Movielens_Analysis Movie Rating Recommendation

Ramesh K Patil

25/05/2020

Executive Summary:

The Netflix's movie recommendation system was an open competition for the best collaborative filtering algorithm to predict user ratings for films,based on previous ratings without any other information about the users or films.The challenge was to improve Netflix's movie recommendation system by 10% and so on. The Netflix movie rating system uses a rating range from 1 to 5 so based on the predictions, the movie with the highest predicted rating for a user is then recommended to the user.

The objective of the project is to predict movie rating by training a machine learning algorithm using inputs from test data set to predict movie ratings in validation data set. As there are different factors pertaining to rate the movies, it depends on different genres, liked or disliked movies where User can rate movies for pecific genres, only movies they liked or rate the movies they disliked.

This brings biases in the movie ratings and should be considered while predicting movie ratings using machine learning models. The Goal of Netflix was to reduce an error as much as possible challenge so,Winner is decided based on the lowest residual mean squared error (RMSE) on a test set where expected reduction in RMSE was 10%. For this project, We will train different models and the best model was chosen based on the RMSE on the validation set using the 10M version of MovieLens dataset, collected by GroupLens Research.

Project Objectives:

MovieLens Rating Prediction Project have below Objectives:

```
1. Train a machine learning algorithm that predicts user ratings using the inputs of edx (test)
dataset to predict movie ratings in validation set.

2. Algorithm performance will be evaluated based on Root Mean Square Error i.e. RMSE
and outcome to defined as lower RMSE is better.

3. Developed models will be compared based on RMSE results for performance and accuracy.
The evaluation of algorithm is RMSE < 0.86490.
```

RMSE is one of the imporatant parameter used to measure differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset. The effect of each error on RMSE is proportional to the size of the squared error.

The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

Finally, the best resulting model based on RMSE will be used to predict the movie ratings.

Dataset Used For Analysis:

The MovieLens dataset will be used for creating test and validation testsets and it will be automatically downloaded from,

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
############################################################
# Create edx set, validation set
############################################################
# Note: Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

To predict the movie ratings, we will split the MovieLens dataset into 2 sub- datasets,

1. The “edx”, a test dataset to develop and train our algorithm,
2. The “validation” a dataset to test final the movie ratings predictions.

```
# 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)
```

So as mentioned above, algorithm will be developed and trainned using "edx" dataset and Movie ratings will be predicted in "validation" dataset as if they were unknown.

Methods/Analysis:

Data Exploration:

As we see our MovieLens dataset was having 10 million ratings initially, After creating the sub datasets, we observe that "edx" test dataset contains 9 miliion and "validation" dataset contains 1 million rows spanned across 6 columns.

Have a look at few rows from training dataset "edx" in tidy format

```
head(edx) %>% as_tibble()
```

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

The "edx" dataset contain the six variables, i.e. columns as "userID", "movieID", "rating", "timestamp", "title", and "genres". Each row in dataset represents a one to one relationship of user providing ratig to movie.

Now , have a look at few rows from validation dataset "validation" in tidy format

```
head(validation) %>% as_tibble()
```

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title                    genres
##    <int>   <dbl>  <dbl>     <int> <chr>                    <chr>
## 1      1     231      5 838983392 Dumb & Dumber (1994)     Comedy
## 2      1     480      5 838983653 Jurassic Park (1993)     Action|Adventure|S~
## 3      1     586      5 838984068 Home Alone (1990)        Children|Comedy
## 4      2     151      3 868246450 Rob Roy (1995)           Action|Drama|Roman~
## 5      2     858      2 868245645 Godfather, The (1972)    Crime|Drama
## 6      2    1544      3 868245920 Lost World: Jurassic Park~ Action|Adventure|H~
```

Looking at summary of "edx" dataset, we can make sure there are no missing values involved.

