capstone_movielens

Savas turkoglu

5/15/2020

##1. Executive summary This project created for Edx HarcardX data Science Certification Program capstone project. This project tries to create a model that estimate people rating by provided dataset that 10m movies and ratings

This document contains data analysis and prediction models.

The next section shows the results of the previous process and then, the conclusions of the project are given.

Code provided by the edx staff to download an create edx dataset.

```
#Create test and validation sets
# Create edx set, validation set, and submission file
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")
# 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")
# 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)
```

```r
library(caret)
library(anytime)
library(tidyverse)
library(lubridate)
```

1. The Dataset The data set contains 9000055 observations of 6 variables.

- `userId`: Unique identification number given to each user. -numneric
- `movieId`: Unique identification number given to each movie. -numeric
- `timestamp`: rating date and time datestams -integrer
- `title`: Title of the movies. -numerşc
- `genres`: Film categories. -char
- `rating`: Rating given by the user -char

2. Analysis Section

2 a Data preperation

Edx gave us prepared data but wee nedd some preperation we have to extract datestams to human readable format and we have to get year and month from date

we have to sepera title into the film name and release year

we have to split genres column

```r
# my computer sources was not enough for proccess 10m rows data,
#therefore i used  dataset partially during development

 edx <- head(edx,1000000)  #if you get memory error use this code
```

Data preparation Edx gave us prepared data but we need some more preparation !!

First we have to dealing with date

Converting date-time to human readable

```r
edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01")
validation$date <- as.POSIXct(validation$timestamp, origin="1970-01-01")

## get year and month from date

edx$rate_year <- format(edx$date,"%Y")
edx$rate_month <- format(edx$date,"%m")

validation$rate_year <- format(validation$date,"%Y")
validation$rate_month <- format(validation$date,"%m")
```

now wee'll deal with title we have to separate title and release year

```r
## + in edx datasets

edx <- edx %>%
  mutate(title = str_trim(title)) %>%
  extract(title, c("title_", "release"), regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = F) %>%
```

```r
  mutate(release = if_else(str_length(release) > 4,
                           as.integer(str_split(release, "-",
                                                simplify = T)[1]),
                           as.integer(release))
  ) %>%
  mutate(title = if_else(is.na(title_), title, title_)
  )
edx <- edx %>% select(-title_)

## + in validaton datasets

validation <- validation %>%
  mutate(title = str_trim(title)) %>%
  extract(title, c("title_", "release"), regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = F) %>%
  mutate(release = if_else( str_length(release) > 4,
                            as.integer(str_split(release, "-",
                            simplify = T)[1]),
                            as.integer(release))) %>%
  mutate(title = if_else(is.na(title_), title, title_))

validation <- validation %>% select(-title_)
```

we have to modify the genres vars in the edx & validation dataset by column_separated

```r
##split daa

genre_split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
genre_split_valid <- validation  %>% separate_rows(genres, sep = "\\|")

genres<- genre_split_edx %>%
  group_by(genres) %>%
  summarize(count = n())

genres<- genres  %>% arrange(desc(count))
genres
```

```
## # A tibble: 20 x 2
##    genres              count
##    <chr>               <int>
##  1 Drama              435586
##  2 Comedy             393165
##  3 Action             284539
##  4 Thriller           259607
##  5 Adventure          213558
##  6 Romance            190509
##  7 Sci-Fi             149767
##  8 Crime              147954
##  9 Fantasy            102669
## 10 Children            82246
## 11 Horror              75760
## 12 Mystery             63487
## 13 War                 56656
## 14 Animation           51693
```

```
## 15 Musical             47991
## 16 Western             20630
## 17 Film-Noir           13229
## 18 Documentary          9679
## 19 IMAX                  805
## 20 (no genres listed)      1
```

```
genre_names <- genres$genres




## remove unnecessary column from datasets

edx <- edx %>% select(-date, -timestamp)
```

```
head(edx)
```

```
##   userId movieId rating                  title release
## 1      1     122      5              Boomerang    1992
## 2      1     185      5               Net, The    1995
## 3      1     231      5          Dumb & Dumber    1994
## 4      1     292      5               Outbreak    1995
## 5      1     316      5               Stargate    1994
## 6      1     329      5 Star Trek: Generations    1994
##                          genres rate_year rate_month
## 1                Comedy|Romance      1996         08
## 2         Action|Crime|Thriller      1996         08
## 3                        Comedy      1996         08
## 4  Action|Drama|Sci-Fi|Thriller      1996         08
## 5       Action|Adventure|Sci-Fi      1996         08
## 6 Action|Adventure|Drama|Sci-Fi      1996         08
```

2.b Data Exploratory Analysis.

