MovieLens Report

Francis Angonesse

6/3/2019

Project Overview

The main goal of this project is to demonstrate that ive acquired skills with R programming language being able to apply on a real dataset, which are composed by millions of data inputs generated by users. By applying machine learning with the right algorithm we can predict some patterns and create a recommender system.

Used Libraries

The following libraries were used in this project. NOTE: Since the author are learning, he tryed to use as many as possible libraries in order to get more knowledge.

```
library(tidyverse)
library(caret)
library(data.table)
library(kableExtra)
library(lubridate)
library(Matrix.utils)
library(DT)
library(wordcloud)
library(RColorBrewer)
library(ggthemes)
library(irlba)
library(recommenderlab)
library(recosystem)
library(h2o)
library("mgcv")
library("nlme")
library("nnet")
library("spatial")
library("survival")
library(lattice)
library(magrittr)
library(dplyr)
library(ggplot2)
library(lattice)
library(plotly)
library(latticeExtra)
library(dplyr)
library(magrittr)
library(knitr)
```

Data Sets

Working with .rds data, provided by edx on 5/6/19 on https://drive.google.com/drive/folders/1IZcBBX0OmL9wu9AdzMBFUG8GoPbGQ38D (https://drive.google.com/drive/folders/1IZcBBX0OmL9wu9AdzMBFUG8GoPbGQ38D) The dataset used are a “shorter” version of the MovieLens public library.

```
edx <- readRDS("~/Desktop/edx/edx.rds")
```

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474
838983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Star
gate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-F
i|Thriller" "Action|Adventure|Sci-Fi" ...
```

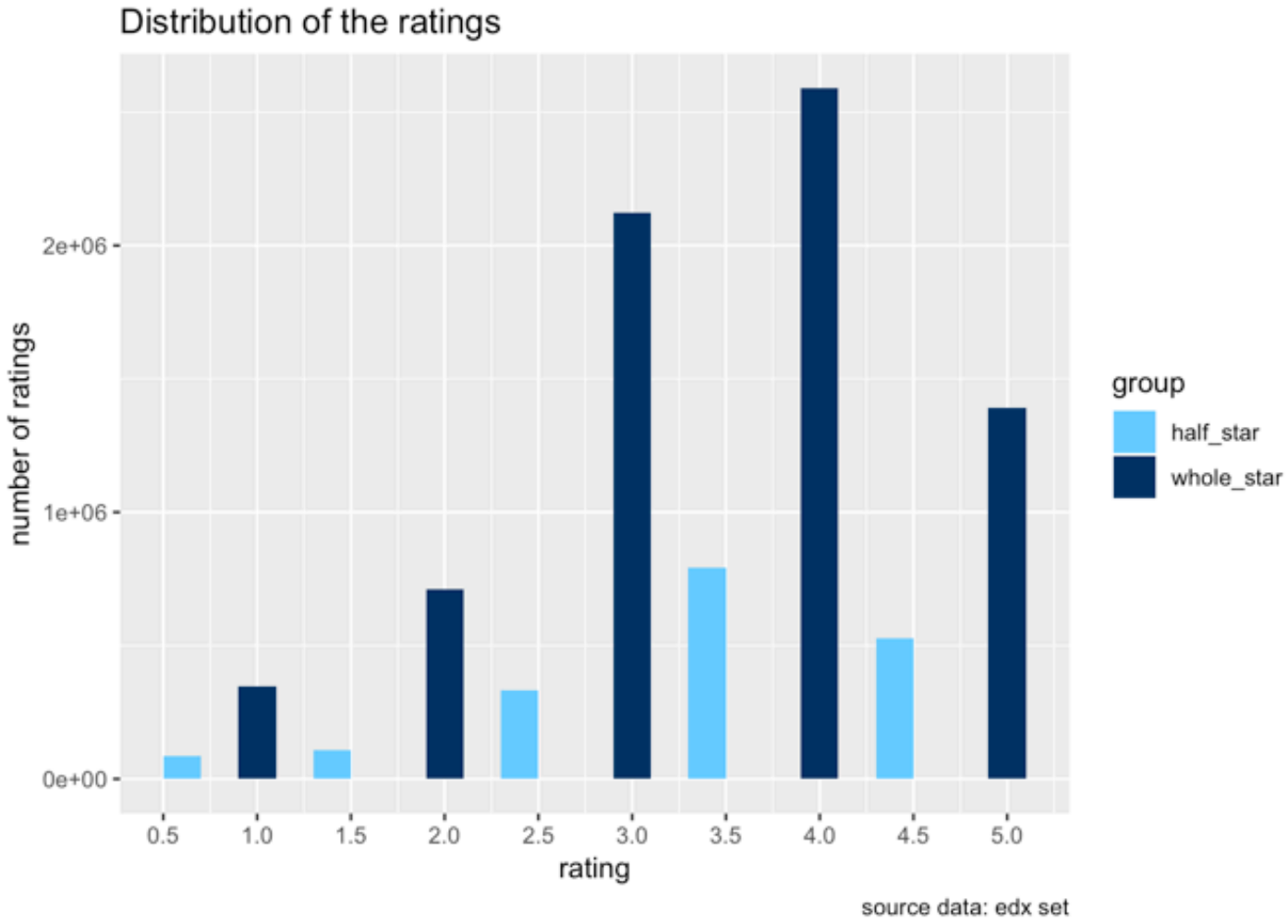
By running str(edx) it will display the internal structure of the R object. We can observ that the object holds 6 “variables”.

Explore ratings

We set the ratings as “complete values” and “half values”