```
summary(edx)
```

```
##      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
##
##
##
```

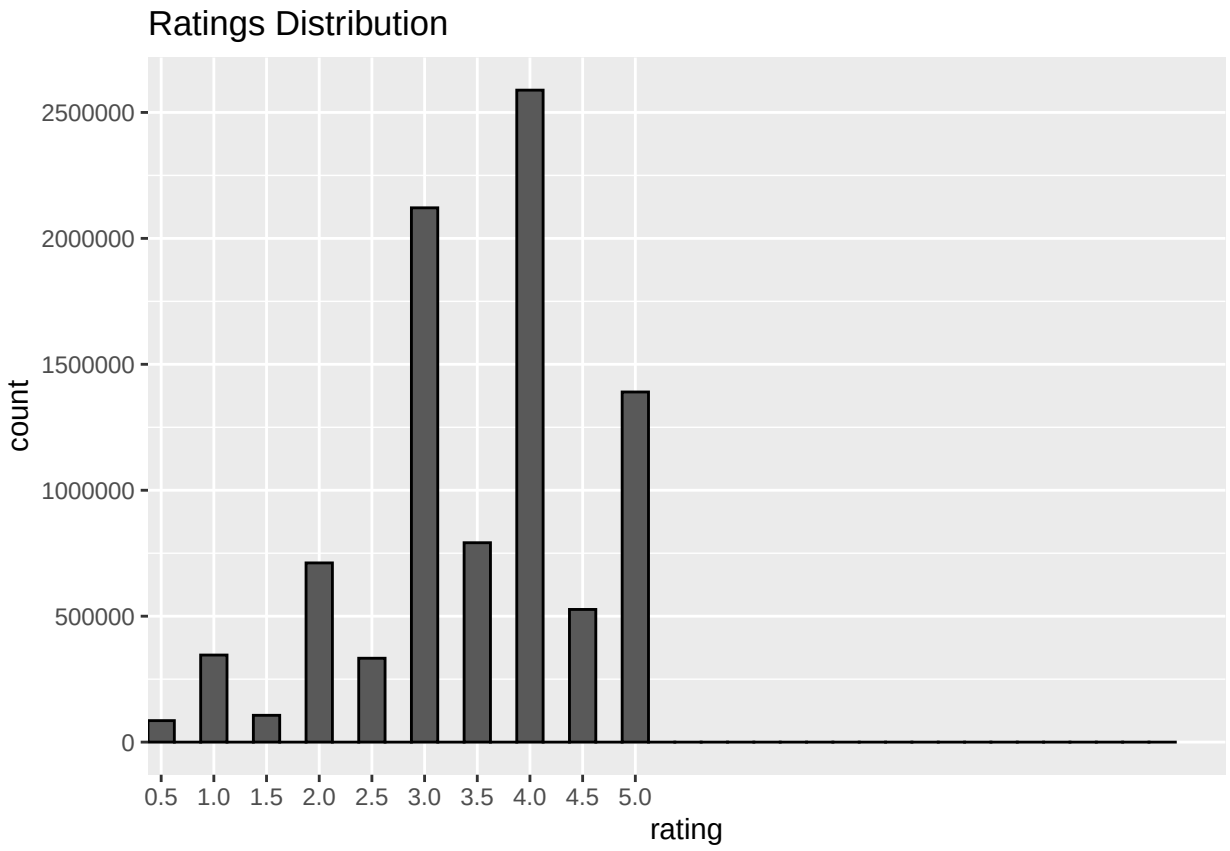
When we tried to identify Unique Users and movies present in “edx” datset using below lines of code, we get around 70K unique users and 10.700 unique movies,

