HarvardX: PH125.9x Data Science
MovieLens Rating Prediction Project

Spyridon Ntokos

May 21, 2020

Contents

1 Overview **2**

1.1 Introduction . . . 2

1.2 Aim of the project . . . 2

1.3 Problem definition . . . 2

1.4 Root Mean Square Error - RMSE . . . 3

1.5 Data ingestion . . . 3

2 Methods and Analysis **5**

2.1 Exploratory Data Analysis . . . 5

2.2 Data analysis strategies . . . 8

2.2.1 Movie bias . . . 8

2.2.2 User bias . . . 9

2.2.3 Genre popularity . . . 10

2.2.4 Rating vs Release year . . . 11

2.2.5 Age of movie vs Rating . . . 12

2.2.6 Age of movie vs Star-rating . . . 13

3 Data analysis: Model Preparation **14**

3.1 I. Naive model: Average movie rating model . . . 14

3.2 II. Movie effect model . . . 15

3.3 III. User & Movie effect model . . . 16

3.4 IV. Regularized model . . . 18

4 RMSE results **20**

5 Conclusion **20**

6 Appendix - Enviroment 20

1 Overview

This project is related to the MovieLens Project of the HarvardX: PH125.9x Data Science: Capstone course. The objective of this project is to build a predictive model of a movie dataset to be evaluated by measuring RMSE.

1.1 Introduction

Recommendation systems use ratings given by different users to items, in order to make specific recommendations. International companies that sell many products to many customers globally and allow these customers to rate their products, for instance Amazon, are able to collect massive datasets that can be used to predict what rating a particular user will give to a specific item. Items for which a high rating is predicted for a given user are then recommended to that user.

The same could be done for other items, as movies for instance in our case. One of the major families of applications of machine learning in the information technology sector is the ability to make recommendations of items to potential users or customers. In fact the success of Netflix is said to be based on its strong recommendation system. In year 2006, Netflix has offered a challenge to the data science community. The challenge was to improve Netflix' recommendation software's accuracy by 10% and win the 1,000,000$ prize.

This capstone project is based on the winner's team algorithm and is a part of the course HarvardX: PH125.9x Data Science: Capstone project. However, the Netflix data is not freely available, so an open source dataset from movieLens '10M version of the MovieLens dataset' has been used instead.

1.2 Aim of the project

The aim of this project is to develop a machine learning algorithm using the inputs in edx subset (90% of the original dataset) to predict movie ratings in the validation subset (the remaining 10%). Several machine learning algorithms have been used and results have been compared to get maximum possible accuracy in prediction.

This report contains problem definition, data ingestion, exploratory analysis, modeling and data analysis and results. Finally the report ends with some concluding remarks.

1.3 Problem definition

This capstone project on "Movie recommendation system" predicts the movie rating by a user based on users past rating of movies. The dataset used for this purpose can be found in the following links:

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

The challenge is not so easy given that there are many different type of biases (a.k.a. effects) present in the movie reviews. It can be different social, psychological, demographic variations that changes the taste of every single users for a given particular movie. However the problem can still be designed to tackle major biases which can be expressed via mathematical equations relatively easily. The idea here is to develop a model which can effectively predict movie recommendations for a given user without our judgement being impaired due to different biases.

1.4 Root Mean Square Error - RMSE

The value used to evaluate algorithm performance is the Root Mean Square Error, or RMSE. 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, a lower RMSE is better than a higher one. 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.

Four models that will be developed will be compared using their resulting RMSE in order to assess their quality. The evaluation criteria for this algorithm is a RMSE expected to be lower than 0.86490.

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

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

1.5 Data ingestion

The code is provided in the edx capstone project module:

- [Create Test and Validation Sets] https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+1T2020/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=last

```
##################################
# Create edx set, validation set #
##################################

# 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")
```

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

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

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

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

Algorithm development is to be carried out on the "edx" subset only, as "validation" subset will be used to test the final algorithm.

2 Methods and Analysis

2.1 Exploratory Data Analysis

To get familiar with the dataset, we find the first rows of "edx" subset as below. The subset contain the six variables "userID", "movieID", "rating", "timestamp", "title", and "genres". Each row represent a single rating of a user for a single movie.

```
##   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
```

A summary of the subset confirms that there are no missing values.

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

Before continuing further into in-depth observation of the dataset, modification of the columns is needed to acquire an appropriate format for exploratory analysis.

Exctract the year of release:

```
edx <- edx %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)),
         title = str_sub(title, 1, -8))
names(edx)[names(edx) == "year"] <- "year_release"
```

Transform timestamp to year:

```
edx <- edx %>%
  mutate(timestamp = as.POSIXct(timestamp,
                                origin = "1970-01-01",
                                tz = "GMT"))
edx$timestamp <- format(edx$timestamp, "%Y")
names(edx)[names(edx) == "timestamp"] <- "year_rated"
edx <- edx %>%
  mutate(year_rated = as.numeric(year_rated))
colnames(edx)
```

```
## [1] "userId"       "movieId"      "rating"       "year_rated"   "title"
## [6] "genres"       "year_release"
```

The above transformations are also executed on the validation dataset. To assure the dataset's consistency a peek into the first rows is needed:

```
head(edx)
```

```
##   userId movieId rating year_rated                 title
## 1      1     122      5       1996              Boomerang
## 2      1     185      5       1996               Net, The
## 3      1     292      5       1996               Outbreak
## 4      1     316      5       1996               Stargate
## 5      1     329      5       1996 Star Trek: Generations
## 6      1     355      5       1996       Flintstones, The
##                          genres year_release
## 1                Comedy|Romance         1992
## 2          Action|Crime|Thriller         1995
## 3  Action|Drama|Sci-Fi|Thriller         1995
## 4       Action|Adventure|Sci-Fi         1994
## 5 Action|Adventure|Drama|Sci-Fi         1994
## 6       Children|Comedy|Fantasy         1994
```

The total of unique movies and users in the edx subset is about 70.000 unique users and about 10.700 different movies:

```
##   n_users n_movies
## 1   69878    10677
```

