Movielens Project

Ha Duc Vinh

April 12, 2019

Summary

This is an R Markdown document for performing analysis of MovieLense Data and to recommend the new / untried movies to users. We explore the the different algorithms and will give conclusion on the best algorithm that can be used

This document is part of the course work for edx Havardx Data Science Capstone

Method/Analysis

There are 4 methods that will be use for our analysis and the best one will be recommended at the conclusion section of this document.

The details steps include:

Step 1: Downloading Movielens Data

Step 2: Splitting Data into 2 set: Training(90% of data), Test(10% of data)

Step 3: Exploring 1st algorithm: Using Average of rating in the training set as a basis to predict rating in the test set

Step 4: Exploring 2nd Algorithm: Calculate Average + Movie Effect on the training set and apply it to predict rating in the test set

Step 5: Exploring 3rd Algorithm: Calculate Average + Movie Effect + User effect on the training set and apply it to predict rating on the test set

Step 6: Exploring 4th Algorithm: Calculaate Average + Movie Effect + User effect + Genres Effect on the training set and apply it to predict rating on the test set

Exploration and Results

Dowloading and Creation of Test set and Training Set

This part help to create the test set and training set from the movielens data set. The test set is 10% of the total data, and the training set is 90% of the total data

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0       v purrr   0.3.1
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- 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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

Training/ Testing Set preparation

The testing set if called Validation and the training set is called edx. The edx training set will be used to train different algorithm and the test set validation will be used to evaluate how effective the algorithm is in determine the predicted rating of users for those movie that they have not watched.

```
# 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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

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

Function to determine RMSE

This section create function called RMSE used in calculating the score RMSE for different algorithm. The best algorithm is the one with lowest RMSE.

```
#------------------------ START OF PROJECT--------------------------



#Define RMSE function used to calculate testing of algorithm
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Exploring 1st Algorithm: Average

This section explores simple algorithm simply by considering average rating accross all movie and predict those movie that does not has rating by any user using that average. The resulting RMSE is around 1.0612

```
#----- 1st Algorithm Just average prediction ---

#Calculate average rating accorss all rating
mu_hat <- mean(edx$rating)

#Calculate naive RMSE if mu_hat is used to predict all rating on validation set
naive_rmse <- RMSE(validation$rating, mu_hat)

#Create RMSE results table to store all RMSE for different algorithm
rmse_results <- data_frame(method = "1st: Just the average", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
#Display RMSE for Naive RMSE alsorithm (using average to predict movie rating)
options(pillar.sigfig = 6)
pillar::pillar(rmse_results)
```

```
## <tibble>
## # A tibble: 1 x 2
##   method                   RMSE
##   <chr>                   <dbl>
## 1 1st: Just the average 1.06120
```

Exploring 2nd Algorithm: Average + Movie effect

We are trying to improve the previous algorithm. This time, we will calculate the movie effect on the training set and then try to predict movie rating of each individual user in the test set using that effect. The improvement is significant, RMSE = 0.9439 compared to 1.0612 in the previous one.

```
#------ 2nd Algorithm Movie Effect prediction ---------------------------

#Calculate the Movie effect which influence the rating
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))

#Using Movie effect to calculate rating prediction on validation set
predicted_ratings <- mu_hat + validation%>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

#Calculate RMSE result and store in RMSE result table
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="2nd: Movie Effect Model",
                                     RMSE = model_1_rmse ))

#Display RMSE result for 1st and 2nd algorithm
#Notice significant better result in 2nd Algoritm
options(pillar.sigfig = 6)
pillar::pillar(rmse_results)
```

```
## <tibble>
## # A tibble: 2 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 1st: Just the average   1.06120
## 2 2nd: Movie Effect Model 0.943909
```

Exploring 3rd Algorithm: Average + Movie effect + User Effect

To improve the result further, we exploring the user effect on the training set.The improvement to our algorithm is now reflected on RMSE with value RMSE = 0.8653

```
#------- 3rd Algorithm Movie Effect + User Effect prediction -----


#Calculate the user effect which affect the final rating:
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

#Using Movie effect and User effect to predict the rating in validation set
predicted_ratings <-mu_hat + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred =  b_i + b_u) %>%
  .$pred

#Calculate RMSE result and store in RMSE result table
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="3rd: Movie + User Effects Model",
                                     RMSE = model_2_rmse ))



#Display RMSE result for 1st, 2nd and 3rd algorithm
#Notice significant better result in 3rd Algoritm
options(pillar.sigfig = 6)
pillar::pillar(rmse_results)
```

```
## <tibble>
## # A tibble: 3 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 1st: Just the average           1.06120
## 2 2nd: Movie Effect Model         0.943909
## 3 3rd: Movie + User Effects Model 0.865349
```

Exploring 4th Algorithm: Average + Movie effect + User Effect + Genres Effect

“COnsidering Genres as annother important factor influence the raing of users. There is additional improvement to RMSE, it is now stand at 0.8649”

```
#-------4th Algorithm Movie, User and Genres Effect prediction------------------------------



#Calculate the genres effect which influence the rating
genres_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs,by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu_hat - b_i-b_u))

#Using Movie effect,User effect and Genres to predict the rating in validation set
temp <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId')

temp2<-left_join(temp,genres_avgs,by = 'genres')
predicted_ratings <- mu_hat +temp2 %>% mutate(pred = b_i+b_u+b_g) %>% .$pred



#Calculate RMSE result and store in RMSE result table
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="4th: Movie + User Effects Model + Genres",
                                     RMSE = model_2_rmse ))

#Display RMSE result for 1st, 2nd, 3rd  and 4th algorithm
#Notice significant better result in 4th Algoritm
options(pillar.sigfig = 6)
pillar::pillar(rmse_results)
```

```
## <tibble>
## # A tibble: 4 x 2
##   method                                       RMSE
##   <chr>                                       <dbl>
## 1 1st: Just the average                     1.06120
## 2 2nd: Movie Effect Model                   0.943909
## 3 3rd: Movie + User Effects Model           0.865349
## 4 4th: Movie + User Effects Model + Genres 0.864947
```

Conclusion:

From above 4 algorithm, the algorithm that perform bestis the 4th one, which account for Movie + User + Genres Effect with RMSE of 0.864