HarvardX: PH125.9x Data Science Movie recommendation system using the MovieLens dataset

Fernando Gripe

January 12, 2023

1. Introduction

Recommendation engines are a sub-class of machine learning which generally deal with ranking or rating products / users. A recommendation system is a system which predicts ratings a user might give to a specific item (product, service, etc). These predictions will then be ranked and returned back to the user.

They have been used by various large name companies like Google, Instagram, Spotify, Amazon, Netflix etc. often to increase engagement with users in the platform. For example, Netflix would recommend movies similar to the ones you've repeatedly watched to or liked so that you can continue using their platform to watch movies.

In fact the success of Netflix is said to be based on its strong recommendation system. The Netflix prize (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) represents the importance of the algorithm for products recommendation system.

2. Overview and purpose of the project

This project is a requirement for the HarvardX Professional Certificate Data Science Program which aims to predict movies ratings using "MovieLens 10M Dataset" https://grouplens.org/datasets/movielens/10m/ (https://grouplens.org/datasets/movielens/10m/).

The objective in this project is to train a machine learning algorithm that predicts user ratings (stars) using the inputs of a provided subset ('edx') to predict movie ratings in a provided validation subset ('final_holdout_test').

3. RMSE

The Root Mean Square Error, or RMSE, is the value used to evaluate algorithm performance. The RMSE is one of the most used measure of the 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 lower RMSE, the better is the model. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers results / numbers.

Three models that will be developed will be compared using their resulting RMSE in order to assess their quality.

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

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

3.1 HarvardX Project Evaluation Criteria for RMSE

The evaluation criteria for this algorithm is a RMSE in the following range with the according grades:

5 points: RMSE >= 0.90000
10 points: 0.86550 <= RMSE <= 0.89999
15 points: 0.86500 <= RMSE <= 0.86549
20 points: 0.86490 <= RMSE <= 0.86499
25 points: RMSE < 0.86490

4. Dataset and data preprocessing

```
##########################################################
# Dataset
# Create edx and final_holdout_test sets
##########################################################

# 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(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(stringi)) install.packages("stringi", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(tinytex)) install.packages("tinytex", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(ggplot2)
library(dplyr)
library(stringi)
library(lubridate)
library(tinytex)

options(timeout = 120)

dl <- "ml-10m.zip"

if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"

if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"

if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)

colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")

ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
```

```
colnames(movies) <- c("movieId", "title", "genres")

movies <- movies %>% mutate(movieId = as.integer(movieId))

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

#additional columns: rating_year, movie_year, movie_age
movielens <- mutate(movielens,
                    movieId = as.numeric(movieId),
                    title = as.character(title),
                    genres = as.character(genres),
                    rating_year= year(as_datetime(timestamp)),
                    movie_year = as.numeric(stri_extract_last(title, regex = "(\\d{4})", comm
ents = TRUE)) %>% as.numeric(),
                    movie_age = 2022 - movie_year) %>%
            select(-timestamp)

#checking if regex worked
movielens %>% filter(movie_year < 1900 || movie_year > 2018) %>%
  group_by(movie_year) %>%
  summarize(n = n())
```

```
0 rows
```

```
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later

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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

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

5. Exploratory Analysis

5.1 Some data transformations were implemented to create new categories for example:

- rating_year: the year the rating was assigned, sourced from the 'timestamp' column of the original datasource
- movie_year: the year the movie was launched, sourced from the movie title column of the original datasource
- movie_age: the number of years between the launch year and the reference year 2022 (last full year)

5.2 Exploratory Analysis was used to get more insights for the recommendation system and have a better understanding of the dataset

Display the Structure of the dataset

```
## 'data.frame':    9000055 obs. of  8 variables:
##  $ userId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId    : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating     : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ title      : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (199
4)" ...
##  $ genres     : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thrille
r" "Action|Adventure|Sci-Fi" ...
##  $ rating_year: num  1996 1996 1996 1996 1996 ...
##  $ movie_year : num  1992 1995 1995 1994 1994 ...
##  $ movie_age  : num  30 27 27 28 28 28 28 28 28 28 ...
```

Data summary

