Harvard University: Data Science Capstone Report

Ferry Edouard

2023-11-27

Summary

The purpose of this project is to predict movie ratings using the MovieLens(10M) dataset, which contains the ratings of many movies given by many users. For this project we begin by importing (downloading) data, cleaning and preparation of the EDA(exploratory data analysis). Hence we explored the dataset in other to get valuable outcomes. The dataset is divided into a 90/10 (ratio) split of training and test datasets. A model is built from the training data, and then applied to the unseen test data. The success metric is root mean square estimate (RMSE). We attempted to predict movie ratings on a scale of 0.5 stars to 5 stars, and RMSE is measured on the same scale. An RMSE of 0 means we are always correct, which is unlikely. An RMSE of 1 means the predicted ratings are off by 1 star. Note that the final_holdout_test set is used only for the final RMSE evaluation. The goal for this project is to achieve RMSE < 0.8649 as computed on the unseen test dataset.

Introduction

Nowadays recommendation systems are omnipresent and trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own. Recommendation systems can also be as: identifying and pursuing key business channels(selling products based upon customer's specific needs/expectations and so on and so forth). Hence (in this vein) comes to mind (life) the idea of building a movie recommendation system: the MovieLens data set as a project using the 10M version, a data set that was collected by GroupLens Research.

Project Goal:

The purpose of choosing MovieLens data set is to predict movie ratings based on user preference, age of movie, genre/category of movies etc… This will be done by training a machine learning algorithm that predicts user ratings (on a scale of 0.5 to 5 stars) using the MovieLens dataset split into training and validation sets to train on and predict movie ratings the validation set.

Note that the final_holdout_test set is used only for the final RMSE evaluation. The goal for this project is to achieve RMSE < 0.8649 as computed on the unseen test dataset.

The first step toward this goal will be to look at the schema/structure of the data set, visualize it and then sequentially (progressively) build a model that will meet the expectations (reach target accurately). Hence, begins my journey.

This project is the last for *Data Science: Capstone* (PH125.9x) course in the HarvardX Professional Certificate Data Science Program. We will be using the methods taught in the program.

Data exploration

```r
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

```r
### The following questions are part of the Quiz which is ###
### 10% of the project. I'll make sure no duplicate answers ######

# How many zeroes & 3s

zeros <- sum(edx$rating == 0)
threes <- sum(edx$rating == 3)

## Q2
# How many zeros were given as ratings in the edx dataset?
# How many threes were given as ratings in the edx dataset?

print(zeros)
```

```
## [1] 0
```

```r
print(threes)
```

```
## [1] 2121240
```

```r
# Q3
# How many different movies are in the edx dataset?
uniqueMovies <- length(unique(edx$movieId))
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```r
# Q4
# How many different users are in the edx dataset?

uniqueUsers <- n_distinct(edx$userId)

print(uniqueUsers)
```

```
## [1] 69878
```

```r
# Q5
# How many movie ratings are in each of the
# following genres in the edx dataset?
List_of_genre <- c('Drama', 'Comedy', 'Thriller', 'Romance')
Nbr_of_genres <- sapply(List_of_genre, function(g){
  edx %>% filter(str_detect(genres, g)) %>% tally()
})
```

```r
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n())
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 (no genres listed)       7
##  2 Action             2560545
##  3 Adventure          1908892
##  4 Animation           467168
##  5 Children            737994
##  6 Comedy             3540930
##  7 Crime              1327715
##  8 Documentary          93066
##  9 Drama              3910127
## 10 Fantasy             925637
## 11 Film-Noir           118541
## 12 Horror              691485
## 13 IMAX                  8181
## 14 Musical             433080
## 15 Mystery             568332
## 16 Romance            1712100
## 17 Sci-Fi             1341183
## 18 Thriller           2325899
## 19 War                 511147
## 20 Western             189394
```

```r
# Q6
# Which movie has the greatest number of ratings?
N_Ratings <- edx %>% group_by(movieId) %>%
  summarize(N_Ratings = n(), movieTitle = first(title)) %>%
  arrange(desc(N_Ratings)) %>%
  top_n(10, N_Ratings)

print(N_Ratings)
```

```
## # A tibble: 10 x 3
##    movieId N_Ratings movieTitle
##      <int>     <int> <chr>
##  1     296     31362 Pulp Fiction (1994)
##  2     356     31079 Forrest Gump (1994)
##  3     593     30382 Silence of the Lambs, The (1991)
##  4     480     29360 Jurassic Park (1993)
##  5     318     28015 Shawshank Redemption, The (1994)
##  6     110     26212 Braveheart (1995)
##  7     457     25998 Fugitive, The (1993)
##  8     589     25984 Terminator 2: Judgment Day (1991)
##  9     260     25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (197~
## 10     150     24284 Apollo 13 (1995)
```

```r
# Q7
# What are the five most given ratings in order from most to least?
N_Ratings <- edx %>% group_by(rating) %>%
  summarize(number = n())

N_Ratings %>% top_n(5) %>% arrange(desc(number))
```

```
## Selecting by number
```

```
## # A tibble: 5 x 2
##   rating  number
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

```r
# Q8
# True or False: In general, half star ratings are less common than whole star
# ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

N_Ratings %>%
  mutate(halfStar = rating %% 1 == 0.5) %>%
  group_by(halfStar) %>%
  summarize(number = sum(number))
```

```
## # A tibble: 2 x 2
##   halfStar  number
##   <lgl>      <int>
## 1 FALSE    7156885
## 2 TRUE     1843170
```

####################End Quiz##############################

2.2 Data Exploration

Exploring the Data Set (EDA)

```
### Before we go any further (build the model), it is important
### to understand the schema of the data, distribution of ratings
### and the relationship of the predictors. Hence my journey toward a better model.

str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
```

```
# From the initial of the EDA, we notice that edx has 6 variables described as follow:

# •  "user_id : int" : a unique identifier of the user who made the rating

# • "movie_id : int ": a unique identifier of the rated movie

# •  "rating  : num ":  the score of the rating on a five-star scale

# •  "timestamp : int": the timestamp of the ratings, represented in seconds since midnight Coordinated

# •  "title : chr": the title of the rated movie with the release year in parentheses

# •  "genres : chr": a sequence of genres to which the rated movie belongs

# •  "year_rated: num":  1996 1996 1996 1996 1996 ...
```

More EDA: Exploratory Data Analysis

Dataset Dimenions

```
###Check Dimensions (rows and columns ) of both final_holdout_test and train tables
cat("\nEdx (it contents):",dim(edx))
```

```
##
## Edx (it contents): 9000055 6
```

```
cat("\nfinal holdout test (it contents):",dim(final_holdout_test))
```

```
##
## final holdout test (it contents): 999999 6
```

```
# The following table shows the schema and content of edx dataset
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
```

```
# Dissecting genres
# 2.2.1 No specific Genres(mix genres) and Genres
# The data set contains 797 different combinations of genres.
# The following table contents the first list of genres.

edx %>% group_by(genres) %>%
  summarise(n=n()) %>%
  head()
```

```
## # A tibble: 6 x 2
##   genres                                                n
##   <chr>                                             <int>
## 1 (no genres listed)                                    7
## 2 Action                                            24482
## 3 Action|Adventure                                  68688
## 4 Action|Adventure|Animation|Children|Comedy         7467
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy   187
## 6 Action|Adventure|Animation|Children|Comedy|IMAX       66
```

```
# Here is the second list of genre in an orderly fashions.
tibble(count = str_count(edx$genres, fixed("|")), genres = edx$genres) %>%
  group_by(count, genres) %>%
  summarise(n = n()) %>%
  arrange(-count) %>%
  head()
```

```
## `summarise()` has grouped output by 'count'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   count [3]
##   count genres                                                       n
##   <int> <chr>                                                    <int>
## 1     7 Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller   256
## 2     6 Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery    10975
## 3     6 Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery      355
## 4     6 Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance    515
## 5     5 Action|Adventure|Animation|Children|Comedy|Fantasy             187
## 6     5 Action|Adventure|Animation|Children|Comedy|IMAX                 66
```

```
#Sui generis genre (unique genre)
unique_genre <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
print(unique_genre)
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

```r
# Noticing that the dataset  displays 20 different genres &
# 7 other movies that have not listed as genres whatsoever
# (7 movies without genres)


# 2.2.2 Date conversion and rating period of time
#Convert Timestamp to year
edx <- mutate(edx, year_rated = year(as_datetime(timestamp)))
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 year_rated
## 1                Comedy|Romance       1996
## 2         Action|Crime|Thriller       1996
## 4  Action|Drama|Sci-Fi|Thriller       1996
## 5       Action|Adventure|Sci-Fi       1996
## 6 Action|Adventure|Drama|Sci-Fi       1996
## 7       Children|Comedy|Fantasy       1996
```

