MovieLens Rating Prediction Project

Alejandro Zarazo

May 15, 2021

Contents

1 Project General Overview **1**
- 1.1 Introduction . . . 2
- 1.2 Objective of the Project . . . 2
- 1.3 Dataset . . . 2
- 1.4 Training and Validation Datasets . . . 3

2 Additional Libraries that Might be Used for the Analysis and Visualization Process **4**

3 Development **4**
- 3.1 Data Analysis . . . 4
- 3.2 The Users' Bias . . . 6

4 Explore the bias of movies. **6**
- 4.1 The Proposed Model . . . 8
  - 4.1.1 I. Model 1: Average Rating . . . 8
  - 4.1.2 II. Model that considers the movie effect . . . 9
  - 4.1.3 III. Model that considers the user-movie effect: . . . 10
  - 4.1.4 IV. Model that incorporates regularization and user-movie effect . . . 11

5 Results **14**

6 Discussion **14**

7 Conclusion **15**

8 Appendice - Operating System Used. **16**

1 Project General Overview

This work was constructed upon the "MovieLens Project" dataset. Its objective is to present a general idea of the elements used in the analysis of a specific situation and evaluate a possible prediction solution. This investigation is organized in the following way:

First, it presents the project's main idea. Second, the studied dataset is prepared, conditioned and organized. Third, an exploratory data analysis is carried out to posteriorly propose an automatic learning algorithm that allows to make predictions of the scoring classification of certain movies, based on the available historical data. Finally, a discussion of the results is made, and several final observations are presented.

1.1 Introduction

The fundamental base of a recommendation system is to use a series of historical data about the scores that users have subjectively reported regarding a specific product or service that they have consumed. It is known that most service providers allow their clients to emit scores that reflect their level of satisfaction with the purchased good. Normally, this type of companies gather large amounts of data, which is the raw material used by data scientists to construct a Machine Learning algorithm capable of predicting "the score that a particular user would give to a particular article".

For this project, a movie recommendation system is created using the 10M version of the MovieLens dataset, gathered by the GroupLens Research group.

1.2 Objective of the Project

As previously discussed, the goal of this project is to develop a Machine Learning algorithm that permits to train, test, and apply this technique to predict the users' recommendations. For this, the given data will be used to predict the movie recommendations in a validation set

The Residual Mean Squared Error, or RMSE, will be used to evaluate the performance of the proposed algorithm. The RMSE is one of the most used measurements to compare the difference between the values predicted by a model and the observed or actual values. Hence, the RMSE is a measure of precision for a model. In general, a low value for RMSE is considered a good measure when compared to high values. The values of RMSE are usually sensible to atypical data values

In this project, the performance of four different models, which will be developed and explained, will be compared using the RMSE, calculated for each one of them, to evaluate their quality. The equation for computing the RMSE can be summarized in the following way:

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

```
options( warn = -1 )
```

Once the different algorithms are evaluated, the technique with the lowest RMSE will be selected and used.

1.3 Dataset

The dataset that will be used can be downloaded in the following link:

https://grouplens.org/datasets/movielens/10m/ http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
# Note: this process could take a couple of minutes for loading required package: tidyverse and package
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")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- read.table(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(movieId),title = as.character(title), g
movielens <- left_join(ratings, movies, by = "movieId")
```

1.4 Training and Validation Datasets

```
# Validation set will be 10% of MovieLens data
set.seed(1)
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)
#View(edx)
```

In the previous code it can be observed that a partition of the original dataset is done, generating a training subset and a validation subset, this last one to validate the accuracy of the proposed models. Also, all unnecessary files are removed from the working directory.

2 Additional Libraries that Might be Used for the Analysis and Visualization Process

```
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

3 Development

3.1 Data Analysis

Let us see which information does the dataset provide. For this, the first rows of the dataset will be displayed (edX):

```
##   userId movieId rating timestamp                          title
## 1      1     122      5 838985046               Boomerang (1992)
## 2      1     185      5 838983525                Net, The (1995)
## 3      1     231      5 838983392           Dumb & Dumber (1994)
## 4      1     292      5 838983421                Outbreak (1995)
## 5      1     316      5 838983392                Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                            genres
## 1                  Comedy|Romance
## 2           Action|Crime|Thriller
## 3                          Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

This sub dataset contains these six variables:

- userID
- movieID
- rating
- timestamp
- title
- genres

Each row represents a unique rating given by a user to a specific movie.

We check for the presence of missing data (NA)

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
```

```
##  Mode  :character   Mode  :character
##
##
##
```