```
##   num_users num_movies
## 1     69878      10677
```

Distributions/Data Visualization:

Ratings Distribution clearly shows that,

1. Users have tendency to rate the movies higher.
2. Half star ratings are less common compared to full star ratings.
3. Rating 4 is most used rating, followed by 3 ,5 and 0.5 is less common.

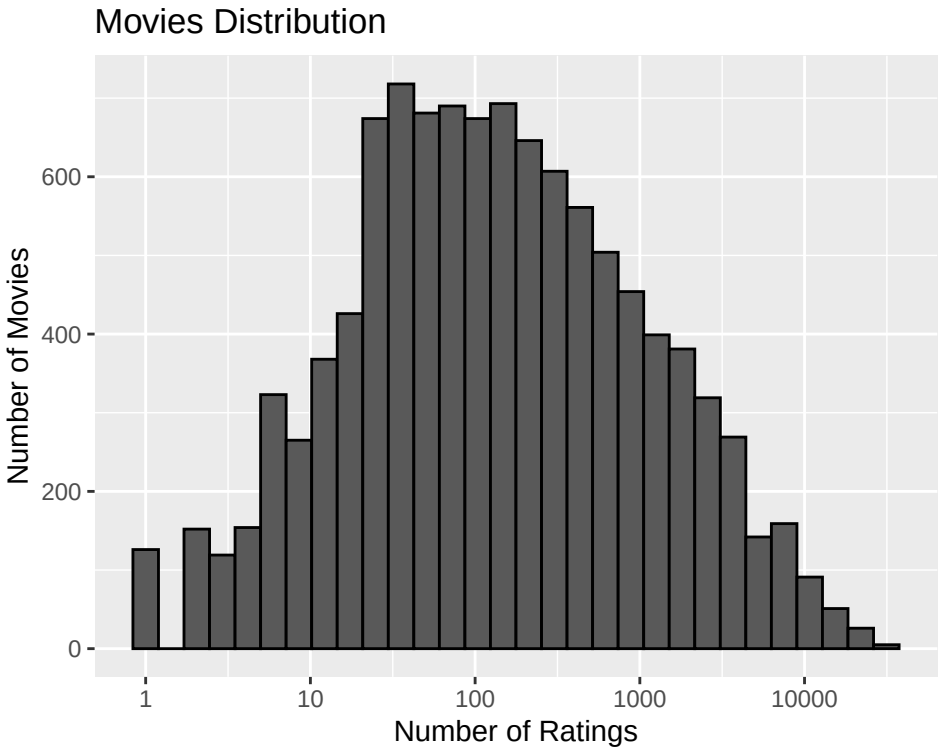


Movies Distribution i.e. number of ratings per movie shows below observations,

1. Few movies are rated much often as compared to others.
2. Few movies are rated few times.
3. Around 125 movies have been rated only once.

To avoid incorrect predictions from models due to less ratings for some movies, we need to consider regularisation and a penalty terms in the models. Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values.

