HarvardX. Module 9: Data Science
MovieLens Rating Prediction Project

Adelaida Fernández Sanz

8/16/2020

Introduction and Aim of the Project

This project is part of the HarvardX Data Science Proffesional Certification: Capstone- MovieLens Project. The MovieLens Project consists in generating a recommendation system based on a given DB.Recommendation systems usually use ratings that users give to items to make specific recommendations. Netflix inspired this project, The Netflix Prize was, as said in Wikipedia:
"an 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, i.e. without the users or the films being identified except by numbers assigned for the contest".

This project aims is to create a movie recommendation system using the 10M version of MovieLens dataset provided by the edx HarvardX course: http://grouplens.org/datasets/movielens/10m/ Training a machine learning algorithm that will predict user ratings taking into account the features and inputs provided in the previous mentioned dataset. The data set will be splited into training dataset [90%] (called: edx) and validation dataset [10%] (called: validation).

For the evaluation of the Machine learing algorithm performance we will use the RMSE (the Root Mean Square Error). The RMSE computes the differences between the model predicted values and the observed values. T

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

Therefore, the lower the RMSE, the better. RMSE is sensitive to outliers. So large erros will get a noisy effect in our prediction. As said by James Moody (2019):
"The random noise here could be anything that our model does not capture (e.g., unknown variables that might influence the observed values). If the noise is small, as estimated by RMSE, this generally means our model is good at predicting our observed data, and if RMSE is large, this generally means our model is failing to account for important features underlying our data".

Analysis

The project analysis was executed by the folowing steps: 1. Split the data (Already done by edx-Harvard)
2. Explore the data (features, distributions ect)
3. Create the RMSE function
4. Creating the ML algoritms and apply the RMSE function to them.

0. Download packages needed

```r
###Packages Download
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")

library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
```

1. Split the Dataset

Dataset downloading and partition for ML: edx set, validation set MovieLens 10M dataset: https://grouplens.org/datasets/movielens/10m/ http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

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

# Validation set will be 10% of MovieLens data
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,]

# UserId and movieId must be in both validation set and 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)
```

2. Dataset Exploration

```
## 'data.frame':    50 obs. of  2 variables:
```

```
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

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

As seen in the basic explorarion of the dataset, modifications of both features: Name of the movie and genres are needed: * Separate the Year from the movie name and create another colum to store it.
* Create a new row for each individual movie genre.

```r
#Year: We substract the last 4 strings of the movie title, omiting the "()",
#and then transform it into numeric form:
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
#Genre: We generete new rows for each genre identify per movie.
edx<- edx  %>% separate_rows(genres, sep = "\\|")
```

Let's take a look to the transformed dataset to make sure we did the correct changes:

```
## # A tibble: 6 x 7
##   userId movieId rating timestamp title            genres    year
##    <int>   <dbl>  <dbl>     <int> <chr>            <chr>    <dbl>
## 1      1     122      5 838985046 Boomerang (1992) Comedy    1992
## 2      1     122      5 838985046 Boomerang (1992) Romance   1992
## 3      1     185      5 838983525 Net, The (1995)  Action    1995
## 4      1     185      5 838983525 Net, The (1995)  Crime     1995
```