In the dataset 'genres' is referred to different categories of movies:

```
# Extract unique combinations of genres in edx
genre_distinct <- edx %>%
  distinct(genres)

# Define all different genre categories as a vector
genre_categories <- genre_distinct %>%
  separate_rows(genres, sep = "\\|") %>%
  distinct(genres) %>%
  .$genres

genre_categories
```

```
##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

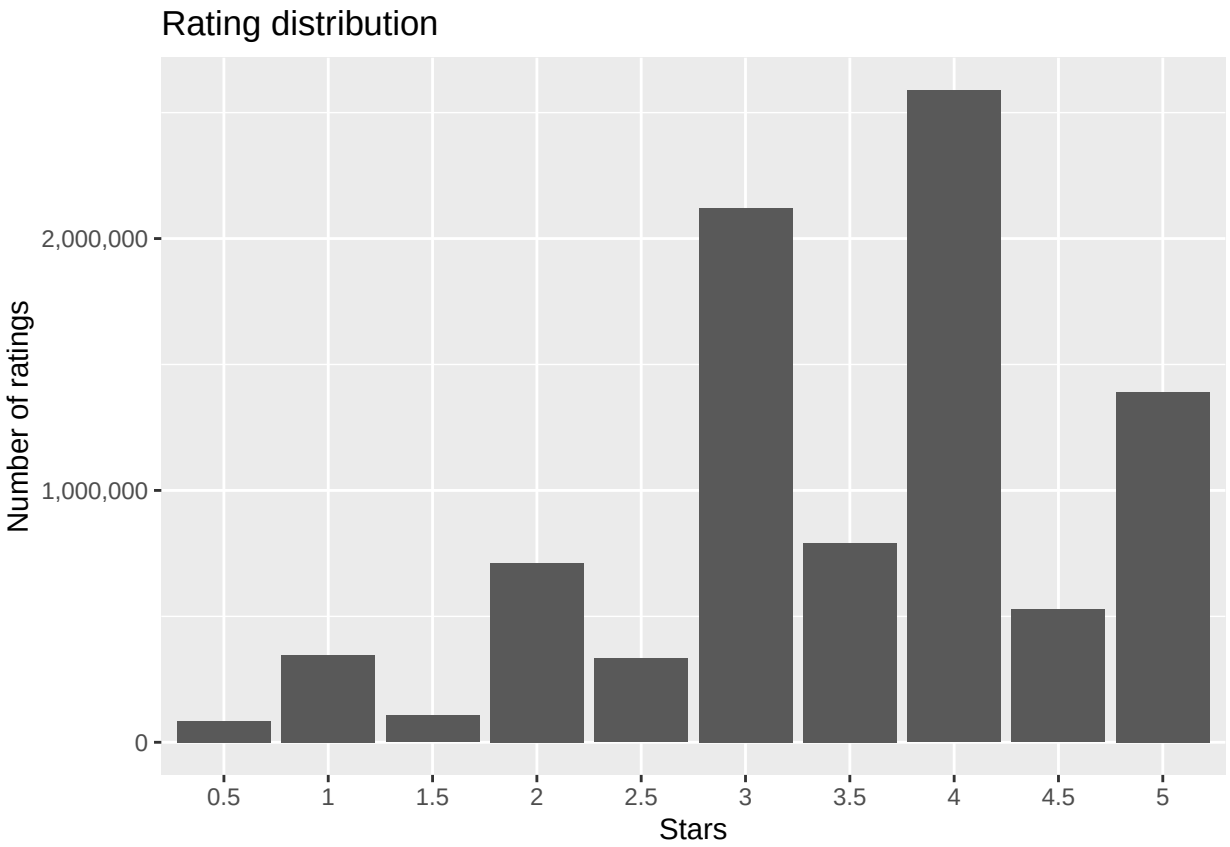
Number of ratings per movie category:

```
sapply(genre_categories, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
##             Comedy            Romance             Action              Crime
##            3540930            1712100            2560545            1327715
##           Thriller              Drama             Sci-Fi          Adventure
##            2325899            3910127            1341183            1908892
##           Children            Fantasy                War          Animation
##             737994             925637             511147             467168
##            Musical            Western            Mystery          Film-Noir
##             433080             189394             568332             118541
##             Horror        Documentary               IMAX (no genres listed)
##             691485              93066               8181                  7
```

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. In general, half rating are less common than whole star ratings.

```
stars <- data.frame(rating = as.factor(edx$rating))