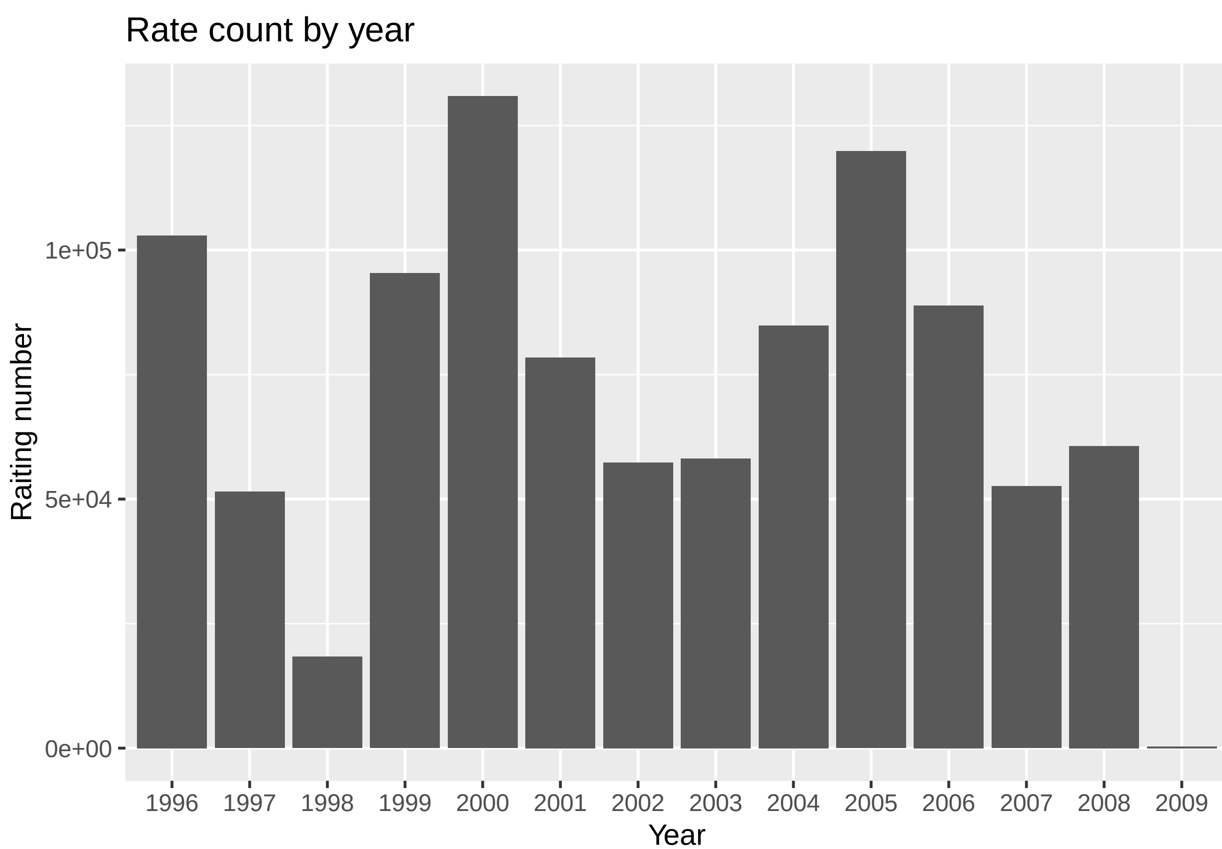
DATA OVERVIEW

```
##  take look to data by date
edx_year <- edx %>% group_by(rate_year) %>%
     summarize(total_rate = sum(rating), count = n())

 ## sort by total rate

 edx_year %>% arrange(desc(total_rate)) %>% knitr::kable()
```

rate_year	total_rate	count
2000	471456.0	130887
2005	408596.0	119829
1996	366700.0	102936
1999	344519.0	95332
2006	307916.0	88858
2004	291100.0	84821

