Capstone Movielens

Ajish Bhaskar

May 25, 2019

1. Introduction

This documentation is my interpretaton of the MovieLens project, a Capstone Project done as part of the Harvard Edx Data Science Course. The purpose of this project is to create a movie rating recommendation system using Machine Learning techniques.Recommendation systems use ratings or preferences that users have given to items to make specific recommendations to the user. In this case, I have used the 10M version of the MovieLens dataset to build a machine learning algorithm using the inputs in one subset known as training set to predict movie ratings in the other subset known as the validation set. RMSE is used to evaluate how close the predictions are to the true values in the validation set.

1.1 About the MovieLens Dataset

Edx has provided the code section to download the dataset and create the training set and the validation set.

```
#Packages and Libraries Used
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)
```

1.2 Additional libraries used

I have also used the below additional libraries.

```
if(!require(gower)) install.packages("gower")
if(!require(backports)) install.packages("backports")
if(!require(GGally)) install.packages("GGally")
library(backports)
library(dslabs)
library(dplyr)
library(ggplot2)
library(tidyr)
library(stringr)
library(lattice)
library(rlang)
library(gower)
library(caret)
library(lubridate)
library(broom)
library(GGally)
```

2. Data Exploration

Let us take a look at the 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
```

Summary Observation:

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Counts:

Number of Movies :

```
n_distinct(edx$movieId)
```

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

Number of Genres :