```
edx %>%   count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Movies") +
  ggtitle("Movies Distribution")
```



Predicting a rating for 125 movies that were rated only once appears to be difficult based on very limited data available. We can look at 20 movies those were rated only once by slicing as follows,

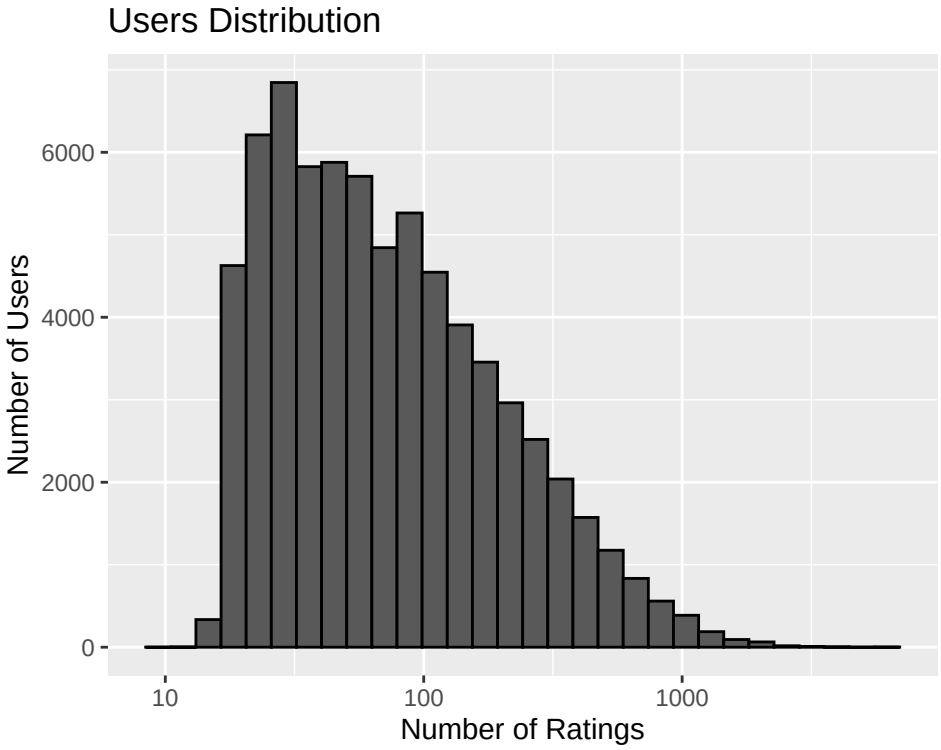
```
edx %>% group_by(movieId) %>%
  summarize(counter = n()) %>%
  filter(counter == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(Rating = first(rating), num_rating = first(counter)) %>%
  slice(1:20) %>%
  knitr::kable()
```

title	Rating	num_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

title	Rating	num_rating
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

After looking at Histogram for Users Distribution, We can observe that the majority of users have rated between 30 and 100 movies.So, Our models should be considering a user penalty term later.

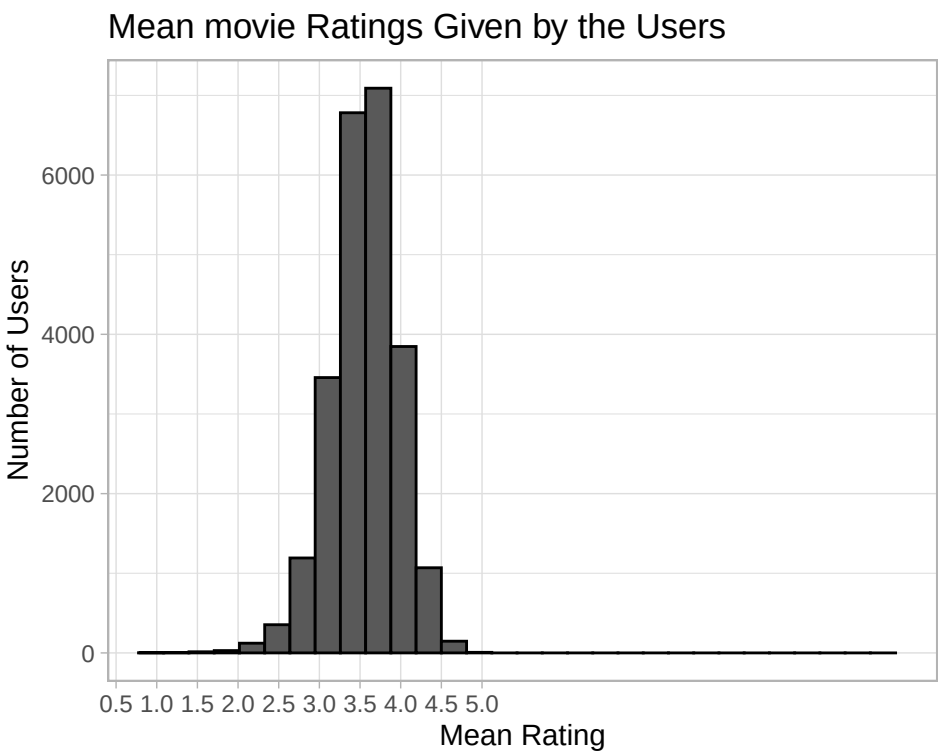
```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Users") +
  ggtitle("Users Distribution")
```



As different Users have different tendency to rate the movie i.e. few rate movies higher and few rate movies lower based on different tendency. If we plot mean movie ratings given by the users where only those Users will be considered who have rated atleast 100 movies, we can clearly observe this.