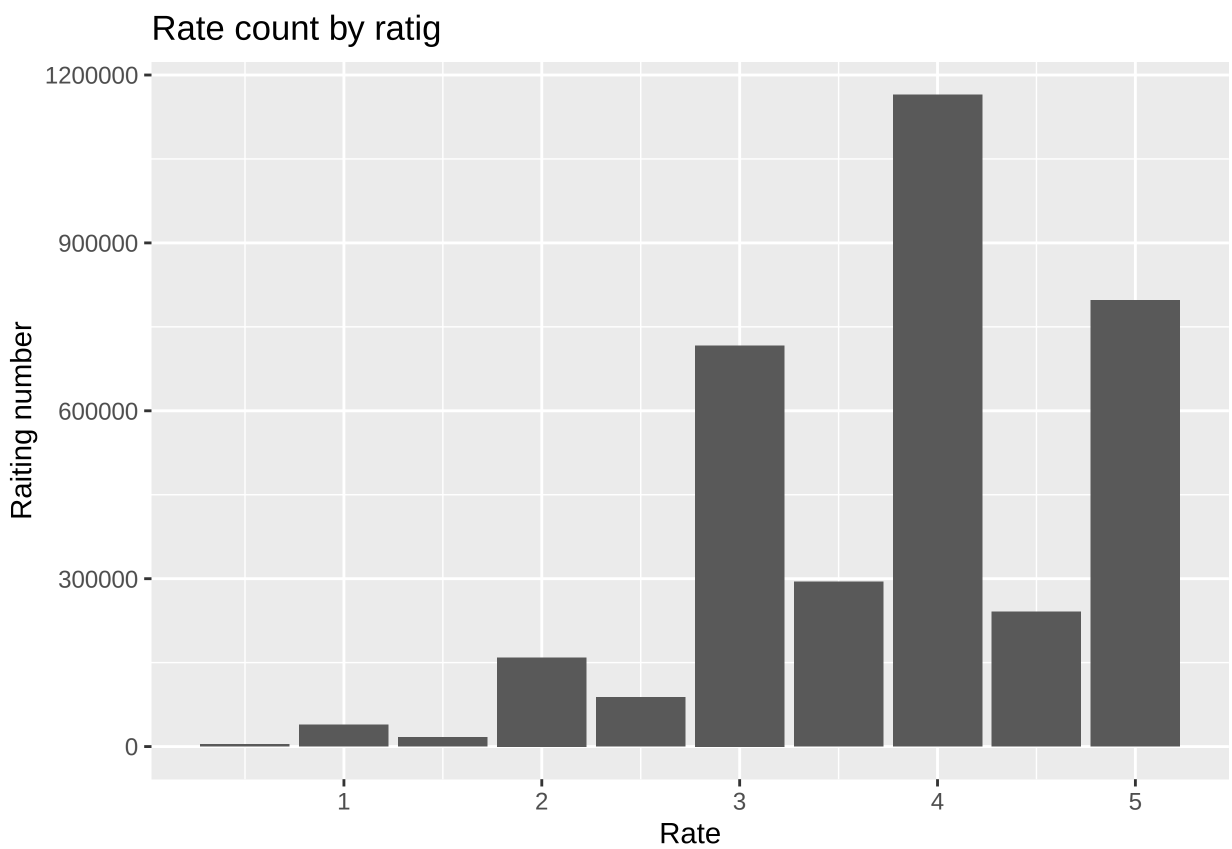
rate_year	total_rate	count
2001	278072.0	78443
2008	214614.5	60687
2003	202813.0	58130
2002	199492.0	57329
2007	185515.0	52583
1997	184625.0	51476
1998	64358.0	18328
2009	1213.5	361

```r
## plot date - raiting count
ggplot(edx_year, aes(y=count, x=rate_year)) +
    geom_bar(stat = "identity")+
    labs(title = "Rate count by year ", x = "Year", y = "Raiting number")
```

