MovieLens Project Report

Andrii Platonov

25 05 2020

INTRODUCTION

The goal of the project is to create a movie recommendation system using the MovieLens data set. The project is based on the 10M version of the MovieLens data set. The link provided in the course was used to download the MovieLens data and generate data sets for further analysis.

For movie recommendation system, a machine learning algorithm will be developed to generate predicted movie ratings and calculate Root Mean Square Error (RMSE). The **edx** set will be used to train a machine learning algorithm to predict movie ratings. For this purpose an additional partition of training and test sets from the provided **edx** data set will be created to experiment with multiple parameters.

For a final test of algorithm, movie ratings in the **validation** set should be predicted as if they were unknown. RMSE will be used to evaluate how close our predictions are to the true values in the **validation** set.

The project report includes the following sections: - Introduction; - Data analysis; - Identification of optimal methods and tools; - Results; - Conclusion.

DATA ANALYSIS

This section presents an analysis of the data structure in order to identify insights that will be used to develop the machine learning algorithm to predict movie ratings.

The MovieLens data set was obtained from the course page. This link was used to download edx set and validation set and make a local copy to the computer.

`ReadRDS function` was used to generate edx set from a local copy of the MovieLens data set.

```
edx = readRDS("/Users/Admin/Documents/edx.rds")
head(edx)
```

```
##   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
```

Validation set was generated from a local copy of the MovieLens data set with the help of `ReadRDS function` as well.

```
validation = readRDS("/Users/Admin/Documents/validation.rds")
head(validation)
```

```
##   userId movieId rating timestamp
## 1      1     231      5 838983392
## 2      1     480      5 838983653
## 3      1     586      5 838984068
## 4      2     151      3 868246450
## 5      2     858      2 868245645
## 6      2    1544      3 868245920
##                                                    title
## 1                                    Dumb & Dumber (1994)
## 2                                    Jurassic Park (1993)
## 3                                       Home Alone (1990)
## 4                                          Rob Roy (1995)
## 5                                    Godfather, The (1972)
## 6 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                  genres
## 1                                 Comedy
## 2        Action|Adventure|Sci-Fi|Thriller
## 3                         Children|Comedy
## 4                 Action|Drama|Romance|War
## 5                           Crime|Drama
## 6 Action|Adventure|Horror|Sci-Fi|Thriller
```

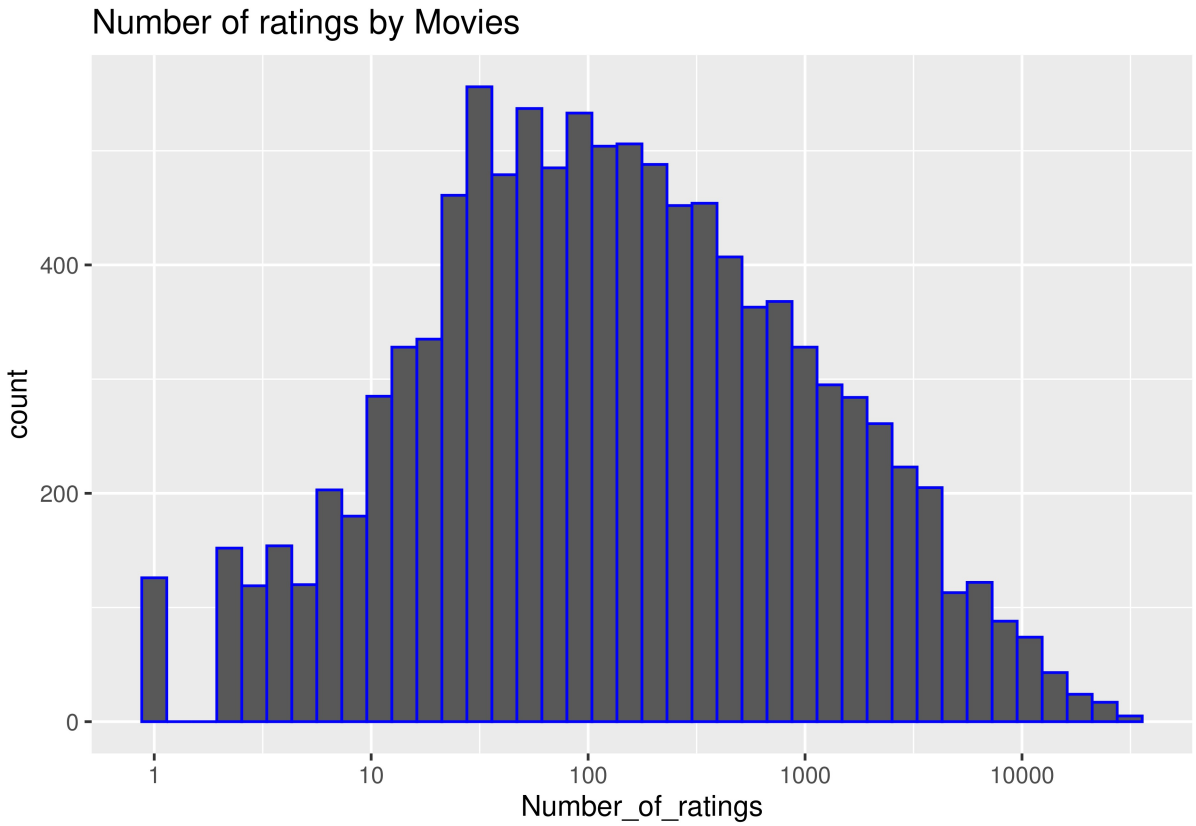
For further analysis some extra R packages will be used:

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(ggplot2)
library(dplyr)
library(colorspace)
```

Movie ratings

First of all, a distribution of ratings among the movies will be analyzed.

