Recommened system to Predict Movie rating- Movilens Dataset

Ladan Foroughi

12/08/2021

Introduction

This project is related to the Movielens project of HarvardX:PH125:9x Data Science Capstone. The MovieLens datasets consist of 10000054 ratings that is applied for 10681 movies by 71567 user between 1937 and 2009.The main goal of this project is predicted the movie rate for suggestion to user based on the Movielens dataset. In order to have the good prediction, the build of algorithm with minimize of Root Mean Sequared Error (RMSE) as close to zero is goal.

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

Methodology

The first step , the required package has to installed from http://cran.us.r-project.org. the list of library that used in this project,

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(stringr)
library(knitr)
library(dplyr)
library(purrr)
library(gridExtra)
```

The movilens datasets was downloaded from this website: https://grouplens.org/datasets/movielens/10m/.

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

Two datasets as ratings and movies have to join to create the movielens dataset.

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

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(movieId),
         title = as.character(title),
         genres = as.character(genres))

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

The six first rows of movielens dataset is shown in Table-1.

```
kable(head(data.frame(movielens)),"pandoc",caption = "Movielens datasets",align = "c")
```

Table 1: Movielens datasets

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

The list of variables in Movielens dataset is shown in Table 2. In this project the UserId, MovieID, timestamp, title, genres are features and rating is outcomes.

```
kable(names(movielens),"pandoc", caption = "List of variables in movielens dataset", align = "c")
```

Table 2: List of variables in movielens dataset

x
userId
movieId
rating
timestamp
title
genres

The timestamp and title have to modified that can be easy used for analyzing.

The timestamp is an integer that shows the date but It has to convert to suitable format of date. The year_rated variable is replaced to the timestamp variable in movielens datasets. The first six rows of Movielens dataset are shown in Table 3 after these modification.

```
movielens <- movielens %>%
  mutate(year_rated = year(as_datetime(timestamp))) %>% select(-timestamp)
kable(head(movielens),"pandoc",caption = "Movielens dataset with replacing year_rated to timestamp", align = "c'
```

Table 3: Movielens dataset with replacing year_rated to timestamp

userId	movieId	rating	title	genres	year_rated
1	122	5	Boomerang (1992)	Comedy|Romance	1996
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996
1	231	5	Dumb & Dumber (1994)	Comedy	1996
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996

In addition, title of movie consist of the released year of movie and the name of movie. It is better the year is separated from title. The year_released variable is placed as new variable in movielens datasets. The first six rows of movielens datasets after this modification are shown in Table 4.

```
movielens <- movielens %>%
  mutate(year_released = as.numeric(str_sub(title,-5,-2)))
kable(head(movielens),"pandoc",caption = "Movielens dataset with Separated year-released from title of monvie",
```

Table 4: Movielens dataset with Separated year-released from title of monvie

userId	movieId	rating	title	genres	year_rated	year_released
1	122	5	Boomerang (1992)	Comedy|Romance	1996	1992
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996	1995
1	231	5	Dumb & Dumber (1994)	Comedy	1996	1994
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994

In order to mimic the ultimate evaluation process, the movielens dataset typically split into two parts as train set (edx) and test set (validation) with proportion of 90% to 10%. We assume the outcome of validation is not known. The building algorithm to minimize the RMSE based on edx data and it will be validated on validation set. It has to be checked that userId and movieId in validation dataset is in edx dataset. Also the removed rows from validation set add to edx dataset.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating,
                                  times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

The dimension of edx and validation datasets is shown in Table 5. The number of rows of each datasets defined the proportion of from movielens dataset. Based on Table 5, It is confirmed that the edx is 90% of 1 million data from movielens (9000055), and validation is 10% (999999). Also the Number of column is defined the number of variable in each datasets.

```
dim_edx <- edx %>% summarise(No_data_edx = nrow(edx),
                                  No_variable_edx = ncol(edx))
dim_validation <- validation %>% summarise(No_data_validation =  nrow(validation),No_variable_validation = ncol(
kable(t(cbind2(dim_edx,dim_validation)),"pandoc",caption = "Dimention of edx and validation datasets", align = '
```

Table 5: Dimention of edx and validation datasets

No_data_edx	9000055
No_variable_edx	7
No_data_validation	999999
No_variable_validation	7

The first six rows of edx dataset is shown in Table 6.

```
kable(head(edx),"pandoc",caption = "The edx dataset", align = "c")
```

Table 6: The edx dataset

userId	movieId	rating	title	genres	year_rated	year_released
1	122	5	Boomerang (1992)	Comedy|Romance	1996	1992
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996	1995

userId	movieId	rating	title	genres	year_rated	year_released
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996	1995
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996	1994
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996	1994
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1996	1994

The outcome of datasets is rating with six features as MovieID, UserID, year_released, title, year_rated and genres.

Analysis of Edx Dataset

In this section, the effect of features as MovieId, UserID, Genres, year_released are studied on movie rating as outcome. Before we start, the number of unique users that provided ratings and how many unique movies and genres were rated in Table 7.

```
edx_unique_info <- edx %>%
  summarise(n_user_edx = n_distinct(userId),
            n_Movie_edx = n_distinct(movieId),
            n_genres_edx = n_distinct(genres))
kable(edx_unique_info,"pandoc",caption = "Number of unique usersID,movieID, genres of edx dataset", align = "c")
```

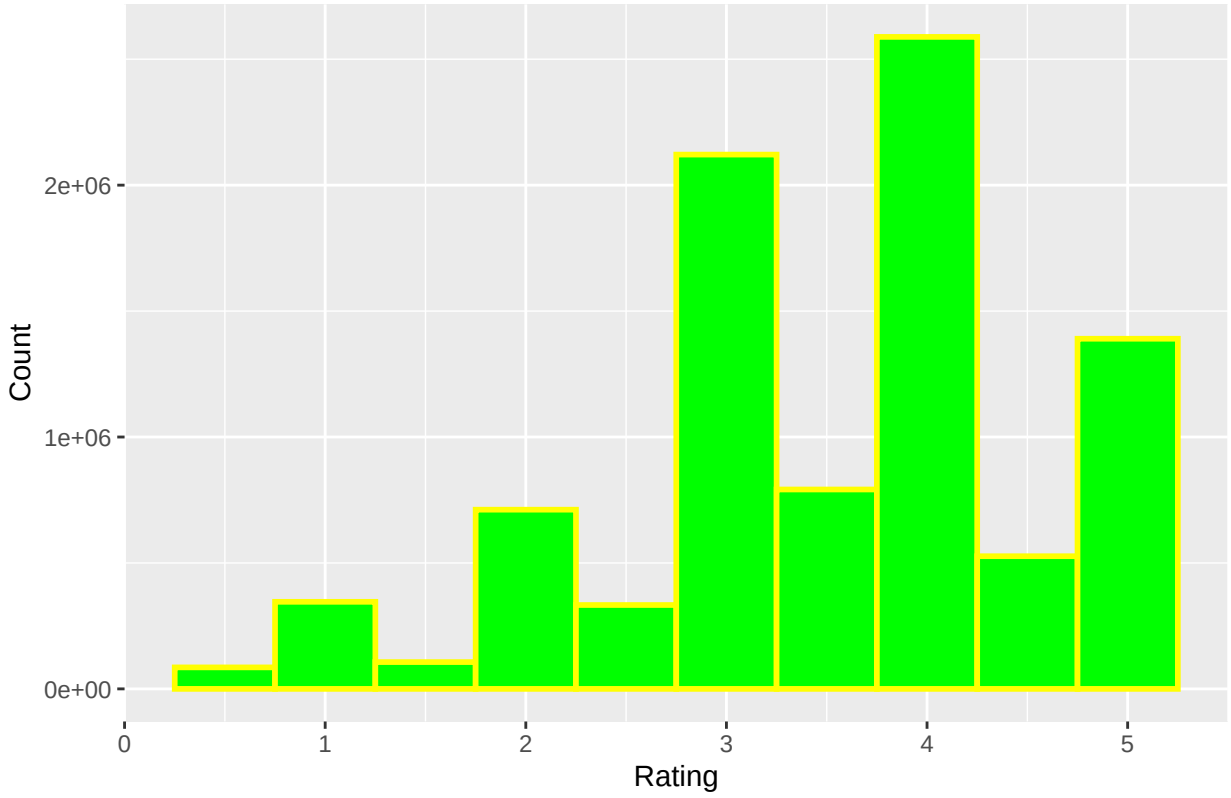
Table 7: Number of unique usersID,movieID, genres of edx dataset

n_user_edx	n_Movie_edx	n_genres_edx
69878	10677	797

The following Figure-1 shows the distribution of movie ratings. It is obvious that the the number of rating 4 is highest followed by 3 and 5. Also half-ratings is less common than whole ratings.