The number of unique movie titles and users in edx is the following:

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

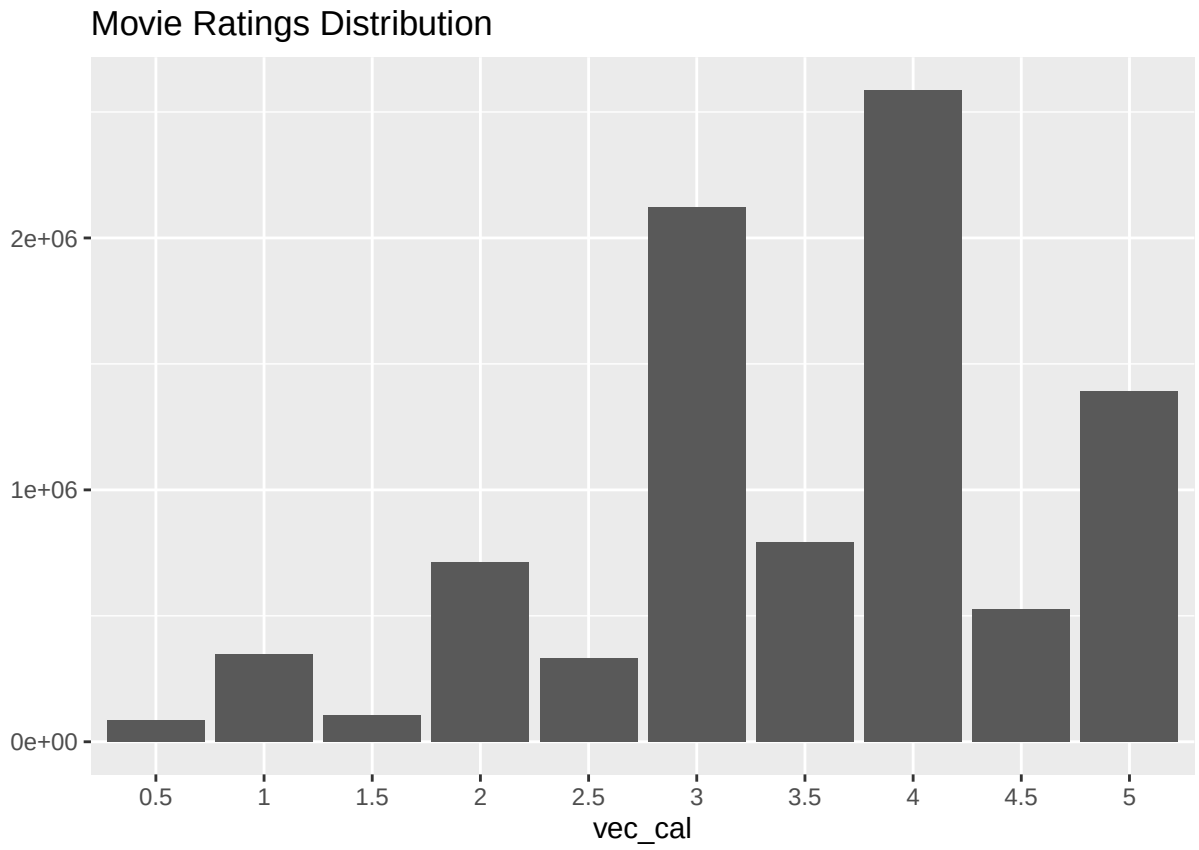
There are nearly 70.000 unique users and 10.700 different movies.

Let us see the types of ratings that are registered within the datasets.

```
##  num [1:10] 5 3 2 4.5 3.5 4 1 1.5 2.5 0.5
```

10 different types of user movie ratings can be seen. To check how these are distributed, a histogram will be used, as follows:

```
vec_cal <- factor(edx$rating)
qplot(vec_cal) +
ggtitle("Movie Ratings Distribution")
```



The ratings distribution presented in the previous graph indicates that users tend to give movie ratings between 3 and 4. Moreover, it is observed that a low percentage of movies has received a score inferior to the category 1.

It is important to note that some movies are rated more frequently than other, for example, blockbusters have a higher number of ratings. This is the reason why we must use strategies that permit to put all data values in similar conditions, to avoid any bias that may impair or affect the performance of the model to be

developed.