```
edx %>% count(movieId) %>% mutate(Number_of_ratings = n) %>%
 ggplot(aes(Number_of_ratings)) +
  geom_histogram( bins=40, color = "blue") +
  scale_x_log10() +
  ggtitle("Number of ratings by Movies")
```



Top-10 movies with the highest ratings presented below:

```
edx %>% group_by(title) %>%
  summarize(mean_rating =  round(mean(rating), digits = 2), Number_of_ratings = n()) %>%
  top_n(10) %>%
  arrange(desc(Number_of_ratings))
```

```
## Selecting by Number_of_ratings
```

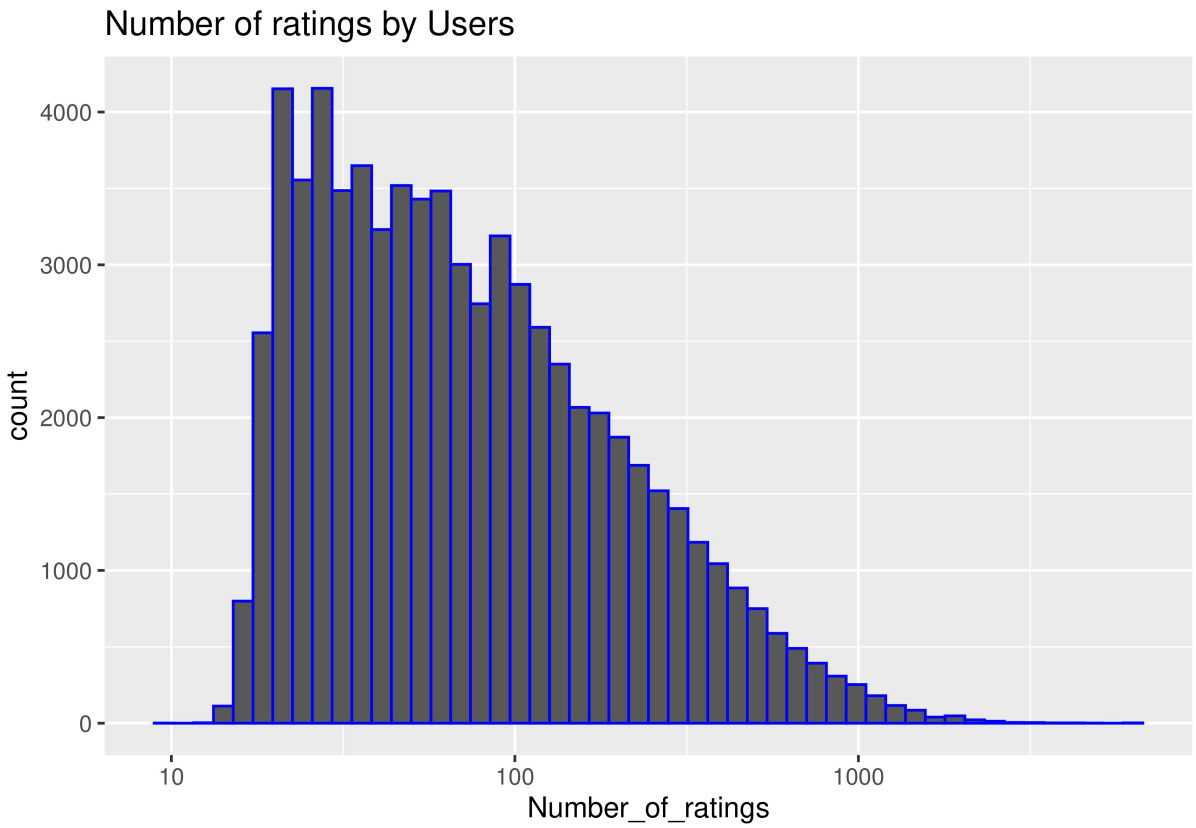
```
## # A tibble: 10 x 3
##    title                                                mean_rating Number_of_ratin~
##    <chr>                                                      <dbl>            <int>
##  1 Pulp Fiction (1994)                                         4.15            31362
##  2 Forrest Gump (1994)                                         4.01            31079
##  3 Silence of the Lambs, The (1991)                            4.2             30382
##  4 Jurassic Park (1993)                                        3.66            29360
##  5 Shawshank Redemption, The (1994)                            4.46            28015
##  6 Braveheart (1995)                                           4.08            26212
##  7 Fugitive, The (1993)                                        4.01            25998
##  8 Terminator 2: Judgment Day (1991)                           3.93            25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star~            4.22            25672
## 10 Apollo 13 (1995)                                            3.89            24284
```

Key findings: The analysis of movie's distribution reveals that each of movies has completely different number of ratings. However, higher number of ratings for the movie does not mean higher average rating of movie. Apart from this, some movies have a way more number of ratings than the other. This effect could be quite significant and therefore can be used in the algorithm to predict movie ratings.

User preferences

User preferences are fundamental for determining the rating of a movie and directly affect the rating of each movie, which may impact on the recommendation system.