```r
# The following code will create an Histogram showing the Rating Distribution Per Year
# Period of collecting rating that started over the years
if(!require(ggthemes))
  install.packages("ggthemes", repos = "http://cran.us.r-project.org")
```

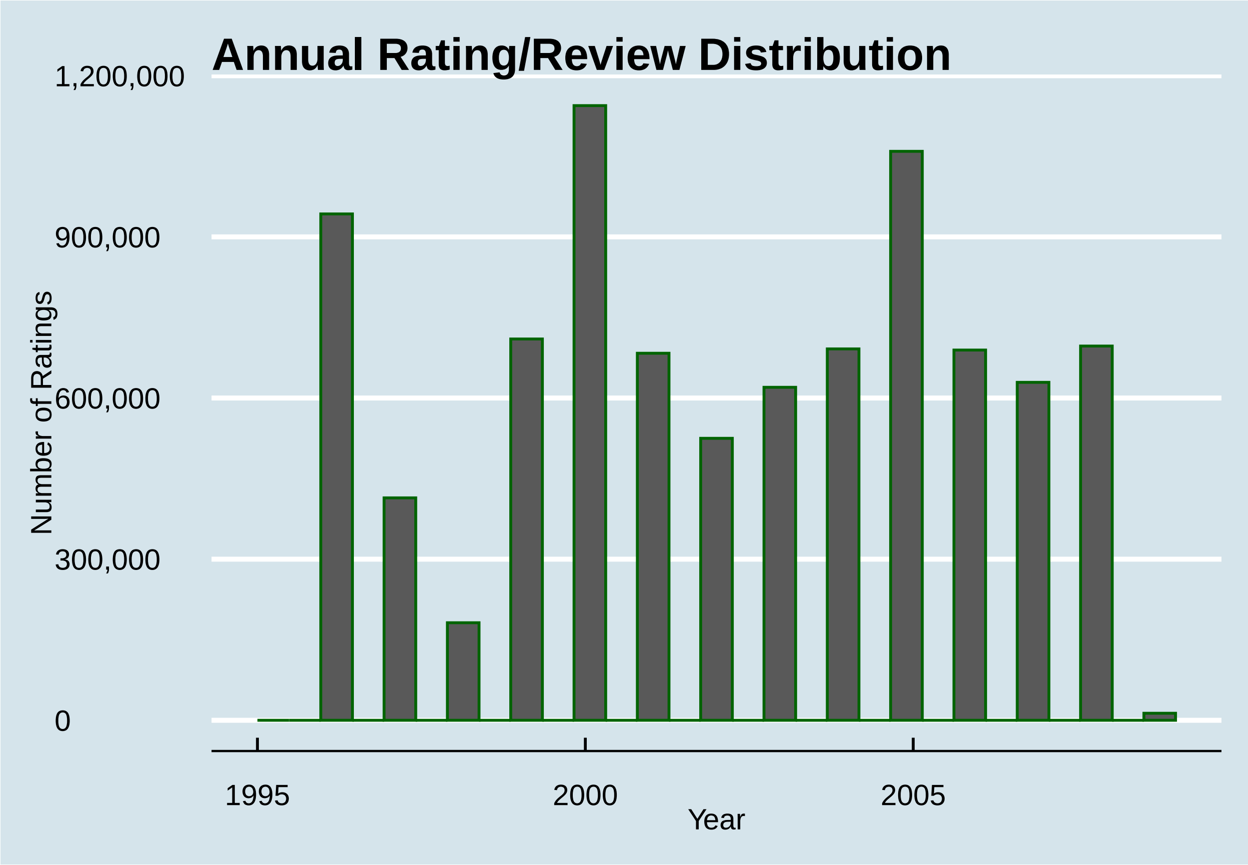
```
## Loading required package: ggthemes
```

```r
library(ggthemes)

if(!require(scales))
  install.packages("scales", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(scales)

edx %>% mutate(year = year(as_datetime(timestamp, origin="1970-01-01"))) %>%
  ggplot(aes(x=year)) +
  geom_histogram(color = "dark green") +
  ggtitle("Annual Rating/Review Distribution") +
  xlab("Year") +
  ylab("Number of Ratings") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



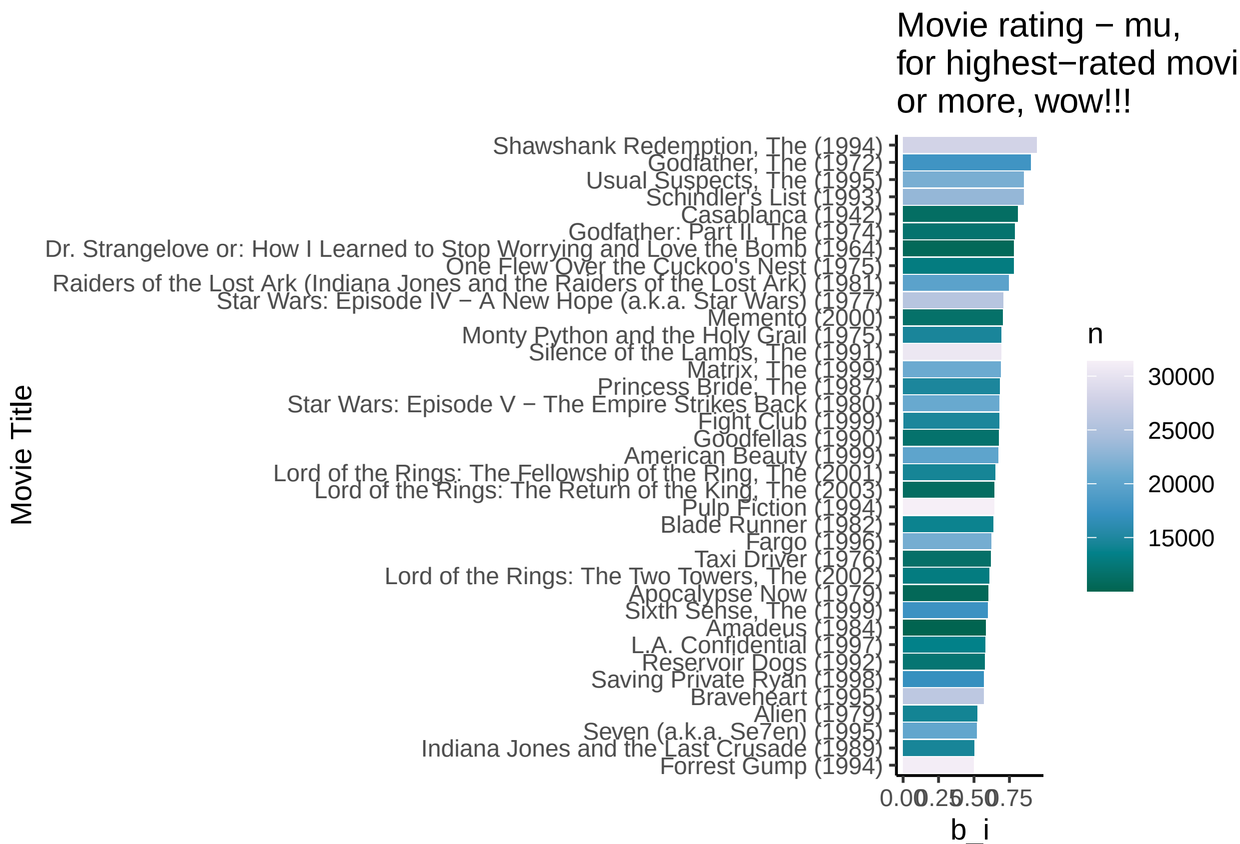

```
#The above plot shows the Rating Distribution Per Year, period which collect rating that started over t
```

```r
#Edx: In depth dissecting (period during which more rating took place)
edx %>% mutate(date = date(as_datetime(timestamp, origin="1970-01-01"))) %>%
  group_by(date, title) %>%
  summarise(count = n()) %>%
  arrange(-count) %>%
  head(15)
```

```
## `summarise()` has grouped output by 'date'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 15 x 3
## # Groups:   date [4]
##    date       title                                                  count
##    <date>     <chr>                                                  <int>
##  1 1998-05-22 Chasing Amy (1997)                                       322
##  2 2000-11-20 American Beauty (1999)                                   277
##  3 1999-12-11 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   254
##  4 1999-12-11 Star Wars: Episode V - The Empire Strikes Back (1980)    251
##  5 1999-12-11 Star Wars: Episode VI - Return of the Jedi (1983)        241
##  6 2005-03-22 Lord of the Rings: The Two Towers, The (2002)            239
##  7 2005-03-22 Lord of the Rings: The Fellowship of the Ring, The (2001)   227
##  8 2000-11-20 Terminator 2: Judgment Day (1991)                        221
##  9 1999-12-11 Matrix, The (1999)                                       210
## 10 2000-11-20 Jurassic Park (1993)                                     201
## 11 2000-11-20 Braveheart (1995)                                        197
## 12 2000-11-20 Star Wars: Episode VI - Return of the Jedi (1983)        194
## 13 2000-11-20 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   193
## 14 2000-11-20 Star Wars: Episode V - The Empire Strikes Back (1980)    192
## 15 2005-03-22 Shrek (2001)                                             192
```

```r
#Graphically speaking let's see which movies have more ratings     than the average mu
mu <- mean(edx$rating)
edx %>% group_by(title) %>%
  summarize(b_i = mean(rating - mu), n = n()) %>% filter(b_i > 0.5, n > 10000) %>%
  ggplot(aes(reorder(title, b_i), b_i, fill = n)) +
  geom_bar(stat = "identity") + coord_flip() + scale_fill_distiller(palette = "PuBuGn") +
  ggtitle("") + xlab("Movie Title") +
  ggtitle("Movie rating - mu,\nfor highest-rated movies that at least 10000 ratings \nor more, wow!!!")
  theme_classic()
```




```r
# The above graph/plot shows the  highest rated movies
```

Training and Testing Sets:

```
#Training and Testing Sets:
set.seed(2023, sample.kind = "Rounding")
```

```
## Warning in set.seed(2023, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <-createDataPartition(y = edx$rating, times = 1, p = 0.1, list = F)

train_data <-edx[-test_index,]
edx_temp <-edx[test_index,]

#Case where userId and movieId are in both the training and         testing sets
test_data <-edx_temp %>% semi_join(train_data, by = "movieId") %>%
  semi_join(train_data, by = "userId")

#Adding removed Rows from the edx_test back into train_data
add_rows_back <-anti_join(edx_temp, test_data)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres,
## year_rated)`
```

```
train_data <-rbind(train_data, add_rows_back)
rm(train_data, test_index, add_rows_back)
```