```
##      userId         movieId          rating         title          
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:9000055    
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   Class :character  
##  Median :35738   Median : 1834   Median :4.000   Mode  :character  
##  Mean   :35870   Mean   : 4122   Mean   :3.512                     
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000                     
##  Max.   :71567   Max.   :65133   Max.   :5.000                     
##     genres           rating_year     movie_year     movie_age     
##  Length:9000055     Min.   :1995   Min.   :1915   Min.   : 14.00  
##  Class :character   1st Qu.:2000   1st Qu.:1987   1st Qu.: 24.00  
##  Mode  :character   Median :2002   Median :1994   Median : 28.00  
##                     Mean   :2002   Mean   :1990   Mean   : 31.78  
##                     3rd Qu.:2005   3rd Qu.:1998   3rd Qu.: 35.00  
##                     Max.   :2009   Max.   :2008   Max.   :107.00
```

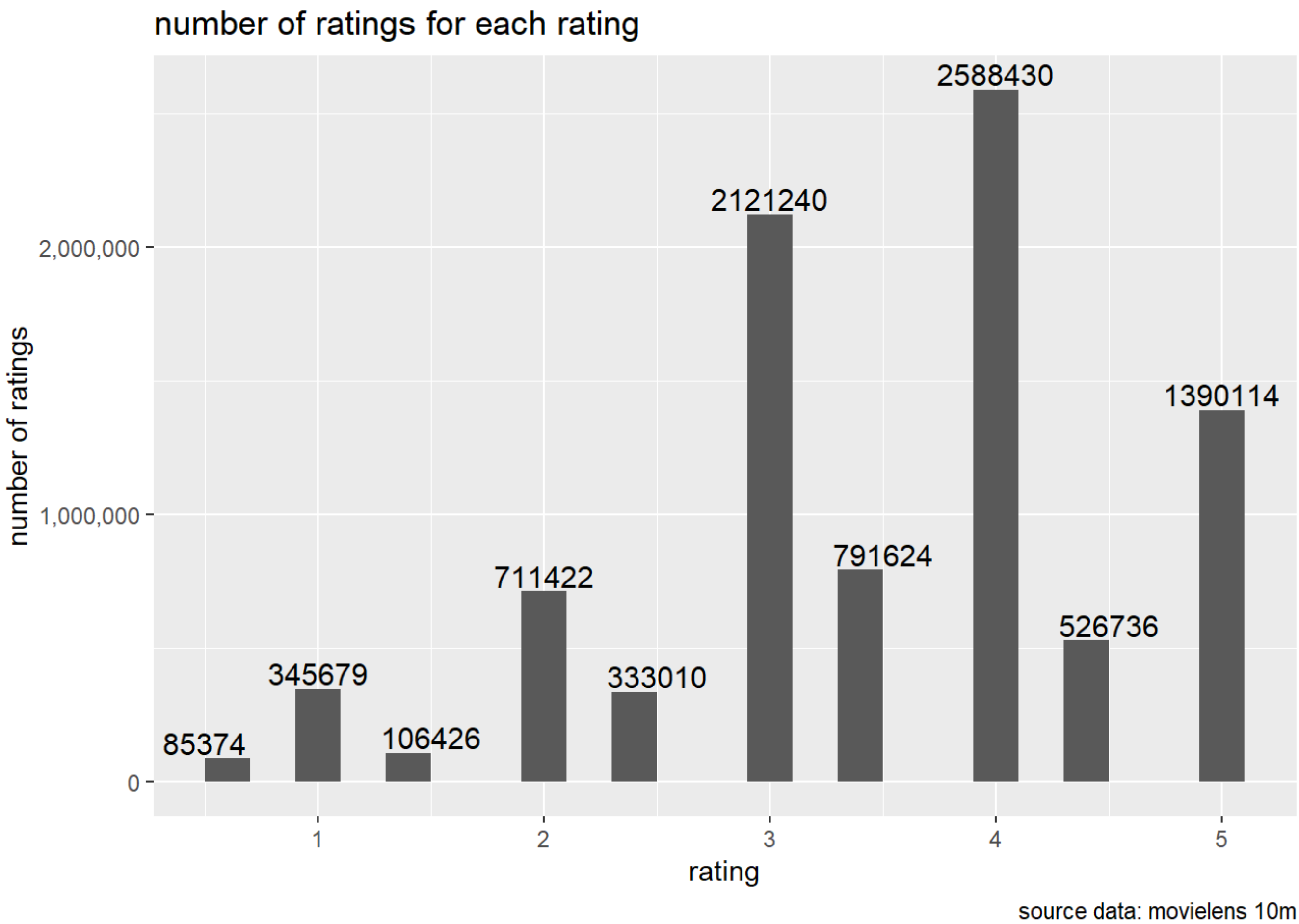
Data Sumary