stars %>%
  ggplot(aes(rating)) +
  geom_bar() +
  ggtitle("Rating distribution") +
  xlab("Stars") +
  scale_y_continuous(name = "Number of ratings", labels = scales::comma)
```



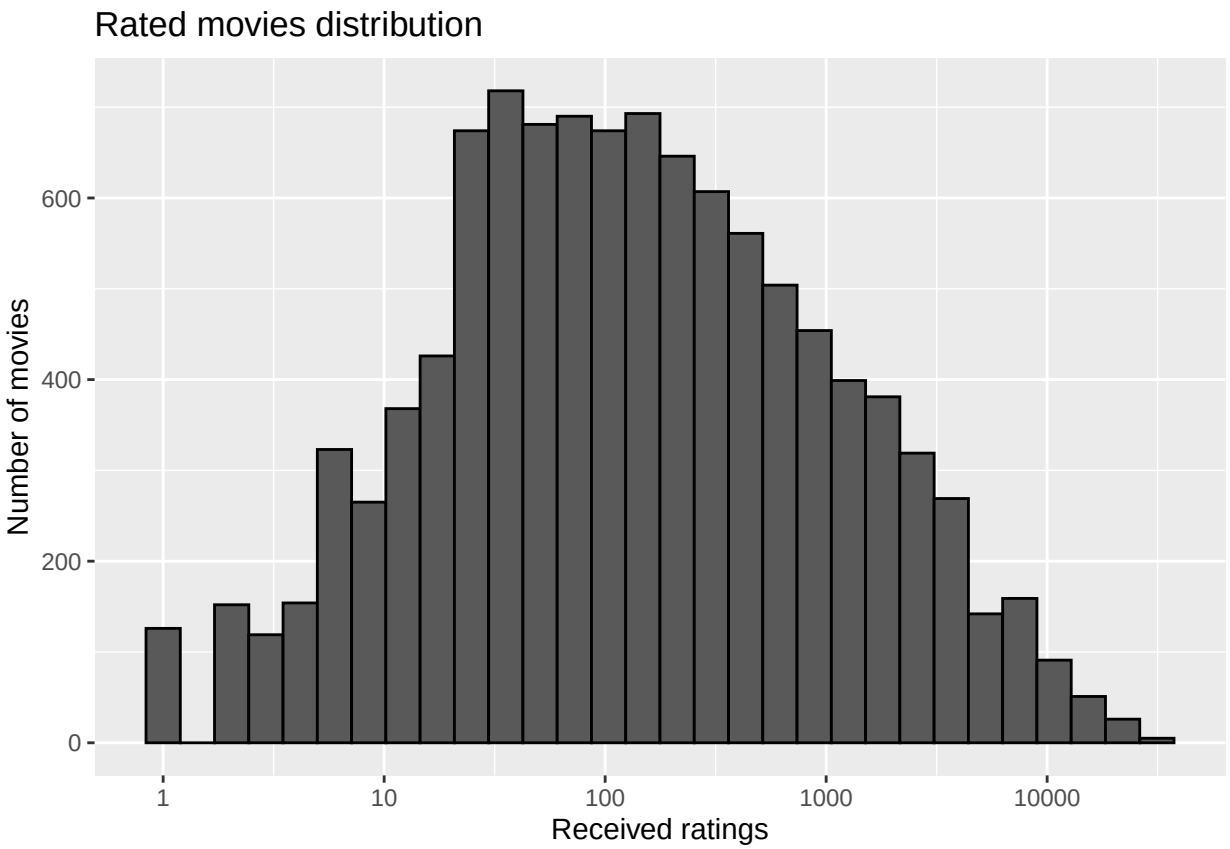
2.2 Data analysis strategies

- Movie bias: some movies are rated more frequently than others (e.g. blockbusters are rated more often).
- User bias: some users have positive where others have negative perspective over a movie due to their own personal liking/disliking.
- Genre popularity over the years: the popularity of the movie genre/category depends strongly on the contemporary issues. So we should also explore the time dependent analysis.
- Rating vs release year: users' mindset may evolve over time. This can also effect the average rating of movies over the years.
- Age of movie vs rating-frequency: users tend to rate a movie as soon as it is available for watching, rather than rating older movies.
- Age of movie vs star-rating: users tend to rate higher older movies which can be referred as "classics".

2.2.1 Movie bias

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10(name = "Received ratings") +
```

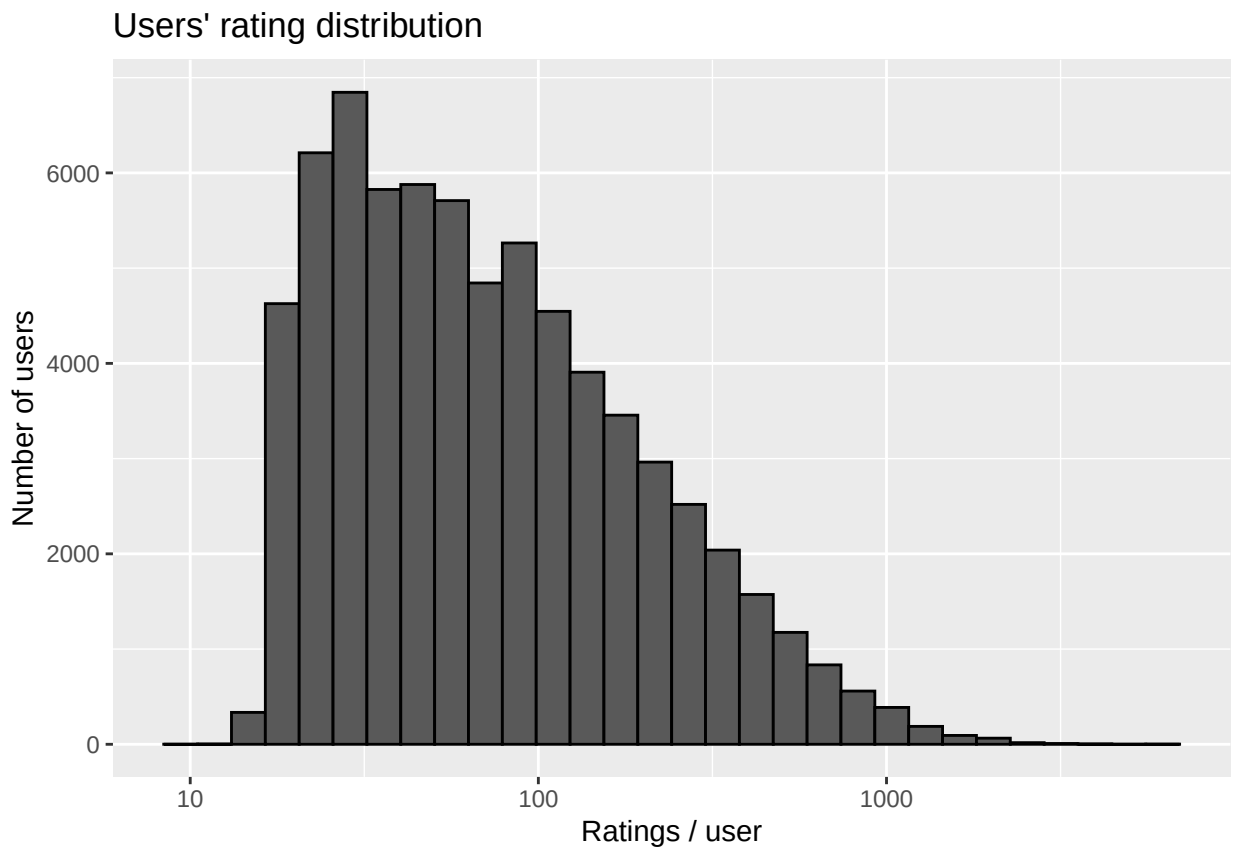
```
  ylab("Number of movies") +
  ggtitle("Rated movies distribution")
```



The histogram shows some movies have been rated very few number of times. These are "noisy" entries, so they should be given lower importance in movie prediction.