```
edx %>% group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(bias_u = mean(rating)) %>%
  ggplot(aes(bias_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean Rating") +
  ylab("Number of Users") +
  ggtitle("Mean movie Ratings Given by the Users") +
```

```
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



Modelling Approach

As discussed previously, We will be using RMSE as measure of an accuracy. So we will compute RMSE for each model and then we have to choose best one i.e. Lower RMSE means higher accuracy of the model.

RMSE is computed using loss function as follows,

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

Where, N is number of user/movie combinations and the sum occurring over all these combinations.

As RMSE is an sort of error we make while predicting which can be considered as Standard Deviation, Higher Value of RMSE means typical error is larger. So, We have to be focusing on minimizing our errors to reach correct or optimal Prediction.

The written function to compute the RMSE for vectors of ratings and their corresponding predictions is,

```
RMSECalc <- function(actual_ratings, predicted_ratings){
  sqrt(mean((actual_ratings - predicted_ratings)^2))
}
```

Model I. Average Movie Rating (Naive Approach):

In our first model for recommendationsystem, we will consider predicting the same rating to all of the movies with no consideration of Users. To do this, we will be using mean rating of the dataset.

This model assumes that ,the same rating will be given for all of the movie with all differences explained by random variation,

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

with $\epsilon_{u,i}$ independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies.

Average movie rating model makes the assumption that all differences in movie ratings are explained by random variation alone. We know that the estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all ratings:

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

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

If we predict all unknown ratings with μ or mu, we obtain the first naive RMSE:

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

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

Here, we represent results table with the RMSE using Average Movie Rating or Naive approach:

```
rmse_results <- tibble(method = "Average Movie Rating : Naive Approach", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average Movie Rating : Naive Approach	1.061202

Now, This is our baseline and RMSE results from other models will be compared to other Model results.

Model II. Movie Effect:

Considering that all movies rated with same rating is unrealistic, we need to start thinking to improve our model. As we know that, some movies will be rated higher as compared to others depending on popularity.Also, some movies will be rated often whereas others rarely.

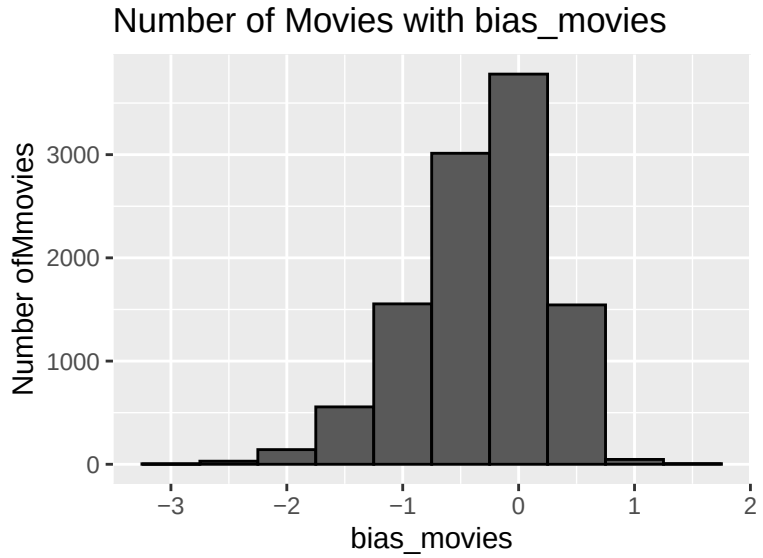
We compute the estimated deviation of each movies's mean rating from the total mean of all movies μ.

The resulting variable is called "bias" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

The histogram is left skewed, implying that more movies have negative effects.

```
movie_avgs <- edx %>% group_by(movieId) %>%
  summarize(bias_movies = mean(rating - mu))
#plot movie bias "bias_movies"
movie_avgs %>% qplot(bias_movies, geom ="histogram", bins = 10, data = ., color = I("black"),
                     ylab = "Number ofMmovies", main = "Number of Movies with bias_movies")
```



This is the penalty term movie effect when we consider that there are bias in rating movies.

Our prediction improves, once we predict using this model.