Cleaning and Analyzing the Data

##Data sui generis (unique). Ensure that data like userIds, movieIds, and genres are not duplicated

```
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 7
##    userId movieId rating timestamp title                         genres year_rated
##     <int>   <int>  <dbl>     <int> <chr>                         <chr>       <dbl>
##  1      1     122      5 838985046 Boomerang (1992)              Comed~       1996
##  2      1     185      5 838983525 Net, The (1995)               Actio~       1996
##  3      1     292      5 838983421 Outbreak (1995)               Actio~       1996
##  4      1     316      5 838983392 Stargate (1994)               Actio~       1996
##  5      1     329      5 838983392 Star Trek: Generations (19~ Actio~       1996
##  6      1     355      5 838984474 Flintstones, The (1994)       Child~       1996
##  7      1     356      5 838983653 Forrest Gump (1994)           Comed~       1996
##  8      1     362      5 838984885 Jungle Book, The (1994)       Adven~       1996
##  9      1     364      5 838983707 Lion King, The (1994)         Adven~       1996
## 10      1     370      5 838984596 Naked Gun 33 1/3: The Fina~ Actio~       1996
## # i 9,000,045 more rows
```

```
#Ensure that data are not duplicated (userIds, movieIds, and enres are: sui generis (unique))
edx %>% summarize(unique_users = length(unique(userId)),
                  unique_movies = length(unique(movieId)),
                  unique_genres = length(unique(genres)))
```

```
##   unique_users unique_movies unique_genres
## 1        69878         10677           797
```

```
################More Ratings: ##################

#Here we go again for more ratings
#Extracting the first date and calculate the age of the movie.
# Find out if the age of the movie effects predicted ratings.

#Bring the first date to light
first_date <- stringi::stri_extract(edx$title, regex = "(\\d{4})", comments = TRUE ) %>% as.numeric()


#Adding the first date
title_dates <- edx %>% mutate(first_date = first_date)
head(title_dates)
```

```
##   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 year_rated first_date
## 1                Comedy|Romance       1996       1992
```

```
## 2         Action|Crime|Thriller       1996       1995
## 4  Action|Drama|Sci-Fi|Thriller       1996       1995
## 5       Action|Adventure|Sci-Fi       1996       1994
## 6 Action|Adventure|Drama|Sci-Fi       1996       1994
## 7       Children|Comedy|Fantasy       1996       1994
```

```
#Get rid of timestamp
title_dates <- title_dates %>% select(-timestamp)

head(title_dates)
```

```
##   userId movieId rating                         title
## 1      1     122      5                Boomerang (1992)
## 2      1     185      5                 Net, The (1995)
## 4      1     292      5                 Outbreak (1995)
## 5      1     316      5                 Stargate (1994)
## 6      1     329      5 Star Trek: Generations (1994)
## 7      1     355      5       Flintstones, The (1994)
##                           genres year_rated first_date
## 1                 Comedy|Romance       1996       1992
## 2          Action|Crime|Thriller       1996       1995
## 4   Action|Drama|Sci-Fi|Thriller       1996       1995
## 5        Action|Adventure|Sci-Fi       1996       1994
## 6  Action|Adventure|Drama|Sci-Fi       1996       1994
## 7        Children|Comedy|Fantasy       1996       1994
```

```
#What is the overall mean rating? Here it is:
overall_mean <- mean(edx$rating)
print(overall_mean)
```

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

```
#Now let see if dates are correct
title_dates %>% filter(first_date > 2018) %>% group_by(movieId, title, first_date) %>% summarize(n = n()
```

```
## `summarise()` has grouped output by 'movieId', 'title'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 6 x 4
## # Groups:   movieId, title [6]
##   movieId title                                        first_date     n
##     <int> <chr>                                             <dbl> <int>
## 1     671 Mystery Science Theater 3000: The Movie (1996)       3000  3280
## 2    2308 Detroit 9000 (1973)                               9000    22
## 3    4159 3000 Miles to Graceland (2001)                    3000   714
## 4    5310 Transylvania 6-5000 (1985)                        5000   195
## 5    8864 Mr. 3000 (2004)                                   3000   146
## 6   27266 2046 (2004)                                       2046   426
```

```
title_dates %>% filter(first_date < 1900) %>% group_by(movieId, title, first_date) %>% summarize(n = n()
```

```
## `summarise()` has grouped output by 'movieId', 'title'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 8 x 4
## # Groups:   movieId, title [8]
##   movieId title                                                first_date     n
##     <int> <chr>                                                     <dbl> <int>
## 1    1422 Murder at 1600 (1997)                                      1600  1566
## 2    4311 Bloody Angels (1732 Høtten: Marerittet Har et Postnu~      1732     9
## 3    5472 1776 (1972)                                                1776   185
## 4    6290 House of 1000 Corpses (2003)                               1000   367
## 5    6645 THX 1138 (1971)                                            1138   464
## 6    8198 1000 Eyes of Dr. Mabuse, The (Tausend Augen des Dr. ~      1000    24
## 7    8905 1492: Conquest of Paradise (1992)                          1492   134
## 8   53953 1408 (2007)                                                1408   466
```

```
#Now let find the age of the movies
title_dates <- title_dates %>% mutate(age_of_movie = 2018 - first_date,
                                      rating_date_range = year_rated - first_date)
head(title_dates)
```

```
##   userId movieId rating                         title
## 1      1     122      5              Boomerang (1992)
## 2      1     185      5               Net, The (1995)
## 4      1     292      5               Outbreak (1995)
## 5      1     316      5               Stargate (1994)
## 6      1     329      5 Star Trek: Generations (1994)
## 7      1     355      5       Flintstones, The (1994)
##                           genres year_rated first_date age_of_movie
## 1                 Comedy|Romance       1996       1992           26
## 2          Action|Crime|Thriller       1996       1995           23
## 4  Action|Drama|Sci-Fi|Thriller       1996       1995           23
## 5        Action|Adventure|Sci-Fi       1996       1994           24
## 6 Action|Adventure|Drama|Sci-Fi       1996       1994           24
## 7        Children|Comedy|Fantasy       1996       1994           24
##   rating_date_range
## 1                 4
## 2                 1
## 4                 1
## 5                 2
## 6                 2
## 7                 2
```

```
# Skip the graph here.....

#Rating average: movies, users, average rating by age of movie,    average rating by year

#Movie rating averages
movie_avgs <- title_dates %>% group_by(movieId) %>% summarize(avg_movie_rating = mean(rating))
user_avgs <- title_dates %>% group_by(userId) %>% summarize(avg_user_rating = mean(rating))
year_avgs <- title_dates%>% group_by(year_rated) %>% summarize(avg_rating_by_year = mean(rating)) #year
age_avgs <- title_dates %>% group_by(age_of_movie) %>% summarize(avg_rating_by_age = mean(rating)) #age
head(age_avgs)
```

```
## # A tibble: 6 x 2
##   age_of_movie avg_rating_by_age
##          <dbl>             <dbl>
## 1        -6982              2.80
## 2        -2982              2.30
## 3         -982              3.48
## 4          -28              3.73
## 5            8              3.37
## 6           10              3.46
```

```
head(user_avgs)
```

```
## # A tibble: 6 x 2
##   userId avg_user_rating
##    <int>           <dbl>
## 1      1            5
## 2      2            3.29
## 3      3            3.94
## 4      4            4.06
## 5      5            3.92
## 6      6            3.95
```

```
# EDA (exploratory Data Analysis)
#cat("\nTrain set dimension :",dim(edx))
#cat("\nNumber of unique movies :",edx$movieId %>% unique() %>% length())
#cat("\nNumber of unique users :",edx$userId %>% unique() %>% length())


#Different movies for different genres
cat("\nDifferent movies for different genres :\n")
```

```
##
## Different movies for different genres :
```

```
genres <- c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
##    Drama   Comedy Thriller  Romance
##  3910127  3540930  2325899  1712100
```

```
#Most rated movies
edx %>% group_by(movieId) %>%
  summarise(n_ratings=n(), title=first(title)) %>%
  top_n(5, n_ratings)
```

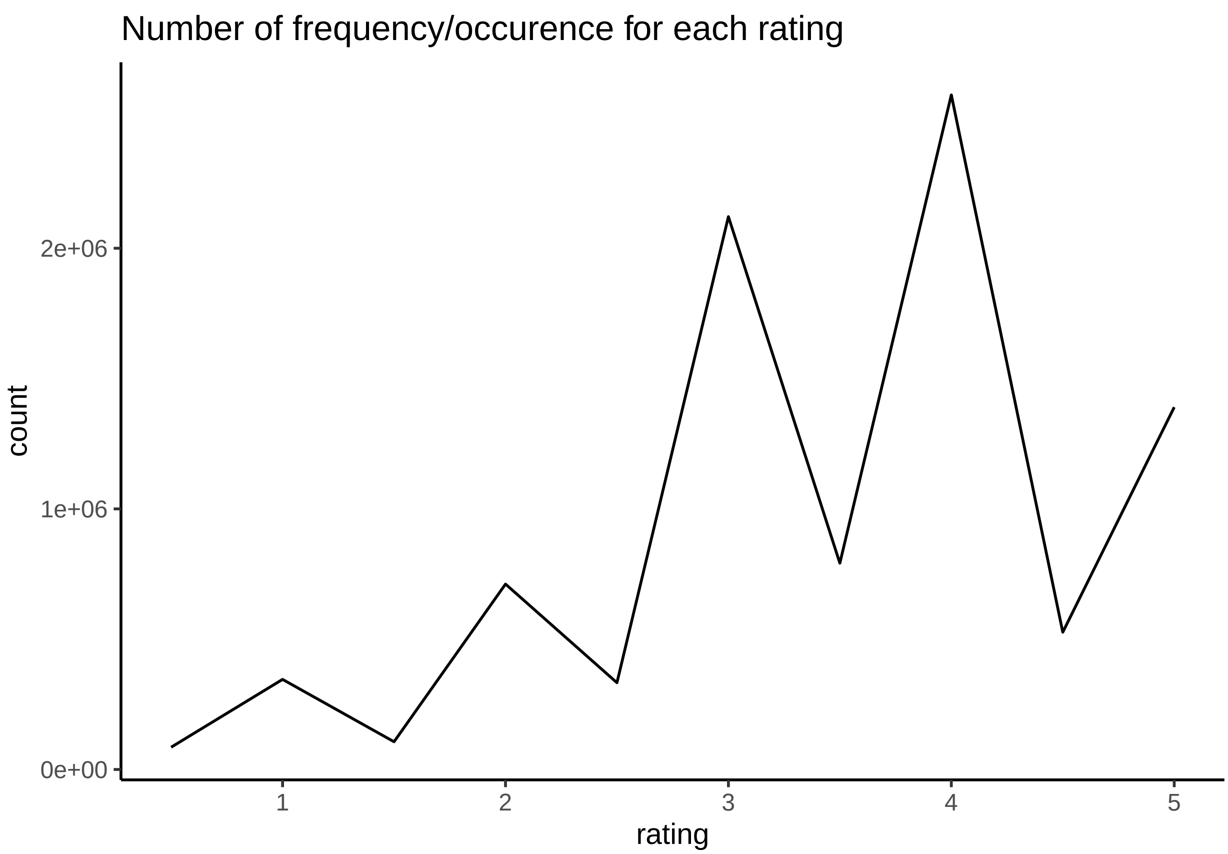
```
## # A tibble: 5 x 3
##   movieId n_ratings title
##     <int>     <int> <chr>
## 1     296     31362 Pulp Fiction (1994)
```

```
## 2     318     28015 Shawshank Redemption, The (1994)
## 3     356     31079 Forrest Gump (1994)
## 4     480     29360 Jurassic Park (1993)
## 5     593     30382 Silence of the Lambs, The (1991)
```

```r
#Most often ratings (10 top ones)
edx %>% group_by(rating) %>%
  summarise(n_ratings=n()) %>%
  top_n(10, n_ratings) %>%
  arrange(desc(n_ratings))
```

```
## # A tibble: 10 x 2
##    rating n_ratings
##     <dbl>     <int>
##  1    4     2588430
##  2    3     2121240
##  3    5     1390114
##  4    3.5    791624
##  5    2      711422
##  6    4.5    526736
##  7    1      345679
##  8    2.5    333010
##  9    1.5    106426
## 10    0.5     85374
```

```r
#Rating Frequency
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line() +
  ggtitle("Number of frequency/occurence for each rating")
```