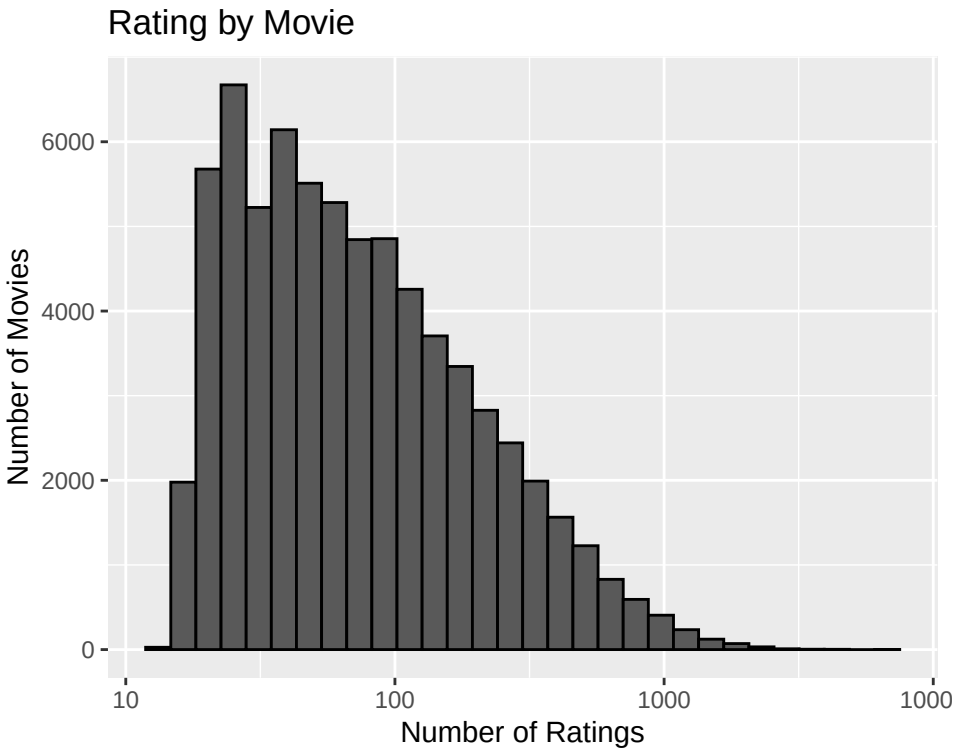
Some Strategies to Consider

- Verify the bias in movie ratings.
- Verify the bias in the rating given by users.
- Verify the popularity of the genre in the time dimension.
- Evaluate the rating in function of the movie release dates.

The idea will be to apply regularization and a penalty factor to reduce the error by appropriately adjusting a function to the given training set and avoid overfitting.

3.2 The Users' Bias

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



4 Explore the bias of movies.

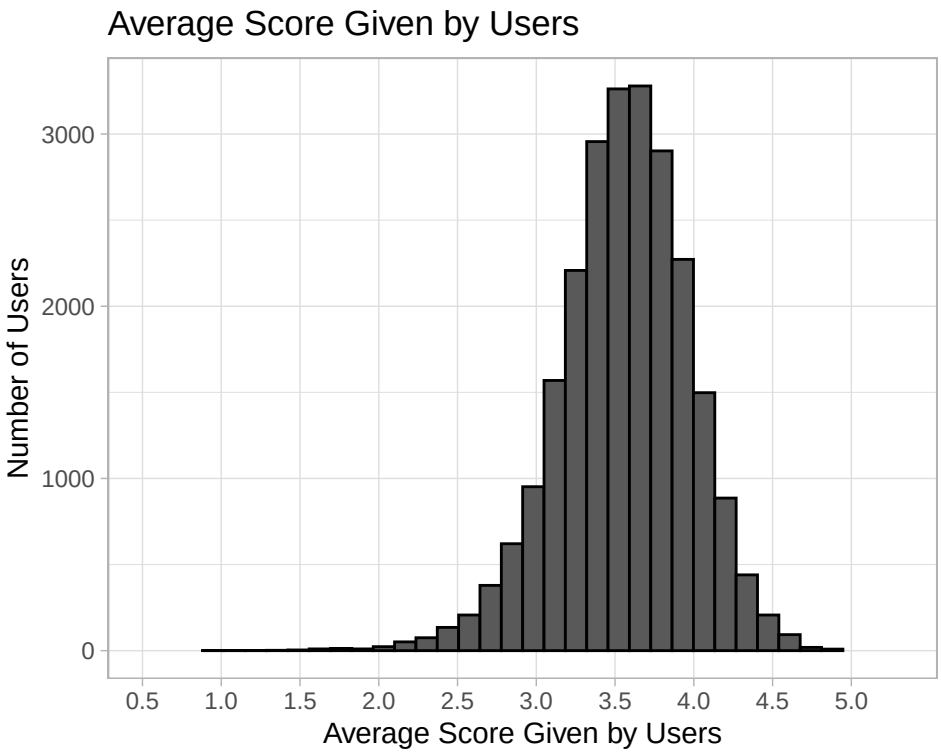
```
tabla <- edx %>%
            group_by(movieId) %>%
            summarize(count = n()) %>%
```

```
            filter(count == 1) %>%
            left_join(edx, by = "movieId") %>%
            group_by(title) %>%
            summarize(rating = rating, n_rating = count) %>%
            slice(1:20) %>%
            knitr::kable()
tabla
```