```
edx %>% ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.5 , fill = "green", color = "yellow", lwd = 1)+
  xlab("Rating") + ylab("Count")+
  ggtitle("Figure 1-Distibution of Movie rating") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 1−Distibution of Movie rating

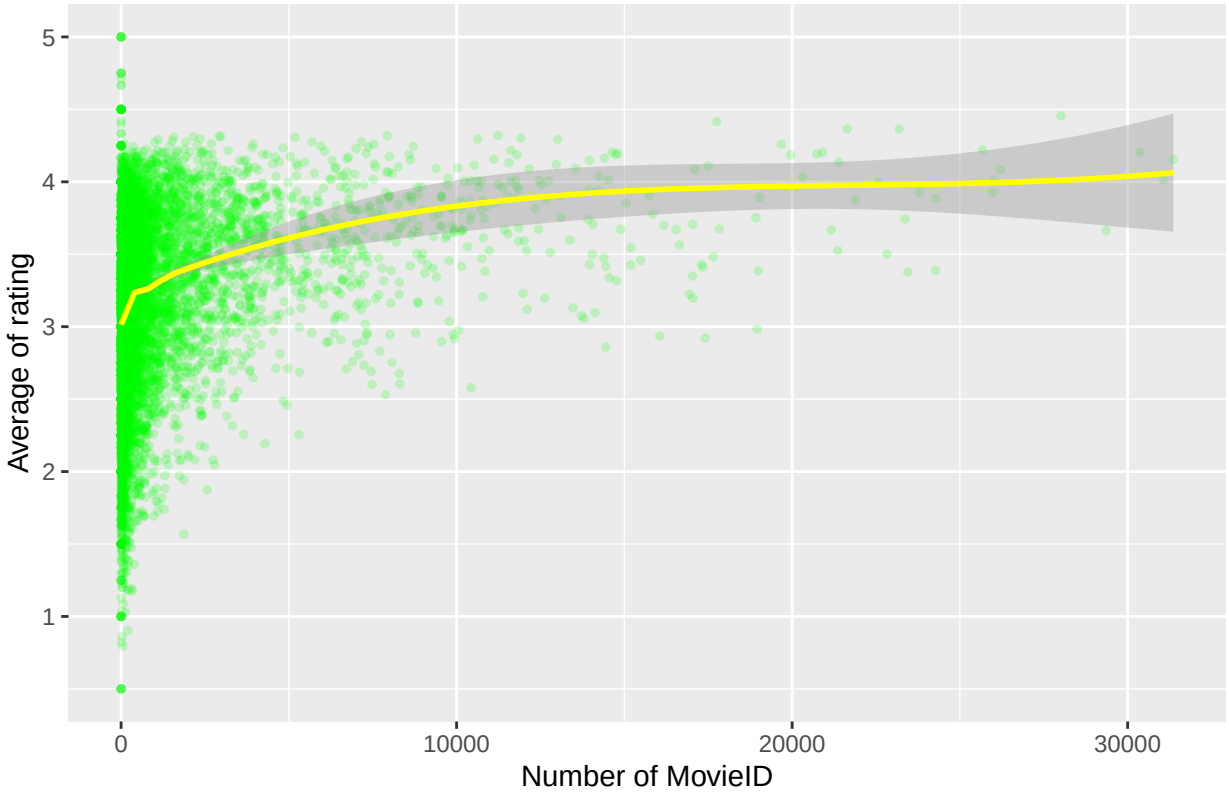


- Effect of MovieID on rating

In edx dataset, each movieId is related to specific title of movie.The Number of movieId (title) and average of rating for each of them are calculated. The Figure-2 shows the relationship between average movie ratings and frequency of ratings. The variation in movie ratings are much higher for movies that have been rated less often.

```
edx %>% group_by(title) %>%
  summarise(number_of_movie_rating  = n(), avg_rating = mean(rating)) %>%
  ggplot(aes(number_of_movie_rating,avg_rating)) +
  geom_point(alpha= 0.2, color = "green", lwd = 1) +
  ggtitle("Figure 2-Average of rating versus number of movie") +
  geom_smooth(method = "loess", color = "yellow") +
  xlab ("Number of MovieID") +
  ylab("Average of rating") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 2−Average of rating versus number of movie



The six movies that has the highest rating are shown in Table 8.

```
Highest_rate_movies <- edx %>% group_by(title) %>% summarise(Avg_rating = mean(rating)) %>% arrange(desc(Avg_rat
kable(head(Highest_rate_movies), "pandoc", caption = "The highest rating movie", align = "c")
```

Table 8: The highest rating movie

title	Avg_rating
Blue Light, The (Das Blaue Licht) (1932)	5
Fighting Elegy (Kenka erejii) (1966)	5
Hellhounds on My Trail (1999)	5
Satan’s Tango (SÃ¡tÃ¡ntangÃ³) (1994)	5
Shadows of Forgotten Ancestors (1964)	5
Sun Alley (Sonnenallee) (1999)	5

The six movies that has the lowest rating are shown in Table 9.

```
lowest_rate_movies <- edx %>% group_by(title) %>% summarise(Avg_rating = mean(rating)) %>% arrange((Avg_rating))
kable(head(lowest_rate_movies), "pandoc", caption = "The lowest rating movie", align = "c")
```

Table 9: The lowest rating movie

title	Avg_rating
Accused (Anklaget) (2005)	0.5000000
Besotted (2001)	0.5000000
Confessions of a Superhero (2007)	0.5000000
Hi-Line, The (1999)	0.5000000
War of the Worlds 2: The Next Wave (2008)	0.5000000
SuperBabies: Baby Geniuses 2 (2004)	0.7946429

- Effect of number of UserId on movie rating

The average movie rating grouped by userId is calculated as follows in Table 10. User is rated movie between 10 to 6616 times, for this reason the number of times of rating more than 100 considered.