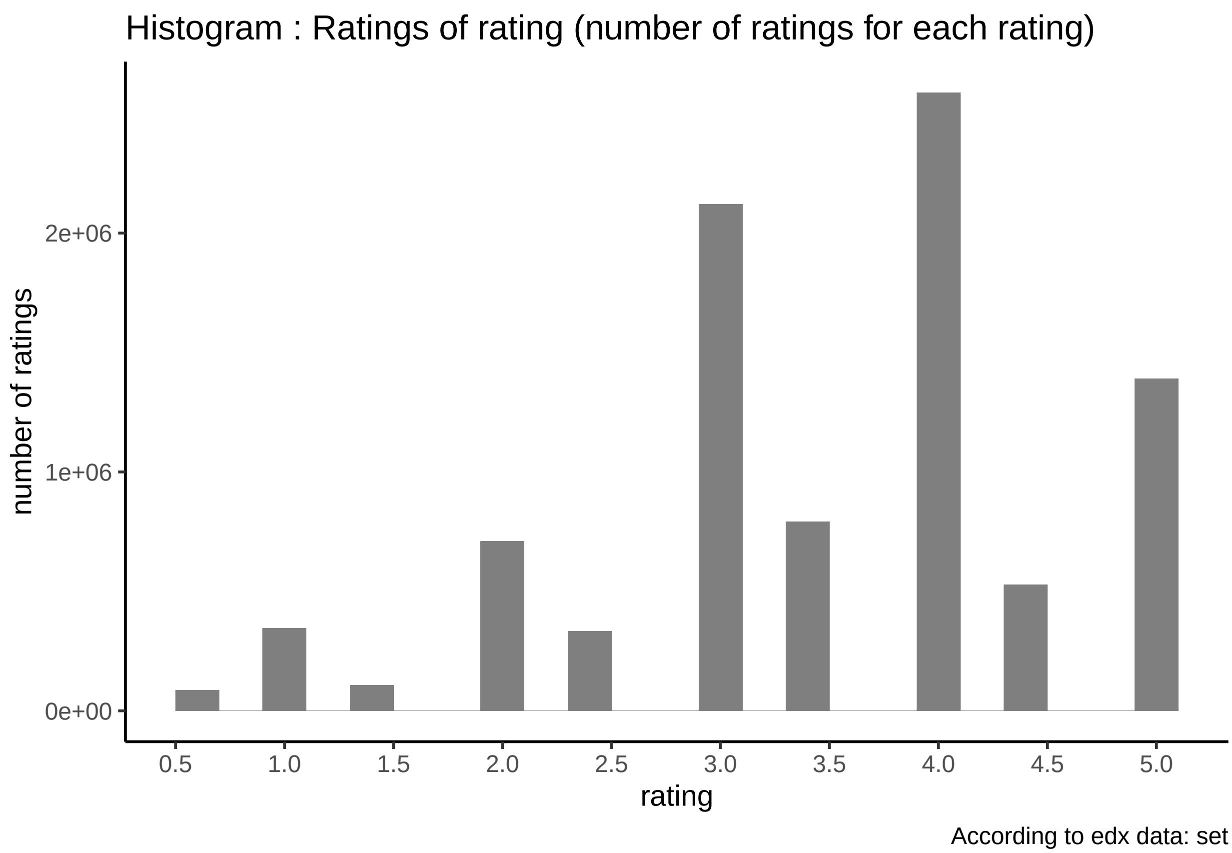
#Noticing that the most common rating is 4, and the least common is 0.

```
# Way of ratings (number of stars)
way <-  ifelse((edx$rating == 1 |edx$rating == 2 | edx$rating == 3 |
                  edx$rating == 4 | edx$rating == 5) ,
               "Full_Star",
               "Half_Star")

ratings_way <- data.frame(edx$rating, way)

head(ratings_way)
```

```
##   edx.rating       way
## 1          5 Full_Star
## 2          5 Full_Star
## 3          5 Full_Star
## 4          5 Full_Star
## 5          5 Full_Star
## 6          5 Full_Star
```

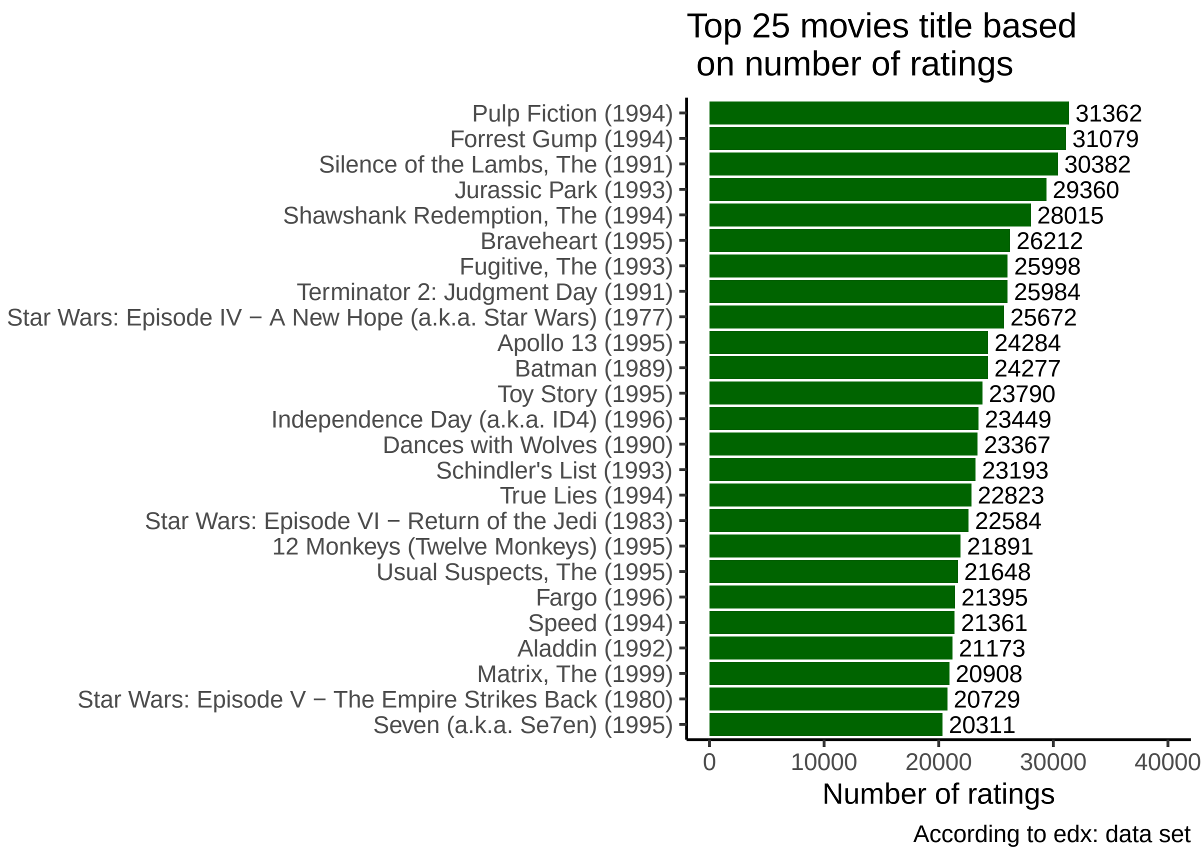
```
#print(ratings_way)


#Plotting/Histogram of ratings
ggplot(ratings_way, aes(x= edx.rating, fill = way)) +
  geom_histogram( binwidth = 0.2) +
```

```
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  scale_fill_manual(values = c("half_star"="yellow", "full_star"="green")) +
  labs(x="rating", y="number of ratings", caption = " According to edx data: set") +
  ggtitle("Histogram : Ratings of rating (number of ratings for each rating)")
```