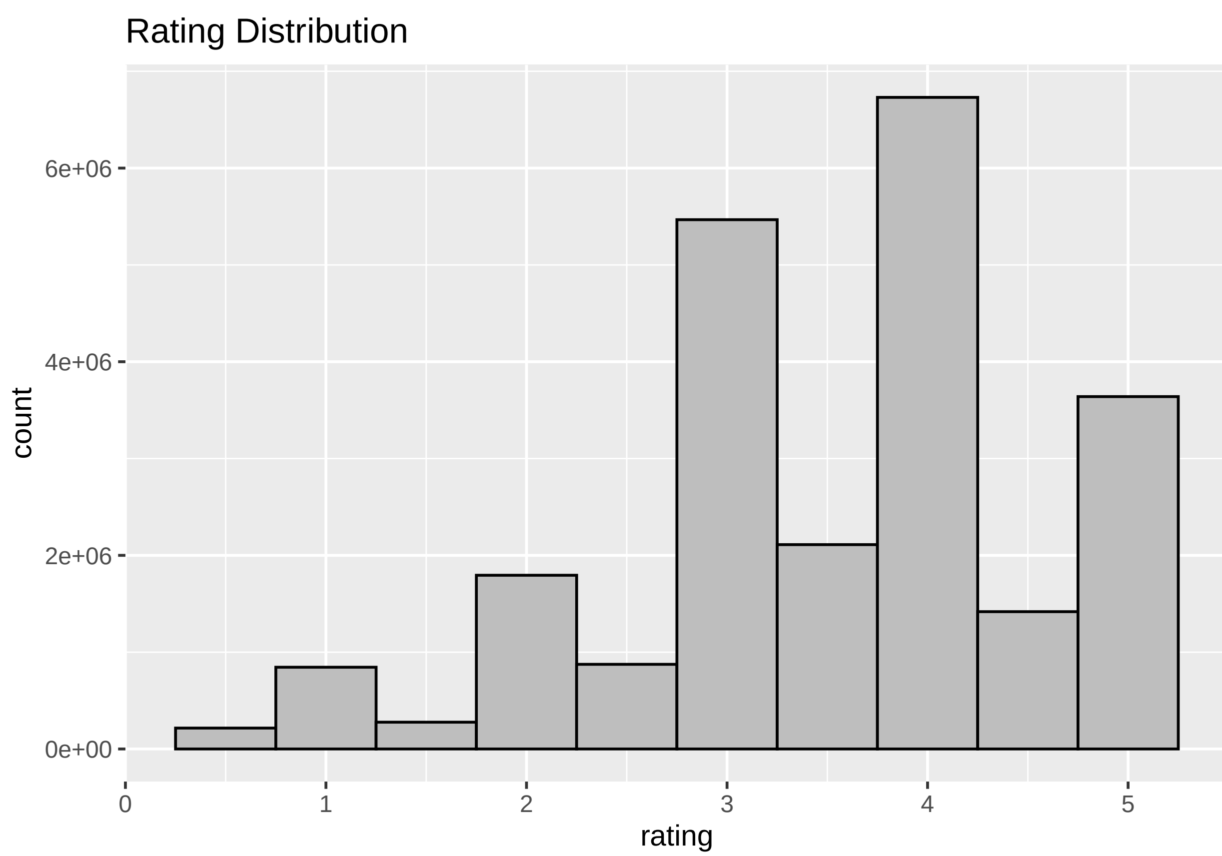
```
## 5      1     185      5 838983525 Net, The (1995)  Thriller  1995
## 6      1     292      5 838983421 Outbreak (1995)  Action    1995
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18140   1st Qu.:  616   1st Qu.:3.000   1st Qu.:9.472e+08
##  Median :35784   Median : 1748   Median :4.000   Median :1.042e+09
##  Mean   :35886   Mean   : 4277   Mean   :3.527   Mean   :1.035e+09
##  3rd Qu.:53638   3rd Qu.: 3635   3rd Qu.:4.000   3rd Qu.:1.131e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title               genres               year
##  Length:23371423    Length:23371423    Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1995
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

Now, let's get some insigths of the transformed dataset:

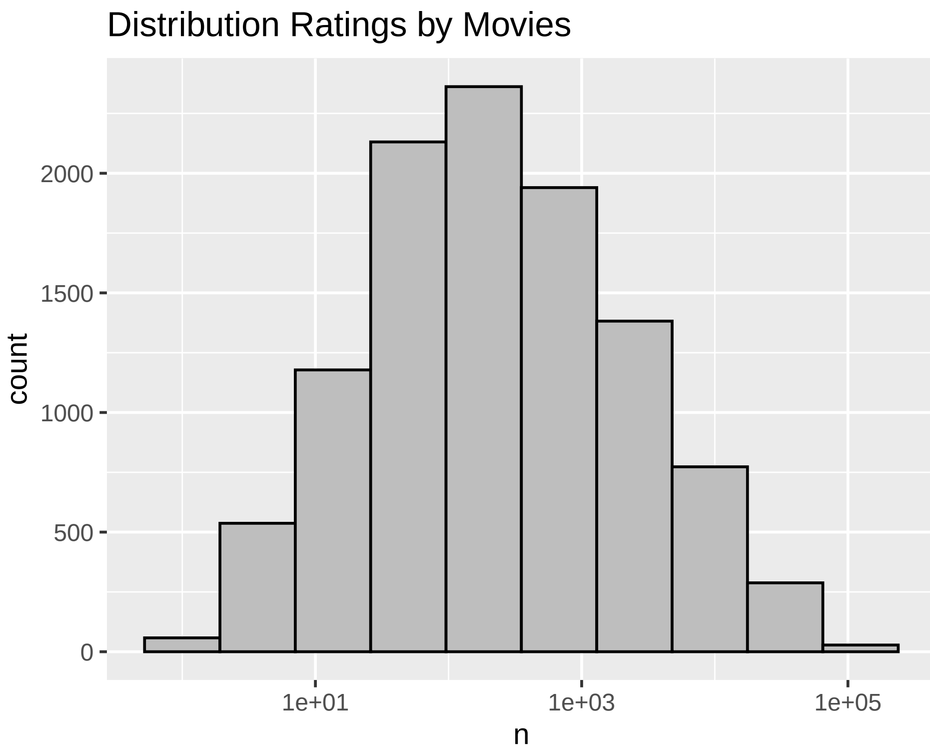
```
## # A tibble: 1 x 2
##   n_users n_movies
##     <int>    <int>
## 1   69878    10677
```

Next, lets plot the distribution of the data set:



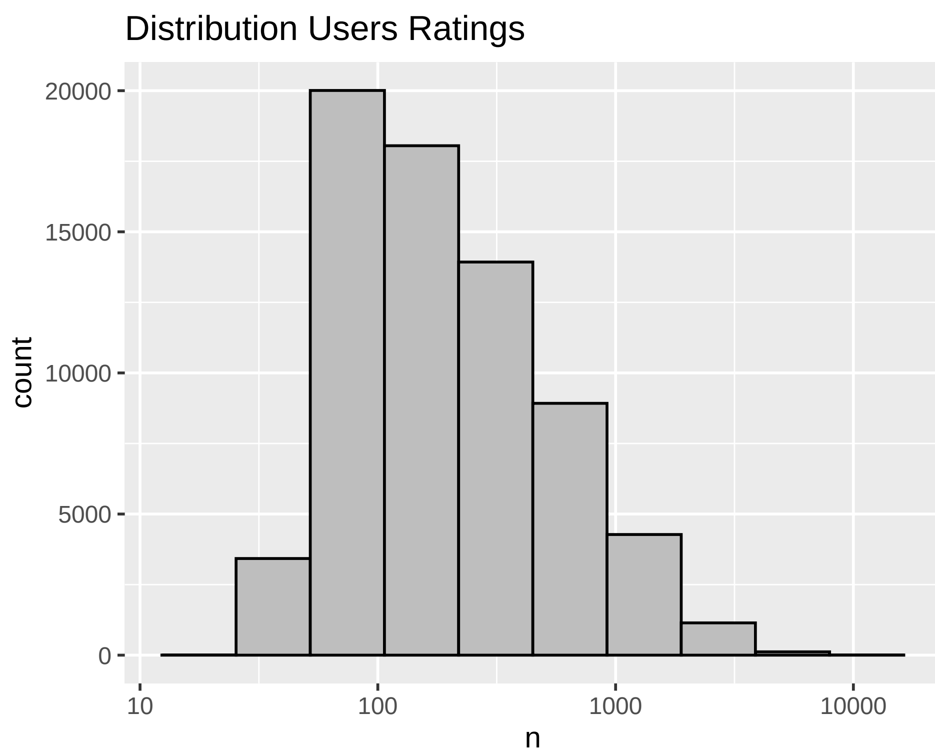
We can see that the ratings given are always between 3 and 4, and the most common rating is 4.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



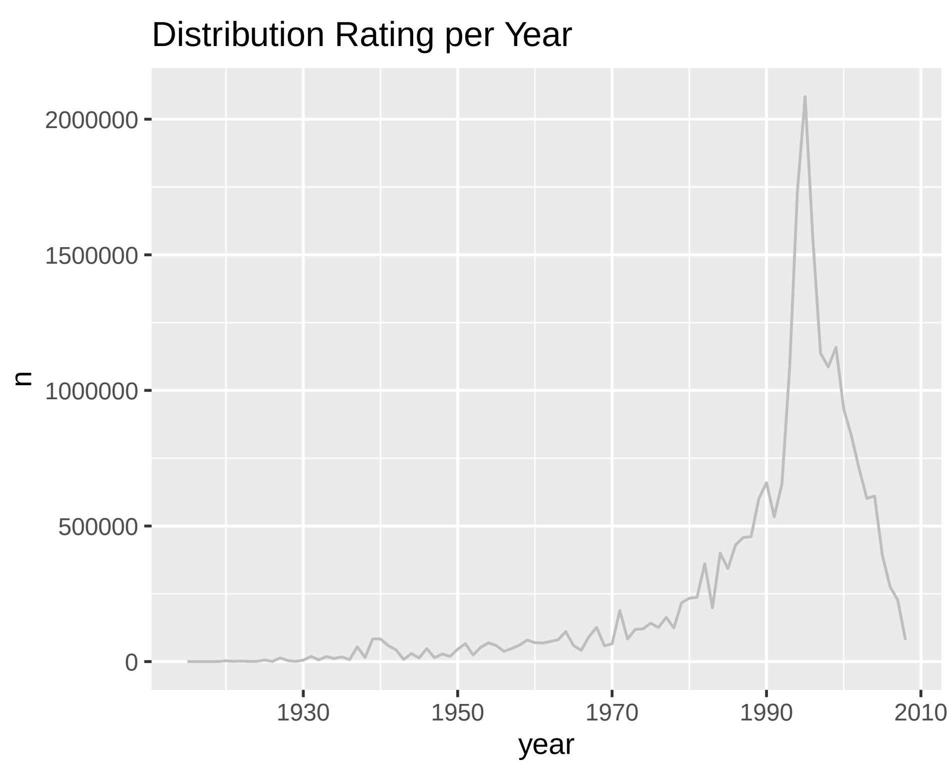
This histogram represent the distribution of the ratings by movies.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



We can observe in the first plots that the ratings are not normally distributed: Most of the users only reated between 20 and 100 movies .

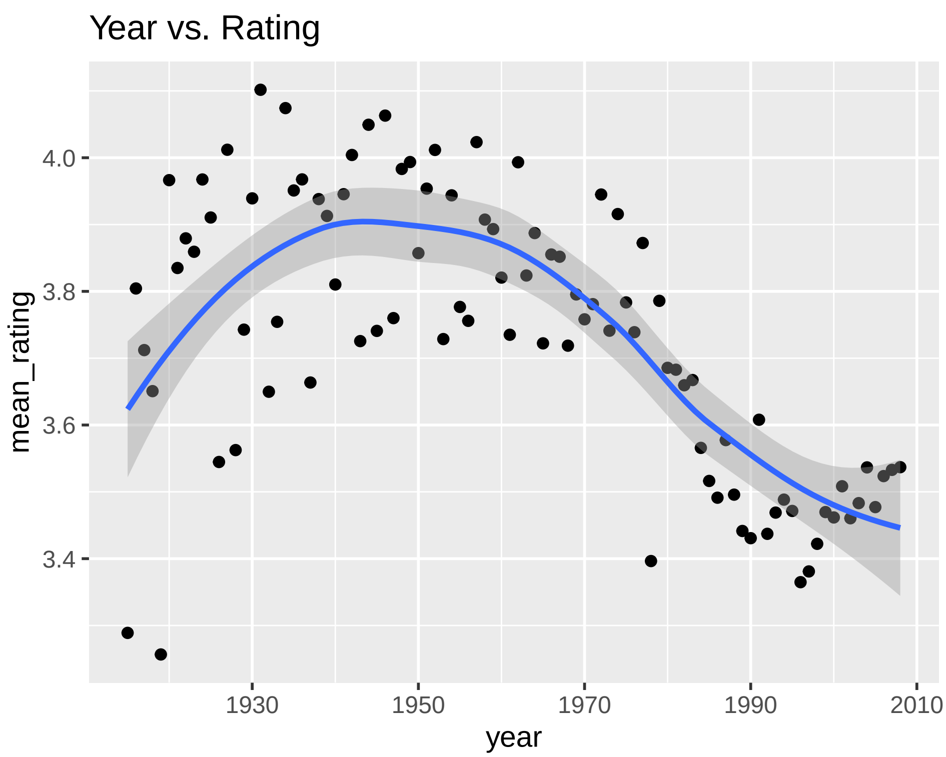
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