n_of_rows <int>	n_of_column <int>	n_of_users <int>	n_of_movies <int>	avg_rating <dbl>	zero_rating <int>	three_rating <int>
9000055	8	69878	10677	3.51	0	2121240

1 row

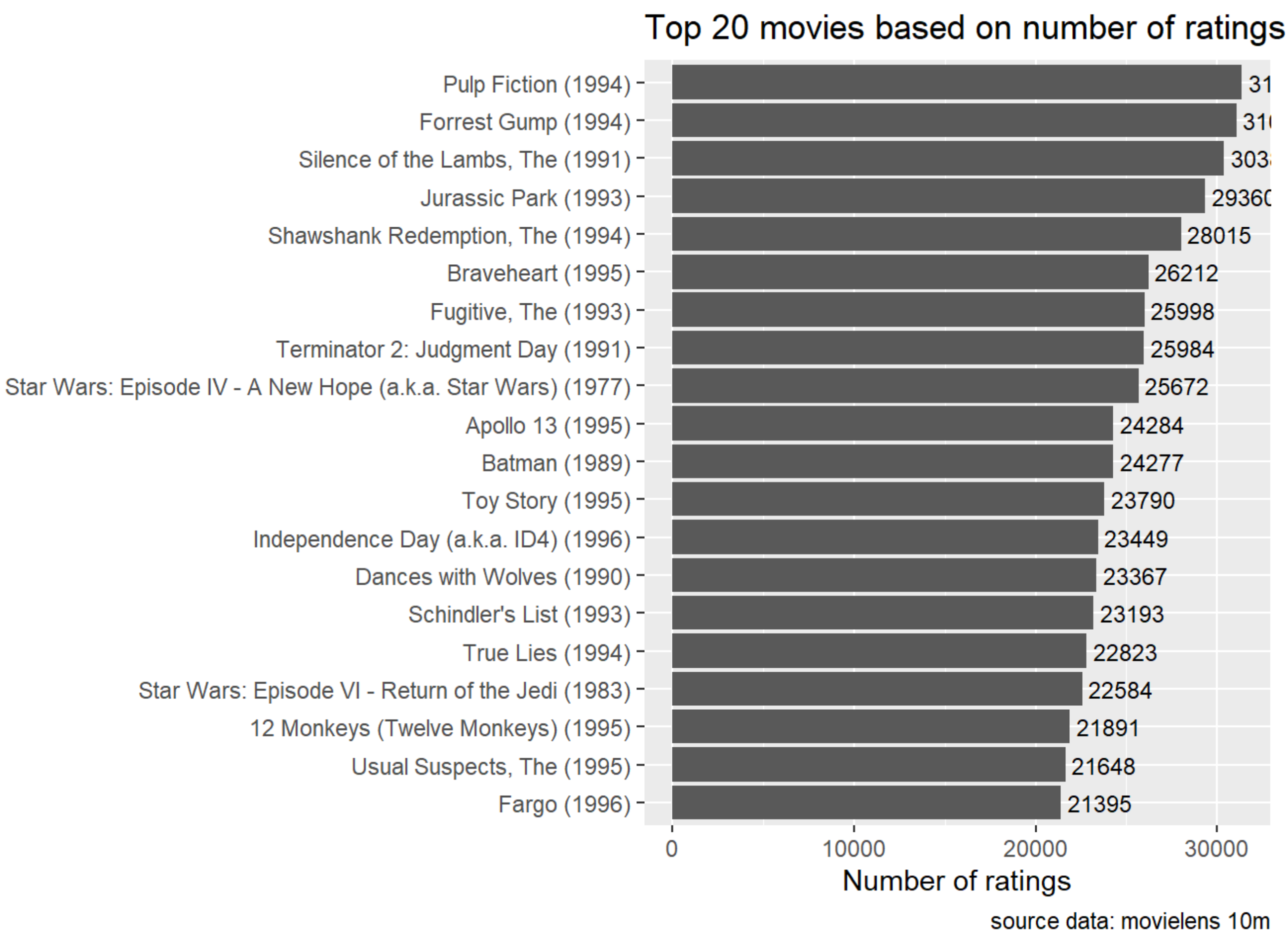
Plot: Number of ratings for each rating

As we can see in the graph below, most grades are assigned between 3 and 4



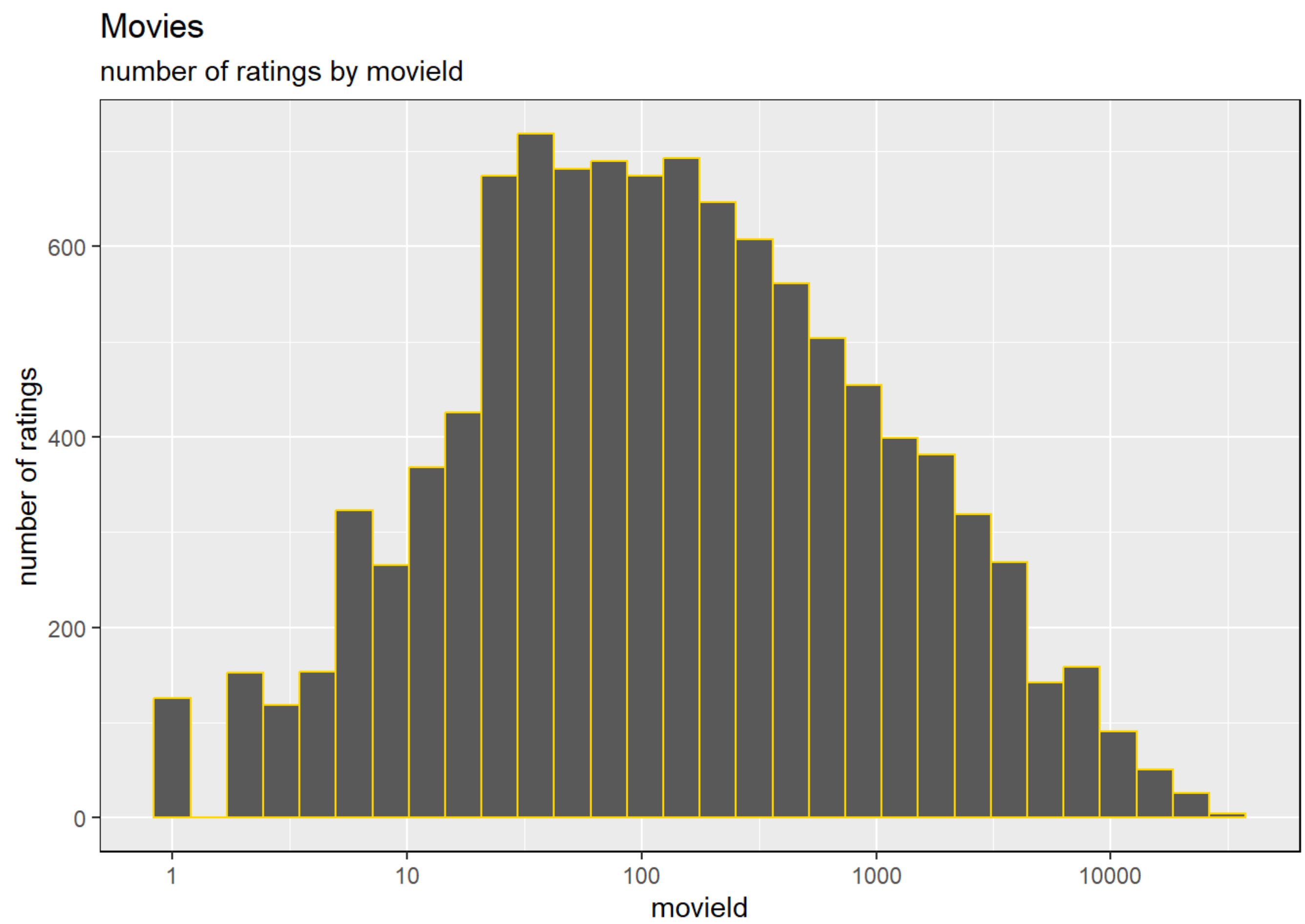
Plot: Top 20 movies based on number of ratings

We can check below the films most frequently awarded with ratings by users.



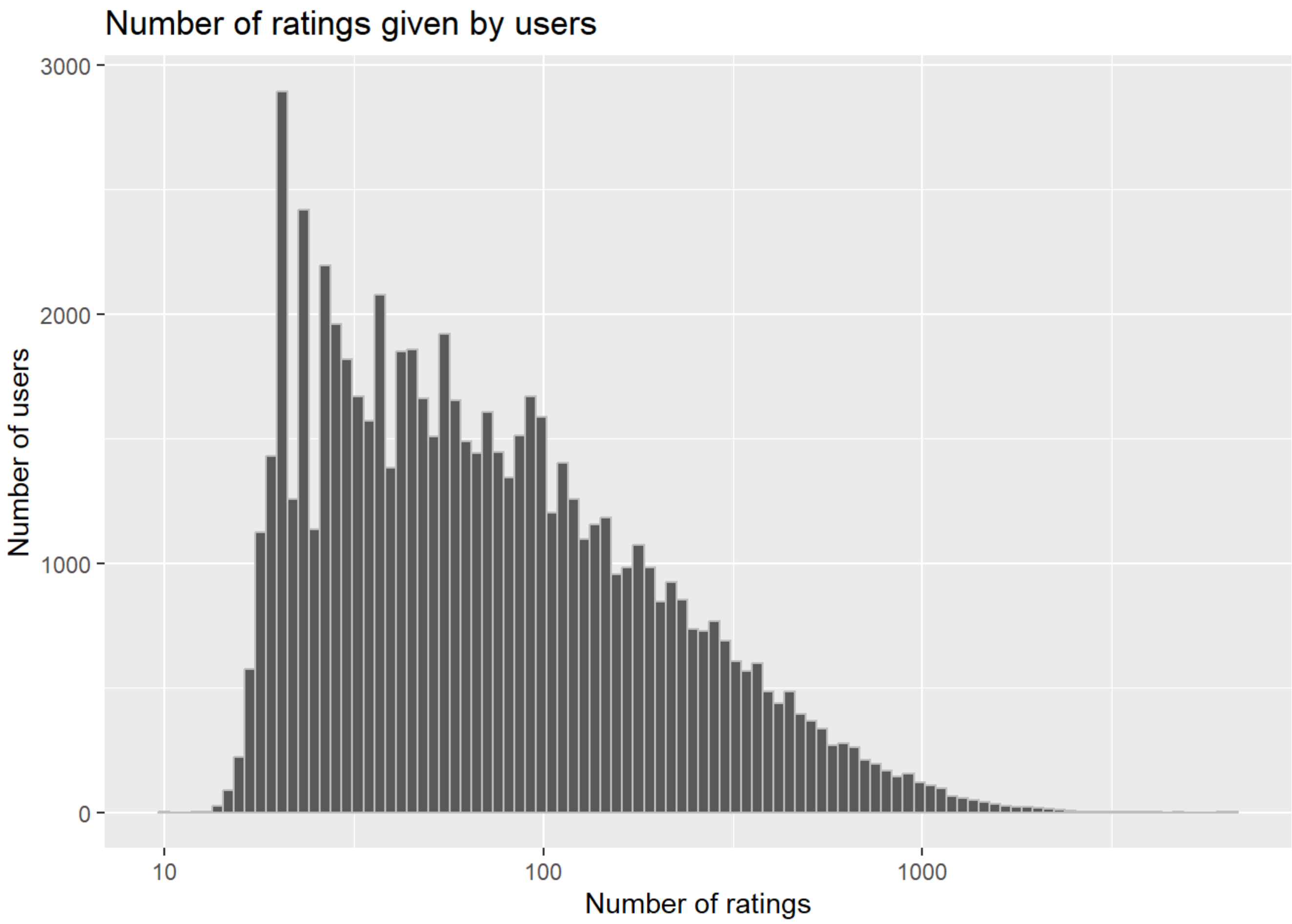