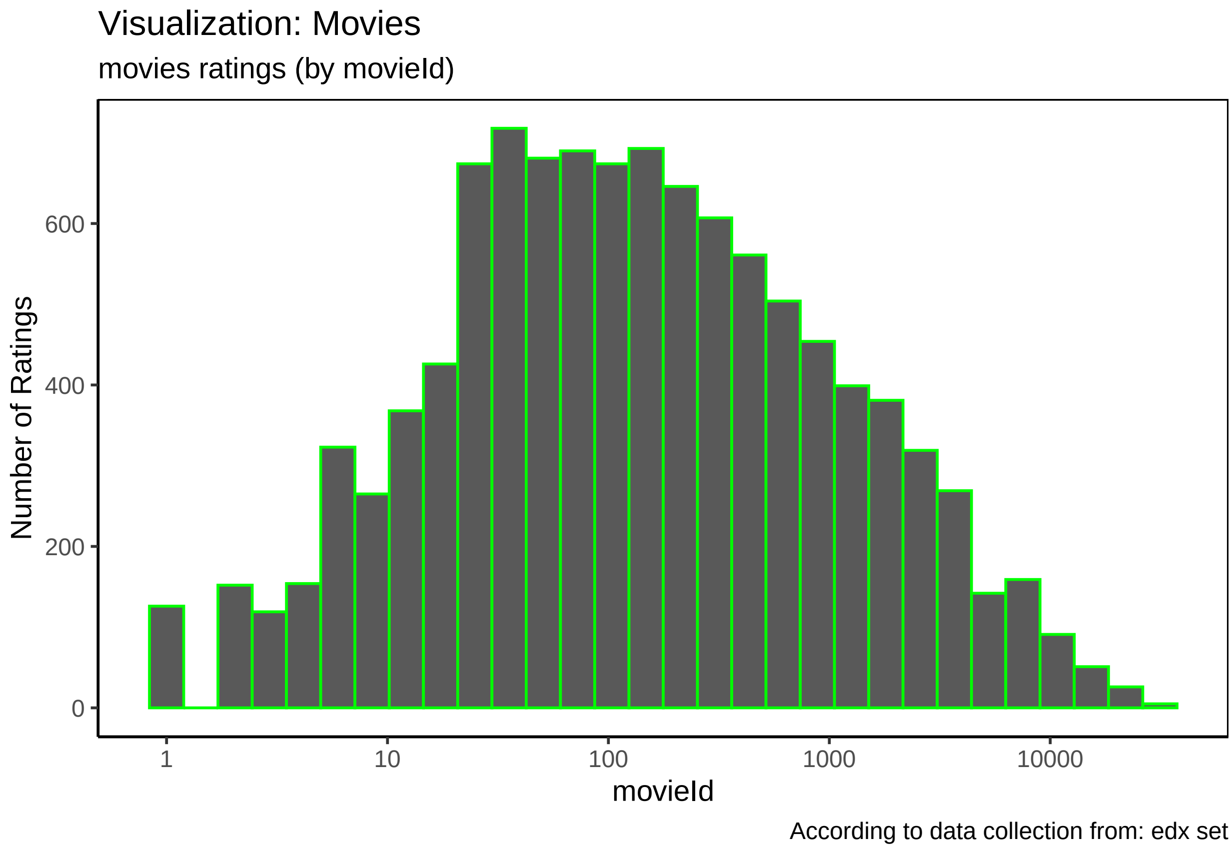


#Plotting/Histogram shows that no zero (0) rating, most ratings are: 4, 3, 5, 3.5 and 2 and the half star ratings are less likely than whole star ratings.

Top Title



#We notice that movies with the highest number of ratings are in the top genres categories such as Jurrasic Park(1993), Pulp fiction (1994), Forrest Gump(1994) which are in the top of movie's ratings number, are part of the Drama, Comedy or Action genres. This is what we call blockbusters movies

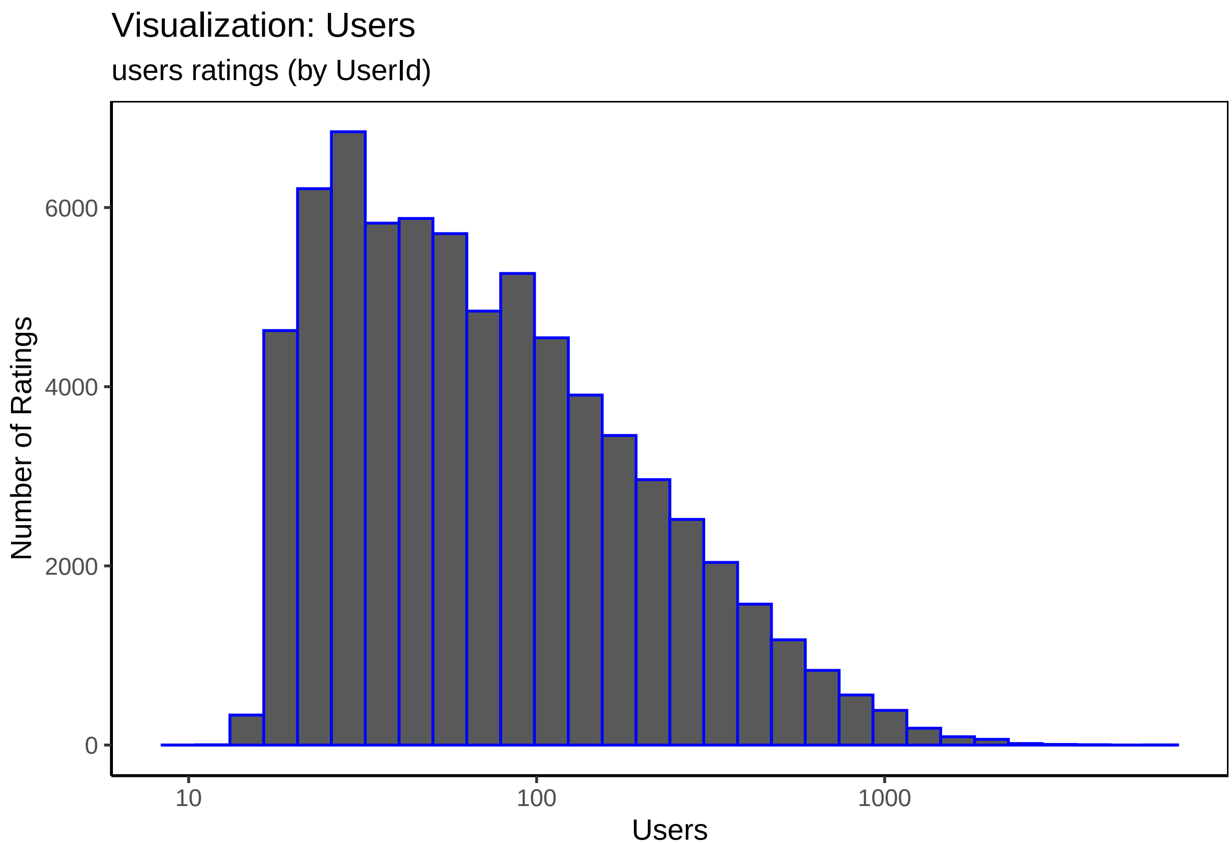


#Noticing that "Reviews" between movies are either less than 1000 or more than 10k. Yes indeed some of them are rated more than others, because many movies are watched by few users and movies like blockbusters have a big impact when it comes to ratings.

```
#The following table is example of how most users rate few movies. few users rate more than a thousand

edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  arrange(n) %>%
  head()
```

```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
## 1  62516    10
## 2  22170    12
## 3  15719    13
## 4  50608    13
## 5    901    14
## 6   1833    14
```

Histogram/Plotting for number of ratings by userId

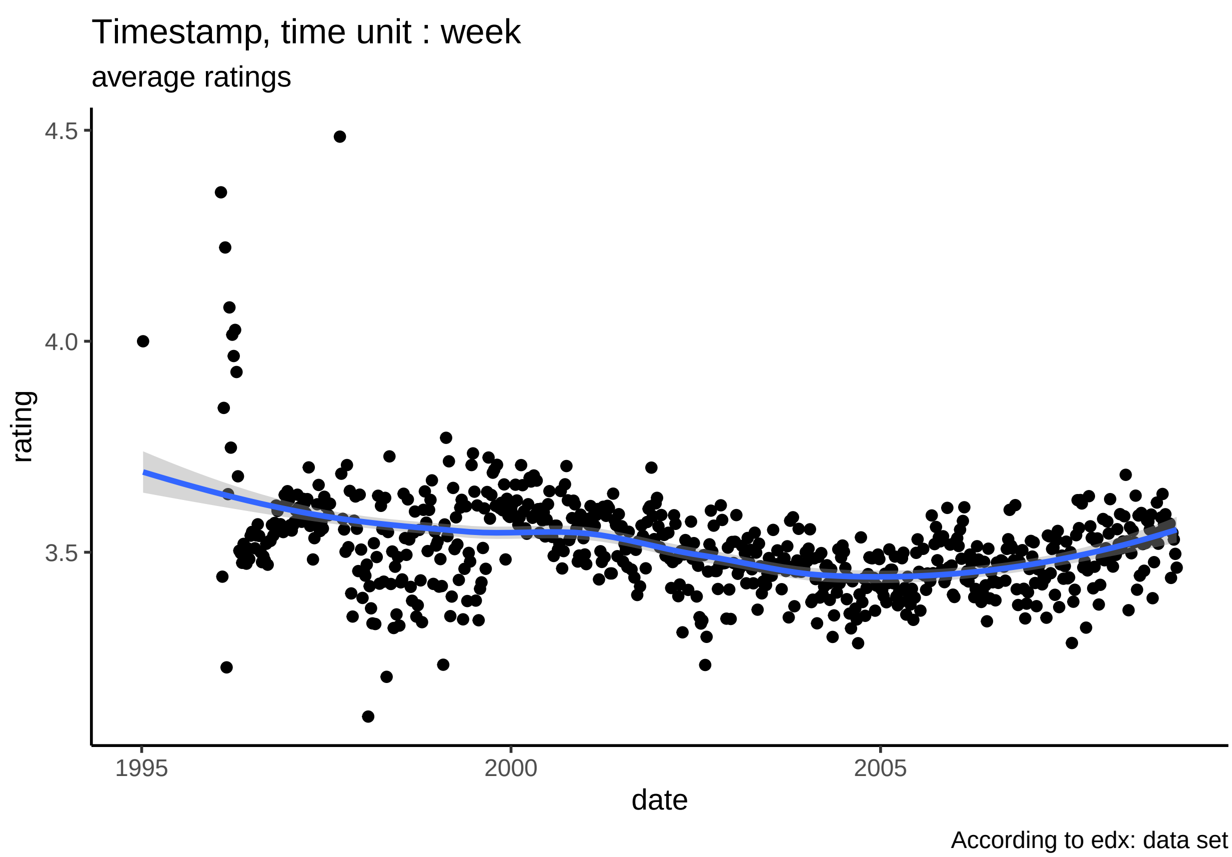


Noticing that some users wrote 100 reviews or less, some 1k or more.

```
# Working with timestamp

#Noticing that the edx set contains the timestamp variable
```

```r
#which represents the time and data in which the rating was
#provided. The units (seconds) are important dated back
#January 1, 1970..

edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Timestamp, time unit : week")+
  labs(subtitle = "average ratings",
       caption = "According to edx: data set")
```