```
user_avg <- edx %>% group_by(userId) %>%
  summarise(number_of_user_rate = n(), avg_user_rating = mean(rating)) %>%
  filter(number_of_user_rate > 100)
kable(head(user_avg),"pandoc", caption = "Number of UserID and average of rating ", align = "c")
```

Table 10: Number of UserID and average of rating

userId	number_of_user_rate	avg_user_rating
8	727	3.386520
10	112	3.830357
13	126	3.301587
18	362	3.454420
19	216	3.736111
30	141	4.503546

The minimum and maximum movie rate based on userID is shown in Table 11. The results show that the minimum rate (1) from userID 24176 and maximum rate (4.934) from userID 5763.

```
min_user_rated <- user_avg %>%
  summarise(Min_rate = min(avg_user_rating),
            userID_min = userId[which.min(avg_user_rating)])
max_user_rated <- user_avg %>%
  summarise(Max_rate = max(avg_user_rating),
            userId_max = userId[which.max(avg_user_rating)])
kable(t(cbind(min_user_rated,max_user_rated)),"pandoc", caption = "Maximum and Minimum rating and UserID", align
```

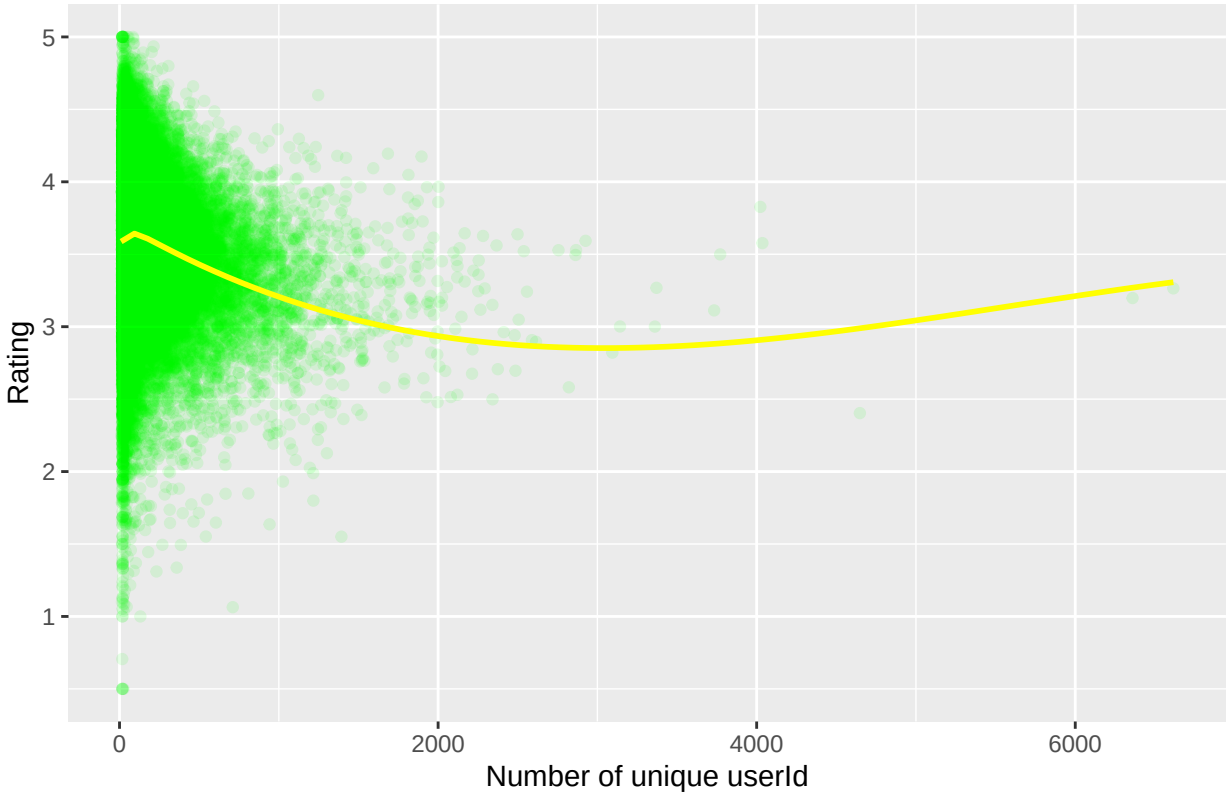
Table 11: Maximum and Minimum rating and UserID

Min_rate	1.000000
userID_min	24176.000000
Max_rate	4.934579
userId_max	5763.000000

In Figure 3, the result shows the relationship between number of user that is rated versus frequency of rating. The variation in movie ratings are much higher for user that watch less movies.

```
edx %>% group_by(userId) %>%
  mutate(rating = mean(rating)) %>% count(userId,rating) %>%
  ggplot(aes(n,rating)) +
  geom_point(color = "green", alpha = 0.1) +
  ggtitle("Figure 3- Effect of Number UserId on rating") +
  geom_smooth(method = "loess", color = "yellow",se = FALSE)+
  xlab("Number of unique userId") +
  ylab ("Rating")+
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 3− Effect of Number UserId on rating



- Effect of Type of genres on Movie Rating

In order to investigate the effect of individual genres on movie ratings, the genre column is separated into single genres as follows in Table 11.

```
single_genres <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(number_of_genres = n(), Avg_rating = mean(rating))
kable(single_genres,"pandoc",
      caption = "Number of each genres based on Number of genres", align = "c")
```