```
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(bias_movies)

model_movieeffect_rmse <- RMSECalc(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results, tibble(method="Movie Effect Model",
                                    RMSE = model_movieeffect_rmse ))

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

method	RMSE
Average Movie Rating : Naive Approach	1.0612018
Movie Effect Model	0.9439087

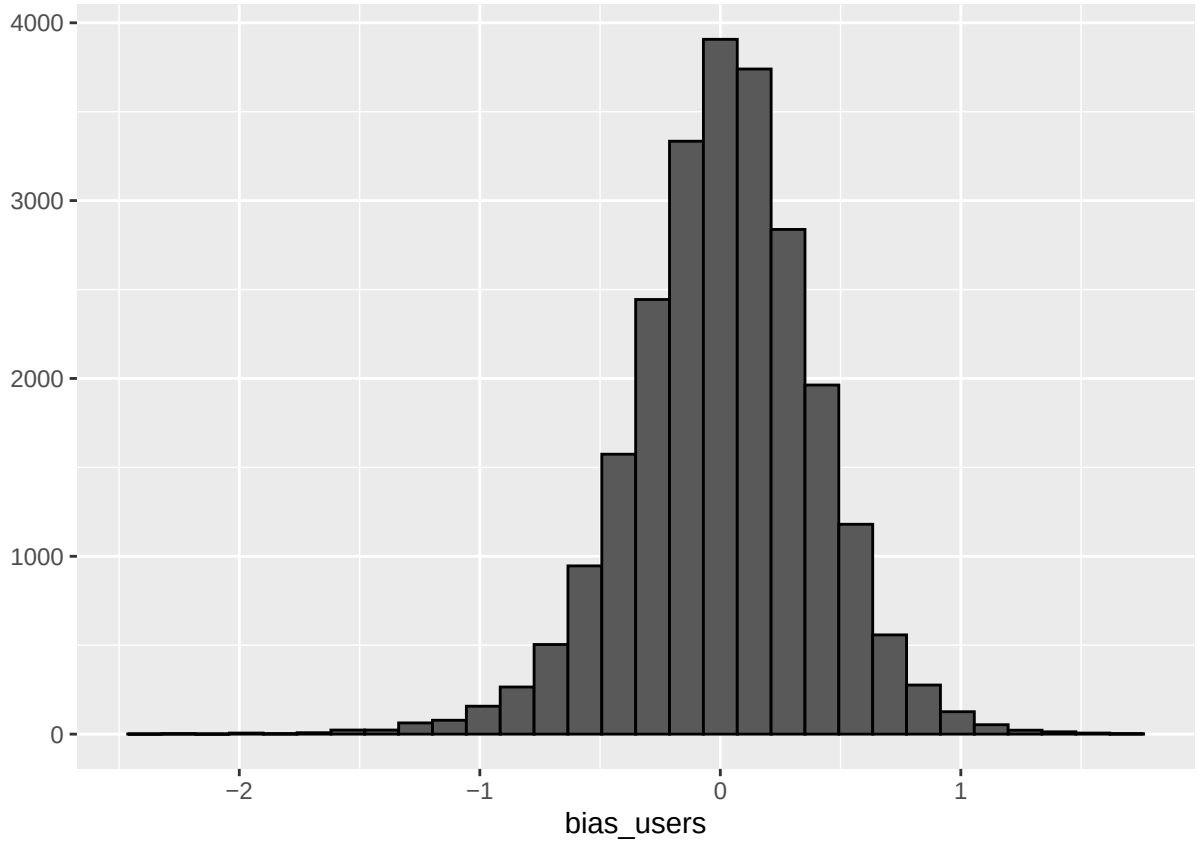
So we have predicted movie rating considering Movie Effect or bias that movies will be rated differently by adding the computed b_i to μ.

Adding movie effect to model improves prediction but does not consider the user rating effect.So in next model we will be adding this user bias to seek another level of improvement in our prediction.