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



Splitting EDX to train and test data

```
### DATA WRANGLING ###

#Splitting EDX to train and test data
set.seed(1998, sample.kind="Rounding")
```

```
## Warning in set.seed(1998, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
B <- 100000
edx_sample <- edx[sample(nrow(edx), B, replace = FALSE),]

# Splitting of edx sample on a 80/20 ratio for training purpose
train_index <- createDataPartition(edx_sample$rating, times=1, p=0.8,list=FALSE)
train <- edx_sample[train_index,]
test <- edx_sample[-train_index,]


# edx <- mutate(edx, year_rated = year(as_datetime(timestamp)))

#Wrangling

train$timestamp <- year(as_datetime(train$timestamp))


#extracting release year from title
p <- "(?<=\\()\\d{4}(?=\\))"
train$release_year <- train$title %>% str_extract(p) %>% as.integer()

#Encode the genres column
# str_split(string, p, n = Inf, simplify = FALSE)
# simplify = TRUE
train$genres <- str_split(train$genres, p="\\|")
genre_of_genres <- enframe(train$genres) %>%
  unnest(value) %>%
  mutate(temp = 1) %>%
  pivot_wider(names_from = value, values_from = temp, values_fill = list(temp = 0))
train <- cbind(train, genre_of_genres) %>% select(-name)
train$genres <- NULL

#Adding average rating  for each movie by subtracting the total average rating
avg_rating <- mean(train$rating)
movie_score <- train %>% group_by(movieId) %>%
  summarise(movie_score = mean(rating-avg_rating))

#Adding  average rating for each user by subtracting the total average rating and movie score
user_score <- train %>% left_join(movie_score, by="movieId") %>%
  mutate(movie_score = ifelse(is.na(movie_score), 0, movie_score)) %>%
  group_by(userId) %>%
  summarise(user_score = mean(rating-avg_rating-movie_score))

train <- train %>% left_join(user_score) %>% left_join(movie_score)
```

```
## Joining with ‘by = join_by(userId)‘
## Joining with ‘by = join_by(movieId)‘
```

```r
head(train)
```

```
##   userId movieId rating timestamp                                 title
## 1   2845     208    3.0      1996                     Waterworld (1995)
## 2  18807   27873    3.5      2005 Metallica: Some Kind of Monster (2004)
## 3   2692   55765    4.0      2008              American Gangster (2007)
## 4    226    1799    4.0      2003                  Suicide Kings (1997)
## 5  61400    4306    4.0      2008                          Shrek (2001)
## 6  65298    4238    4.0      2008            Along Came a Spider (2001)
##   year_rated release_year Action Adventure Sci-Fi Documentary Crime Drama
## 1       1996         1995      1         1      1           0     0     0
## 2       2005         2004      0         0      0           1     0     0
## 3       2008         2007      0         0      0           0     1     1
## 4       2003         1997      0         0      0           0     1     1
## 5       2008         2001      0         1      0           0     0     0
## 6       2008         2001      1         0      0           0     1     0
##   Thriller Comedy Mystery Animation Children Fantasy Romance Horror War Musical
## 1        0      0       0         0        0       0       0      0   0       0
## 2        0      0       0         0        0       0       0      0   0       0
## 3        1      0       0         0        0       0       0      0   0       0
## 4        1      1       1         0        0       0       0      0   0       0
## 5        0      1       0         1        1       1       1      0   0       0
## 6        1      0       1         0        0       0       0      0   0       0
##   Western Film-Noir IMAX (no genres listed)  user_score movie_score
## 1       0         0    0                  0  0.79156046 -0.68357845
## 2       0         0    0                  0 -0.38204225 -0.26813727
## 3       0         0    0                  0 -0.02195946  0.60686273
## 4       0         0    0                  0  0.25960936  0.32801657
## 5       0         0    0                  0  0.17085714  0.41043416
## 6       0         0    0                  0  0.50443485 -0.08480394
```

```r
#Apply same scenario(wrangling) to the test set

#Convert timestamp to datetime by using only the year
test$timestamp <- year(as_datetime(test$timestamp))

#extracting release year from title
p <- "(?<=\\()\\d{4}(?=\\))"
test$release_year <- test$title %>% str_extract(p) %>% as.integer()

#Encoding genres column
test$genres <- str_split(test$genres, p="\\|")
genre_of_genres <- enframe(test$genres) %>%
  unnest(value) %>%
  mutate(temp = 1) %>%
  pivot_wider(names_from = value, values_from = temp, values_fill = list(temp = 0))
test <- cbind(test, genre_of_genres) %>% select(-name)
train$genres <- NULL
```

```r
#Adding & removing data (add missing columns of genres that are absent  in test set, and
#get rid of those that are not in the train set)
for(col in names(train)){
  if(!col %in% names(test)){
    test$newcol <- 0
    names(test)[names(test)=="newcol"] <- col
  }
}
for(col in names(test)){
  if(!col %in% names(train)){
    test[,col] <- NULL
  }
}

#Average scores on the train set of each movie and user
test$user_score <- NULL
test$movie_score <- NULL
test <- test %>% left_join(user_score, by="userId") %>% left_join(movie_score, by="movieId")


test <- test %>% mutate(user_score = ifelse(is.na(user_score), 0, user_score)) %>% mutate(movie_score =




#Reordering .....
test <- test %>% select(names(train))
head(test)
```

```
##   userId movieId rating timestamp
## 1   4431    1084    4.0      1999
## 2  67235    1210    4.0      2006
## 3  29930    2611    4.5      2005
## 4  37568    7983    4.0      2004
## 5   8739    3094    3.0      2001
## 6   8651    3253    3.0      2004
##                                                  title year_rated release_year
## 1                               Bonnie and Clyde (1967)       1999         1967
## 2 Star Wars: Episode VI - Return of the Jedi (1983)       2006         1983
## 3                               Winslow Boy, The (1999)       2005         1999
## 4                            Broadway Danny Rose (1984)       2004         1984
## 5                                        Maurice (1987)       2001         1987
## 6                                   Wayne's World (1992)       2004         1992
##   Action Adventure Sci-Fi Documentary Crime Drama Thriller Comedy Mystery
## 1      0         0      0           0     1     1        0      0       0
## 2      1         1      1           0     0     0        0      0       0
## 3      0         0      0           0     0     1        0      0       0
## 4      0         0      0           0     0     0        0      1       0
## 5      0         0      0           0     0     1        0      0       0
## 6      0         0      0           0     0     0        0      1       0
##   Animation Children Fantasy Romance Horror War Musical Western Film-Noir IMAX
## 1         0        0       0       0      0   0       0       0         0    0
## 2         0        0       0       0      0   0       0       0         0    0
```

```
## 3         0        0        0        0      0   0        0        0          0    0
## 4         0        0        0        0      0   0        0        0          0    0
## 5         0        0        0        1      0   0        0        0          0    0
## 6         0        0        0        0      0   0        0        0          0    0
##    (no genres listed)  user_score movie_score
## 1                  0  0.00000000  0.20408495
## 2                  0  0.00000000  0.52464348
## 3                  0 -0.57317073 -0.10904636
## 4                  0 -0.26083520  0.31519606
## 5                  0 -0.05831583  0.28186273
## 6                  0  0.54817874 -0.02722818
```

BUILDING of ML MODELS

```
# BUILDING of ML MODELS

# Baseline comparison .......
y_hat <- mean(train$rating)
result <- RMSE(test$rating, y_hat)
cat("RMSE :", result)
```

```
## RMSE : 1.051226
```

```
# RMSE is 1.051226 meaning on average the prediction is
# OBOE (off-by-one error) which is not so good


#Building of ML (Machine Learning) Models

#Linear Model

#?
#Linear Model using the following: timestamp, user_score, release_year, and movie_score

control <- trainControl(method = "none")
fit_lm <- train(rating~user_score+movie_score+timestamp+
                  release_year, data=train, method="lm",                    trControl=control)
print(fit_lm$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##  (Intercept)    user_score   movie_score     timestamp  release_year
##    4.7574378     1.0003619     0.9757425    -0.0001211    -0.0005009
```

```
y_hat <- predict(fit_lm, test)
result2 <- RMSE(test$rating, y_hat)
cat("RMSE :", result2)
```

```
## RMSE : 1.042983
```

```
# So let's see how is it for LM without movieId, userId & title
t2 <- train %>% select(-c("userId", "movieId", "title"))
control <- trainControl(method = "none")
fit_lm <- train(rating~., data=t2, method="lm", trControl=control)
print(fit_lm$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
```

```
##
## Coefficients:
##               (Intercept)                  timestamp
##                 5.1836910                 -0.0002859
##                year_rated               release_year
##                        NA                 -0.0005509
##                    Action                  Adventure
##                -0.0151763                 -0.0023558
##            `\\`Sci-Fi\\``                Documentary
##                -0.0009051                  0.0479658
##                     Crime                      Drama
##                 0.0141191                  0.0158736
##                  Thriller                     Comedy
##                 0.0005888                  0.0010364
##                   Mystery                  Animation
##                 0.0048191                  0.0219648
##                  Children                    Fantasy
##                -0.0339020                  0.0136872
##                   Romance                     Horror
##                -0.0126802                  0.0049235
##                       War                    Musical
##                -0.0043971                  0.0082115
##                   Western          `\\`Film-Noir\\``
##                -0.0054717                  0.0134549
##                      IMAX  `\\`(no genres listed)\\``
##                -0.0417882                 -0.0287400
##                user_score                 movie_score
##                 1.0008362                  0.9683767
```

```r
y_hat <- predict(fit_lm, test)
result3 <- RMSE(test$rating, y_hat)
cat("RMSE :", result3)
```

```
## RMSE : 1.042491
```

```r
# Now LM without movieId, userId & title
t2 <- train %>% select(-c("userId", "movieId", "title"))
control <- trainControl(method = "none")
fit_lm <- train(rating~., data=t2, method="lm", trControl=control)
print(fit_lm$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
##               (Intercept)                  timestamp
##                 5.1836910                 -0.0002859
##                year_rated               release_year
##                        NA                 -0.0005509
##                    Action                  Adventure
##                -0.0151763                 -0.0023558
##            `\\`Sci-Fi\\``                Documentary
```

```
##                   -0.0009051                      0.0479658
##                        Crime                          Drama
##                    0.0141191                      0.0158736
##                     Thriller                         Comedy
##                    0.0005888                      0.0010364
##                      Mystery                      Animation
##                    0.0048191                      0.0219648
##                     Children                        Fantasy
##                   -0.0339020                      0.0136872
##                      Romance                         Horror
##                   -0.0126802                      0.0049235
##                          War                        Musical
##                   -0.0043971                      0.0082115
##                      Western              `\\`Film-Noir\\``
##                   -0.0054717                      0.0134549
##                         IMAX  `\\`(no genres listed)\\``
##                   -0.0417882                     -0.0287400
##                   user_score                    movie_score
##                    1.0008362                      0.9683767
```

```r
y_hat <- predict(fit_lm, test)
result3 <- RMSE(test$rating, y_hat)
cat("RMSE :", result3)
```

```
## RMSE : 1.042491
```