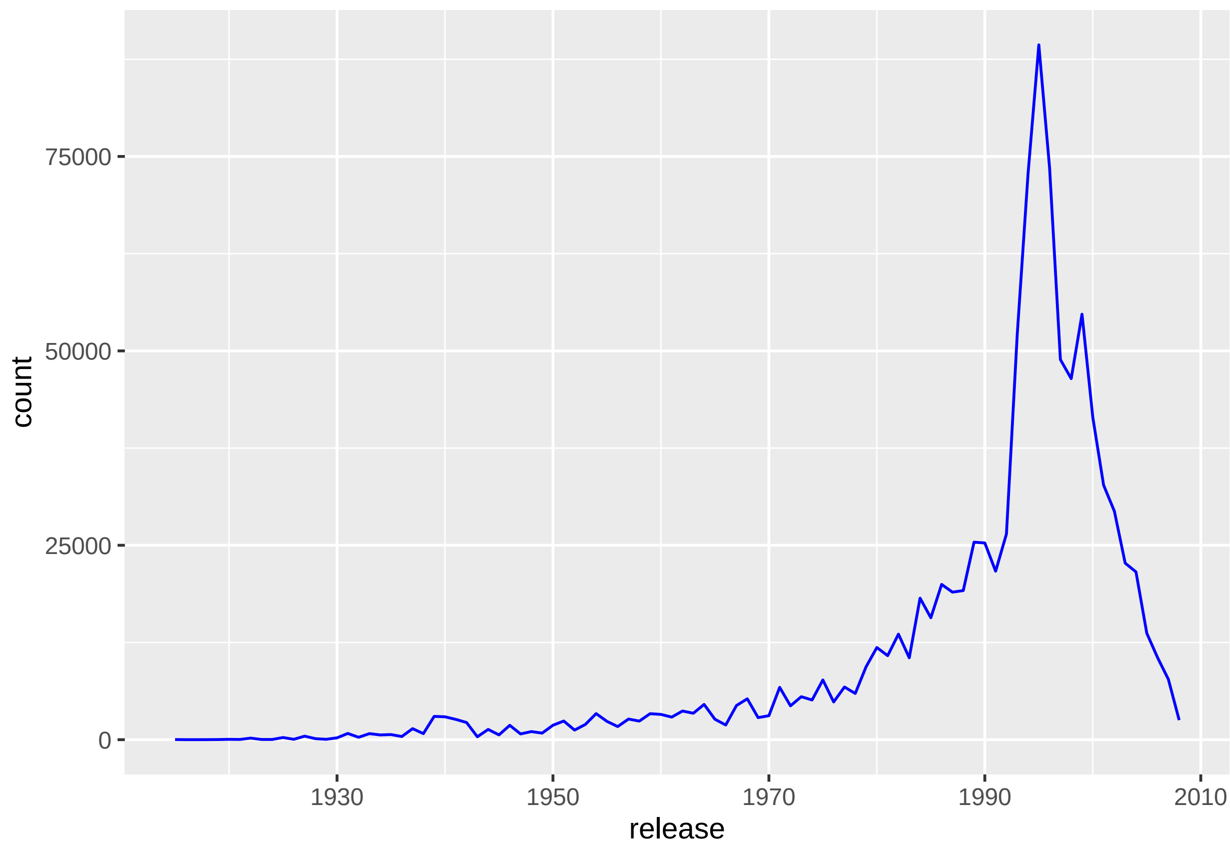



```r
## take look by rate  (1, 1.5, 2)
edx_by_rating <- edx %>% group_by(rating) %>%
  summarize(total_rate = sum(rating), count = n())



## plot rate - raiting count
ggplot(edx_by_rating, aes(x=rating, y=total_rate)) +
  geom_bar(stat = "identity")+
```