Model III. Movie And User Effect:

As ratings will be affected positively or negatively by Users likes/dislikes. Considering penalty term user effect, we will calculate average rating for an user μ, for those that have rated over 100 movies.

```
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(bias_users = mean(rating - mu - bias_movies))
#Plot average ratings by users
user_avgs%>% qplot(bias_users, geom ="histogram", bins = 30, data = ., color = I("black"))
```



As there is wide variability across users where few likes very few movies and others like each and every movie, so it depicts that our model can be improved like,

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

where b_u is a user-specific effect. If a cranky user (negative b_u rates a great movie (positive b_i), the effects counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

We compute an approximation by computing μ and b_i, and estimating b_u, as the average of

$$Y_{u,i} - \mu - b_i$$

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

Now construct the predictors and verify if RMSE improves,

```
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(prediction = mu + bias_movies + bias_users) %>%
  pull(prediction)


#Calculate RMSE considering Movie and Users effect in th model
model_movieusereffect_rmse <- RMSECalc(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method="Movie and User Effect Model",
                                     RMSE = model_movieusereffect_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.
```

```
#Save the results to table
rmse_results %>% knitr::kable()
```

method	RMSE
Average Movie Rating : Naive Approach	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

Here, we can observe that rating prediction has intended effect on RMSE i.e. reduced. Another point to remember that we used only movies in our first model. This considers that "best " and "worst" movie were rated by few users, in most cases just one user. These movies were mostly obscure ones. This is because with a few users, we have more uncertainty. Therefore larger estimates of b_i, negative or positive. As we know that Large errors can increase our RMSE.

Till this model, we have computed standard error and constructed confidence intervals to account for different levels of uncertainty. However, while making predictions , we should be having a single number and prediction and not an interval. To take this into account, we will be using regularization, that permits to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for large values of b_i to the sum of squares equation that we minimize. So having many large b_i, make it harder to minimize.

Model IV. Regularizing Movie And User Effect:

Now we will be using Regularization, which is a method used to reduce the effect of overfitting.

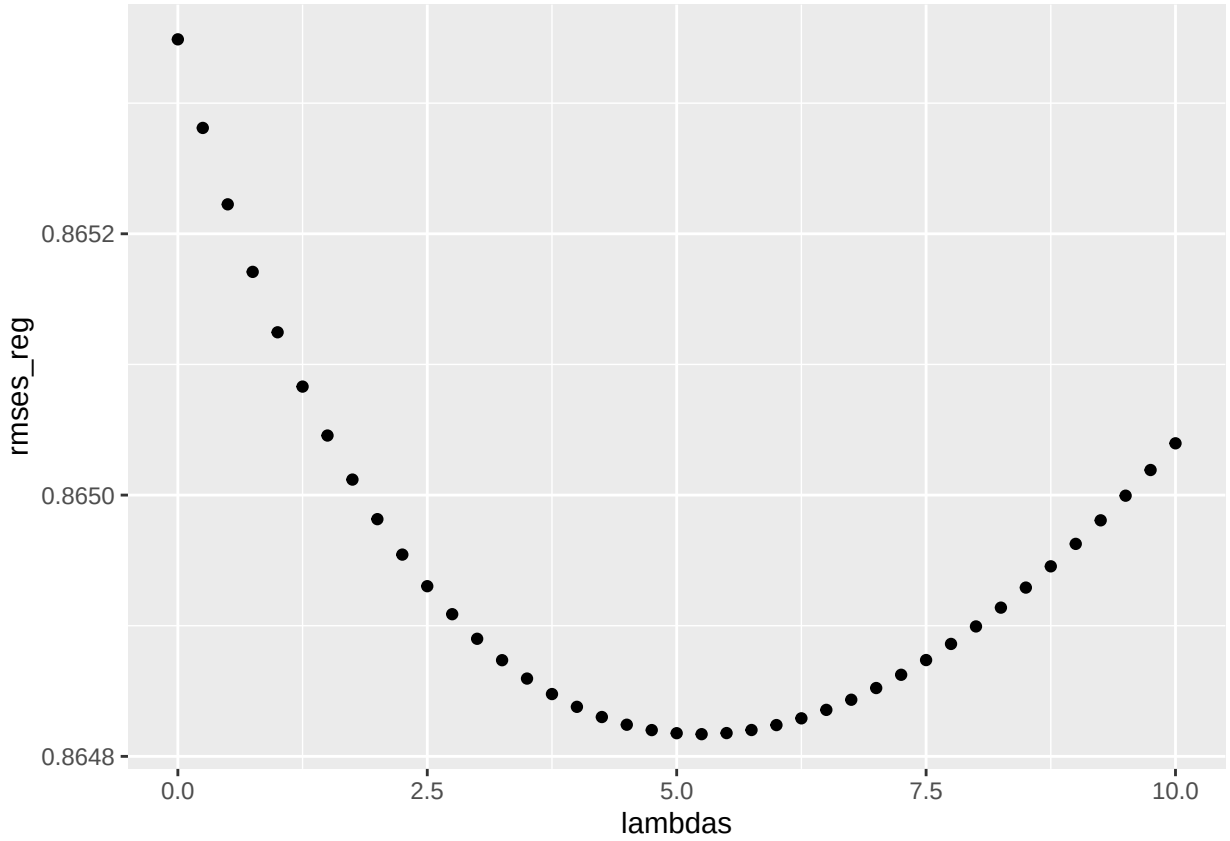
So estimates of b_i and b_u are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these aspects. We should find the value of lambda (that is a tuning parameter) that will