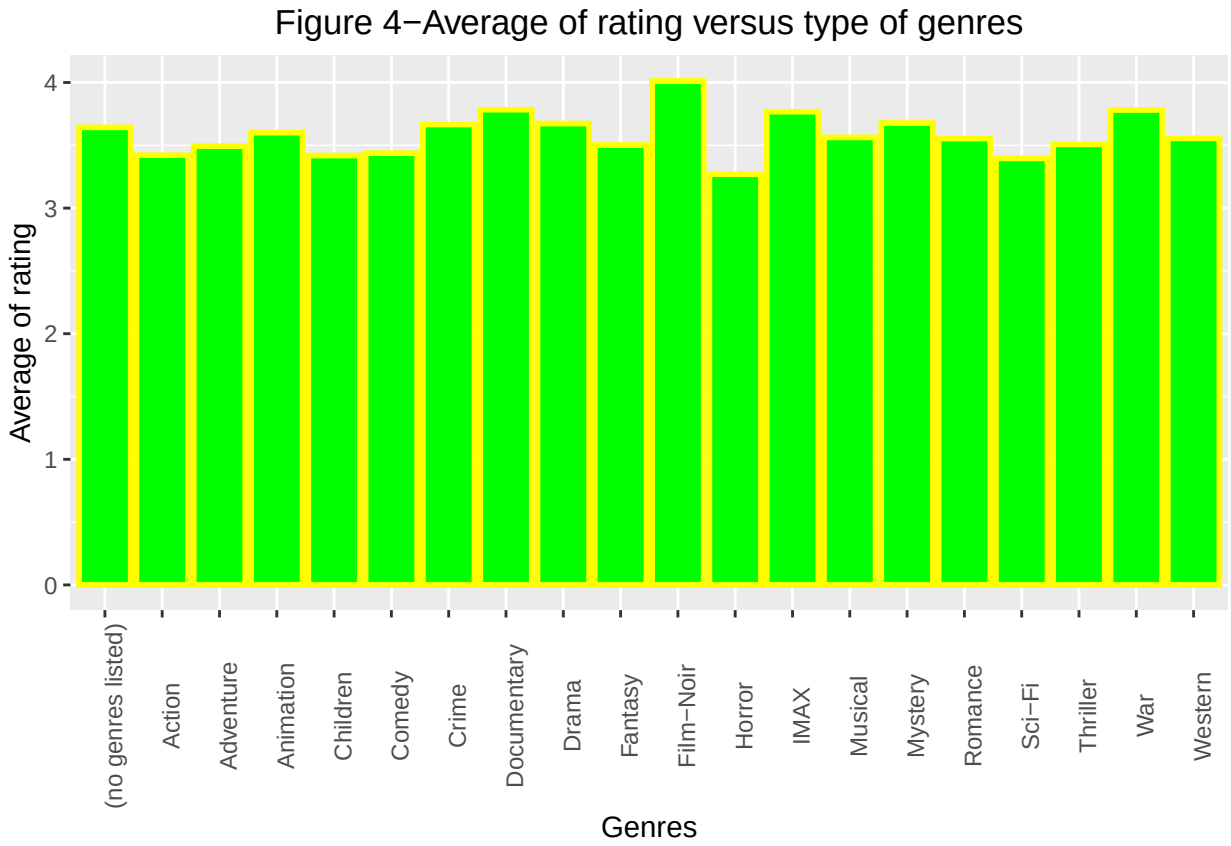
Table 12: Number of each genres based on Number of genres

genres	number_of_genres	Avg_rating
(no genres listed)	7	3.642857
Action	2560545	3.421405
Adventure	1908892	3.493544
Animation	467168	3.600644
Children	737994	3.418715
Comedy	3540930	3.436908
Crime	1327715	3.665925
Documentary	93066	3.783487
Drama	3910127	3.673131
Fantasy	925637	3.501946
Film-Noir	118541	4.011625
Horror	691485	3.269815
IMAX	8181	3.767693
Musical	433080	3.563305
Mystery	568332	3.677001
Romance	1712100	3.553813

genres	number_of_genres	Avg_rating
Sci-Fi	1341183	3.395743
Thriller	2325899	3.507676
War	511147	3.780813
Western	189394	3.555918

Distribution of ratings per genre is shown in Figure 4. It is shown the minimum rate related to Horror film and maximum rate related to Film Noir.

```
single_genres %>%
  ggplot(aes(genres,Avg_rating)) +
  geom_col(color = "yellow", fill = "green" , size = 1) +
  ggtitle("Figure 4-Average of rating versus type of genres") +
  xlab("Genres") +
  ylab("Average of rating") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90 , hjust = 0.5))
```

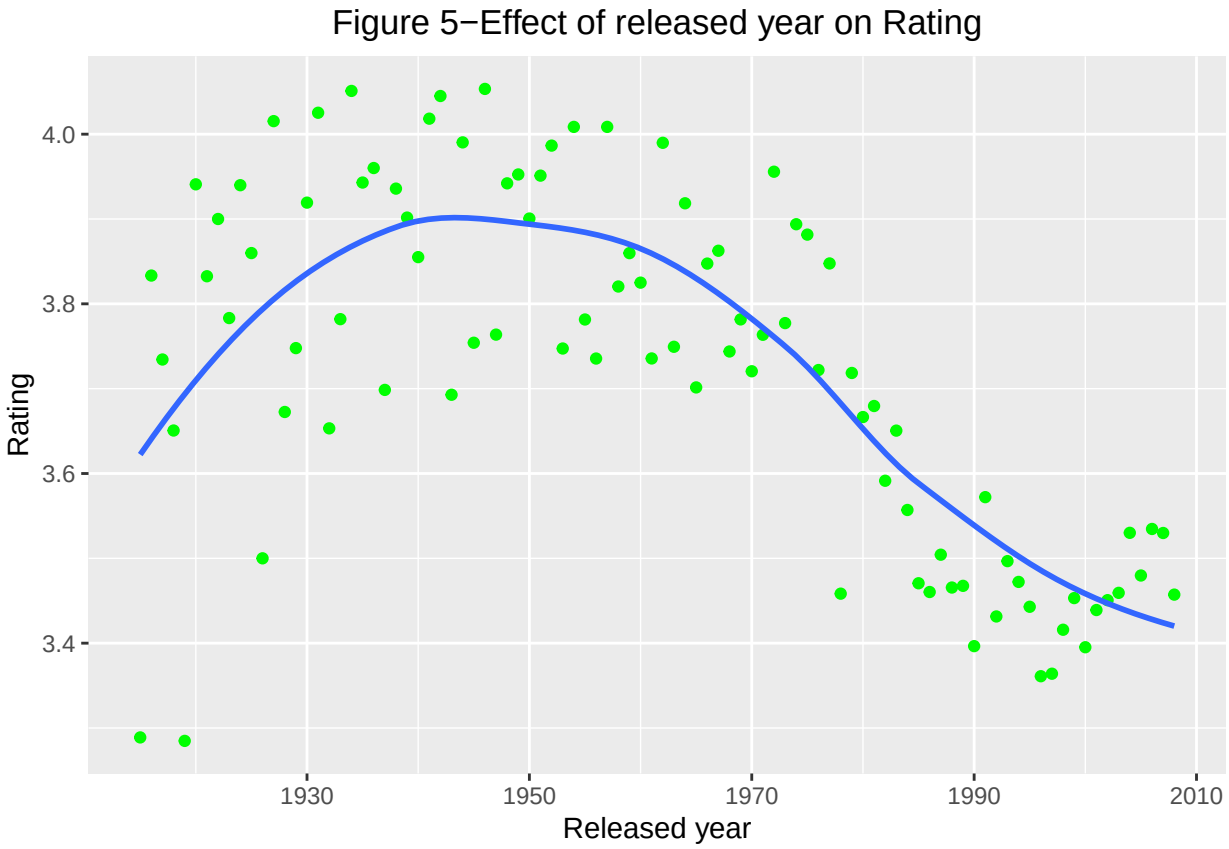


- Effect of Year-Released on Movie Rating

The year_released of movie is also affected on movie rating. As shown in Figure 5 the average movie rate between 1930 to 1980 is higher than recently.

```
seperate_year_released <- edx %>% group_by(year_released) %>%
  summarise(year_released_rating = mean(rating)) %>% arrange(desc(year_released_rating))

seperate_year_released %>%
  ggplot(aes(year_released,year_released_rating)) +
  geom_point(color = "green", alpha = 1) +
  geom_smooth(method = "loess", se = FALSE) +
```

```
  ggtitle("Figure 5-Effect of released year on Rating") +
  xlab("Released year") +
  ylab("Rating") +
  theme(plot.title = element_text(hjust = 0.5))
```



Results

As previously explained, in order to have recommended system for movilens dataset, the dataset is slice to edx and validation. All the models applies on edx dataset and the best model is validated with validation dataset.