```
edx %>% count(userId) %>% mutate(Number_of_ratings = n) %>%
  ggplot(aes(Number_of_ratings)) +
  geom_histogram( bins=50, color = "blue") +
  scale_x_log10() +
  ggtitle("Number of ratings by Users")
```

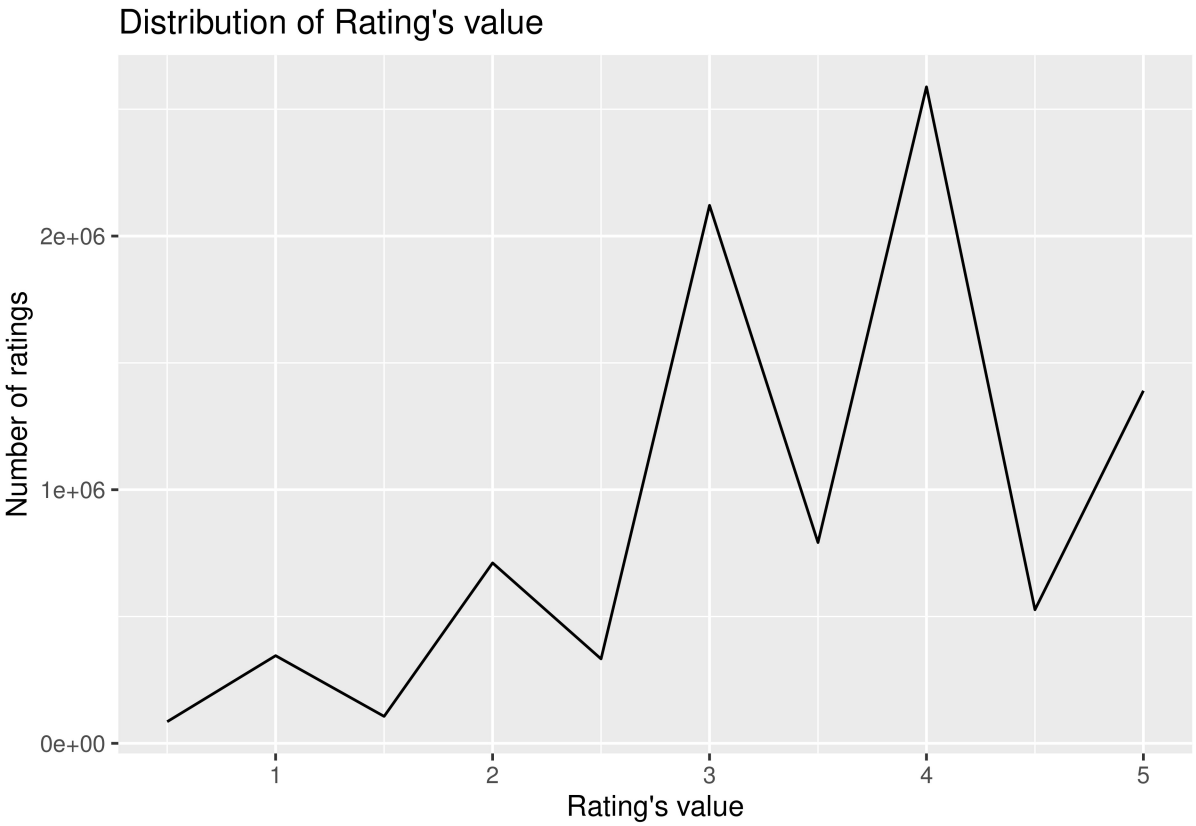


Key findings: User's distribution supports the idea that the activity of users is not the same. Moreover, some of users are extremely active in terms of providing ratings. This effect is definitely strong and should be taken into account when developing the predict algorithm.

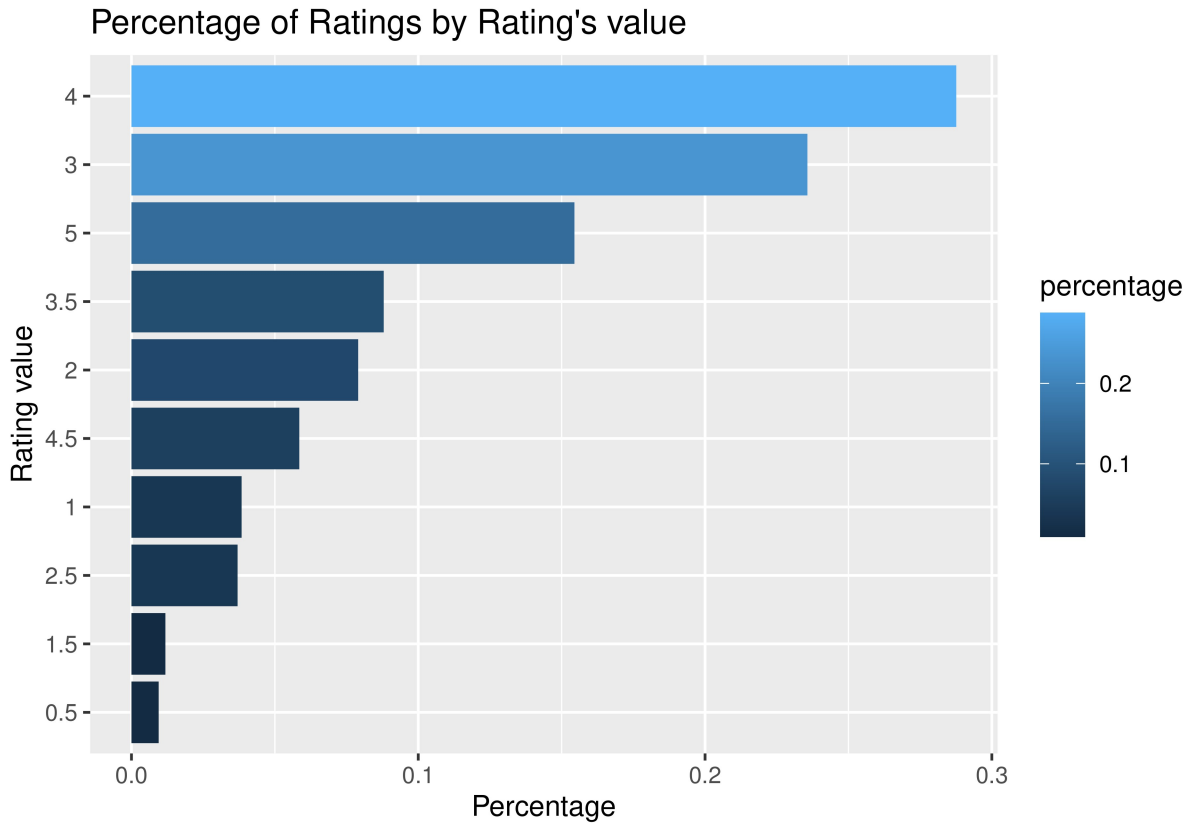
Rating's value

Each user may select any rating value (from 0 to 5) for any movie. However, some of them are more popular among users.