2.2.2 User bias

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10(name = "Ratings / user") +
  ylab("Number of users") +
  ggtitle("Users' rating distribution")
```

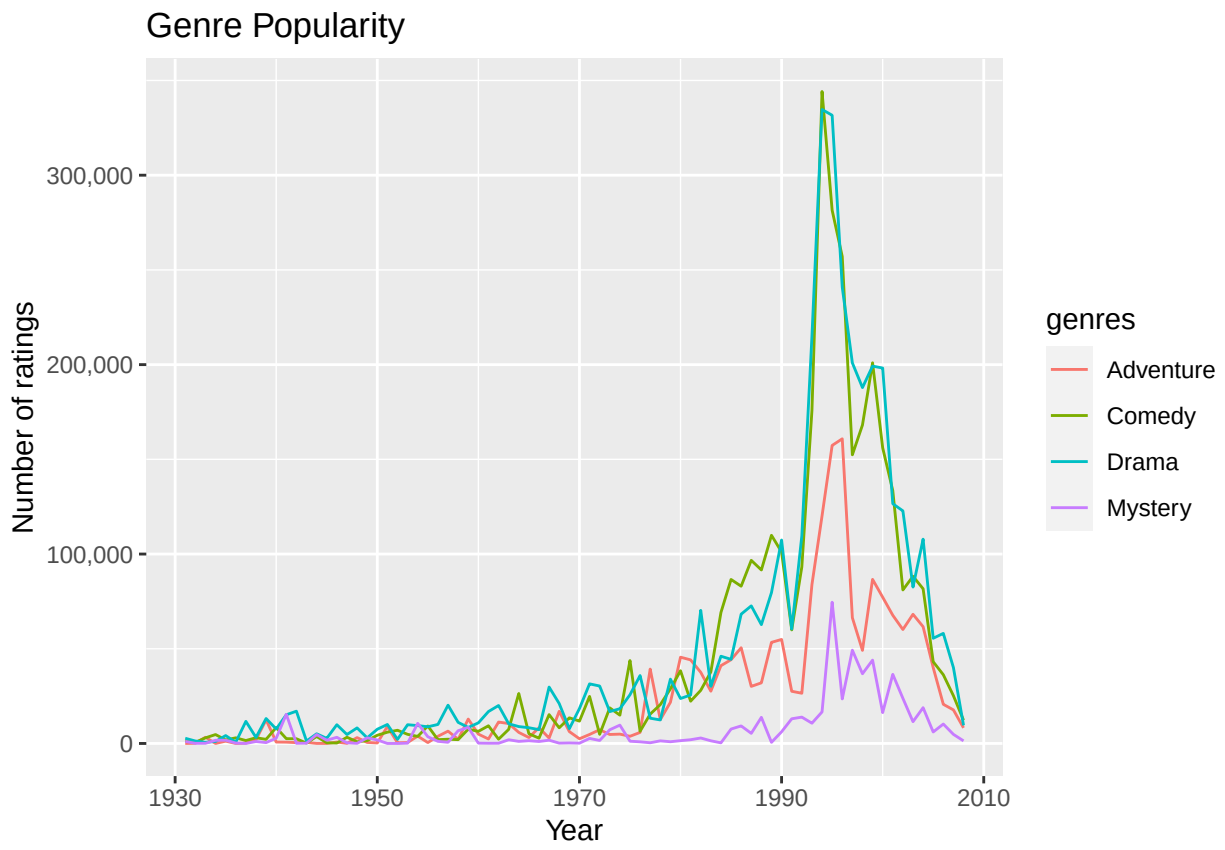


The plot above shows that not every user is equally active. Some users have rated very few movies and their opinion may bias the prediction results.

2.2.3 Genre popularity

```
# create a List of Tibbles (lot) with each distinctive category
lot <- lapply(genre_categories, function(g) {
  edx %>%
    filter(str_detect(genres, g)) %>%  # choose category
    mutate(genres = g) %>%             # rename genre to specific category
    select(movieId, year_release, genres) %>%  # select columns of interest
    group_by(year_release, genres) %>% # group data by year and category
    summarize(number = n())            # number of ratings
})

# transform lot to data frame by binding list element rows
genre_popularity <- do.call(rbind, lapply(lot, data.frame, stringsAsFactors = FALSE)) %>%
  # fill with zero for every release-year without rating in specific category
  complete(year_release = full_seq(year_release, 1), genres, fill = list(number = 0))

