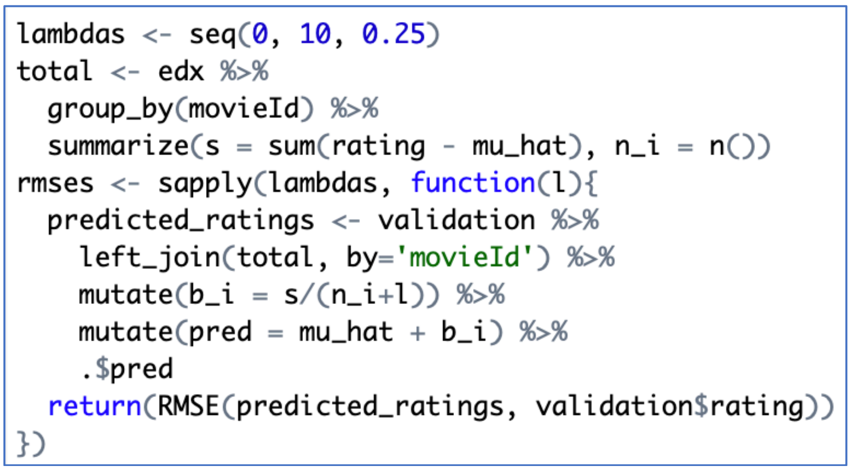
$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2$$

This term represents the residual sum of squares.

The second term in this equation is the penalty term and it increases as the b_is are large enough. This term has the parameter λ which is a tuning parameter.

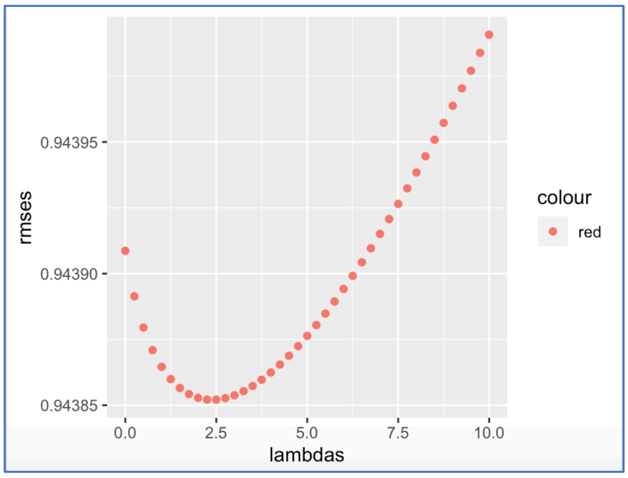
$$\lambda\sum_i b_i^2$$

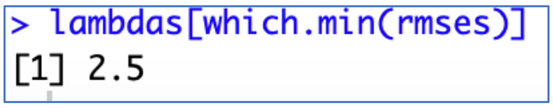
Cross validation is used in order to pick the optimal λ that minimizes the RMSE as follows



```r
lambdas <- seq(0, 10, 0.25)
total <- edx %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu_hat), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- validation %>%
    left_join(total, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu_hat + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, validation$rating))
})
```

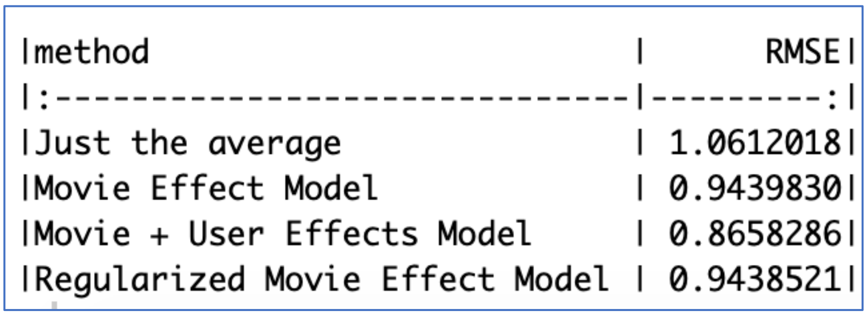
If we plot the RMSE that we obtained for each obtained, we get the following plot which shows that the λ that will allow us to obtain the minimum RMSE if 2.5.





```
> lambdas[which.min(rmses)]
[1] 2.5
```

This λ yields an RMSE of 0**.9438521** which is a slight improvement to the movie effect model before being regularized.