```
n_distinct(edx$genres)
```

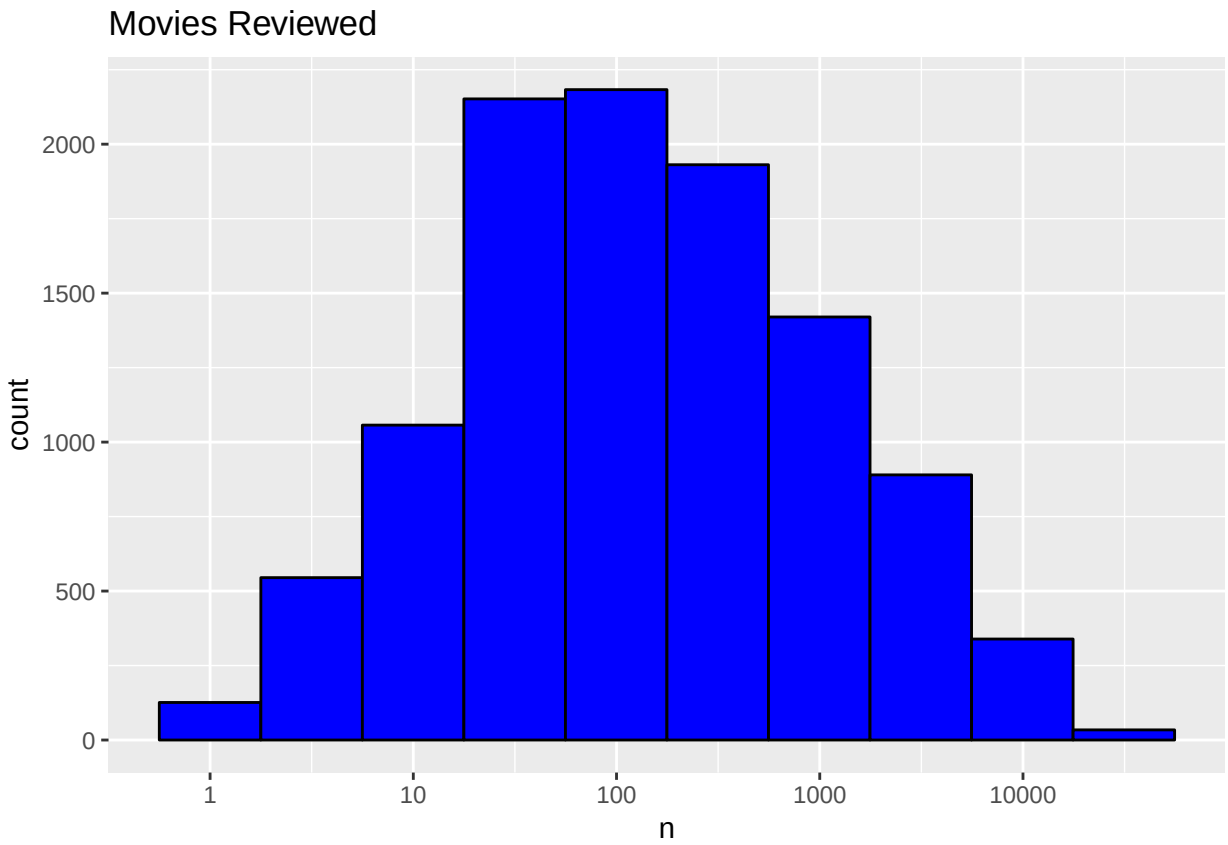
```
## [1] 797
```

Number of Users:

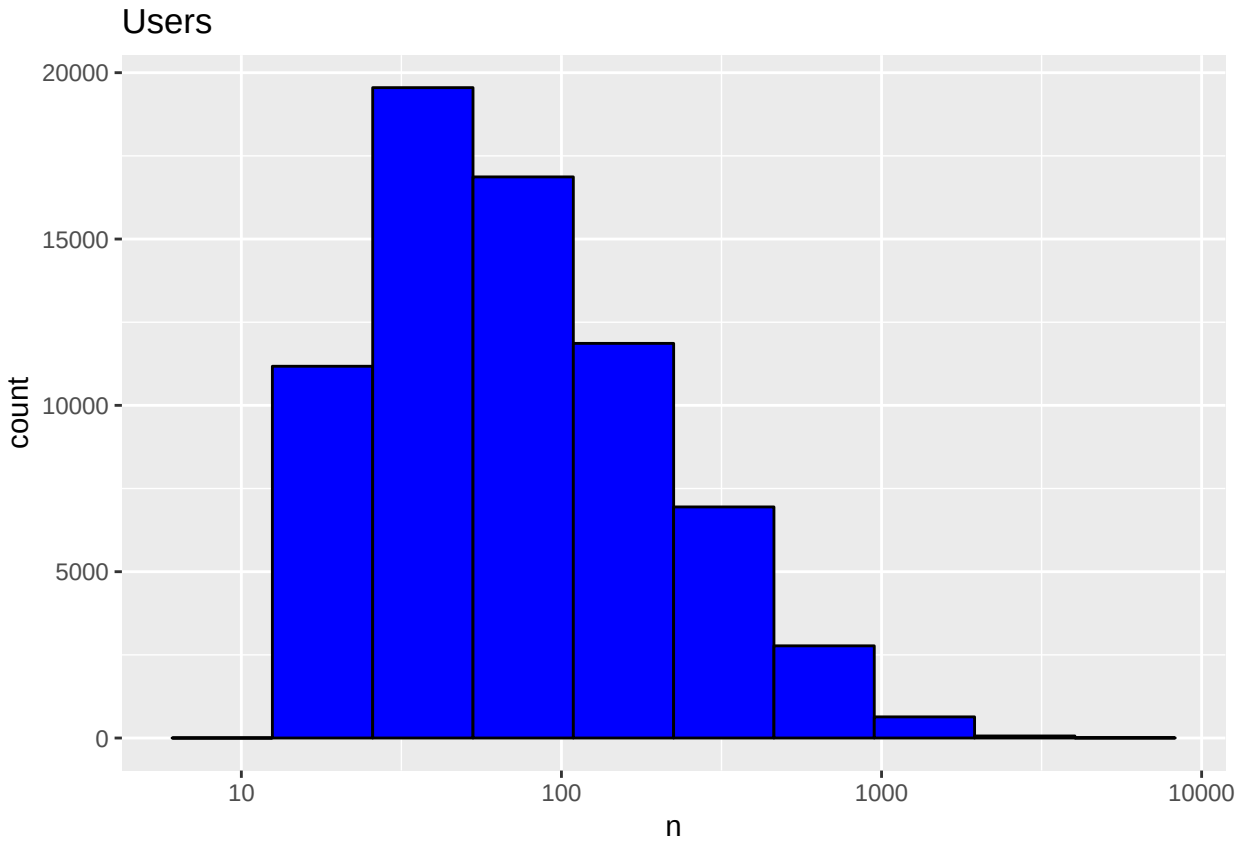
```
n_distinct(edx$userId)
```

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

How many times each movies are reviewed? Lets plot a simple histogram



How about the users?



Now that we got a basic understanding of the dataset, let's move on to the next stage.

3. Data Wrangling

After a brief look at the dataset, it appears that the timestamp field, the datetime the movie was reviewed is not in a readily understandable format. Let's extract the year 'rating_year' and the date 'rating_date' of review and add as new columns.