```
edx %>%
  group_by(rating) %>%
  summarize(Number_of_ratings = n()) %>%
  ggplot(aes(x = rating, y = Number_of_ratings)) +
  geom_line() +
  labs(y = "Number of ratings", x = "Rating's value") +
  ggtitle("Distribution of Rating's value")
```




```
#Percentage of Ratings by Rating's value
edx %>%
  group_by(rating) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  mutate(sumrates = sum(n), percentage = n/sumrates) %>%
  arrange(-percentage) %>%
  ggplot(aes(reorder(rating, percentage), percentage, fill= percentage)) +
  geom_bar(stat = "identity") + coord_flip() +
  labs(y = "Percentage", x = "Rating value") +
  ggtitle("Percentage of Ratings by Rating's value")
```

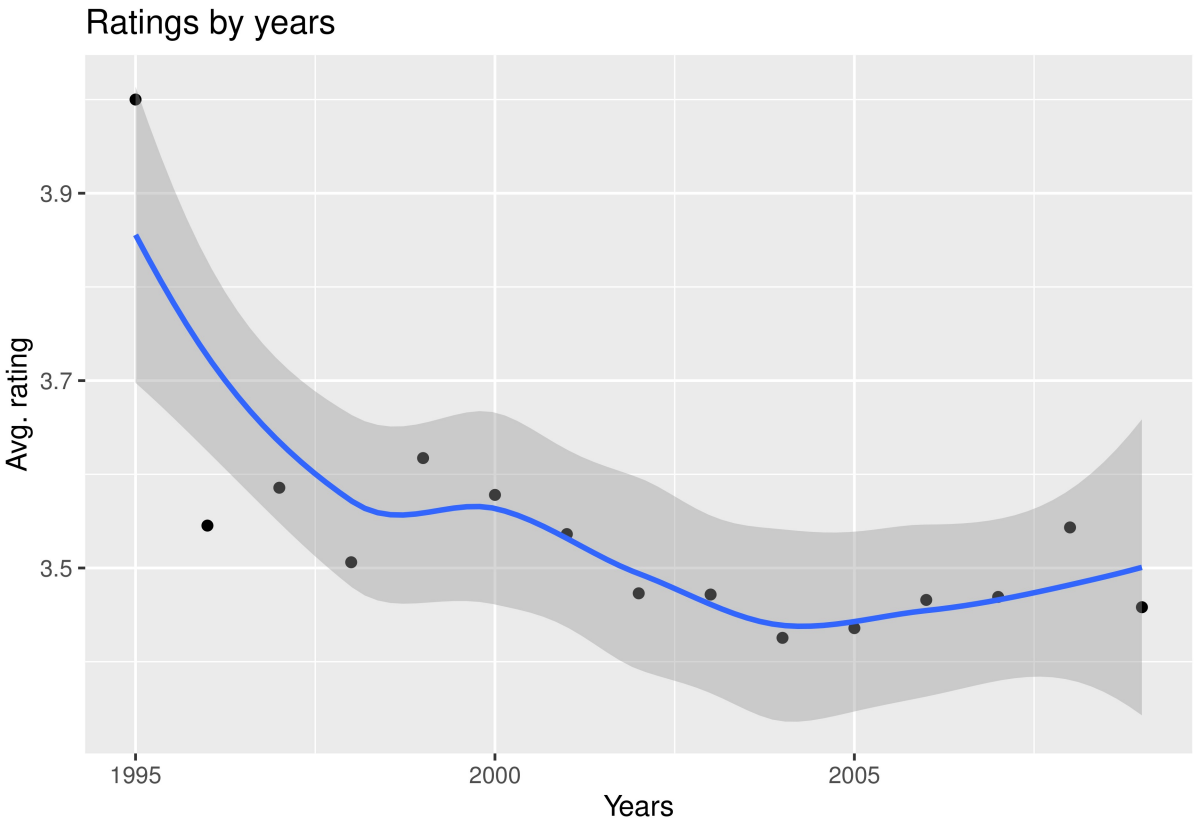


Key findings: The analysis shows that the distribution of the value of rating in uneven. The most popular ratings are 4.0 and 3.0 which are above 50% of all ratings.

Timestamp analysis

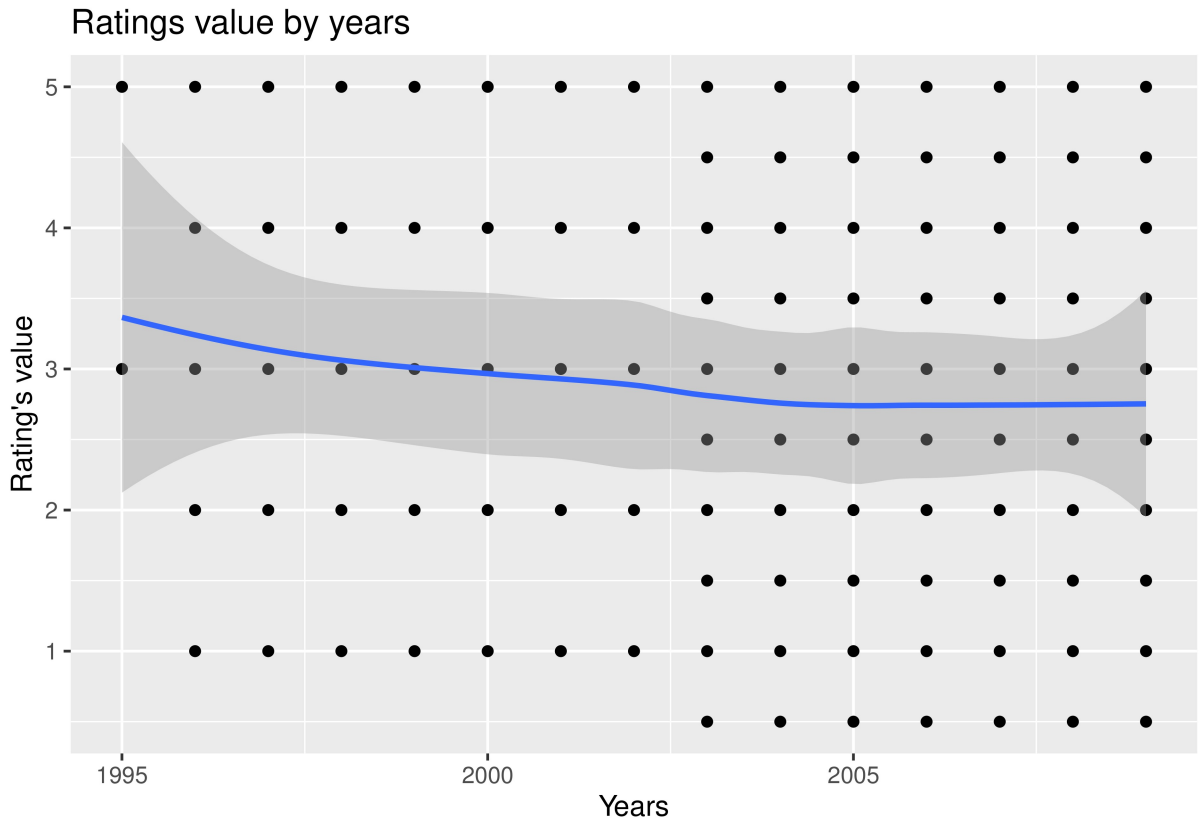
The average rating is experienced a declining trend within the timeline.