title	rating	n_rating
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
5 Centimeters per Second (ByÃ´soku 5 senchimÃªtoru) (2007)	3.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di...) (1971)	1.5	1
Africa addio (1966)	3.0	1
Archangel (1990)	2.5	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bell Boy, The (1918)	4.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Boys Life 4: Four Play (2003)	3.0	1
Brothers of the Head (2005)	2.5	1
CaÃ³tica Ana (2007)	4.5	1
Chapayev (1934)	1.5	1

Most users have rated between 30 and 100 movies, so it is necessary to include a penalty factor. The following graph shows the users that have rated less than 100 movies.

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Average Score Given by Users") +
  ylab("Number of Users") +
  ggtitle("Average Score Given by Users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



4.1 The Proposed Model

The value of RMSE may be considered as the typical error that in which we incur when predicting a rating for a determined movie. If the RMSE is greater than 1, it means that the typical error could be higher than 1 star, which is far from being a satisfactory result.

As it was mentioned at the beginning of this document, a function for computing the RMSE can be defined as follows:

```
RMSE <- function(recomendaciones_reales, recomendaciones_predichas){
  sqrt(mean((recomendaciones_predichas - recomendaciones_reales)^2))
}
```

4.1.1 I. Model 1: Average Rating

The first proposal will be an extremely simple model, based on the average of the movie ratings. This means that the model predicts the same rating for all movies, independently of the user giving the score. Therefore, the expected rating would be between 3 and 4. In general, the model could be expressed as follows:

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

This model tries to explain all the differences between movie ratings by a random variation. The relevant terms of this model would be $\epsilon_{u,j}$ corresponding with the sample of the error independent of the same distribution centered in 0, and μ being the "true" rating for all movies.

Computing the value of μ

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

```
## [1] 3.512464
```

First naive RMSE By applying the model with μ o mu, the first RMSE would be obtained:

```r
RMSE_naive <- RMSE(validation$rating, mu)
RMSE_naive
```

```
## [1] 1.060651
```

A table is created to store this RMSE and the RMSE for the additional models to be proposed.

```r
tabla_rmse <- data.frame(method = "Average Rating Model", RMSE = RMSE_naive)
tabla_rmse %>% knitr::kable()
```

method	RMSE
Average Rating Model	1.060651

4.1.2 II. Model that considers the movie effect

To improve the previous naïve model, I will incorporate some of the insights obtained during the exploratory data analysis. For this, I will focus on the fact that, by experience, it is known that some movies generally have higher ratings than other. These ratings are mainly related to movies that were blockbusters upon release, in contrast to those that were not

Therefore, I propose to compute the bias for the ratings given by the users. For this, I determine the estimated deviation for the mean rating for each movie. This parameter will be called 'sp', for each movie sp_j, which represents the average rating for movie j. In this way, the second proposed model would be expressed as follows:

$$Y_{u,j} = \mu + sp_j + \epsilon_{u,j}$$

We compute the parameter s_j

```r
movie_med <- edx %>%
      group_by(movieId) %>%
      summarize(sp_j = mean(rating - mu))
```