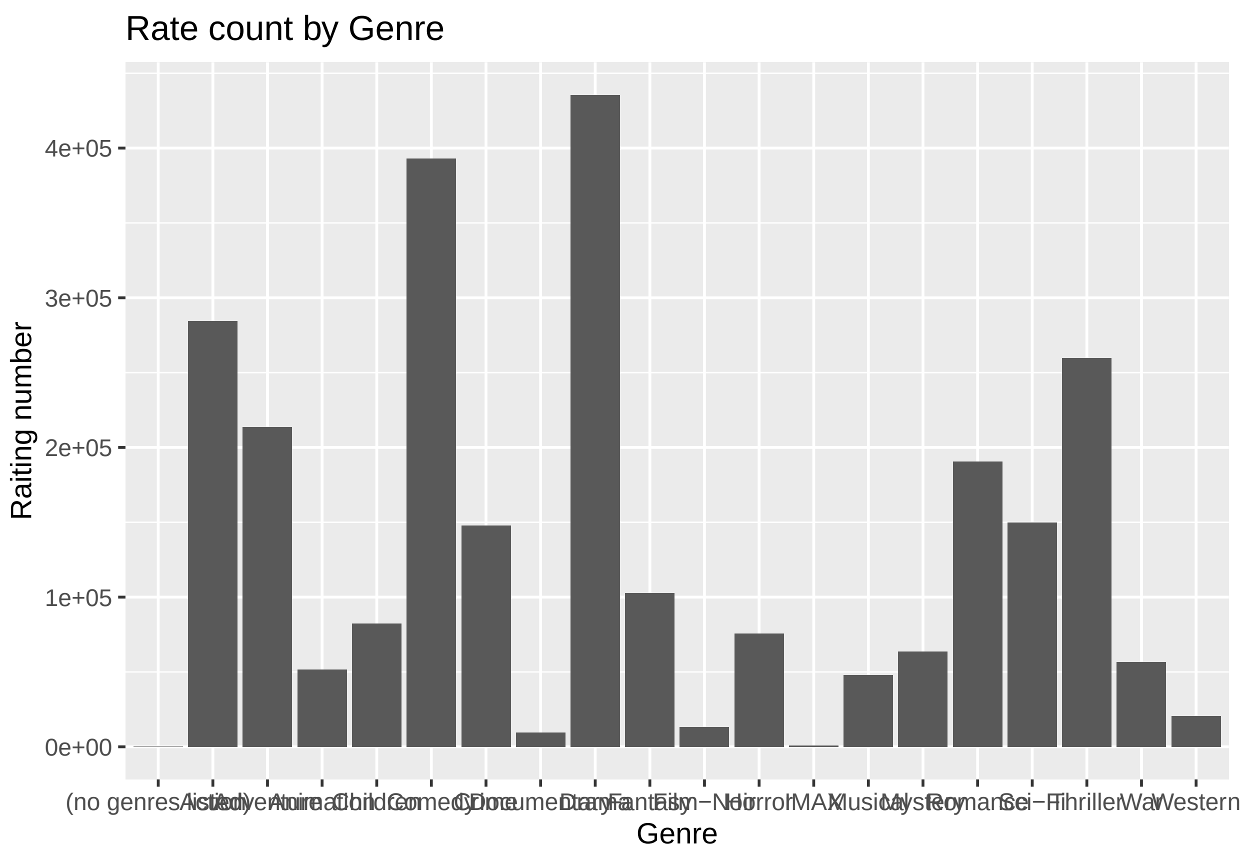
```
  labs(title = "Rate count by ratig ", x = "Rate", y = "Raiting number")
```




```
## rateing count by release year

movies_per_year <- edx %>%
  select(movieId, release) %>%
  group_by(release) %>%
  summarise(count = n())  %>%
  arrange(release)

# plot
movies_per_year %>%
  ggplot(aes(x = release, y = count)) +
  geom_line(color="blue")
```




```r
## take look by genre
edx_by_genre <- genre_split_edx %>% group_by(genres) %>%
  summarize(total_rate = sum(rating), count = n())



## plot genre  - raiting count
ggplot(edx_by_genre, aes(y=count, x=genres)) +
  geom_bar(stat = "identity")+
  labs(title = "Rate count by Genre ", x = "Genre", y = "Raiting number")
```



3-Modelling

```
# we'll use this RMSE
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2,na.rm = T))
}


## Dataset's mean rating is used to predict the same rating for all movies
mu <- mean(edx$rating)
mu
```

```
## [1] 3.52099
```

```
### BASELINE MODEL ###
baseline_model_rmse <- RMSE(validation$rating,mu)
## Test results
baseline_model_rmse
```

```
## [1] 1.060688
```

```
## create table that we vcollect methods and RMSEs

rmse_results <- data_frame(method = "Baseline model", RMSE =  baseline_model_rmse)
```

```r
#check
rmse_results
```

```
## # A tibble: 1 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Baseline model  1.06
```

```r
### MOVIE  MODEL ###
#rating  are different effected from different movies


movie_av <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu))


#rediction

pred_movie_av <- validation %>%
   left_join(movie_av, by='movieId') %>%
   mutate(pred = mu + b_i)


# calculate RMSE
model_movie_av_rmse <- RMSE(validation$rating,pred_movie_av$pred)
model_movie_av_rmse
```

```
## [1] 0.9485734
```

```r
# add to table
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE =  model_movie_av_rmse ))
# check results
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline model	1.0606883
Movie Effect Model	0.9485734

USER AND MOVIE MODEL

different users can interest different movies and this could effect to rating

```r
user_av <- edx %>%
   left_join(movie_av, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu - b_i))
```

```r
# prediction
pred_movie_user <- validation %>%
   left_join(movie_av, by='movieId') %>%
   left_join(user_av, by='userId') %>%
   mutate(pred = mu + b_i + b_u) %>% .$pred


# calculate RMSE
model_user_movie_rmse <- RMSE(validation$rating,  pred_movie_user)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="User & Movie Model",
                                     RMSE = model_user_movie_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline model	1.0606883
Movie Effect Model	0.9485734
User & Movie Model	0.8694943