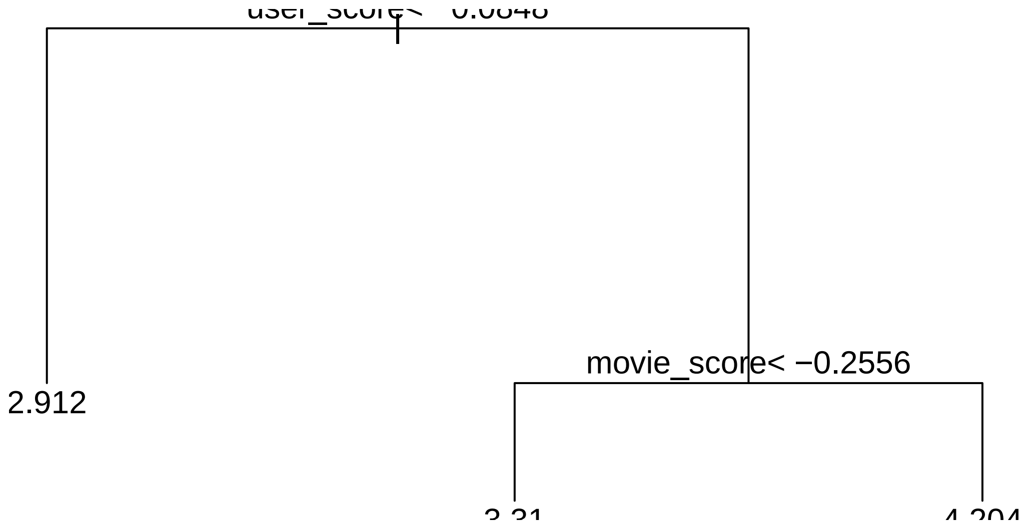
```r
#RMSE = 1.042491. It is slightly better than the baseline score, but not good enough.


#Decision Tree
fit_tree <- train(rating~user_score+movie_score+timestamp+release_year, data=train, method="rpart")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```r
print(fit_tree$results)
```

```
##           cp      RMSE  Rsquared       MAE     RMSESD  RsquaredSD      MAESD
## 1 0.07660190 0.8463595 0.3588611 0.6614992 0.02436197 0.037277092 0.01072485
## 2 0.08038463 0.8864266 0.2967925 0.6762078 0.02470256 0.039419851 0.01075139
## 3 0.24239825 0.9872902 0.2391572 0.7683068 0.07022719 0.002803921 0.08427557
```

```r
plot(fit_tree$finalModel)
text(fit_tree$finalModel)
```



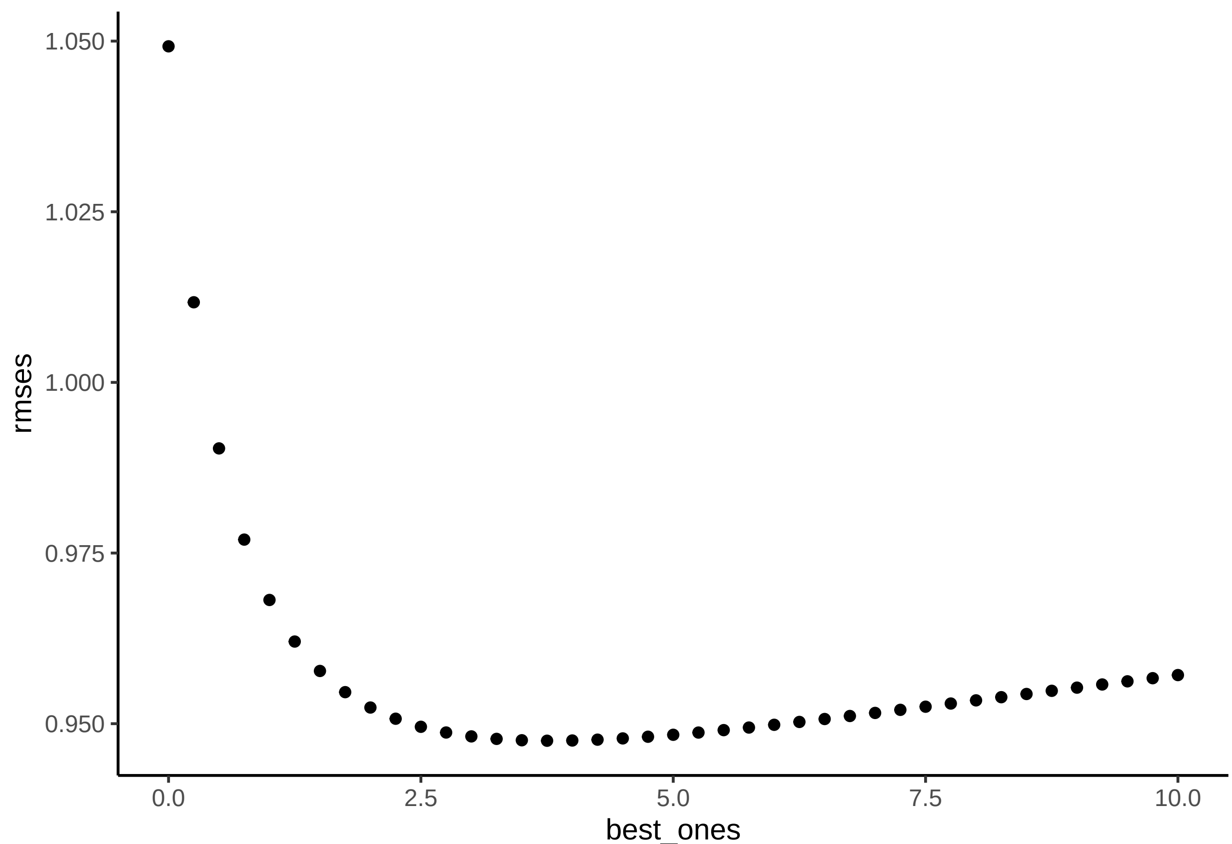

```r
# Noticing that prediction using Decision Tree is way too simple, with
# only 3 predicted ratings and few conditions.

y_hat <- predict(fit_tree, test)
result4 <- RMSE(test$rating, y_hat)
cat("RMSE :", result4)
```

```
## RMSE : 1.062504
```

```r
#The outcome is even worse than the baseline model. So let
#try the linear model with regularization


#Linear Model with regularizationwith only user_score and movie_score

#splitting the train set into 2 to calculate the best of the two
indx <- createDataPartition(train$rating, times=1, p=0.8, list=FALSE)
tpart_1 <- train[indx, ]
tpart_2 <- train[-indx, ]

#calculating the best ones
best_ones <- seq(0, 10, 0.25)
rmses <- sapply(best_ones,  function(l){
  avg_rating <- mean(tpart_1$rating)
  movie_score <- tpart_1 %>%
```

```
    group_by(movieId) %>%
    summarize(b_m = sum(rating - avg_rating)/(n()+l))
  user_score <- tpart_1 %>%
    left_join(movie_score, by="movieId") %>%
    mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_m - avg_rating)/(n()+l))
  predicted_ratings <-
    tpart_2 %>%
    left_join(movie_score, by = "movieId") %>%
    left_join(user_score, by = "userId") %>%
    mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
    mutate(b_u = ifelse(is.na(b_u), 0, b_u)) %>%
    mutate(pred = avg_rating + b_m + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, tpart_2$rating))
})

b1 <- best_ones[which.min(rmses)]
qplot(best_ones, rmses)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```r
print(b1)
```

```
## [1] 3.75
```

```r
# b1 = 3.75

# The b1 which minimizes the RMSE is 3.75, so let use it to
#train the model and predict the test set


#Prediction
b1 <- 3.75
avg_rating <- mean(train$rating)
movie_score <- train %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - avg_rating)/(n()+b1))
user_score <- train %>%
  left_join(movie_score, by="movieId") %>%
  mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - avg_rating)/(n()+b1))
predicted_ratings <-
  test %>%
  left_join(movie_score, by = "movieId") %>%
  left_join(user_score, by = "userId") %>%
  mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
  mutate(b_u = ifelse(is.na(b_u), 0, b_u)) %>%
  mutate(pred = avg_rating + b_m + b_u) %>%
  .$pred

result5 <- RMSE(test$rating, predicted_ratings)
cat("RMSE :", result5)
```

```
## RMSE :  0.9383446
```

```r
# The result of the RMSE < 1, which is much better than
# the other models. Now let use the genres columns to see how
# the predictions will be. So what is the effect of
# genre on the ratings?

not_genres <- c("userId", "movieId", "rating", "timestamp", "title", "release_year", "user_score", "movi
genres <- colnames(train)[!colnames(train) %in% not_genres]
genres
```