```
# Formating column "timestamp" for further analysis
edx_timeline <- mutate(edx, year = year(as_datetime(timestamp)))


# The average rating is experienced a declining trend.
edx_timeline %>% group_by(year) %>%
  summarize(avarage_rating = mean(rating)) %>%
  ggplot(aes(year, avarage_rating)) +
  geom_point() +
  geom_smooth() +
      labs(x = "Years", y = "Avg. rating") +
  ggtitle("Ratings by years")
```



New values of ratings with half stars appeared in 2003.

```
# Distribution of Ratings value by years
edx_timeline %>% group_by(year, rating) %>%
  summarize(n()) %>%
  ggplot(aes(year, rating)) +
  geom_point() +
  geom_smooth()+
    labs(x = "Years", y = "Rating's value") +
  ggtitle("Ratings value by years")
```



Key findings: Timestamp analysis shows that average rating has declining trend after 2003 driven by ratings with half stars. In fact, Timestamp effect has no significant evidence and may not affect the recommendation system.

IDENTIFICATION OF OPTIMAL METHODS AND TOOLS

This section covers the application of different methods and tools in order to identify an optimal way for the machine learning algorithm which should be able to predict movie ratings with RMSE < 0.86490.

RMSE function

For evaluating predictions of algorithm, Root Mean Square Error (RMSE) will be used under the following code:

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

Edx data split

The edx data was spit into separate training and test sets to design and test an algorithm.

```
# Dataset preparation: generating train and test sets from edx set
y <- edx$rating

set.seed(755)
test_index <- createDataPartition(y, times = 1, p = 0.2, list = FALSE)
test_set <- edx[test_index, ]
train_set <- edx[-test_index, ]
```

```r
# Make sure userId and movieId in train set are also in test set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Average rating Model

The idea of this method is to use an average rating to predict the rating of movies.

```r
# Idemtidying avarage effect
mu <- mean(train_set$rating)

# Evaluating the RMSE of the model on test set
model_1 <- RMSE(test_set$rating, mu)
model_1
```