In this plot we can observe an exponential grow from the 70s until the mid 90s and the completly drop off in 2010.

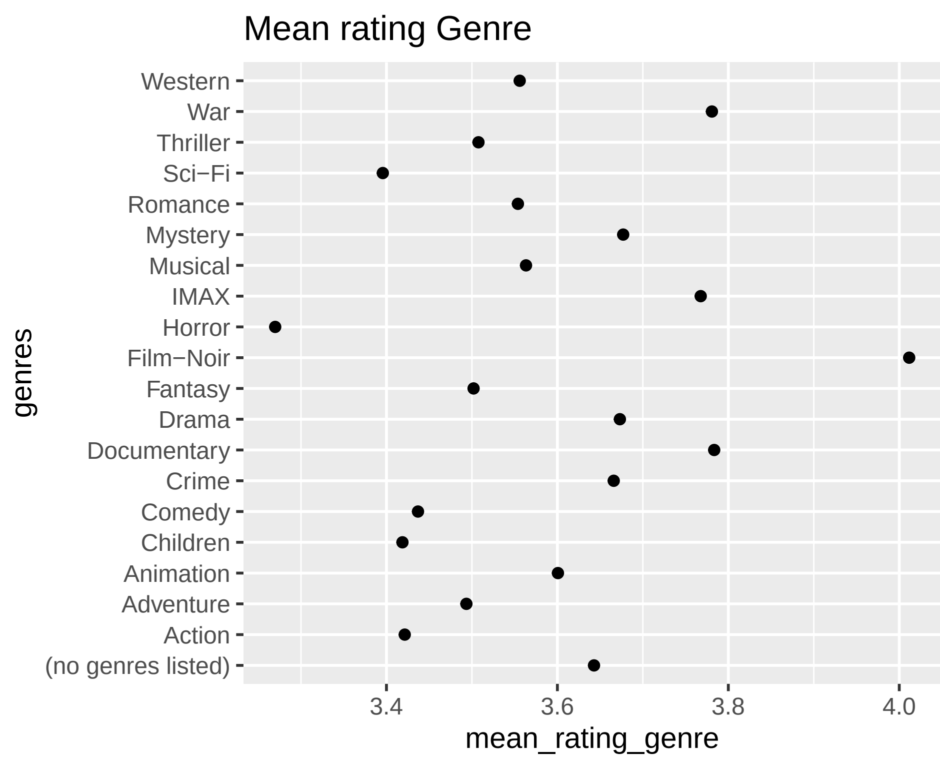
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In this plot we can observe the realtionship between the realease year and the rating: Older movies tend to have better ratings.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Conclusion: We covered most of the features in the data set, and the ones that seems to have a bigger impact in the rating prediction are: Num movies efect, users effect and year effect. We will build our ML modeltaking into account those insigths.

3. RMSE Function

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

4. ML algoritms + apply the RMSE function to them.