Also, the edx dataset is slice to train and test, to optimum the best model.

```
y <- edx$rating
index <- createDataPartition(y, times = 1 , p = 0.2 , list = FALSE)
train <- edx[-index,]
test <- edx[index,]
test <- test %>%
  semi_join(train , by = "movieId") %>%
  semi_join(train , by = "userId")
memory.limit(200000)
```

```
## [1] 8e+05
```

The goal of this project is predicted the best model to have the Root mean square error (RMSE) less than 0.8649.

BASE MODEL

The basic model predicts the same rating for all movies by all users (i.e., calculating mean rating for entire dataset). That is our base model and is represented by the following formula:

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

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

In this model all differences in movie ratings are explained by random variation alone, and is calculated by averaging all movie ratings in the entire dataset. The result shows in Table 12, the RMSE is 1.06237, it is too far from our goal (RMSE = 0.8649).

```
mu <- mean(train$rating)
rmse_naive <- RMSE(train$rating , mu)
RMSE_Result <- tibble(method = "Base Method" ,
                      RMSE_on_Train_set = rmse_naive,
                      RMSE_on_Validation_set = NA)
kable((RMSE_Result[1:1,]), "pandoc", caption = "RMSE Results",
      align = "c")
```

Table 13: RMSE Results

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.060237	NA

##- Effect of Features on RMSE

In order to improve, it is considered the effect of features as MovieID, UserID, genres and released year on rating to calculate the RMES.

- Effect of MovieID on RMSE

Because popular movies usually rate higher than non-popular movies, it's not correct to average ratings for all movies altogether, rather, movie rating bias b_i should be taken into account. This bias is calculated as follows:

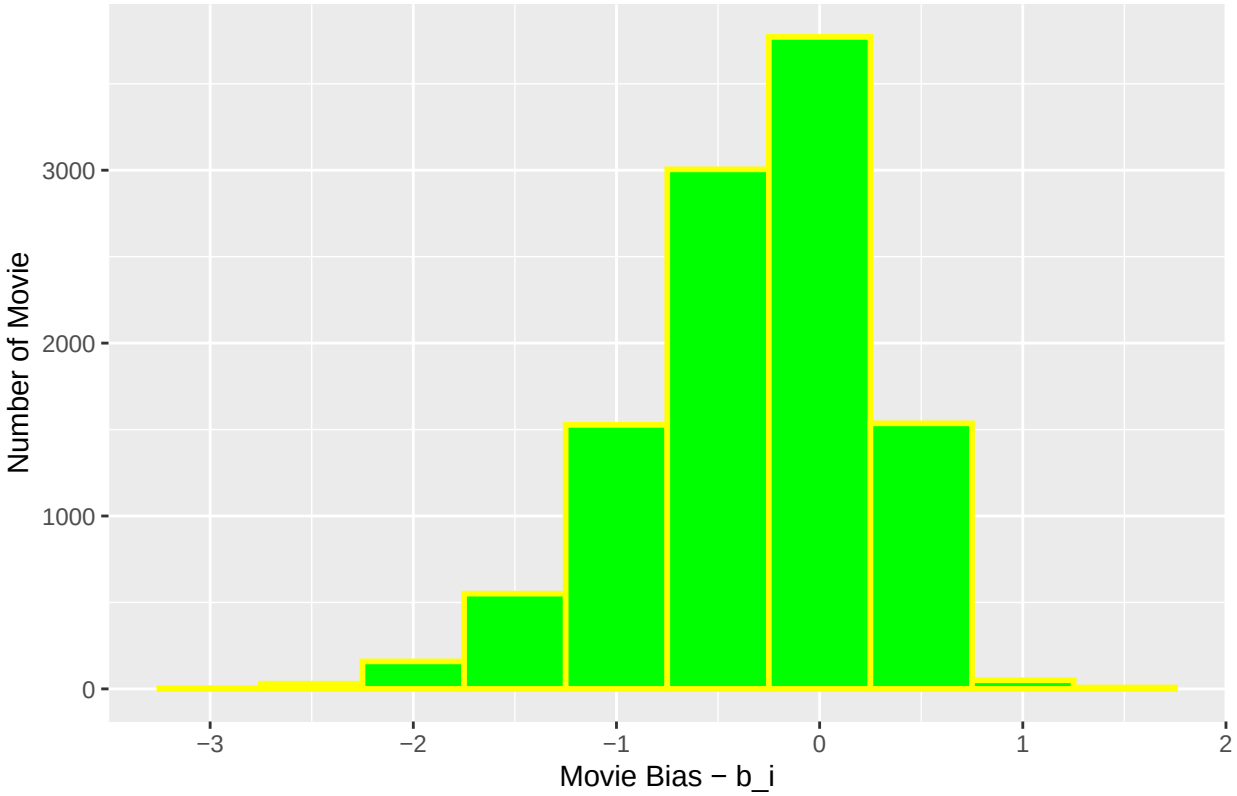
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

```
movie_avg <- train %>% group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))
```

The distribution of movie bias is plotted in Figure 6.

```
movie_avg %>% ggplot(aes(b_i)) +
  geom_histogram(color = "yellow" , fill = "green", binwidth = 0.5, lwd = 1) +
  ggtitle("Figure 6 - Ditribution of Movie Bias") +
  xlab("Movie Bias - b_i") +
  ylab("Number of Movie") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 6 − Ditribution of Movie Bias



Now it is better to calculate that how much prediction improves. The effect of movie bias is validated with test set (the part of edx dataset).

```
predicted_rating <- mu + test %>%
  left_join(movie_avg , by = "movieId") %>%
  .$b_i

Effect_of_Movie <- RMSE(test$rating,predicted_rating)
```

The RMSE is calculated around 0.94371 (Table 13). Although the value of RMSE improved but it is far from a target.