Histogram of number of ratings by movieId

We can see that in the available dataset most movies awarded between 50 and 150 ratings



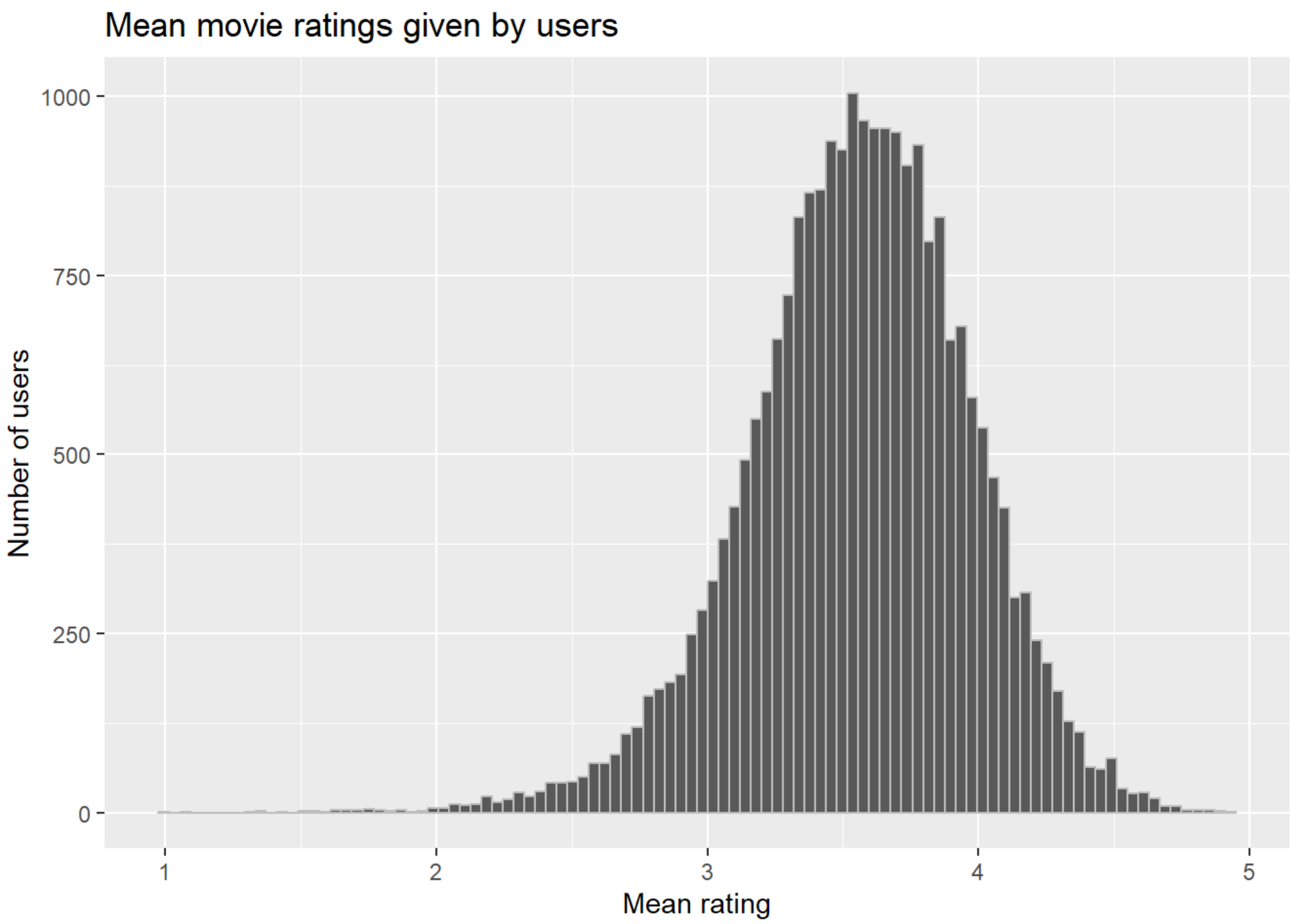
Histogram of number of ratings by userId

We can see that in the available dataset most users awarded grades up to 100 times



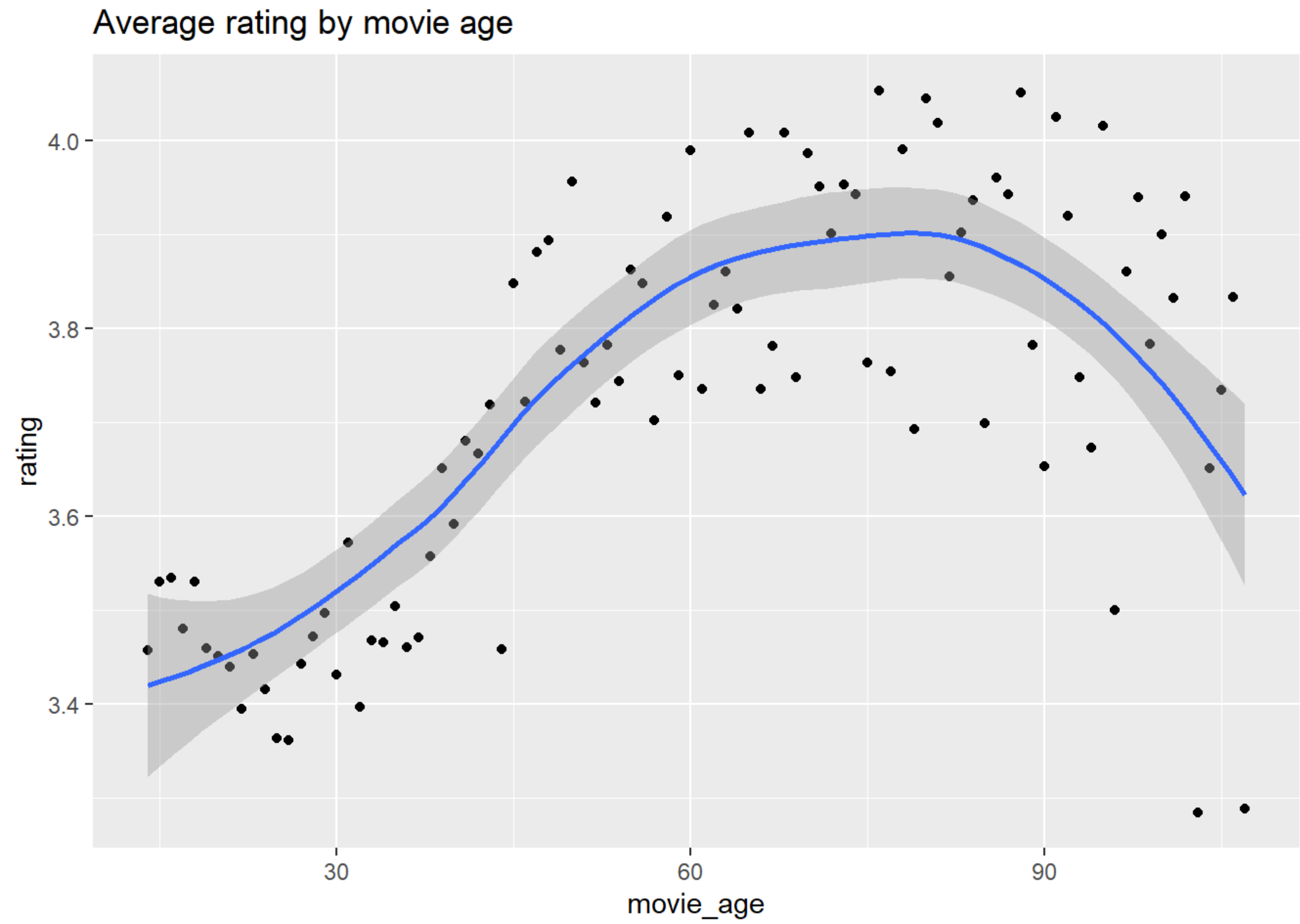
Plot mean movie ratings given by users

We can see that in the available dataset most users gave ratings close to 3.6



Average rating by movie age

We can see that in the available dataset the average score given changes between 3.4 to 3.9, depending on the age of each movie.



Results

Models were built from the training data (edx), and then assessed to the test data (final_holdout_test) in three different approaches as following

5.3.1 Average Movie Rating (mu):

Assuming that some films tend to be better or worse graded, we can use this effect to generalize to the next grade assignments using the average rating of each movie.

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

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