Penalty for the Movie Effect

If we evaluate this model using the penalty of the movie effect, we would proceed as follows:

```r
pred_cali_modelo2 <- validation %>%
  left_join(movie_med, by='movieId') %>%
  mutate(predictor = mu + sp_j)
rmse_modelo_2 <- RMSE(validation$rating,pred_cali_modelo2$predictor)
tabla_rmse <- bind_rows(tabla_rmse,
                        data.frame(method="Movie Effect Model",
                                   RMSE = rmse_modelo_2))
tabla_rmse %>% knitr::kable()
```

method	RMSE
Average Rating Model	1.0606506
Movie Effect Model	0.9437046

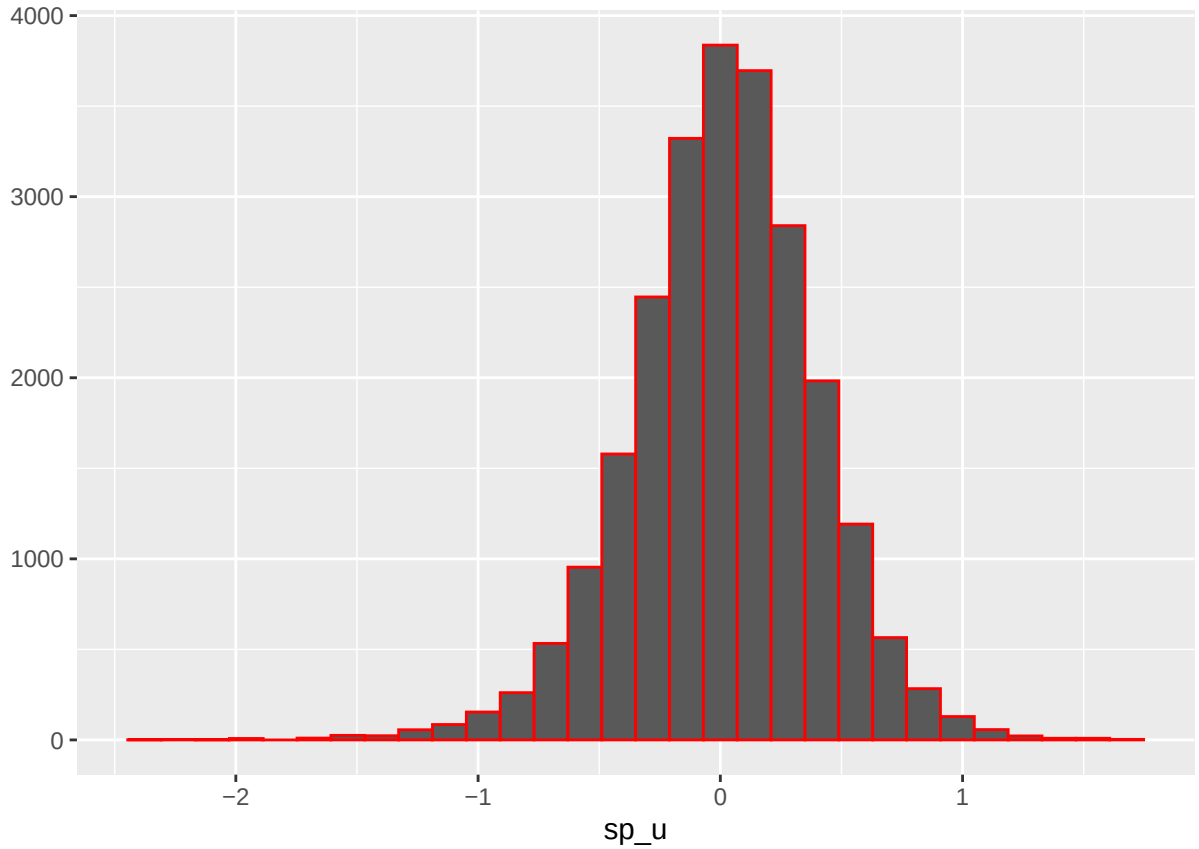
method	RMSE

The error has decreased by approximately 5 percentage points. However, this model has not considered the individual effect of user ratings.