```
RMSE_Result <- add_row(RMSE_Result,method = "Movie Effect",
                       RMSE_on_Train_set = Effect_of_Movie,
                       RMSE_on_Validation_set = NA)
kable((RMSE_Result[1:2,]), "pandoc",caption = 'RMSE Result', align = "c")
```

Table 14: RMSE Result

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.0602372	NA
Movie Effect	0.9437144	NA

- Effect of UserId on RMSE

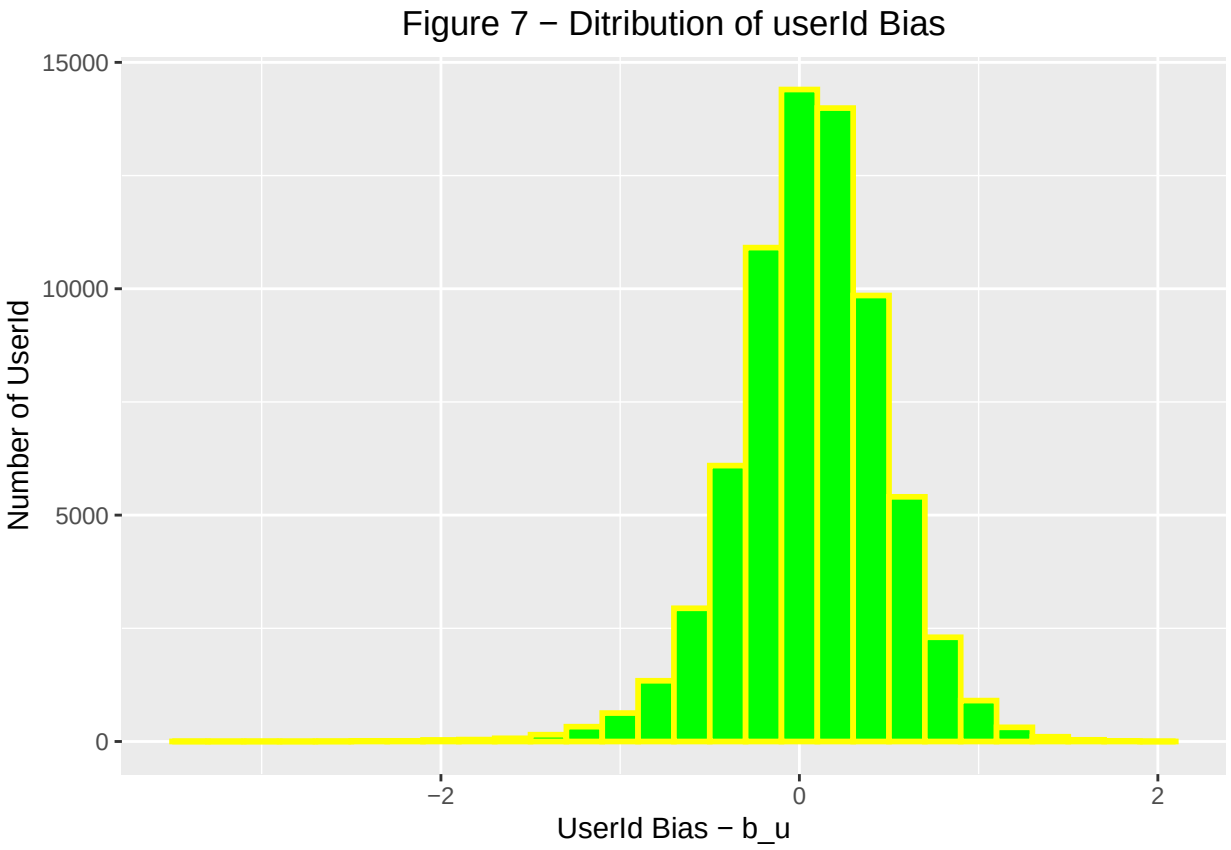
Some users give higher rating to movies in general than others, which will also creates a bias. This bias needs to be added into account. This shows that further improvement to model:

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

b_u is bias of user. Distribution of user bias is shown in Figure 7.

```
userid_avg <- train %>%
  left_join(movie_avg , by = "movieId")  %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))
```

```
userid_avg %>% ggplot(aes(b_u)) +
  geom_histogram(color = "yellow" , fill = "green", binwidth = 0.2, lwd =1) +
  ggtitle("Figure 7 - Ditribution of userId Bias") +
  xlab("UserId Bias - b_u") +
  ylab("Number of UserId") +
  theme(plot.title = element_text(hjust = 0.5))
```



The calculated value of RMSE is 0.86616 (Table 14). It seems by adding the effect of userId to movieId, the value of RMSE improved.

```
predicted_rating <- test %>%
  left_join(movie_avg , by = "movieId") %>%
  left_join(userid_avg , by = "userId")  %>%
  mutate(pred = mu + b_i + b_u) %>% pull(pred)

Effect_of_Movie_User <- RMSE(test$rating,predicted_rating)
```

```
RMSE_Result <- add_row(RMSE_Result,method = "Movie_User Effect",
                       RMSE_on_Train_set = Effect_of_Movie_User,
                       RMSE_on_Validation_set = NA)
kable((RMSE_Result[1:3,]), "pandoc",caption = 'RMSE Result', align = "c")
```

Table 15: RMSE Result

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.0602372	NA
Movie Effect	0.9437144	NA
Movie_User Effect	0.8661625	NA

- Effect of Genres on RMSE

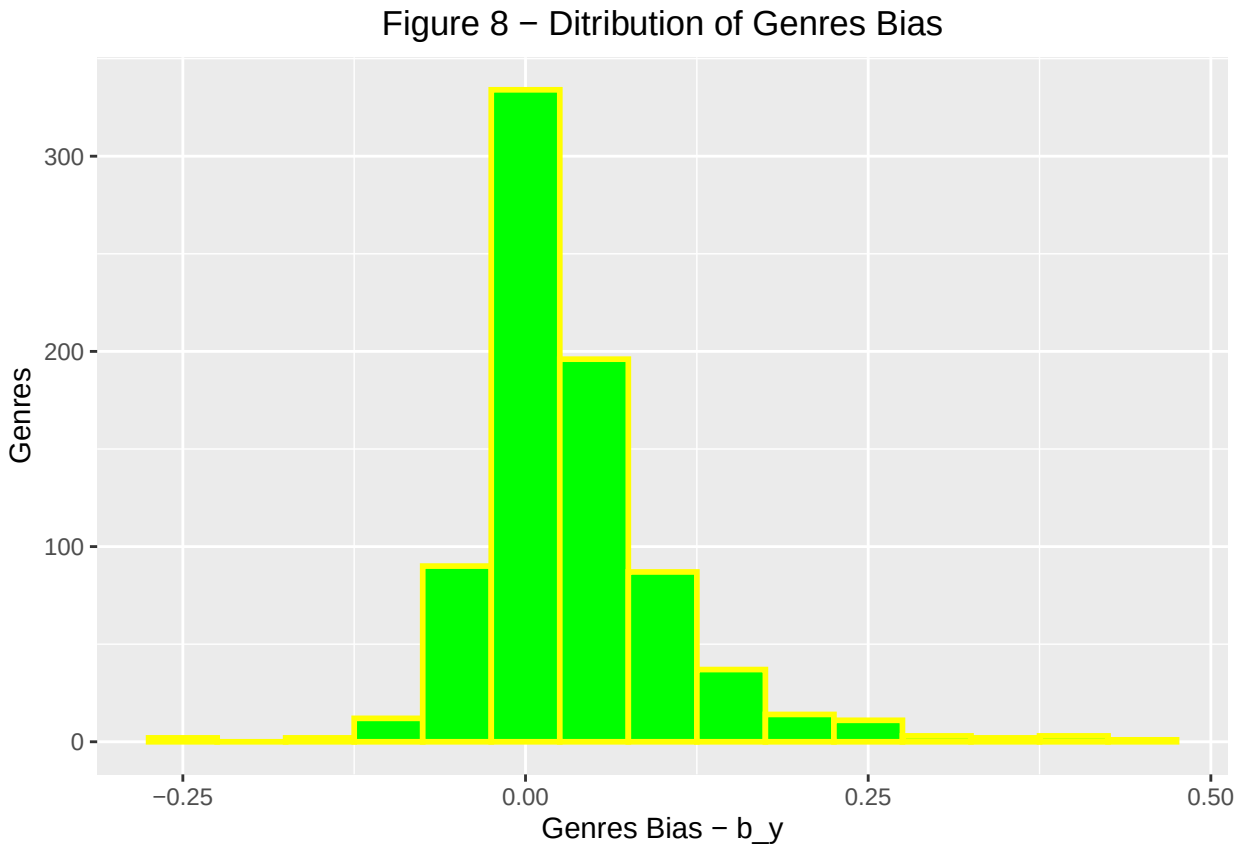
The next feature is genres to be added the model.

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

```
genres_avg <- train %>%
  left_join(movie_avg , by = "movieId")  %>%
  left_join(userid_avg , by = "userId")  %>%
  group_by(genres) %>%
  summarise(b_g = mean(rating - mu - b_i - b_u))
```

Distribution of genres bias b_g is shown in Figure 8.