```
#1.Baseline (simplest one)
mu <- mean(edx$rating)
naive_RMSE <- RMSE(validation$rating, mu)
naive_RMSE
```

```
## [1] 1.061308
```

```
#2.Adding Movie effect model (Taking into account b_i(movie effect))

  #The movie effect:The mean of substracting the mean to the rating (b_i)
  movie_effect <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))
```

```r
  #Predictions: We compute the predicted movie effect in the validation dataset
  predicted_movie_effect <- validation %>%
    left_join(movie_effect, by='movieId') %>%
    mutate(prediction = mu + b_i)
  movie_RMSE <- RMSE(validation$rating,predicted_movie_effect$prediction)


#3.Adding User effect model to previous model (Taking into account b_i(movie effect))
  #First, we add the movie effect, then group by userId, and calculate b_u:
  #The mean of substracting the mean and b_i to the rating.
  user_effect <- edx %>%
    left_join(movie_effect, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - mu - b_i))

  #Predictions: We compute the predicted user+ movie effect in the validation dataset
  predicted_movie_user_effect  <- validation %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
    mutate(prediction = mu + b_i + b_u)

  movie_user_RMSE <- RMSE(validation$rating,predicted_movie_user_effect$prediction)

#4.Adding Year effect model to previous model (Taking into account b_i(movie effect))
  #First, we add the movie and user effect, then group by year, and calculate b_y
  year_effect <- edx %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = mean(rating - mu - b_i - b_u))
  #Predictions: We compute the predicted user+movie+year effect in the validation dataset
  predicted_movie_user_year_effect  <- validation %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
    left_join(year_effect, by='year') %>%
    mutate(prediction = mu + b_i + b_u + b_y)

  movie_user_year_RMSE <- RMSE(validation$rating,predicted_movie_user_year_effect$prediction)


# Data is extremely infuenced by noisy estimate as we notice during the data exploration
#(Ex: Users that made few reviews, Movies with few reviews..)
#We need to remove  the effect of these noise effect as possible in order to
#improve our RMSE. Therefore we must chose a lambda that fits better our model.

lambdas <- seq(0,10,0.2)
RMSEs <- sapply(lambdas, function(lambda){
  mu <- mean(edx$rating)