4.1.3 III. Model that considers the user-movie effect:

Based on the fact that users affect ratings in a positive or negative way, a model that penalizes this effect is proposed. The following graph confirms this:

```
cliente_media<- edx %>%
  left_join(movie_med, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(sp_u = mean(rating - mu - sp_j))

cliente_media%>% qplot(sp_u, geom ="histogram", bins = 30, data = ., color = I("red"))
```



An improvement to the previous model consists in introducing a penalty for the user-movie effect. In a general way, the model would be as follows:

$$Y_{u,j} = \mu + sp_j + sp_u + \epsilon_{u,j}$$

Where sp_u, corresponds to the parameter that considers the rating effect given by a specific user to a determined movie. In this case, the estimated computation for sp_u could be expressed as $Y_{u,j} - \mu - sp_j$.

```
cliente_med <- edx %>%
  left_join(movie_med, by='movieId') %>%
  group_by(userId) %>%
  summarize(sp_u = mean(rating - mu - sp_j))
```

Penalty for the user-movie effect

```
predic_cal_modelo_3 <- validation%>%
  left_join(movie_med, by='movieId') %>%
  left_join(cliente_med, by='userId') %>%
  mutate(predictor = mu + sp_j + sp_u) %>%
  pull(predictor)
modelo_3_rmse <- RMSE(predic_cal_modelo_3, validation$rating)
tabla_rmse <- bind_rows(tabla_rmse,
                        data.frame(method="User-Movie Effect Model",
                                   RMSE = modelo_3_rmse))
tabla_rmse %>% knitr::kable()
```

method	RMSE
Average Rating Model	1.0606506
Movie Effect Model	0.9437046
User-Movie Effect Model	0.8655329

The RMSE keeps decreasing with the penalties introduced in this model, but there is room for further improvement. Now, the bias introduced by the rating of movies by few users will be considered, since higher uncertainty arises when this happens

In the following approach, the concept of regularization will be introduced, which permits to penalize big estimations that come from small sample sizes.

4.1.4 IV. Model that incorporates regularization and user-movie effect

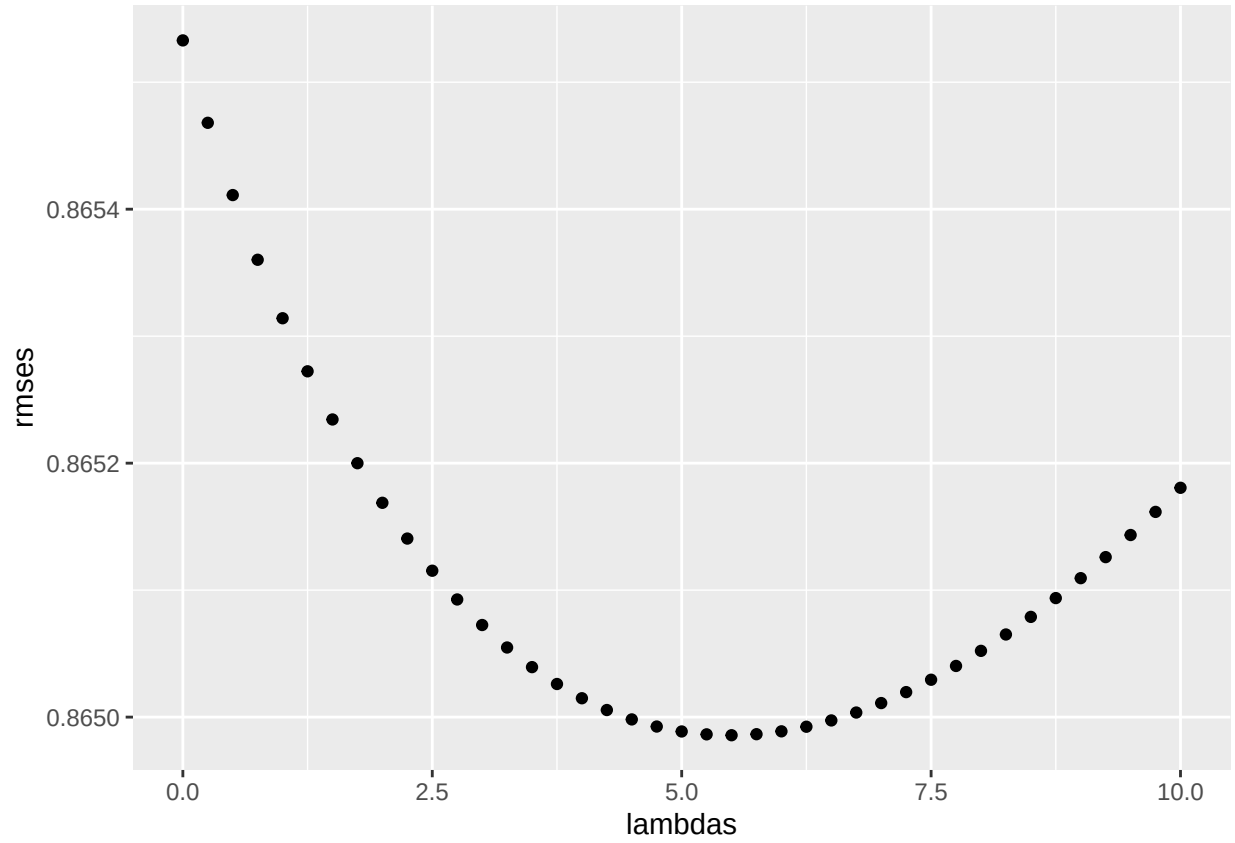
The facts that there are movies with very few ratings and/or that there are some users that rated a few number of movies may affect the performance of the model.

