Project Submission: Movilelense

```
Title: "Movielens Project"
Author: "Ashraf Masadeh"
Date: "November 23, 2020"
```

Introduction

This report is part of the capstone project for the HarvardX Online Program Professional Certificate in Data Science The purpose of this project is to demonstrate that I acquired the skills in R language in the field of data science throughout the courses in this series , I will be creating a movie recommendation system using the MovieLens dataset.

Part(1/3): Create Train and Validation Sets

The original datatset consists of 10M version of the movielens dataset, and beause of the limited ablitiltis to our computer we will be downloading part of it, therefore we will use the following code to generate the datasets

```
################################
# Create edx set, validation set
################################

# 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 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- 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
```

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

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

```
# 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 <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
 colnames(movies) <- c("movieId", "title", "genres")
 movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

```
#save the and load the tables
#write.table(edx,file = 'edx.csv')
#data(edx)
```

Part(2/3): Data Exploration

Q1. How many rows and columns are there in the edx dataset? The follwoing small table shows the dimention of the data set, which is the number of rows and columns

```
edx %>% summarize(rows = nrow(edx), columns= ncol(edx)) %>%
  select(rows, columns)
```

```
##      rows columns
## 1 9000055       6
```

Q2. How many zeros were given as ratings in the edx dataset? there is a number of `sum(edx$rating[] == 0)` movies with zero star rating below is the code:

```
sum(edx$rating[] == 0)
```

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

Q3.How many threes were given as ratings in the edx dataset? there is a number of `sum(edx$rating[] == 3)` movies with 3 star rating below is the code:

```
sum(edx$rating[] == 3)
```

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

Q4. How many different movies are in the edx dataset? there is `summarize(c, n_movies = n_distinct(movieId))` movies in the edx dataset, below is the code:

```
summarize(edx, n_movies = n_distinct(movieId))
```

```
##   n_movies
## 1    10677
```

Q5.How many different users are in the edx dataset?? there is a number of `summarize(edx, n_users = n_distinct(userId))` users in the dataset, below is the code:

```
summarize(edx, n_users = n_distinct(userId))
```

```
##   n_users
## 1   69878
```

Q7. Which movie has the greatest number of ratings?

```
edx %>%
  group_by(movieId, title) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

```
## `summarise()` regrouping output by 'movieId' (override with `.groups` argument)
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title count
##      <dbl> <chr> <int>
##  1     296 <NA>  31362
##  2     356 <NA>  31079
##  3     593 <NA>  30382
##  4     480 <NA>  29360
##  5     318 <NA>  28015
##  6     110 <NA>  26212
##  7     457 <NA>  25998
##  8     589 <NA>  25984
##  9     260 <NA>  25672
## 10     150 <NA>  24284
## # ... with 10,667 more rows
```

Q8. What are the five most given ratings in order from most to least?

```
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
    arrange(desc(count))
```

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

```
## # A tibble: 10 x 2
##    rating   count
##     <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
```

Q9. 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.)

```
edx %>%
  group_by(rating) %>%
  summarize(count = n())
```

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

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

Part(3/3): RMSE Model

first we divide the edx dataset into a training and test set, we apply the model on the training set and we test the RMSE on the test set and finally we validate our final model with the validation set to reach the minimum RMSE