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

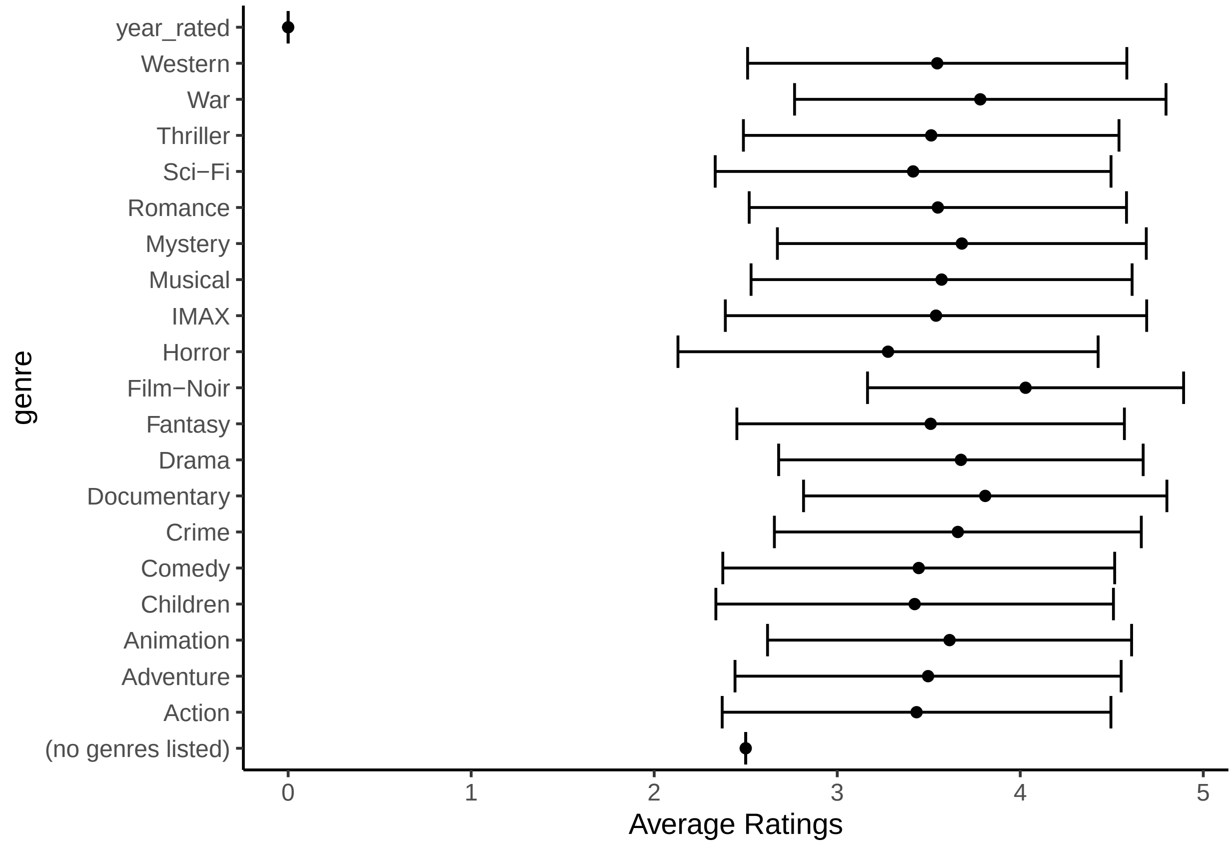
```r
#What is the average ratings for each genre? Let see.
genre_scores <- data.frame(genre="",m=0, sd=0)
for(genre in genres){
  results <- train %>% filter(train[colnames(train)==genre]==1) %>%
    summarise(m=mean(rating), sd=sd(rating))
  genre_scores <- genre_scores %>% add_row(genre=genre, m=results$m, sd=results$sd)
}
```

```
## Warning: Using one column matrices in ‘filter()‘ was deprecated in dplyr 1.1.0.
## i Please use one dimensional logical vectors instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
genre_scores <- genre_scores[-1,]
genre_scores[is.na(genre_scores)] <- 0
genre_scores
```

```
##                 genre        m        sd
## 2          year_rated 0.000000 0.0000000
## 3              Action 3.433409 1.0619524
## 4           Adventure 3.496405 1.0548618
## 5              Sci-Fi 3.414611 1.0812908
## 6         Documentary 3.808537 0.9926102
## 7               Crime 3.658984 1.0023646
## 8               Drama 3.675710 0.9958227
## 9            Thriller 3.513454 1.0260652
## 10             Comedy 3.445359 1.0705303
## 11            Mystery 3.680841 1.0078153
## 12          Animation 3.613767 0.9945636
## 13           Children 3.423029 1.0861241
## 14            Fantasy 3.510395 1.0586236
## 15            Romance 3.549792 1.0307855
## 16             Horror 3.277596 1.1477909
## 17                War 3.781634 1.0147717
## 18            Musical 3.570237 1.0409776
## 19            Western 3.546312 1.0360351
## 20          Film-Noir 4.029081 0.8635001
## 21               IMAX 3.539683 1.1510920
## 22 (no genres listed) 2.500000 0.0000000
```

```r
#Plotting genres
genre_scores %>% ggplot(aes(x=m, y=genre)) +
  geom_point() +
  xlab("Average Ratings") +
  geom_errorbarh(aes(xmin=m-sd, xmax=m+sd))
```




```
# In the plot we notice that different genres have different
# ratings average. So it is ideal to use the average of the genres
# of a movie to predict the ratings if the movie and the user
# in the test are not seen in the training set.

#The actual issue with the regularized model is if there
# is a case in the test data that has new movie and new user,
# the model can only predict with the average of all ratings.
# With the added feature of genres,  it will probably change the landscape for better
# Minimize the RMSE.

#Regularized model with genre feature
b1 <- 3.75
avg_rating <- mean(train$rating)
movie_score <- train %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - avg_rating)/(n()+b1))
user_score <- train %>%
  left_join(movie_score, by="movieId") %>%
  mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - avg_rating)/(n()+b1))
genre_score <- as.matrix(test[, genres]) %*% genre_scores$m
n_genres <- rowSums(test[,genres])
genre_score <- genre_score / n_genres
```

```r
#What is the effect of using the genre_scores if the user and
#movie are unknown?
predicted_ratings <-
  test %>%
  left_join(movie_score, by = "movieId") %>%
  left_join(user_score, by = "userId") %>%
  cbind(genre_score) %>%
  mutate(pred = genre_score) %>%
  mutate(pred = ifelse(!is.na(b_m)|!is.na(b_u),
                       avg_rating + replace_na(b_m,0) + replace_na(b_u,0),
                       pred))

result6 <- RMSE(test$rating, predicted_ratings$pred)
cat("RMSE :", result6)
```

```
## RMSE : 0.9491269
```

```r
# We notice that the improvement is very slim in RMSE
# when using genres for prediction purposes


#Put them together (table of RMSE Results)
data.frame(
  method=c("Naive Prediction", "Linear Model (with 4 features)", "Linear Model (with all features)", "De
  rmse=c(result, result2, result3, result4, result5, result6))
```

```
##                                                                    method      rmse
## 1                                                        Naive Prediction 1.0512258
## 2                                          Linear Model (with 4 features) 1.0429830
## 3                                        Linear Model (with all features) 1.0424912
## 4                                                           Decision Tree 1.0625044
## 5 Linear Model with Regularisation(only using movie and user scores) 0.9383446
## 6    Linear Model with Regularisation(movie, user, and genre scores) 0.9491269
```

Using the model on the final_holdout_test data

```
#Training the final model

# Now let apply the final model to validation set
# After dissecting to come up with different models, we can use
# the best performing model in the previous section, which is the regularized model.
# The validation data is set in a way so the users and movies in the data are all present in the
# edx data.
# So, it is not  recommended using the following:
#genre fech has both unknown user and unknown movie, not the best choice



#Using the model on the final_holdout_test data
b1 <- 3.75
avg_rating <- mean(edx$rating)
movie_score <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - avg_rating)/(n()+b1))
user_score <- edx %>%
  left_join(movie_score, by="movieId") %>%
  mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - avg_rating)/(n()+b1))
predicted_ratings <-
  final_holdout_test %>%
  left_join(movie_score, by = "movieId") %>%
  left_join(user_score, by = "userId") %>%
  mutate(b_m = ifelse(is.na(b_m), 0, b_m)) %>%
  mutate(b_u = ifelse(is.na(b_u), 0, b_u)) %>%
  mutate(pred = avg_rating + b_m + b_u) %>%
  .$pred


final_result <- RMSE(final_holdout_test$rating, predicted_ratings)
cat("RMSE :", final_result)
```

```
## RMSE : 0.8648477
```

The final RMSE is 0.8648477 which is the required RMSE (0.8649) to get the maximum point for the EDX Capstone Movielens projects.

Conclusion

This MovieLens dataset model provided by edx is a beautiful project to work on. Indeed outcomes can be quite different, close to perfect meaning RMSE could almost be second to none if average raters(users) don't play bias, meaning user doesn't rate a particularly good/popular movie with a large margin bi, and vice versa. Probabilistically speaking if my statement about RMSE is not impossible, then it has to be probable, even if the chance is infinitesimally small.

Polymorphically speaking, I used quite a few machine learning algorithms to come up with predictions movie ratings for this MovieLens dataset. As expected results are different from one another for RMSE. Among those steps (algorithms) used. The regularized model would be ideal with the users side effect to lower RMSE.

Is there a better way or much room for improvement in a nutshell? Off course yes, because life is polymorphic but I haven't come across it yet.

Hence this model can be improved by adding other accoutrement(age, year, genre,...) and on how users should/could rate movies. Also we can apply different machine learning models to improve, hence a better polished outcome.