The objective now is to find a value λ to make an adjustment to the model to be able to decrease the error in movie ratings.

```
# lambda is the parameter to be configured
# We apply cross-validation to choose lambda
# Executing this code could take several minutes
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

  sp_u <- edx %>%
    left_join(sp_j, by="movieId") %>%
    group_by(userId) %>%
```

```
    summarize(sp_u = sum(rating - sp_j - mu)/(n()+l))

  predic_cal <-
    validation %>%
    left_join(sp_j, by = "movieId") %>%
    left_join(sp_u, by = "userId") %>%
    mutate(predictor = mu + sp_j + sp_u) %>%
    pull(predictor)

  return(RMSE(predic_cal, validation$rating))
})
```

```
# Plot rmses vs lambdas to find the optimal lamda
qplot(lambdas, rmses)
```



The graph shows that the best value for lambda is close to 5.5, but we can know the exact value applying the following code:

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

```
## [1] 5.5
```

Let us test the model using the obtained parameter lamda.

Applying regularization

```
# Compute regularized estimates of b_i using lambda
regula_media_peli <- edx %>%
```

```
  group_by(movieId) %>%
  summarize(sp_j = sum(rating - mu)/(n()+lambda), n_i = n())

# Compute regularized estimates of b_u using lambda
regula_media_cliente <- edx %>%
  left_join(regula_media_peli, by='movieId') %>%
  group_by(userId) %>%
  summarize(sp_u = sum(rating - mu - sp_j)/(n()+lambda), n_u = n())

# Predict ratings
regula_predic_cali <- validation %>%
  left_join(regula_media_peli, by='movieId') %>%
  left_join(regula_media_cliente, by='userId') %>%
  mutate(predictor = mu + sp_j + sp_u) %>%
  pull(predictor)

# Test and save results
rmse_modelo_4 <- RMSE(regula_predic_cali,validation$rating)
tabla_rmse <- bind_rows(tabla_rmse,
                        data.frame(method="Regularization User-Movie Effect",
                                   RMSE = rmse_modelo_4))
tabla_rmse %>% knitr::kable()
```

method	RMSE
Average Rating Model	1.0606506
Movie Effect Model	0.9437046
User-Movie Effect Model	0.8655329
Regularization User-Movie Effect	0.8649857