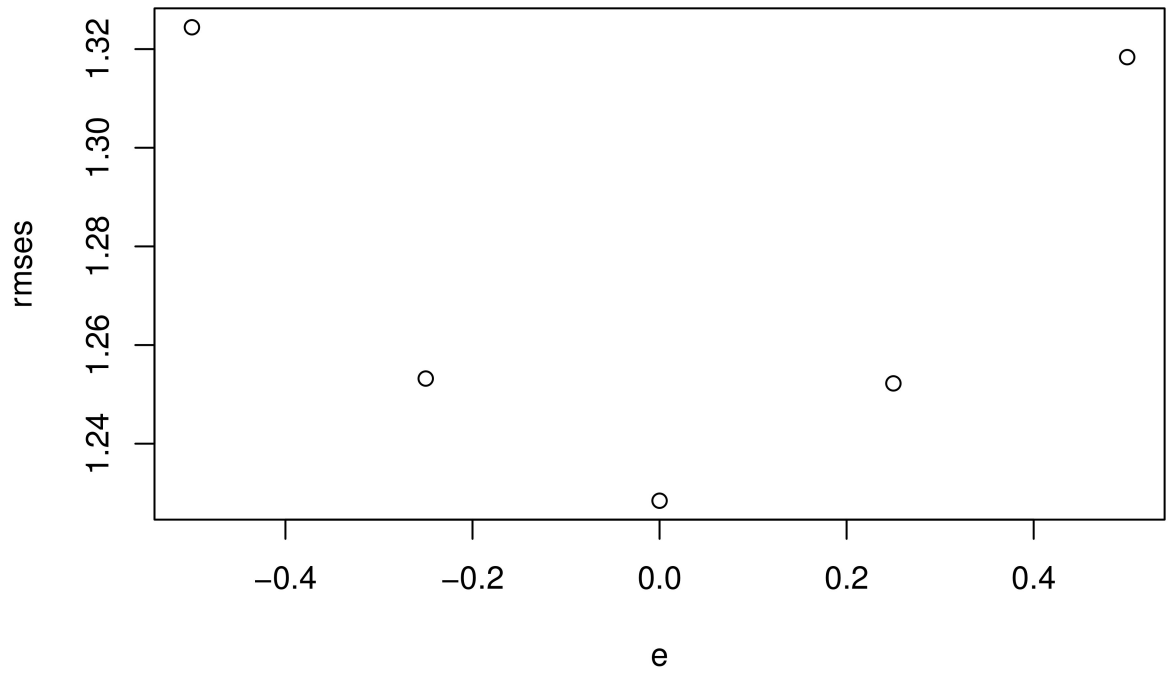
```
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

#to make sure we do not include users and movies in the test set that do not appear in the trainin set

test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

```
#Root Mean Square Error Loss Function
RMSE <- function(true_ratings, predicted_ratings){
        sqrt(mean((true_ratings - predicted_ratings)^2))
      }

e <- seq(-0.5, 0.5, 0.25)

rmses <- sapply(e,function(i){

  #Calculate the mean of ratings from the edx training set
  mu <- mean(train_set$rating)
```

```r
  #Adjust mean by movie effect and penalize low number on ratings
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))

  #ajdust mean by user and movie effect
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - b_i - mu))

  #predict ratings in the training set to derive optimal penalty value 'lambda'
  predicted_ratings <-
    train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = ifelse(mu + b_i + b_u + i > 5, 5,
                  ifelse ( mu + b_i + b_u + i<1, 1,mu + b_i + b_u + i ))) %>%
    .$pred

  return(RMSE(predicted_ratings, test_set$rating))
})
```

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

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length

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

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length

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

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length

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

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length

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

## Warning in true_ratings - predicted_ratings: longer object length is not a
## multiple of shorter object length
```

```r
plot(e, rmses)
```




```r
e <- e[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',e)
```

```
## [1] "Optimal RMSE of 1.22844212160821 is achieved with Lambda 0"
```

Apply Lamda on Validation set for Data-Export

```r
lambda <- e

pred_y_lse <- sapply(lambda,function(i){

  #Derive the mearn from the training set
  mu <- mean(edx$rating)

  #Calculate movie effect with optimal lambda
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))

  #Calculate user effect with optimal lambda
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
```

```
    group_by(userId) %>%
    summarize(b_u = mean(rating - b_i - mu))

  #Predict ratings on validation set
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u + i) %>%
    .$pred #validation

  return(predicted_ratings)

})
```

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

```
#the final RMSE is:
RMSE(validation$rating, pred_y_lse)
```

```
## [1] 0.8653488
```