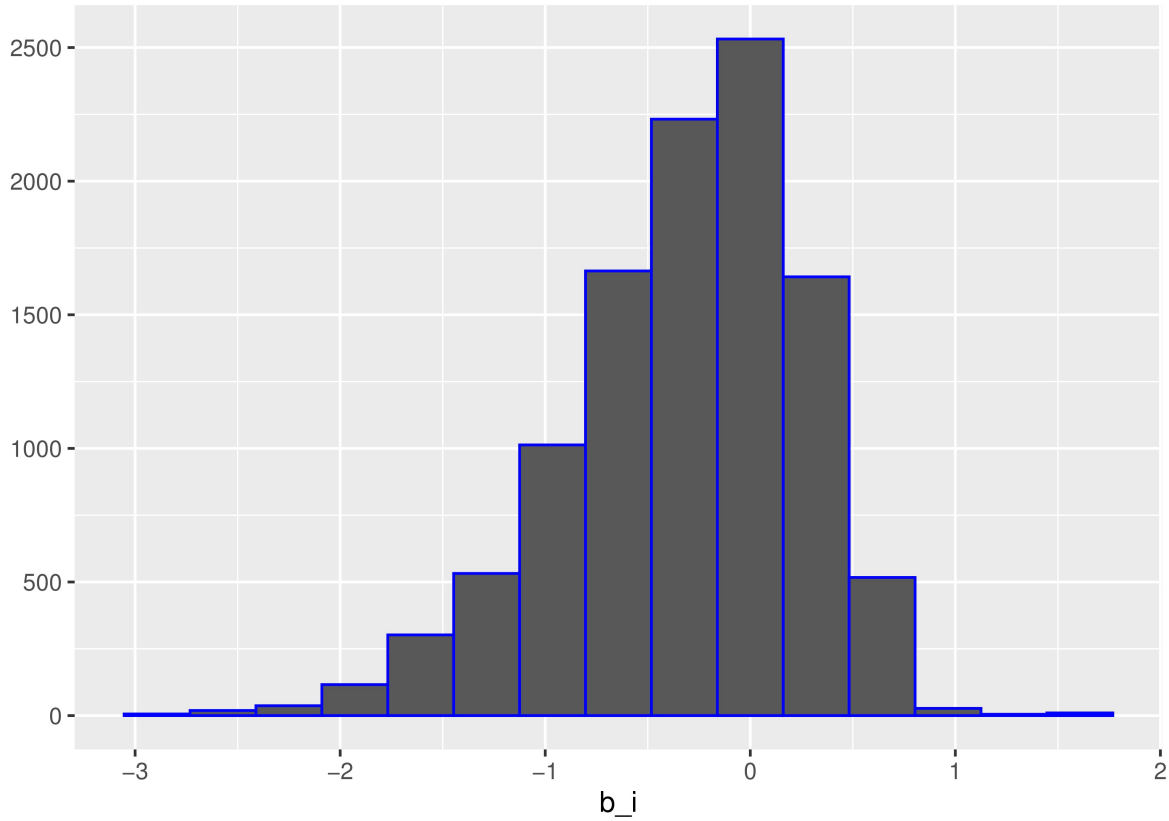
```
## [1] 1.06053
```

The result of RMSE for this model is not sufficient for the algorithm.

Movie Effect Model

This model is based on the utilization of the movie effect in order to improve the quality of the prediction of movie ratings.

```r
# Idemtidying Movie effect
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_avgs %>% qplot(b_i, geom ="histogram", bins = 15, data = ., color = I("blue"))
```



Movie effect has non-central distribution which is affected by downward bias due to negative ratings.

```
# Predicting rating results on test set
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

# Evaluating the RMSE of the model on test set
model_2 <- RMSE(test_set$rating, predicted_ratings)
model_2
```

```
## [1] 0.9440004
```

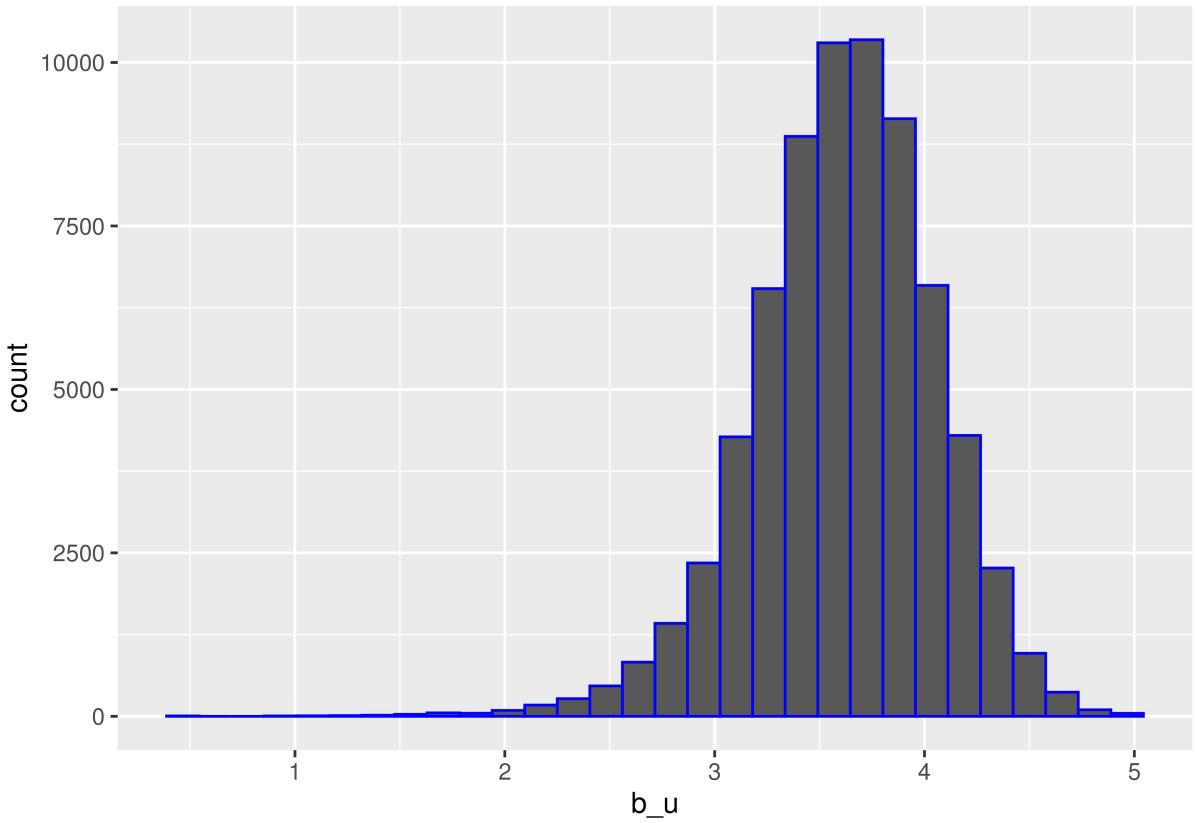
The RMSE results of Movie Effect Model are lower by 0.12 in comparison with model which is based on average ratings only. However, it is not enough for the predictions of algorithm.