```
edx <- mutate(edx, rating_year = year(as_datetime(timestamp)))
edx <- mutate(edx, rating_date = as.Date(as_datetime(edx$timestamp), "%m/%d/%Y"))
```

Similarly the movie title has the year of release. Let's extract that to a new column 'title_year'

```
title_year <- as.numeric(stringi::stri_extract(edx$title, regex = "(\\d{4})", comments = TRUE ))

edx_with_dates <- edx %>% mutate(release_year = title_year)
```

Now take a look at the new columns:

```
head(edx_with_dates)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
```

```
##                          genres rating_year rating_date release_year
## 1                Comedy|Romance        1996  1996-08-02         1992
## 2         Action|Crime|Thriller        1996  1996-08-02         1995
## 3  Action|Drama|Sci-Fi|Thriller        1996  1996-08-02         1995
## 4       Action|Adventure|Sci-Fi        1996  1996-08-02         1994
## 5 Action|Adventure|Drama|Sci-Fi        1996  1996-08-02         1994
## 6       Children|Comedy|Fantasy        1996  1996-08-02         1994
```

Now let's see the oldest movie that we have.

```
movies_grouped_by_year <- edx_with_dates %>% group_by(release_year) %>% summarize(n = n()) %>% arrange(
movies_grouped_by_year
```

```
## # A tibble: 107 x 2
##    release_year      n
##           <dbl>  <int>
##  1         9000     22
##  2         5000    195
##  3         3000   4140
##  4         2046    426
##  5         2010   2133
##  6         2008  26741
##  7         2007  75322
##  8         2006 103870
##  9         2005 128606
## 10         2004 204239
## # ... with 97 more rows
```

This reveals that the title year that we extracted has some invalid years. Let's see how many invalid years we have

```
junk_dates <- edx_with_dates %>% filter(release_year > 2010 | release_year < 1900)
junk_dates %>% group_by(movieId, title, release_year) %>% summarize(n = n())
```

```
## # A tibble: 14 x 4
## # Groups:   movieId, title [14]
##    movieId title                                          release_year     n
##      <dbl> <chr>                                                 <dbl> <int>
##  1     671 Mystery Science Theater 3000: The Movie (199~         3000  3280
##  2    1422 Murder at 1600 (1997)                                 1600  1566
##  3    2308 Detroit 9000 (1973)                                   9000    22
##  4    4159 3000 Miles to Graceland (2001)                        3000   714
##  5    4311 Bloody Angels (1732 HÃ¸tten: Marerittet Har ~         1732     9
##  6    5310 Transylvania 6-5000 (1985)                            5000   195
##  7    5472 1776 (1972)                                           1776   185
##  8    6290 House of 1000 Corpses (2003)                          1000   367
##  9    6645 THX 1138 (1971)                                       1138   464
## 10    8198 1000 Eyes of Dr. Mabuse, The (Tausend Augen ~         1000    24
## 11    8864 Mr. 3000 (2004)                                       3000   146
## 12    8905 1492: Conquest of Paradise (1992)                     1492   134
## 13   27266 2046 (2004)                                          2046   426
## 14   53953 1408 (2007)                                          1408   466
```

This shows that we have only 14 movies that have invalid release date assigned. We can easily fix them and get a clean dataset.