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

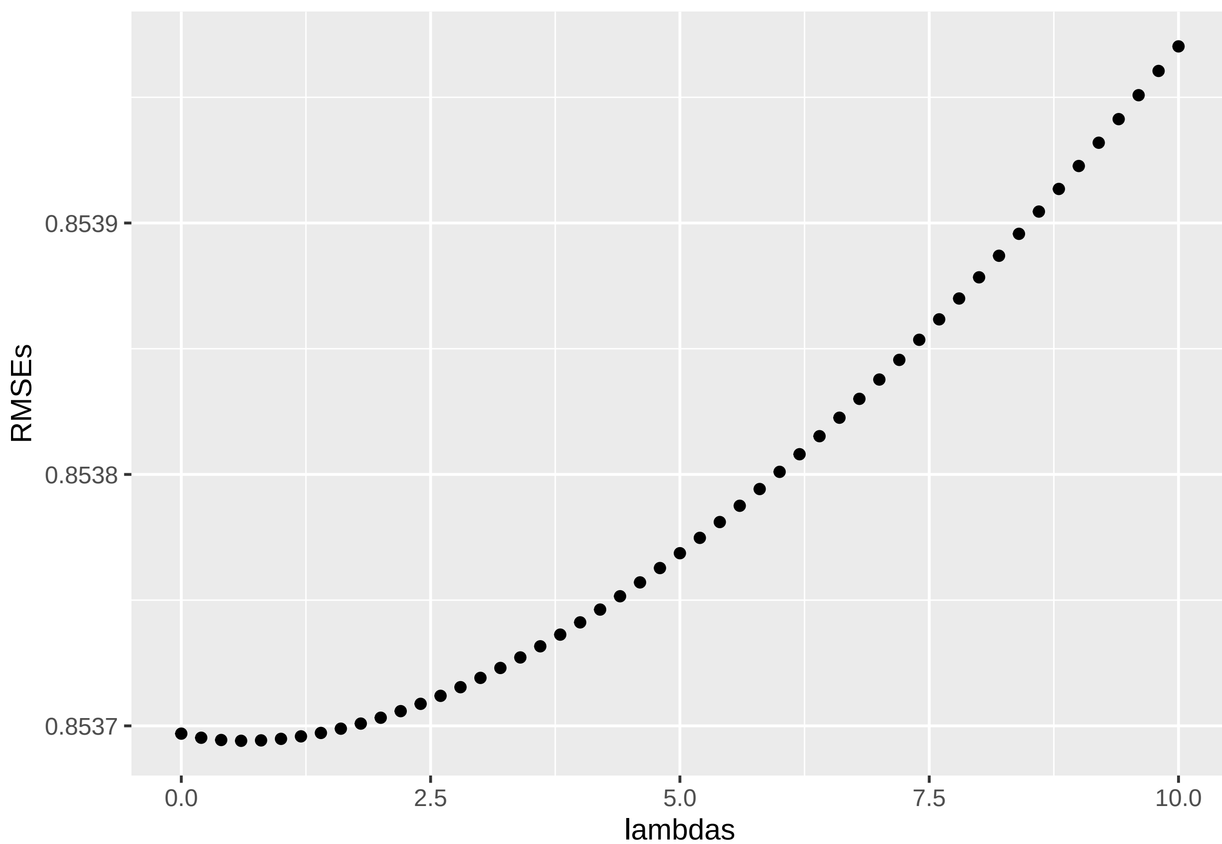
```
  b_u <- edx%>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() +lambda))

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

  predicted_ratings <- edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_y, by = "year") %>%
    mutate(prediction = mu + b_i +b_u + b_y) %>% .$prediction

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

qplot(lambdas, RMSEs)
```



Results

```
rmse_results <- data_frame(Model=c("Naive","Movie Effect","Movie+User Effect","Movie+User+Year Effect",
                           RMSE = c(naive_RMSE,movie_RMSE,movie_user_RMSE,movie_user_year_RMSE,min(RMSEs
```

Model	RMSE
Naive	1.0613075
Movie Effect	0.9439087
Movie+User Effect	0.8665960
Movie+User+Year Effect	0.8662406
Regularized Movie+User+Year Effect	0.8536941

Conclusions:

As we can see in the previous dataframe, we applied 5 different ML models to our dataset beeing the Regularized Movie + User + Year Effect the one that shows the least RMSE, and therefore, the best fit for our project aim. We can also see that the improvement in the RMSE from model 3 to model 4 is low. So by appling Occam's razor (Parsimony Principle), we can just take into account the Regularize Movie + User Effect.

Thanks!!!

Thanks for your time reviewing my project. I really apreciate your feedback!

——Adelaida Fernández——