# Genre category vs Release year visualization
genre_popularity %>%
  filter(year_release > 1930) %>%
  # only 4 randomly picked categories are chosen for readability
```

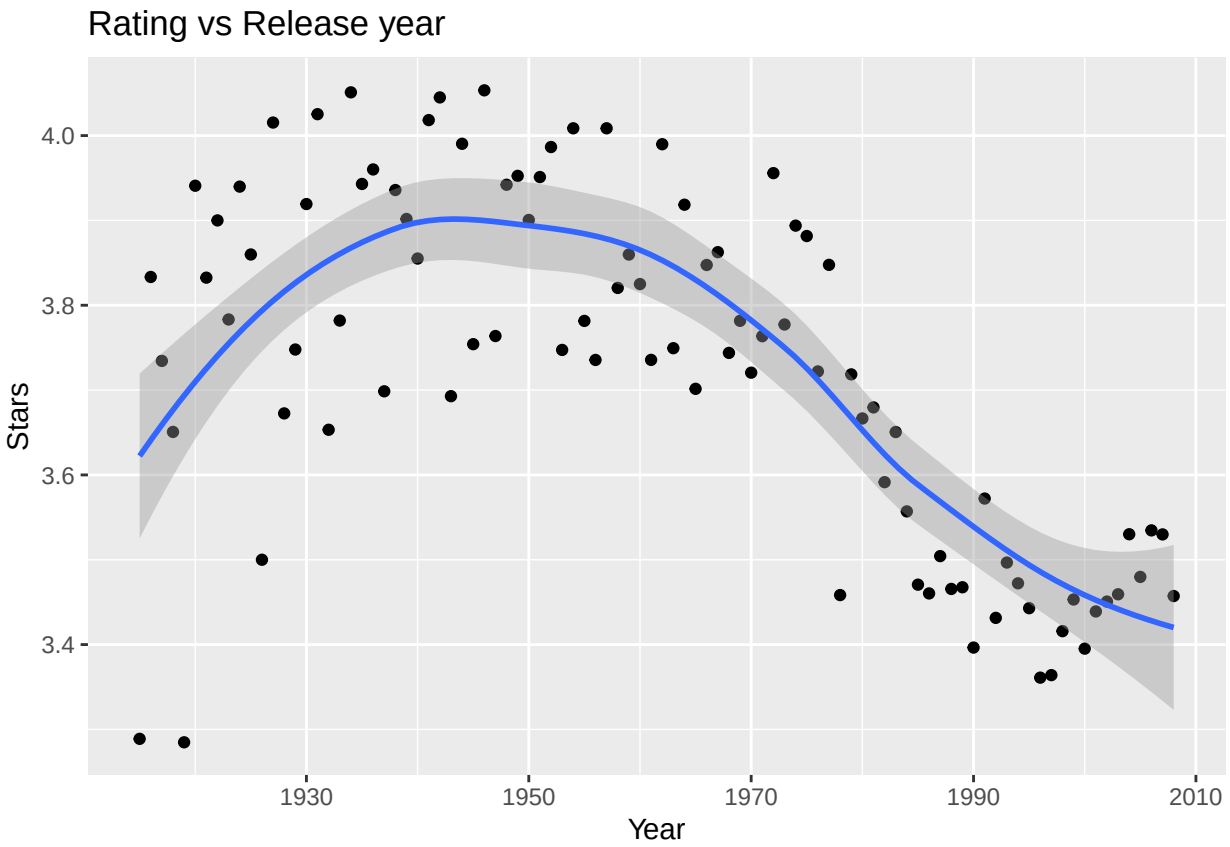
```
  filter(genres %in% c("Comedy", "Adventure", "Drama", "Mystery")) %>%
  ggplot(aes(year_release, number, col = genres)) +
  geom_line() +
  scale_y_continuous(name = "Number of ratings", labels = scales::comma) +
  xlab("Year") +
  ggtitle("Genre Popularity")
```



This plot depicts some genres/categories becoming more popular over others for different periods of time.

2.2.4 Rating vs Release year

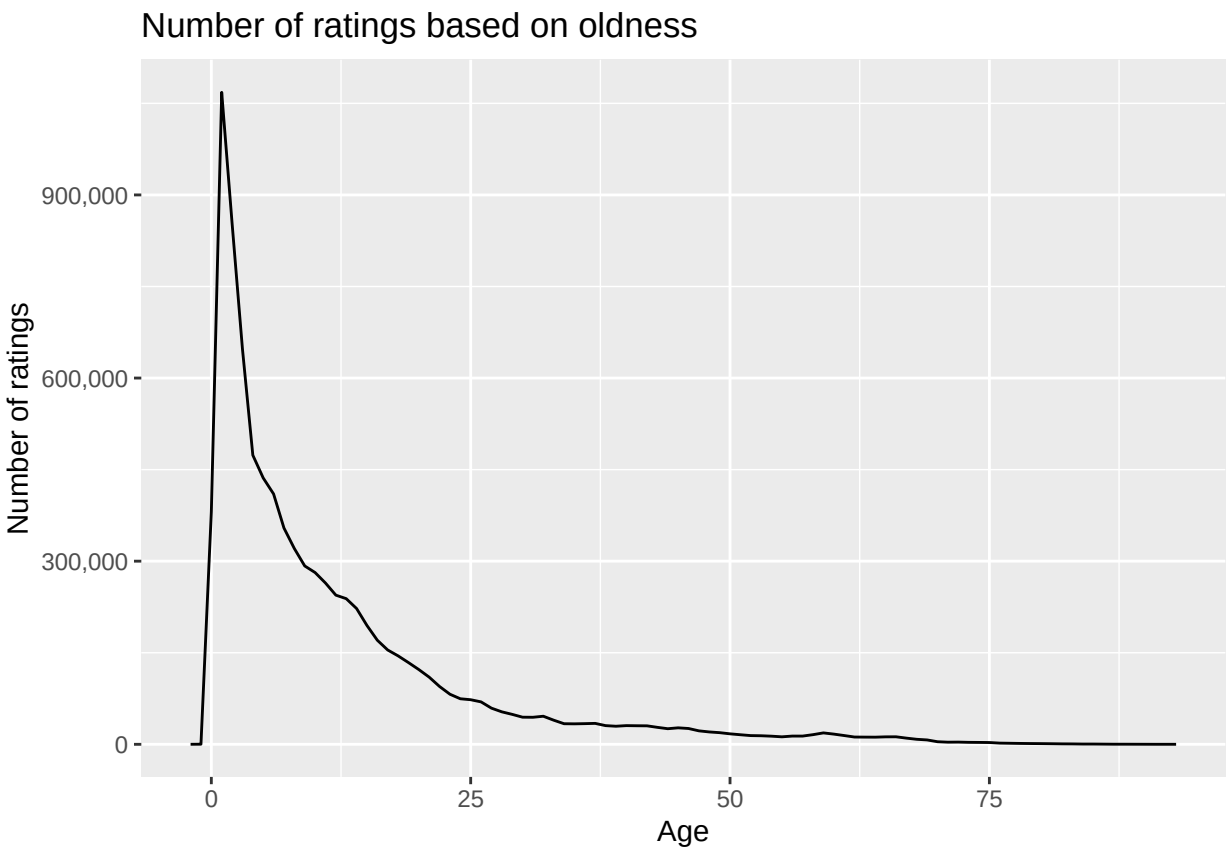
```
edx %>%
  group_by(year_release) %>%
  summarize(stars = mean(rating)) %>%
  ggplot(aes(year_release, stars)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Rating vs Release year") +
  xlab("Year") +
  ylab("Stars")
```



The general trend shows users relatively rate modern movies lower.

2.2.5 Age of movie vs Rating

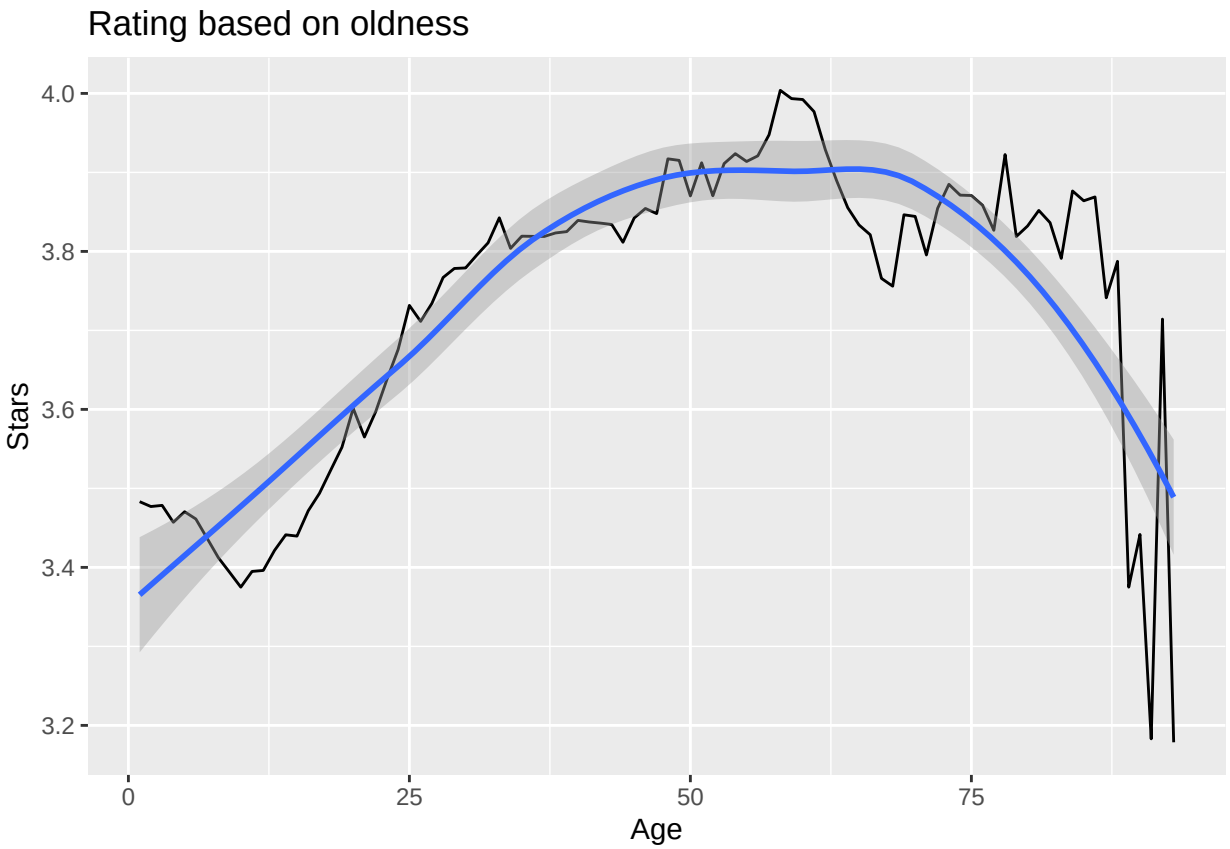
```
edx %>%
  mutate(age = year_rated - year_release) %>% # define age as the age of movie when rated
  group_by(age) %>%
  summarize(number = n()) %>%
  ggplot(aes(age, number)) +
  geom_line() +
  ggtitle("Number of ratings based on oldness") +
  xlab("Age") +
  scale_y_continuous(name = "Number of ratings", labels = scales::comma)
```



According to the plot above, users tend to rate a movie as soon as it is available to watch.

2.2.6 Age of movie vs Star-rating