```
start <- edx$rating
genres <-(edx$genres)


# Create a dataframe to sort half and full ratings
group <-  ifelse((start == 1 |start == 2 | start == 3 |
                    start == 4 | start == 5) ,
                 "whole_star",
                 "half_star")

explore_ratings <- data.frame(start, group)
```

```
ggplot(explore_ratings, aes(start, fill = group)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  scale_fill_manual(values = c("half_star"="#66CCFF", "whole_star"="#003366")) +
  labs(x="rating", y="number of ratings", caption = "source data: edx set") +
  ggtitle("Distribution of the ratings")
```



We can observ that users have the tendency to give full ratings more than half ratings, and the rate “4” are the most present.

Separating combinated genres

Since the data frame has combine genres of movies, we write this script to separate it and understand each genre rate separated

```
top_genr <- edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
top_genr
```

```
## # 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
```

```
knitr::kable(head(top_genr, 10))
```

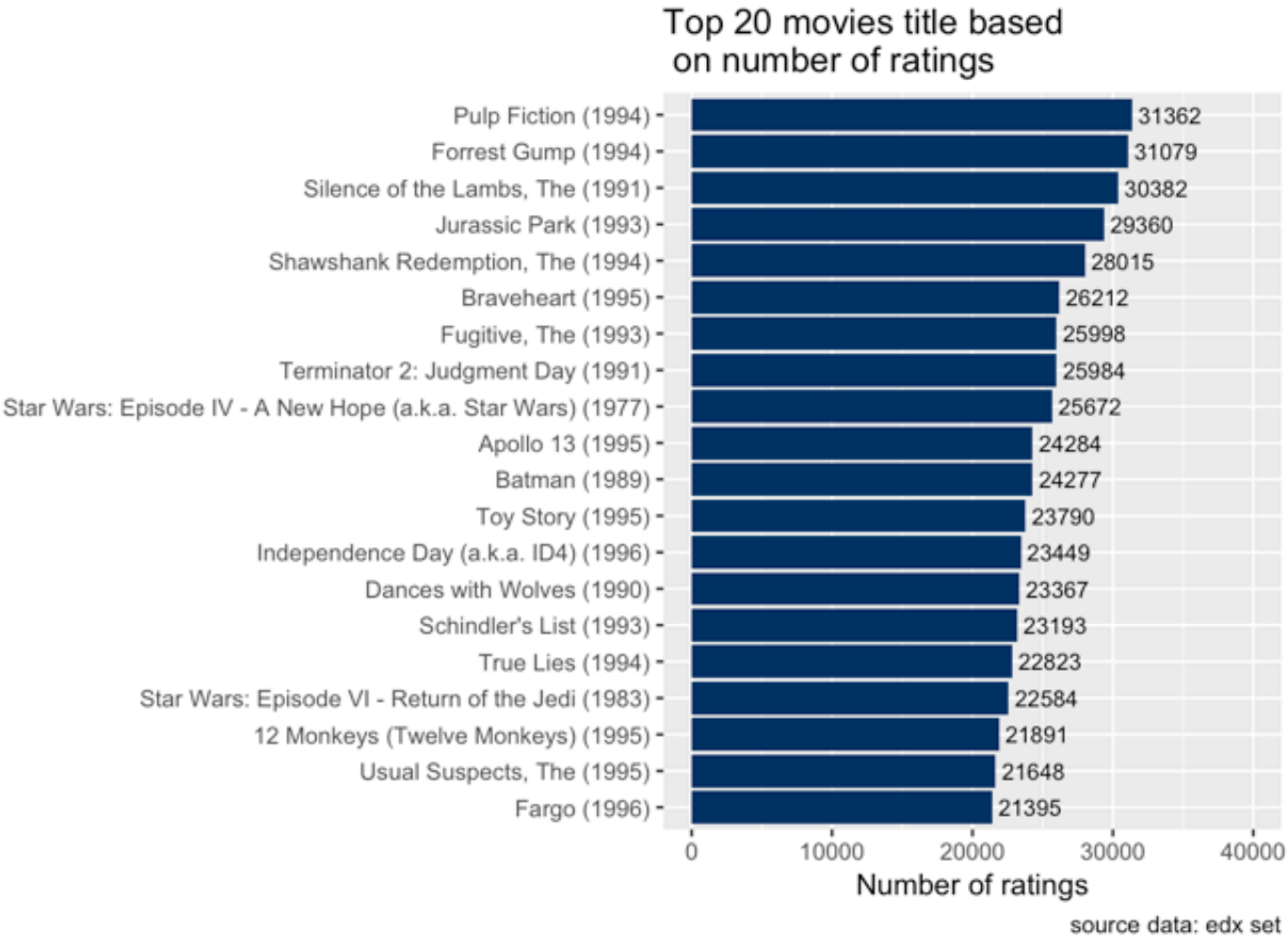
genres	count
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994

Top 20 Movies based on user rating