minimize the RMSE. This shrinks the b_i and b_u in case of small number of ratings.

```
lambdas <- seq(0, 10, 0.25)


rmses_reg <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  bias_movie <- edx %>%
    group_by(movieId) %>%
    summarize(bias_movie = sum(rating - mu)/(n()+l))

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

  predicted_ratings <-
    validation %>%
    left_join(bias_movie, by = "movieId") %>%
    left_join(bias_user, by = "userId") %>%
    mutate(prediction = mu + bias_movie + bias_user) %>%
    pull(prediction)

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

To Select optimal lambda, we will be plotting RMSE against RMSE as follows,

```
qplot(lambdas, rmses_reg)
```



Optimal lambda is as follows for the full model and compusted as,

```
  lambda <- lambdas[which.min(rmses_reg)]
lambda
```

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

Optimal lambda is 5.25 for the full model.

The RMSE results will be as follows considering regularization,

```
rmse_results <- bind_rows(rmse_results, data_frame(method="Regularized Movie and User Effect model",
                                     RMSE = min(rmses_reg)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average Movie Rating : Naive Approach	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect model	0.8648170

Data Analysis Results:

We can list down the RMSE values for all the model defined as below,

method	RMSE
Average Movie Rating : Naive Approach	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect model	0.8648170

Looking at the results , we confirm that we have lowest RMSE value as 0.8648170.

Conclusion

As per the analysis performed, we can see that final model which gives promising results i.e. lowest RMSE is,

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

Looking at analysis results, we can confirm that the lowest RMSE achieved is 0.8648170 when used project model mentioned above. This model has regularized variations of movies and users and hence efficient model compared to other models. When we did only considered same rating for movies, we got RMSE value as 1.0612018 which is quite high and depicts higher error. Then we did considered, movie effect as some movies are rated high as compared to other and we got RMSE as 0.9439087 with some improvement comapred to first model. While doing so, we still missed user effect and hence after adding user effect along with movie effect , we got lower RMSE as 0.8653488 compared to second model. Regularizing Movie effect for those have very few ratings and user efftect where some users only rated very few movies has helped us improving our model to get lower RMSE. We may be able to further improve the model to achieve lower RMSE by conisdering year effect, genre effect etc. whereas dataset wrangling needed for genre column to have single value of genre which calls for splitting dataset genere wise for movies having more than one genre, which could be complicated to perform as it increases number of records in dataset and may cause hardware limitations as well.

References:

1. https://en.wikipedia.org/wiki/Netflix_Prize
2. http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
3. https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
4. https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52