```
edx %>%
  mutate(age = year_rated - year_release) %>% # define age as the age of movie when rated
  filter(age > 0) %>%
  group_by(age) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(age, avg_rating)) +
  geom_line() +
  geom_smooth() +
  ggtitle("Rating based on oldness") +
  xlab("Age") +
  ylab("Stars")
```



"Classics" tend to get a higher rating than modern movies, although really old movies are not as much preferred.

3 Data analysis: Model Preparation

```
# Define an 'RMSE results' table to store results from various models
rmse_results <- data_frame()
```

```
## 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.
```

3.1 I. Naive model: Average movie rating model

Dataset's mean rating is used to predict the same rating for all movies, regardless of the user and movie. A model based approach assumes the same rating for all 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.

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

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

If we predict all unknown ratings in the validation dataset 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 first RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.061202

This will be the baseline that needs to be improved.

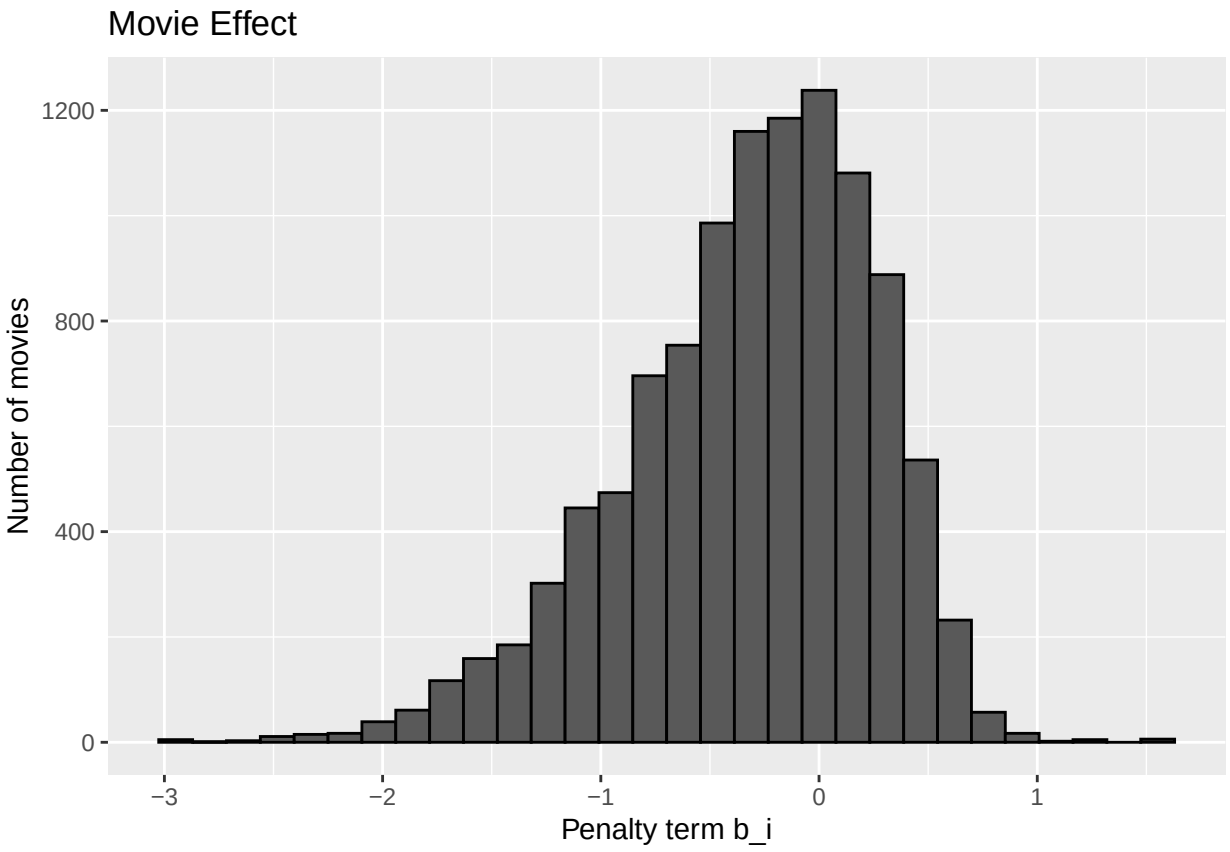
3.2 II. Movie effect model

Different movies are rated differently as of our experience. Higher ratings are mostly linked to popular movies among users and the opposite holds true for unpopular movies.

We compute the estimated deviation of each movies' mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

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

movie_effect %>%
  ggplot(aes(b_i)) +
  geom_histogram(bins = 30, color = "black") +
  ggtitle("Movie Effect") +
  xlab("Penalty term b_i") +
  ylab("Number of movies")
```