```
edx_with_dates[edx_with_dates$movieId == "671", "release_year"] <- 1996
edx_with_dates[edx_with_dates$movieId == "1422", "release_year"] <- 1997
edx_with_dates[edx_with_dates$movieId == "2308", "release_year"] <- 1973
edx_with_dates[edx_with_dates$movieId == "4159", "release_year"] <- 2001
edx_with_dates[edx_with_dates$movieId == "4311", "release_year"] <- 1998
edx_with_dates[edx_with_dates$movieId == "5310", "release_year"] <- 1985
edx_with_dates[edx_with_dates$movieId == "5472", "release_year"] <- 1972
edx_with_dates[edx_with_dates$movieId == "6290", "release_year"] <- 2003
edx_with_dates[edx_with_dates$movieId == "6645", "release_year"] <- 1971
edx_with_dates[edx_with_dates$movieId == "8198", "release_year"] <- 1960
edx_with_dates[edx_with_dates$movieId == "8864", "release_year"] <- 2004
edx_with_dates[edx_with_dates$movieId == "8905", "release_year"] <- 1992
edx_with_dates[edx_with_dates$movieId == "27266", "release_year"] <- 2004
edx_with_dates[edx_with_dates$movieId == "53953", "release_year"] <- 2007
```

Now take a look at the cleansed dataset.

```
head(edx_with_dates)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                          genres rating_year rating_date release_year
## 1                Comedy|Romance        1996  1996-08-02         1992
## 2         Action|Crime|Thriller        1996  1996-08-02         1995
## 3  Action|Drama|Sci-Fi|Thriller        1996  1996-08-02         1995
## 4       Action|Adventure|Sci-Fi        1996  1996-08-02         1994
## 5 Action|Adventure|Drama|Sci-Fi        1996  1996-08-02         1994
## 6       Children|Comedy|Fantasy        1996  1996-08-02         1994
```

Now that the dataset is in reasonably good shape, lets do some data analysis.

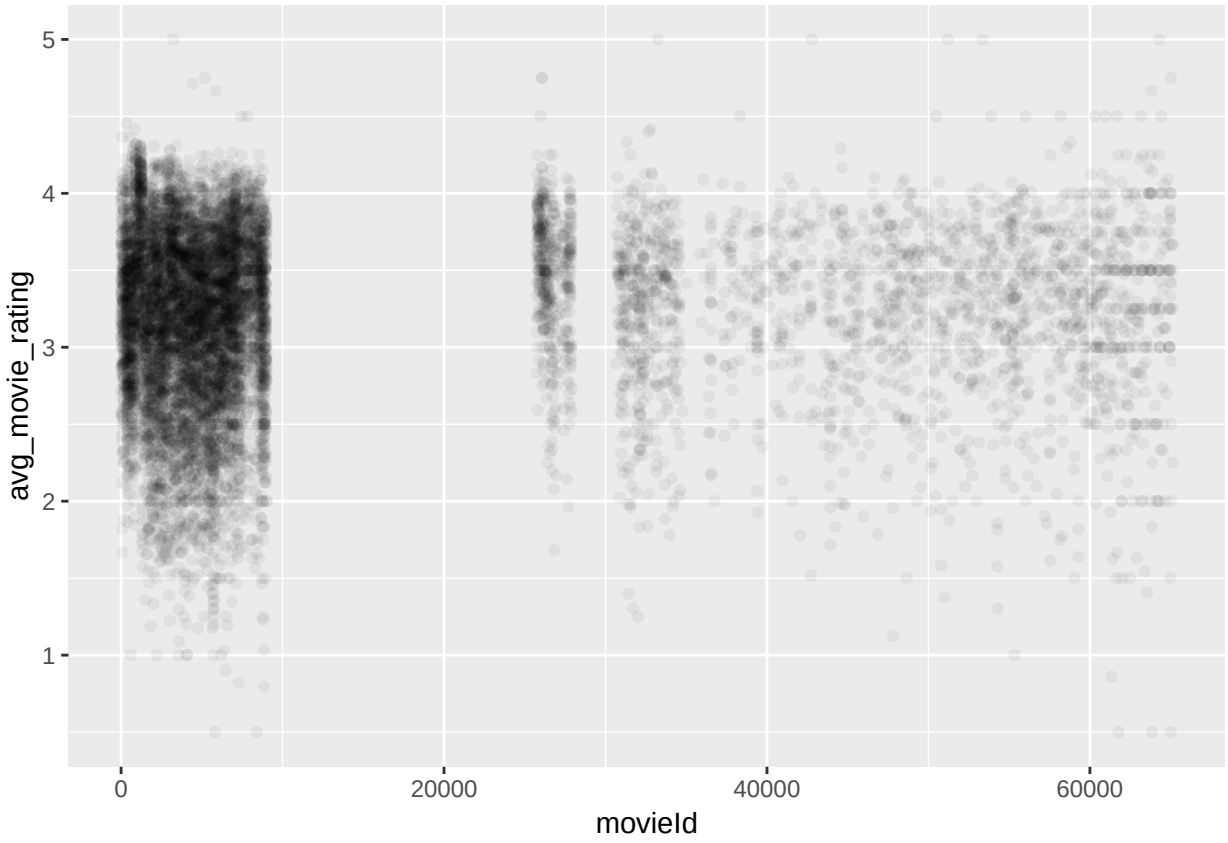
4. Data Analysis

Is there a relationship between the movieID and the average ratings. What are the ratings that movies get consistently?