```
AMR <- RMSE(final_holdout_test$rating, mu)

rmse_r <- tibble(method = "Average movie rating model", RMSE = AMR)

AMR
```

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

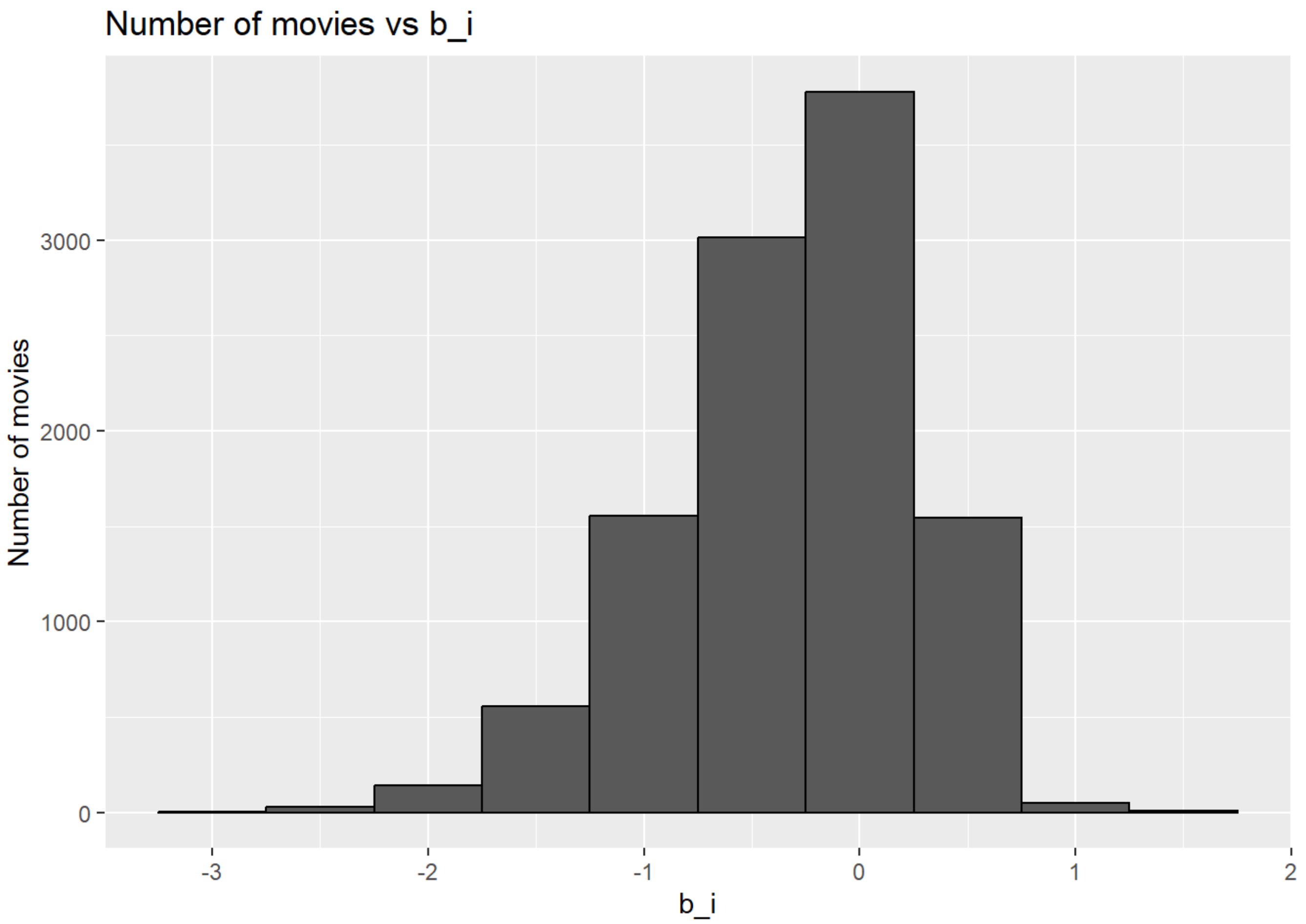
5.3.2 Movie Effect (b_i):

Variability may be related to the fact that each movie will have a different rating distribution.

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

predicted_ratings <- mu + final_holdout_test %>% left_join(movie_avgs, by='movieId') %>% pull
(b_i)

MRE <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_r <- bind_rows(rmse_r, tibble(method="Movie rating effect",  RMSE = MRE ))

MRE
```

```
## [1] 0.9439087
```