Movie + User Effects Model

This model will modify previous model by adding user's effect in order to increase the level of the rating prediction.

```
# Idemtidying Movie effect
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Idemtidying User effect (based on the average rating for user with more than 100 movies ratings)
train_set %>%
```

```
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "blue")
```




```
# Culculating user effect
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

User preferences have essential volatility due to very picky users and those who are easygoing in terms of movie evaluation.

```
# Predicting rating results on test set
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

# Evaluating the RMSE of the model on test set
model_3 <- RMSE(test_set$rating, predicted_ratings)
model_3
```

```
## [1] 0.8667329
```

The result of Movie + User Effects Model improved the RMSE by 0.0773 in comparison with previous model.

Movie + User + Genres Effects Model

There is another indicator in the data set which might have an effect on our prediction model. It is genre. However, a lot of movies have more than one different genre.

```
head(test_set)
```

```
##   userId movieId rating timestamp                           title
## 1      1     329      5 838983392    Star Trek: Generations (1994)
## 2      1     355      5 838984474          Flintstones, The (1994)
## 3      1     420      5 838983834     Beverly Hills Cop III (1994)
## 4      1     466      5 838984679      Hot Shots! Part Deux (1993)
## 5      1     520      5 838984679 Robin Hood: Men in Tights (1993)
## 6      1     616      5 838984941           Aristocats, The (1970)
##                             genres
## 1 Action|Adventure|Drama|Sci-Fi
## 2       Children|Comedy|Fantasy
## 3  Action|Comedy|Crime|Thriller
## 4             Action|Comedy|War
## 5                        Comedy
## 6            Animation|Children
```

To analyze the effect of genre, train and test sets should be modified. The movies with multiple genres have to be split in several rows. This may cause dramatic increase of size of data set. The following code will split "genres" column in training and test sets respectively.

```
# Modify the genres colomn in train and test sets (column splitting)

train_set_g <- train_set  %>% separate_rows(genres, sep = "\\|")
test_set_g <- test_set  %>% separate_rows(genres, sep = "\\|")
```

Unfortunately, running of the code caused the error massage about limits of computer memory to perform this code. In this case, we will try to identify Genre Effect without splitting "genres" column.

```
# Idemtidying Movie effect
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Idemtidying User effect
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Idemtidying Genre effect
genres_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

# Predicting rating results on test set
predicted_ratings <- test_set %>%
```

```
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  mutate(preddd = mu + b_i + b_u + b_g) %>%
  .$preddd

# Evaluating the RMSE of the model on test set
model_4 <- RMSE(test_set$rating, predicted_ratings)
model_4
```

```
## [1] 0.8663809
```

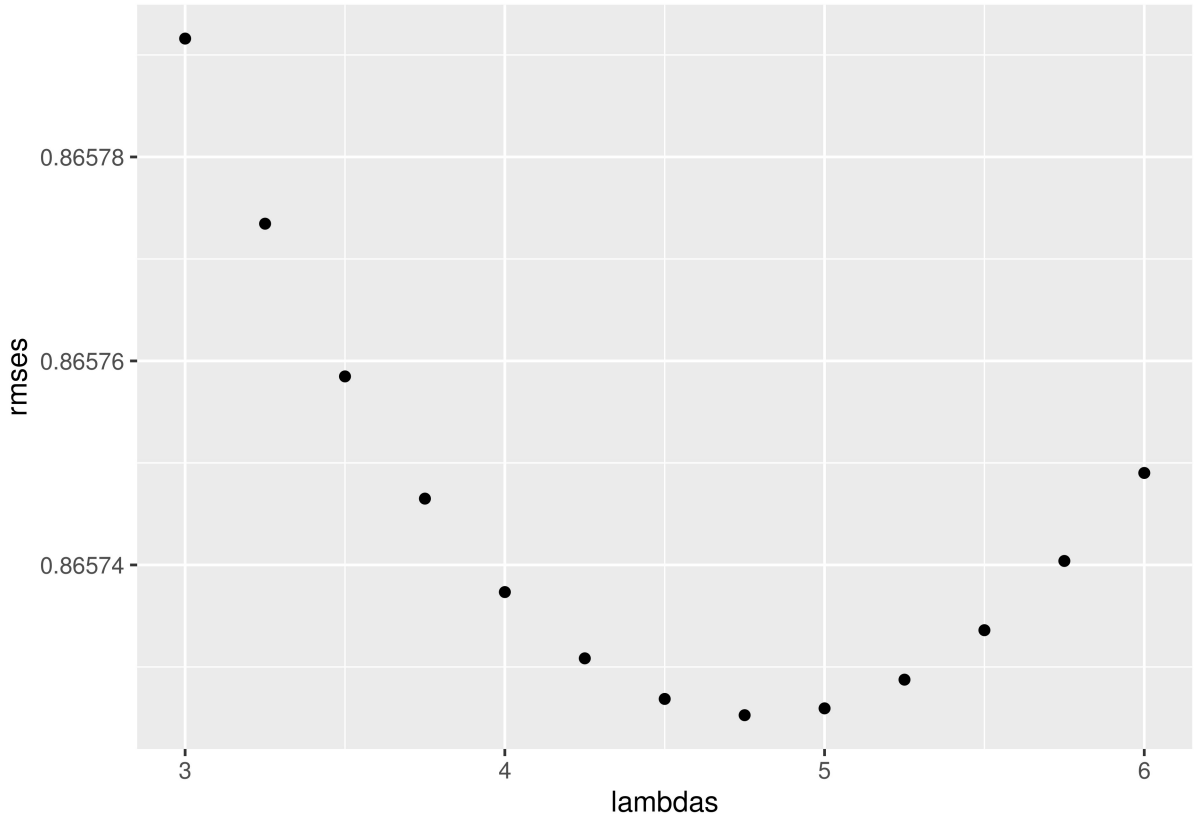
The RMSE of the Movie + User + Genres Effects Model is 0.8662908 which slightly better (by 0.00035) than previous one.

Regularized Movie + User + Genres Effects Model

Let's try to optimize the Movie + User + Genres Effects Model with Regularization. This approach will use the penalty lambda as tuning parameter. To compute the regularized estimates we need to find the best lambda which can provide the minimum RMSE.