```
avg_mov_rating <- edx_with_dates %>% group_by(movieId) %>% summarize(avg_movie_rating = mean(rating))
head(avg_mov_rating)
```

```
## # A tibble: 6 x 2
##   movieId avg_movie_rating
##     <dbl>            <dbl>
## 1       1             3.93
## 2       2             3.21
## 3       3             3.15
## 4       4             2.86
## 5       5             3.07
## 6       6             3.82
```

```
avg_mov_rating %>% ggplot(aes(movieId, avg_movie_rating)) +
  geom_point(alpha = 0.05)
```



Now let's see the release year that we got above has any significants? In order to do that let's introduce the age of the movie, which is the number of years from the release of the movie to the year of our analysis 2019. Then determine the average rating by the age of the movie and do a simple geom point plot.

```
#Let's calculate the age of the movie
edx_movies_and_age <-  edx_with_dates %>% mutate(movie_age = 2019 - release_year)

head(edx_movies_and_age)
```

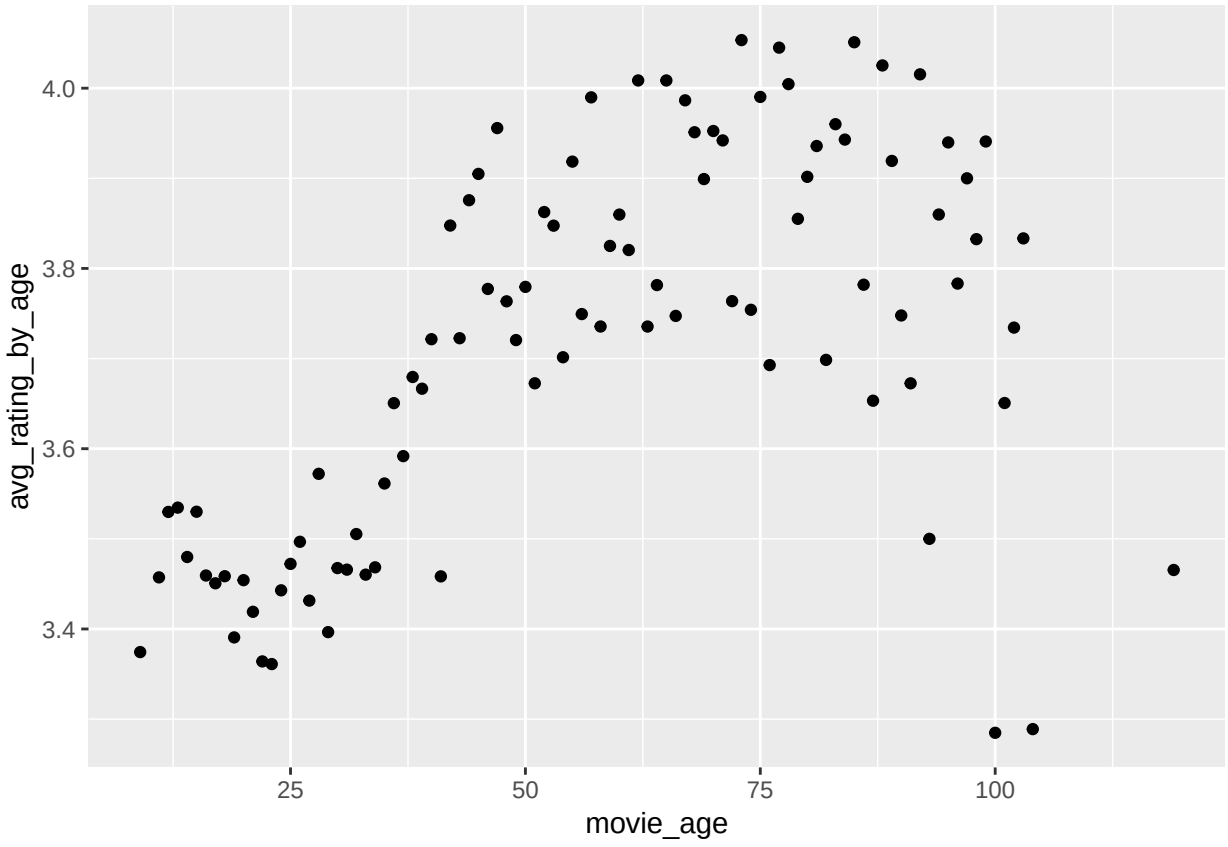
```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                         genres rating_year rating_date release_year
## 1                Comedy|Romance        1996  1996-08-02         1992
## 2         Action|Crime|Thriller        1996  1996-08-02         1995
## 3  Action|Drama|Sci-Fi|Thriller        1996  1996-08-02         1995
## 4       Action|Adventure|Sci-Fi        1996  1996-08-02         1994
## 5 Action|Adventure|Drama|Sci-Fi        1996  1996-08-02         1994
## 6       Children|Comedy|Fantasy        1996  1996-08-02         1994
##   movie_age
## 1        27
```