In the following histogram we can note that more movies have negative effects than positive



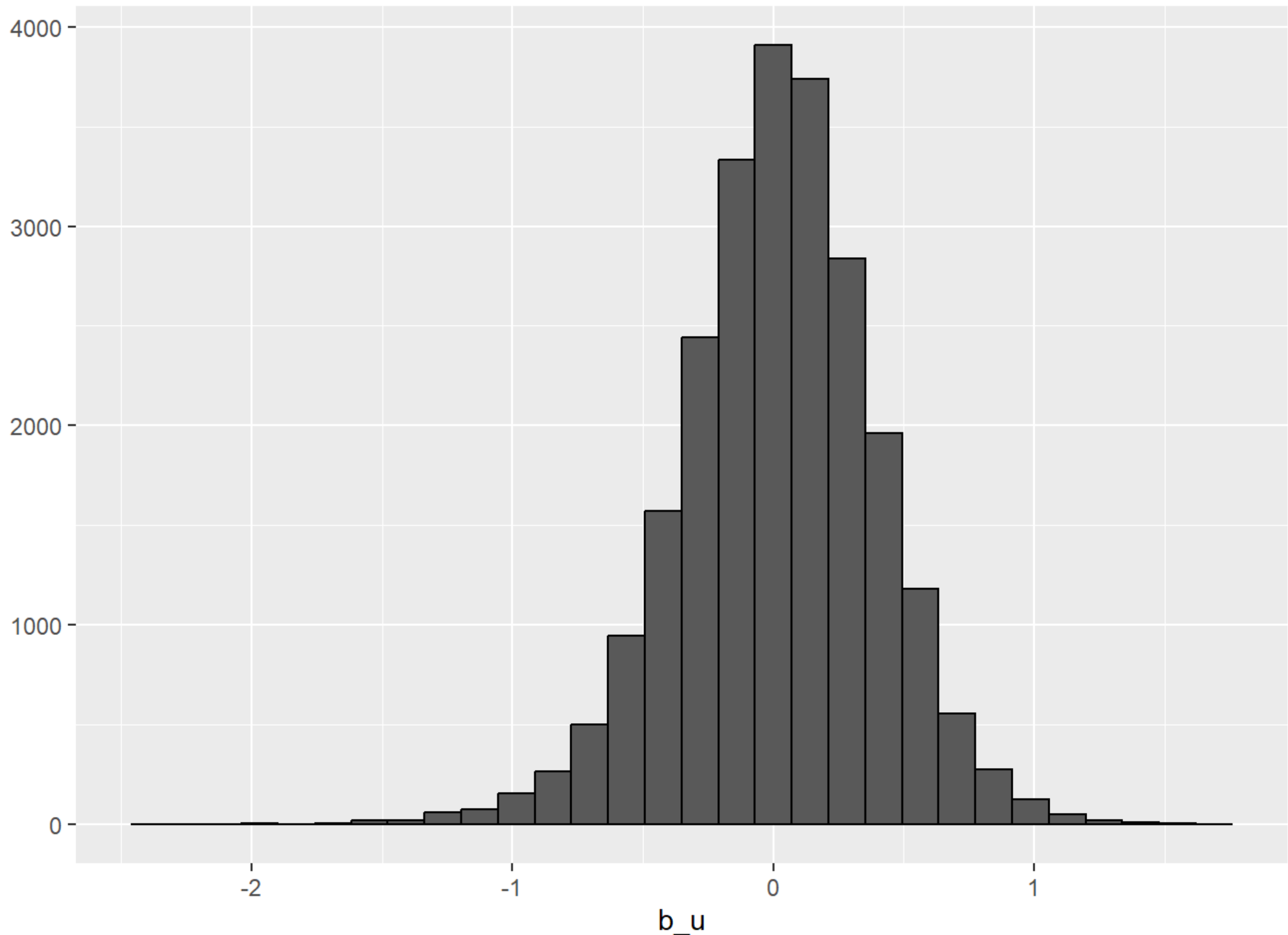
5.3.3 Adding User Effect Movie (b_u):

Variability may be related to the fact that each user will have a different rating distribution.

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

predicted_ratings <- final_holdout_test%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

MURE <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_r <- bind_rows(rmse_r, data_frame(method="Movie and user ratting effect",  RMSE = MURE))

MURE
```

```
## [1] 0.8653488
```

We can see in the following plot that there is a great variability across all users.



6 Final Result and Conclusion

the objective of this work was to apply the knowledge acquired in the course to build a recommendation system.

in the end, three approaches were used, with the best result being Movie rating effect's and user effect, achieving an RMSE of 0.8653488.

As limitations of the project we can mention that there is the possibility of testing new approaches to try to improve the RMSE such as the use of other variables such as film genre, temporal effects or applying more complex models such as Matrix Factorization.

method <chr>	**RMSE** <dbl>
Average movie rating model	1.0612018
Movie rating effect	0.9439087
Movie and user ratting effect	0.8653488

3 rows