REGULARİZATİON USİNG USER AND MOVİE

We have learned on Edx Machine learning course #Regularization is very usefull method to imprıve our results we'll use this method #using user and movie

```r
lambdas <- seq(0, 10, 0.25)

# this loop will take time
rmses <- sapply(lambdas, function(l){

   mu <- mean(edx$rating)

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

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

   pred_rating <- validation %>%
      left_join(b_i, by = "movieId") %>%
      left_join(b_u, by = "userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      .$pred

   return(RMSE(validation$rating,pred_rating))
})

# now we'ill get lowest value from rmses
```

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

```
## [1] 3.5
```

```r
# now we can compute regularized estimates of b_i using this lowest lambda
mov_av_reg <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

# also we can compute regularized estimates of b_u using  this lowest lambda
user_av_reg <- edx %>%
   left_join(mov_av_reg, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())


# Prediction
pred_ratings_reg <- validation %>%
   left_join(mov_av_reg, by='movieId') %>%
   left_join(user_av_reg, by='userId') %>%
   mutate(pred = mu + b_i + b_u) %>%     .$pred


#  get rmse
reg_rmse <- RMSE(validation$rating,pred_ratings_reg)

# add to table
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regulariz User & Movie Model",
                                     RMSE = reg_rmse ))

#check results
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline model	1.0606883
Movie Effect Model	0.9485734
User & Movie Model	0.8694943
Regulariz User & Movie Model	0.8677770

REGULARİZATİON USER , MOVIE , YEAR, GENRE ###

```r
lambdas <- seq(0, 10, 0.5)
# this part will take time
rmses <- sapply(lambdas, function(l){

   mu <- mean(edx$rating)

   b_i <-  genre_split_edx %>%
      group_by(movieId) %>%
```

```r
      summarize(b_i = sum(rating - mu)/(n()+l))

   b_u <-  genre_split_edx %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu)/(n()+l))

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

   b_g <-  genre_split_edx %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      left_join(b_y, by = 'release') %>%
      group_by(genres) %>%
      summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda), n_g = n())

   # prediction
   reg_predict <- genre_split_valid %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      left_join(b_y, by = 'release') %>%
      left_join(b_g, by = 'genres') %>%
      mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
      .$pred

   return(RMSE(genre_split_valid$rating,reg_predict))
})


# now we'll get lowest value from rmses
lambda_all <- lambdas[which.min(rmses)]
lambda_all
```

```
## [1] 8
```

```r
# now we can copmute rmse with tihs lowet lambda value


movie_reg_av_ <- genre_split_edx %>%
   group_by(movieId) %>%
   summarize(b_i = sum(rating - mu)/(n()+lambda_all), n_i = n())

user_reg_av_ <- genre_split_edx %>%
   left_join(movie_reg_av_, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = sum(rating - mu - b_i)/(n()+lambda_all), n_u = n())

year_reg_av_ <- genre_split_edx %>%
   left_join(movie_reg_av_, by='movieId') %>%
```

```r
  left_join(user_reg_av_, by='userId') %>%
  group_by(release) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda_all), n_y = n())

genre_reg_av_ <- genre_split_edx %>%
  left_join(movie_reg_av_, by='movieId') %>%
  left_join(user_reg_av_, by='userId') %>%
  left_join(year_reg_av_, by = 'release') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u - b_y)/(n()+lambda_all), n_g = n())


#prediction
req_pred <- genre_split_valid %>%
  left_join(movie_reg_av_, by='movieId') %>%
  left_join(user_reg_av_, by='userId') %>%
  left_join(year_reg_av_, by = 'release') %>%
  left_join(genre_reg_av_, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  .$pred

#compute rmse
model_4_rmse <- RMSE(genre_split_valid$rating,req_pred)


rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regulariztion User, Year,  Movie, Genre  Model",
                                     RMSE = model_4_rmse ))
```

4. Results

```r
rmse_results %>% knitr::kable()
```

method	RMSE
Baseline model	1.0606883
Movie Effect Model	0.9485734
User & Movie Model	0.8694943
Regulariz User & Movie Model	0.8677770
Regulariztion User, Year, Movie, Genre Model	0.8644141

##5. Conclusion The variables `userId` and `movieId` have sufficient predictive power to permit us to predict how a user will rate a movie. Therefore, the user could decide to spend more time using the service.

My project Github repository is **in this link**