```
genres_avg %>% ggplot(aes(b_g)) +
  geom_histogram(color = "yellow" , fill = "green", binwidth = 0.05, lwd = 1) +
  ggtitle("Figure 8 - Ditribution of Genres Bias") +
  xlab("Genres Bias - b_y") +
  ylab("Genres") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
predicted_rating <- test %>%
  left_join(movie_avg , by = "movieId") %>%
  left_join(userid_avg , by = "userId")  %>%
  left_join(genres_avg, by = "genres")  %>%
  mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

Effect_of_Movie_User_genres <- RMSE(test$rating,predicted_rating)
```

The value of RMSE is calculate around 0.86582 (Table 15). It shows the adding the genres can lead to improve the RMSE.

```
RMSE_Result <- add_row(RMSE_Result,method = "Movie_User_genres Effect",
                       RMSE_on_Train_set = Effect_of_Movie_User_genres,
                       RMSE_on_Validation_set = NA)
kable((RMSE_Result[1:4,]), "pandoc",caption = 'RMSE Result', align = "c")
```

Table 16: RMSE Result

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.0602372	NA
Movie Effect	0.9437144	NA
Movie_User Effect	0.8661625	NA
Movie_User_genres Effect	0.8658242	NA

- Effect of Year_Released on RMSE

The last features that added to our model is year_released.

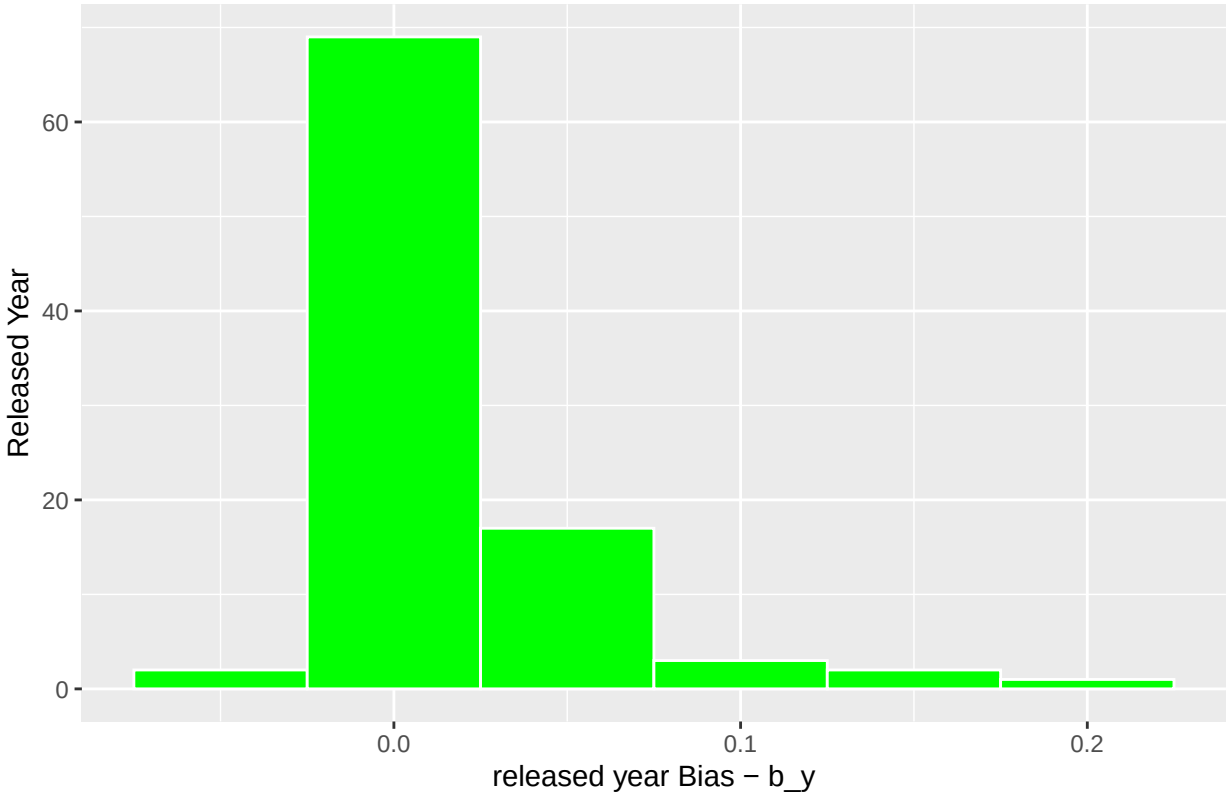
$$Y_{u,i} = \mu + b_i + b_u + b_g + b_y + \epsilon_{u,i}$$

```
released_year_avg <- train %>%
  left_join(movie_avg , by = "movieId")  %>%
  left_join(userid_avg , by = "userId")  %>%
  left_join(genres_avg, by = "genres")  %>%
  group_by(year_released) %>%
  summarise(b_y = mean(rating - mu - b_i - b_u - b_g))
```

Distribution of year_released bias is shown in Figure 9.

```
released_year_avg %>% ggplot(aes(b_y)) +
  geom_histogram(color = "white" , fill = "green", binwidth = 0.05) +
  ggtitle("Figure 9 - Ditribution of released year Bias") +
  xlab("released year Bias - b_y") +
  ylab("Released Year") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 9 − Ditribution of released year Bias



The value of RMSE is calculated around 0.86564 with effect of movieId, UserId, year_released and genres, as shown in Table 16.

```
predicted_rating <- test %>%
  left_join(movie_avg , by = "movieId") %>%
  left_join(userid_avg , by = "userId")  %>%
  left_join(genres_avg, by = "genres")  %>%
  left_join(released_year_avg, by = "year_released")  %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>% pull(pred)

Effect_of_Movie_User_genres_releasedYear <- RMSE(test$rating,predicted_rating)
```

```
RMSE_Result <- add_row(RMSE_Result,method = "Movie_User_genres_releasedYear Effect",
                       RMSE_on_Train_set = Effect_of_Movie_User_genres_releasedYear,
                       RMSE_on_Validation_set = NA)
kable((RMSE_Result[1:5,]), "pandoc",caption = 'RMSE Result', align = "c")
```

Table 17: RMSE Result

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.0602372	NA
Movie Effect	0.9437144	NA
Movie_User Effect	0.8661625	NA
Movie_User_genres Effect	0.8658242	NA
Movie_User_genres_releasedYear Effect	0.8656468	NA

Regularized Method

In order to improve our predictions, Regularization method used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting. The general idea behind regularization is to constrain the total variability of the effect sizes.