```
# Indentification of lambda as a tuning parameter for Regularized Model

lambdas <- seq(3, 6, 0.25)

# Cross-validation will be used for the best lambda indentification

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

  # idemtidying avarage effect on training set
  mu <- mean(train_set$rating)

  # idemtidying movie effect on training set
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  # idemtidying user effect on training set
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  # Idemtidying Genre effect on training set
  b_g <- train_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u))

  # Predicting rating results on test set
  predicted_ratings <- test_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_g, by='genres') %>%
    mutate(preddd = mu + b_i + b_u + b_g) %>%
```

```
    .$preddd

  return(RMSE(test_set$rating, predicted_ratings))
})

# Plot rmses vs lambdas to identify the optimal lambda
qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
lambda   # the best lambda to be used for the regularized model
```

```
## [1] 4.75
```

The best lambda is 4.75.

```
# Compute regularized model using the best lambda (= 4.75)

lambda_2 <- 4.75

# idemtidying avarage effect on training set
mu <- mean(train_set$rating)
# Compute regularized estimates of b_i using lambda on training set
movie_reg <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_2), n_i = n())
# Compute regularized estimates of b_u using lambda on training set
user_reg <- train_set %>%
```

```
  left_join(movie_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_2), n_u = n())
# Compute regularized estimates of b_g using lambda on training set
genre_reg <- train_set %>%
  left_join(movie_reg, by='movieId') %>%
  left_join(user_reg, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda_2), n_g = n())
# Predict ratings on test set
predicted_ratings <- test_set %>%
  left_join(movie_reg, by='movieId') %>%
  left_join(user_reg, by='userId') %>%
  left_join(genre_reg, by = 'genres') %>%
  mutate(pred_reg = mu + b_i + b_u + b_g) %>%
  .$pred_reg
# Evaluating the RMSE of Regularized model's algorithm
model_5r <- RMSE(test_set$rating,predicted_ratings)
model_5r
```

```
## [1] 0.8657234
```

The RMSE of Regularized Movie + User + Genres Effects Model using the best lambda is 0.865651. This result is the lowest among all models and improves the prediction model by 0.00006 in comparison with the RMSE results of previous Model without Regularization.

Since the effects of all available predictors in the database were explored, the algorithm of Regularized Movie + User + Genres Effects Model (with lambda 4.75 as tuning parameter) will be used **as the final algorithm** due to the lowest RMSE.

Final Algorithm Evaluation

The final algorithm of "Regularized Movie + User + Genres Effects Model" with lambda 4.75 will be used for evaluation the RMSE on the **validation** data.

```
# Regularized Movie + User + Genre Effects Model on validaion set

# Final Algorithm evaluation using the best lambda

lambda_2 <- 4.75

# idemtidying avarage effect on edx set
mu <- mean(edx$rating)
# Compute regularized estimates of b_i using lambda on edx set
movie_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_2), n_i = n())
# Compute regularized estimates of b_u using lambda on edx set
user_reg <- edx %>%
  left_join(movie_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_2), n_u = n())
# Compute regularized estimates of b_g using lambda on edx set
genre_reg <- edx %>%
  left_join(movie_reg, by='movieId') %>%
  left_join(user_reg, by='userId') %>%
```

```
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda_2), n_g = n())
# Predict ratings on validation set
predicted_ratings <- validation %>%
  left_join(movie_reg, by='movieId') %>%
  left_join(user_reg, by='userId') %>%
  left_join(genre_reg, by = 'genres') %>%
  mutate(pred_reg = mu + b_i + b_u + b_g) %>%
  .$pred_reg
# Evaluating the RMSE of final algorithm
model_6_VAL <- RMSE(validation$rating,predicted_ratings)
model_6_VAL
```

```
## [1] 0.8644514
```

The RMSE result of the final algorithm is 0.8644514.

RESULTS

According to the results on validation set, the RMSE of the final algorithm (based on Regularized Model) is 0.8644514 which is lower than target RSME (0.86490). **The validation data set was used to evaluate the RMSE of the final algorithm only.**

CONCLUSION

The aim of the project was to practice obtained knowledge in the course to generate the machine learning algorithm to predict movie ratings based on the MovieLens data set. Root Mean Square Error (RMSE) was used to evaluate how close our predictions are to the true values. The RMSE results demonstrate an improvement of models based on different assumptions. Regularization approach was used in the final algorithm. The RMSE of the final algorithm is 0.8644514.

At the same time, due to limited capacity of my computer I was not successful in splitting genres column to analyze this effect for model optimization. Therefore, there is an opportunity for future work to provide additional analysis of genre effect in order to find the minimum RSME.