As shown above, the histogram is not symmetric and is skewed towards negative rating effect. This is called the penalty term movie effect.

Our prediction improve once we predict using this model:

```
movie_effect_model <- validation %>%
  left_join(movie_effect, by = 'movieId') %>%
  mutate(pred = mu + b_i)

rmse_movie <- RMSE(validation$rating, movie_effect_model$pred)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Movie Effect Model",
                                     RMSE = rmse_movie))
rmse_results %>% knitr::kable()
```

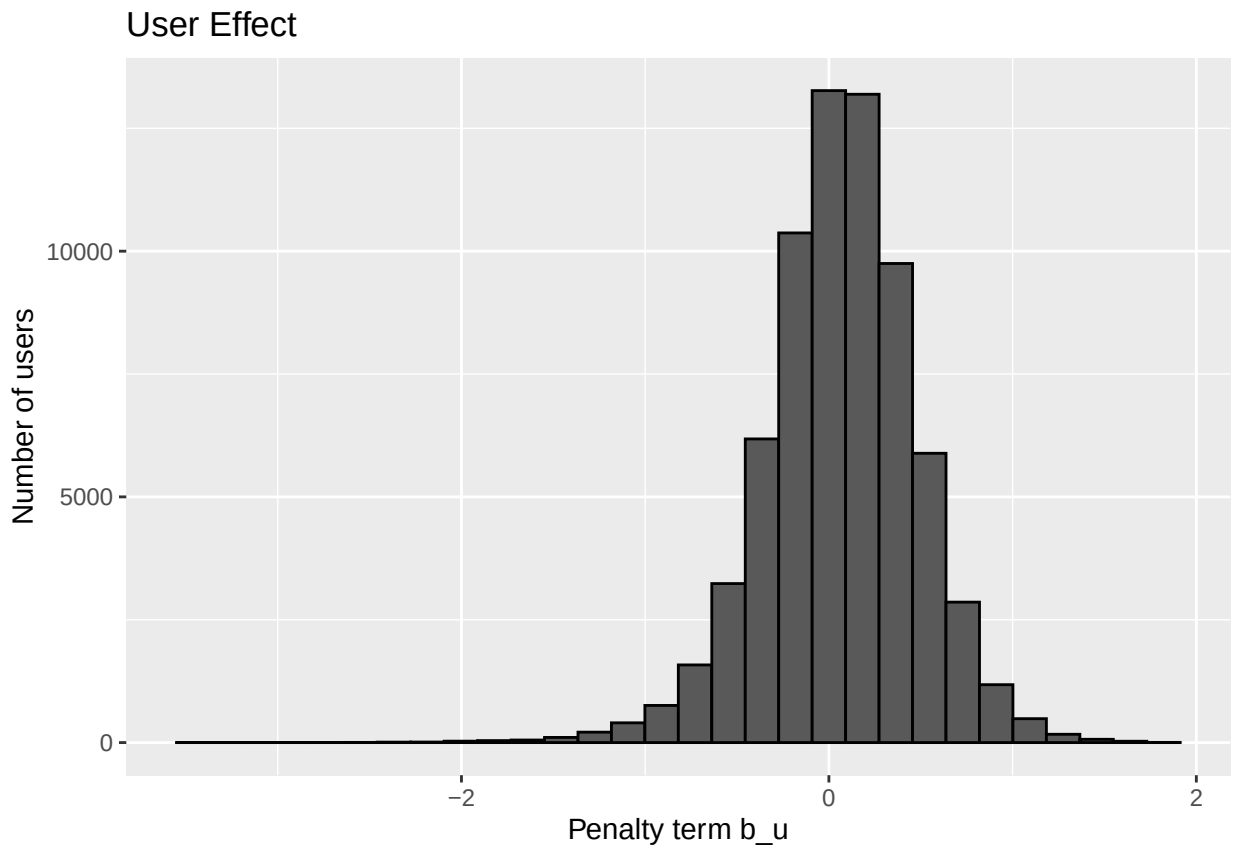
method	RMSE
Average movie rating model	1.0612018
Movie Effect Model	0.9439087

3.3 III. User & Movie effect model

Different users are different in terms of how they rate movies. Some cranky users may rate a good movie lower or some users may be fans of a specific movie / genre or even every movie and will be very generous

in their rating.