```
top_title <- edx %>%

  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(20,count) %>%
  arrange(desc(count))

# with the head function i output the top 5

kable(head(edx %>%
             group_by(title,genres) %>%
             summarize(count=n()) %>%
             top_n(20,count) %>%
             arrange(desc(count)) ,
           5)) %>%
  kable_styling(bootstrap_options = "bordered", full_width = F , position ="center
") %>%
  column_spec(1,bold = T ) %>%
  column_spec(2,bold =T) %>%
  column_spec(3,bold=T)
```

title	genres	count
Pulp Fiction (1994)	**Comedy|Crime|Drama**	**31362**
Forrest Gump (1994)	**Comedy|Drama|Romance|War**	**31079**
Silence of the Lambs, The (1991)	**Crime|Horror|Thriller**	**30382**
Jurassic Park (1993)	**Action|Adventure|Sci-Fi|Thriller**	**29360**
Shawshank Redemption, The (1994)	**Drama**	**28015**

```
top_title %>%
  ggplot(aes(x=reorder(title, count), y=count)) +
  geom_bar(stat='identity', fill="#003366") + coord_flip(y=c(0, 40000)) +
  labs(x="", y="Number of ratings") +
  geom_text(aes(label= count), hjust=-0.1, size=3) +
  labs(title="Top 20 movies title based \n on number of ratings" , caption = "sour
ce data: edx set")
```



We can see the number of unique users that provided ratings and how many unique movies were rated:

```
edx %>%
  summarize(n_users = n_distinct(userId),
  n_movies = n_distinct(movieId))
```

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

The RMSE is then defined as:

<

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

```
# i calculate the average of all ratings of the edx set
mu <- mean(edx$rating)
movieId <- (edx$movieId)
# i calculate b_i on the training set
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# predicted ratings
predicted_ratings_bi <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i


#b.movie + user effect

#i calculate b_u using the training set
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#predicted ratings
predicted_ratings_bu <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred


#c.movie + user + time effect

#i create a copy of validation set , valid, and create the date feature which is t
he timestamp converted to a datetime object  and  rounded by week.

valid <- validation
valid <- valid %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week"))

# i calculate time effects ( b_t) using the training set
temp_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u))

# predicted ratings
  predicted_ratings_bt <- valid %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  left_join(user_avgs, by='userId') %>%
  left_join(temp_avgs, by='date') %>%
  mutate(pred = mu + b_i + b_u + b_t) %>%
  .$pred

#d.  i calculate the RMSE for movies, users and time effects

rmse_model1 <- RMSE(validation$rating,predicted_ratings_bi)
rmse_model1
[1] 0.9437046

rmse_model2 <- RMSE(validation$rating,predicted_ratings_bu)
rmse_model2
[1]0.8653488

rmse_model3 <- RMSE(valid$rating,predicted_ratings_bt)
rmse_model3
[1]0.8652511
```