```
## 2         24
## 3         24
## 4         25
## 5         25
## 6         25
```

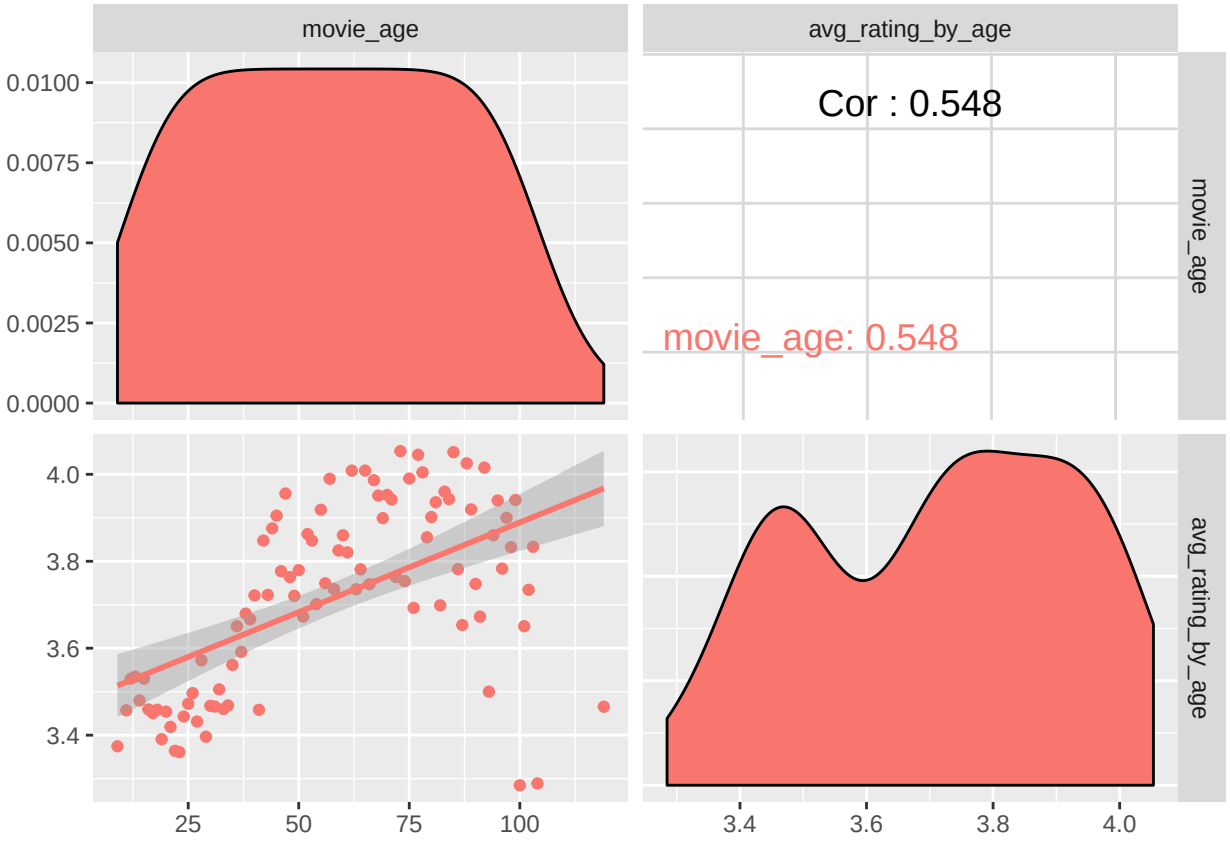
```
#Now let's if age of the movie has any relationship.. like classics have better rating?
avg_rating_age <- edx_movies_and_age %>% group_by(movie_age) %>% summarize(avg_rating_by_age = mean(rat

head(avg_rating_age)
```

```
## # A tibble: 6 x 2
##   movie_age avg_rating_by_age
##       <dbl>             <dbl>
## 1         9              3.37
## 2        11              3.46
## 3        12              3.53
## 4        13              3.53
## 5        14              3.48
## 6        15              3.53
```

```
#Now lets plot a simple geom point
avg_rating_age %>%
  ggplot(aes(movie_age, avg_rating_by_age)) + geom_point()
```



Let's see if there is a corelation between the age of the movie and the average rating? We can plot a ggpairs mapping.



Now let's see if we can use linear model to get the coefficient.

```
lm_stat <- lm(avg_rating_by_age ~ movie_age, data = avg_rating_age)
tidy_data_4plot <- tidy(lm_stat, conf.int = TRUE)
lm_stat %>% .$coef
```