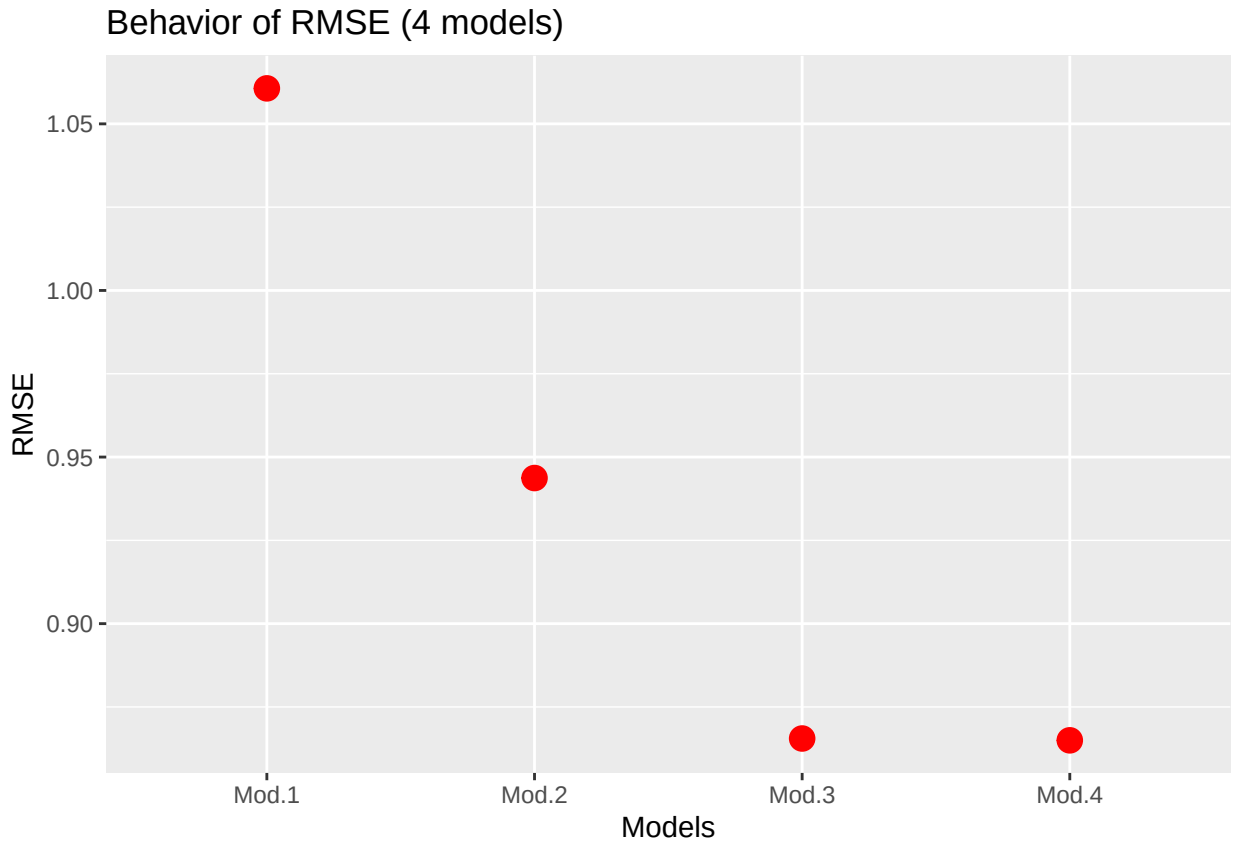
5 Results

The following table presents the values of RMSE for the different proposed models:

method	RMSE
Average Rating Model	1.0606506
Movie Effect Model	0.9437046
User-Movie Effect Model	0.8655329
Regularization User-Movie Effect	0.8649857

The last proposed model, Regularization User-Movie Effect, gives an RMSE of 0.8648170, which is far lower than the one of the initial (naïve) model, about 18 percentage points. The following graph shows the behavior of the RMSE along the proposed models.

```
qplot(c('Mod.1','Mod.2','Mod.3','Mod.4'), tabla_rmse$RMSE,main = 'Behavior of RMSE (4 models)',ylab = '
```



6 Discussion

The model that better behaves with respect to the reduction of the rating error can be expressed as follows:

$$Y_{u,j} = \mu + sp_j + sp_u + \epsilon_{u,j}$$

For the final model, the concept of regularization has been introduced to optimize the model, through a computing process for the best value of *lambda*.

7 Conclusion

We have studied the gaps between four prediction models with the goal of developing a Machine Learning algorithm to predict the movie ratings, based on the given dataset. Finally, an improvement of more than 18 percentage points was achieved with respect to the first proposed model, a quite naive one.

The RMSE table shows an improvement of the model over different approaches. The simplest model, which uses only the average rating has an RMSE greater than 1, meaning that a mistake of about one star could be made when predicting a rating. Based on that initial approach, several parameters were tuned according to the biases that could be found in a movie recommendation system. These tuned parameters were then integrated to two more robust models, this allowing to achieve substantial improvements in the RMSE with respect to the first model. Finally, a deeper approach was used, now applying regularization to penalize the bias present in the previous models.

8 Appendice - Operating System Used.

```
print("SO:")
```

```
## [1] "SO:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.5
## year           2021
## month          03
## day            31
## svn rev        80133
## language       R
## version.string R version 4.0.5 (2021-03-31)
## nickname       Shake and Throw
```