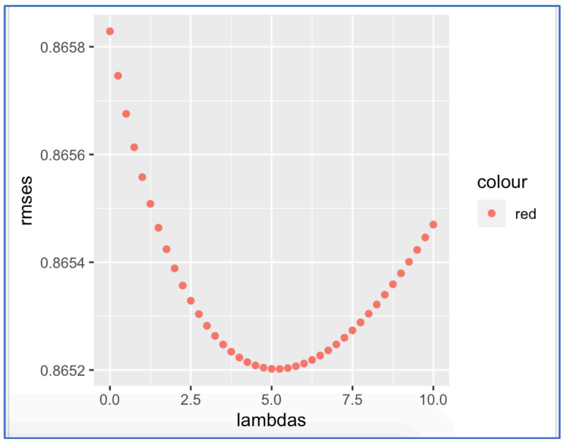
```
|method                          |      RMSE|
|:-------------------------------|---------:|
|Just the average                | 1.0612018|
|Movie Effect Model              | 0.9439830|
|Movie + User Effects Model      | 0.8658286|
|Regularized Movie Effect Model  | 0.9438521|
```

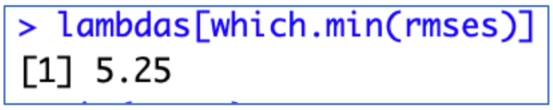
Regularized Movie and Users Effects Model

Regularization can also be used to estimate the user effect, and this requires minimizing the following equation

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

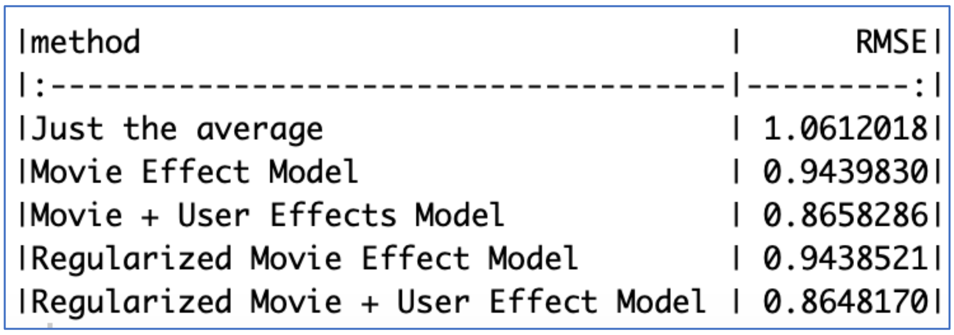
Similar to what was done in the Regularized Movie Effect Model, is chosen by cross validation and we obtain that the λ that will allow us to have the minimum RMSE is 5.25.





```
> lambdas[which.min(rmses)]
[1] 5.25
```

Accordingly, choosing λ equals to 5.25, will allow us to reach an RMSE of **0.864817** which is also considered to be an improvement on the Movie and User Effects Model and it yields the lowest RMSE compared to the 4 other models.



```
|method                              |      RMSE|
|:-----------------------------------|---------:|
|Just the average                    | 1.0612018|
|Movie Effect Model                  | 0.9439830|
|Movie + User Effects Model          | 0.8658286|
|Regularized Movie Effect Model      | 0.9438521|
|Regularized Movie + User Effect Model | 0.8648170|
```

Results

We have considered 5 models as described above and each model resulted in a specific value of RMSE and we saw that the least values of the RMSE were due to regularization. We were able to reach an RMSE less than 0.86490 when using the regularized movie and user effects model. The values of the RMSEs across the 5 models are displayed in the table below

Method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439830
Movie and User Effects Model	0.8658286
Regularized Movie Effect Model	0.9438521
Regularized Movie and User Effects Model	0.8648170

Conclusion

After considering several models, from naïve to considering movieId effect alone and combining it with the userId whether it has been regularized or not, we have seen that the regularized model considering both the movieId and userId effects resulted in obtaining the least RMSE having a value of 0.864817. This is a huge improvement from the baseline model which resulted in an RMSE of 1.0612018 meaning that the rating error is more than one star. So we were able to continue on improving our predictions leading to lower errors until we reached a minimum of 0.864817 with the final regularized model. Further addition and consideration of genre and age effects in regularized models are also expected to result in decrease in RMSE to further lower values than the minimum obtained. Alternative machine learning models can be considered and could result in further reductions of the RMSE but some limitations such as computer power and ability might be a challenge for running such algorithms and models.