Tuning parameters are added to calculation of bias as follows:

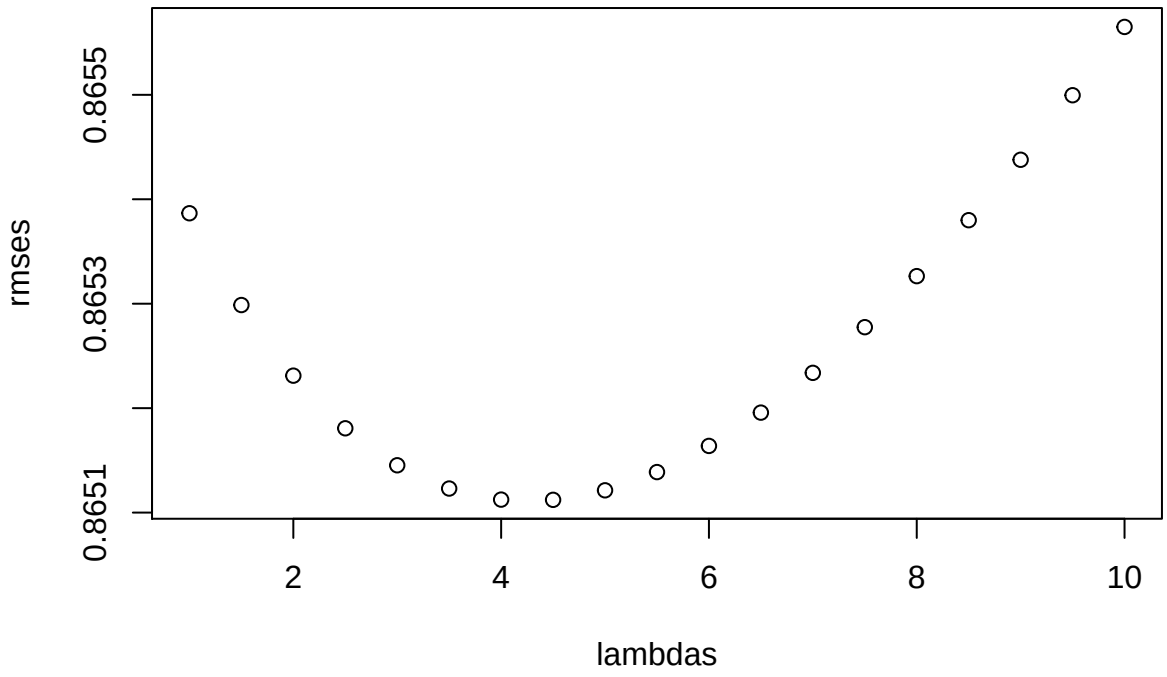
$$b_{i,u,g,y}(\lambda) = \frac{1}{\lambda + n_{i,u}} \sum_{i,u=1}^{n} (Y_{i,u} - \mu)$$

This shrinks the b_i,b_u,b_g and b_y in case of small number of ratings.

In order to determine the tuning parameter, the plot of RMSE versus lambdas is constructed as follows (Figure 10):

```
lambdas <- seq(1,10,0.5)
 rmses <- sapply(lambdas, function(l){
   mu<- mean(train$rating)

   b_i <- train %>% group_by(movieId) %>%
     summarise(b_i = sum(rating-mu)/(n() +l))

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

   b_g <- train %>%
     left_join(movie_avg , by = "movieId")  %>%
     left_join(userid_avg , by = "userId")  %>%
     group_by(genres) %>%
     summarise(b_g = sum(rating - mu - b_i - b_u)/(n()+l))

   b_y <- train %>%
     left_join(movie_avg , by = "movieId")  %>%
     left_join(userid_avg , by = "userId")  %>%
     left_join(genres_avg, by = "genres")  %>%
     group_by(year_released) %>%
     summarise(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+l))

   predicted_ratings <- test %>%
     left_join(b_i , by = "movieId") %>%
     left_join(b_u , by = "userId") %>%
     left_join(b_g , by = "genres") %>%
     left_join(b_y , by = "year_released") %>%
     mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
     pull(pred)

   return(RMSE(predicted_ratings, test$rating))

 })

 plot(lambdas, rmses , main = "Figure 10- Plot of RMSE versus lambda")
```

Figure 10− Plot of RMSE versus lambda




```
optimum_lambdas <- lambdas[which.min(rmses)]

Regularized_Movie_User_genres_releasedYear = min(rmses)
```

The optimum lambdas to have minimum RMSE is 4.5. The minimum RMSE regarding to this value of lambdas is 0.86511

Check the validation set

The value of optimum lambdas (4.5) is used to apply the Regularization method to validate the validation dataset.

```
  lambda = lambdas[which.min(rmses)]
  mu<- mean(validation$rating)

  b_i <- validation %>% group_by(movieId) %>%
    summarise(b_i = sum(rating-mu)/(n() + lambda))

  b_u <- validation %>% left_join(b_i , by = "movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - mu - b_i)/(n()+lambda))

  b_g <- validation %>%
    left_join(b_i , by = "movieId")  %>%
    left_join(b_u , by = "userId")  %>%
    group_by(genres) %>%
    summarise(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda))

  b_y <- validation %>%
   left_join(b_i , by = "movieId")  %>%
    left_join(b_u , by = "userId")  %>%
    left_join(b_g, by = "genres")  %>%
    group_by(year_released) %>%
```

```
  summarise(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+lambda))

predicted_ratings <- validation %>%
  left_join(b_i , by = "movieId") %>%
  left_join(b_u , by = "userId") %>%
  left_join(b_g , by = "genres") %>%
  left_join(b_y , by = "year_released") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

Regularized_Movie_User_genres_releasedYear_validate <- RMSE(predicted_ratings, validation$rating)
```

The results of validation dataset based on Regularization method is shwon in Table 17. The value of RMSE for validation dataset is 0.83884 that is meet our target < 0.8649

```
RMSE_Result <- add_row(RMSE_Result,
                       method =  "Regularized_Movie_User_genres_releasedYear Effect",
                       RMSE_on_Train_set = Regularized_Movie_User_genres_releasedYear,
                       RMSE_on_Validation_set = Regularized_Movie_User_genres_releasedYear_validate)
kable((RMSE_Result[1:6,]), "pandoc",caption = 'RMSE Result', align = "c")
```

Table 18: RMSE Result

method	RMSE_on_Train_set	RMSE_on_Validation_set
Base Method	1.0602372	NA
Movie Effect	0.9437144	NA
Movie_User Effect	0.8661625	NA
Movie_User_genres Effect	0.8658242	NA
Movie_User_genres_releasedYear Effect	0.8656468	NA
Regularized_Movie_User_genres_releasedYear Effect	0.8651122	0.8388471

Conclusion

In this assignment a machine learning algorithm was successfully build in order to predict movie ratings with a subset of MovieLens dataset. It is determined that the regularized model for combined movie, user, genres and released year effect is sufficient to reduce RSME to 0.83884. Therefore the final model for this project is:

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