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

user_effect %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  ggtitle("User Effect") +
  xlab("Penalty term b_u") +
  ylab("Number of users")
```



This implies that further improvement to our model my be:

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

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_movie_effect_model <- validation %>%
  left_join(movie_effect, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u)

rmse_user_movie <- RMSE(validation$rating, user_movie_effect_model$pred)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "Movie and User Effect Model",
                                     RMSE = rmse_user_movie))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

Our rating predictions further reduced the RMSE.

3.4 IV. Regularized model

We have noticed in our data exploration, some users are more actively participated in movie reviewing. There are also users who have rated very few movies (less than 30 movies). On the other hand, some movies are rated very few times (say 1 or 2). These are basically noisy estimates that we should not trust.

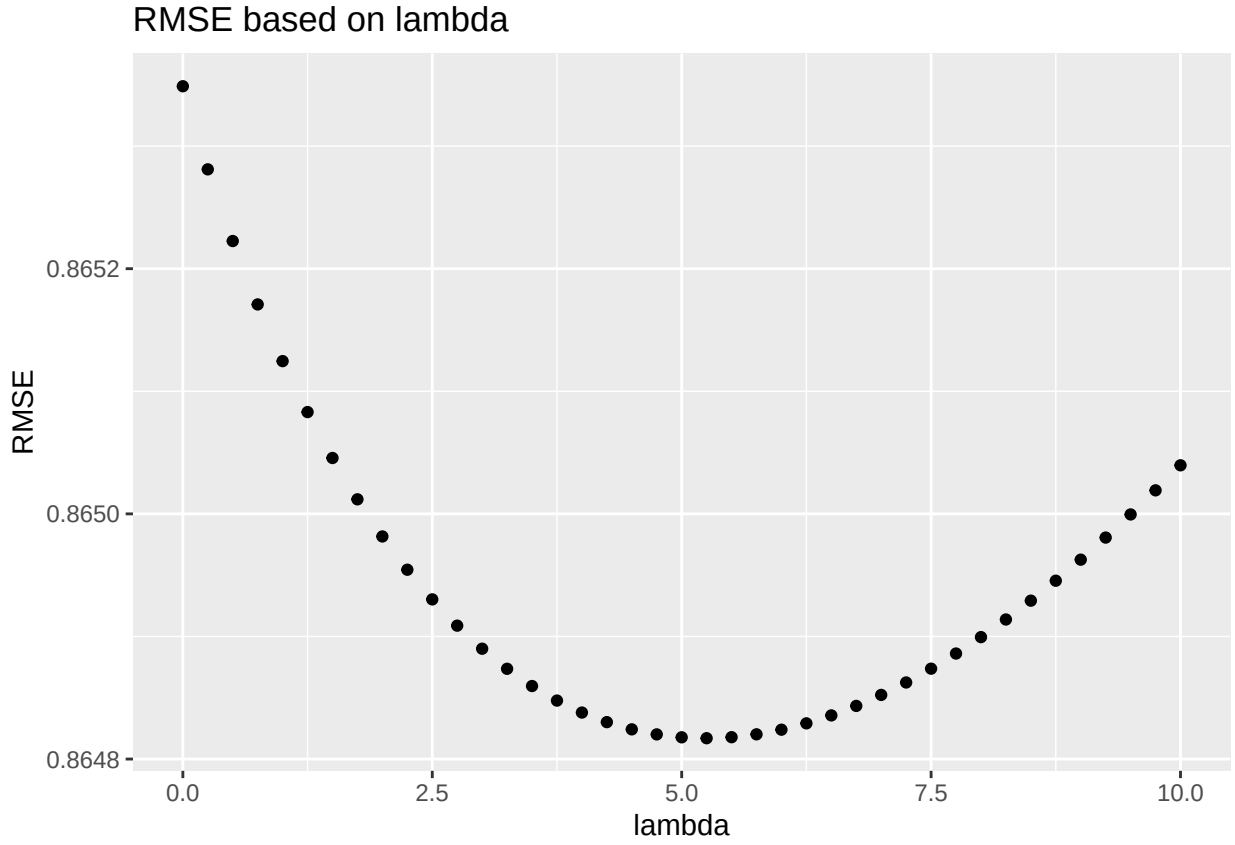
So estimates of b_i and b_u are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these aspects. We should find the value of lambda (that is a tuning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

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  ratings_pred <- validation %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
```

```
  rmse <- RMSE(validation$rating, ratings_pred)
  rmse
})
```

We plot lambdas to choose the optimal one:

```
rmses <- data.frame(rmse = rmses, lambda = lambdas)
rmses %>%
  ggplot(aes(lambda, rmse)) +
  geom_point() +
  ggtitle("RMSE based on lambda") +
  ylab("RMSE")
```




```
lambda <- rmses$lambda[which.min(rmses$rmse)]
lambda
```

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

For the full model, the optimal lambda is: 5.25

The new results will be:

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

method	RMSE
Average movie rating model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User effect model	0.8648170

4 RMSE results

All in all the final RMSE result table look like this:

method	RMSE
Average movie rating model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User effect model	0.8648170

Therefore the lowest RMSE value is the one obtained with the 4th regularized model and is equal to 0.8648170.

Thus we can confirm that the final model for our project is the following:

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

5 Conclusion

The RMSE table shows an improvement of the model over different assumptions. The simplest model "Average rating model" calculates the RMSE more than 1, which means we may miss the rating by one star (not good). Then incorporating "Movie effect" and "Movie and User effect" on the initial model we achieve an improvement by ~11% and ~18% respectively. This is substantial improvement given the simplicity of the model. A deeper insight into the data revealed some data point in the features have large effect on errors. So a regularization model was used to penalize such data points. The final RMSE is 0.8648 with an improvement over 18.5% with respect to the baseline model. This implies we can trust our prediction for movie rating given by a users.

We can affirm to have built a machine learning algorithm to predict movie ratings with MovieLens dataset. The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project. The optimal model characterised by the lowest RMSE value (0.8648170) slightly lower than the initial evaluation criteria (0.86490) given by the goal of the present project. We could also affirm that improvements in the RMSE could be achieved by adding other effect (genre, year, age,..). Other different machine learning models could also improve the results further, but hardware limitations, as the RAM, are a constraint.

6 Appendix - Enviroment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          6.3
## year           2020
## month          02
## day            29
## svn rev        77875
## language       R
## version.string R version 3.6.3 (2020-02-29)
## nickname       Holding the Windsock
```