```
## (Intercept)   movie_age
## 3.476820177 0.004124101
```

Now that we have a reasonably good understanding of the dataset, let's move on to the next phase.

5. Prediction & RMSE

I will be utilizing the mu, b_i and b_u parameters in the my prediction as outlined in the Machine Learning course.

To begin with lets do the RMSE setup.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

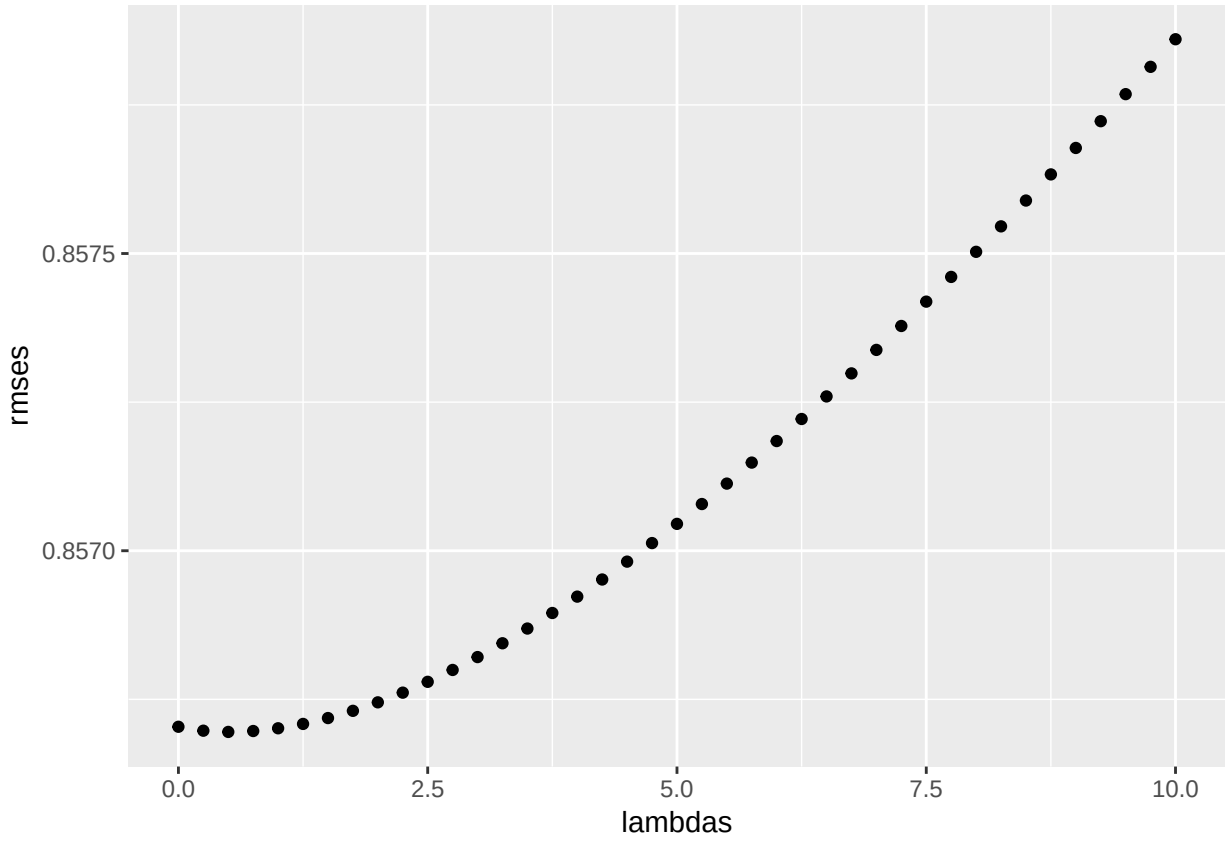
Now let's see use regularization and tune lamdas and see if we can pick the best lambda for our prediction.

```
# Tuning Parameter Lamda
lambdas <- seq(0, 10, 0.25)

#Let's see if we can pick the best rmses
```

```
rmses <- sapply(lambdas, function(l){

# mu as the mean rating
mu <- mean(edx_with_dates$rating)

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

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

#pred using mu, b_i and b_u
predicted_ratings <- edx_with_dates %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i +  b_u) %>% pull(pred)

return(RMSE(predicted_ratings, edx_with_dates$rating))

})
```

Now let's plot the lamdas that we used and the rmses returned.

```
#let's plot the lamdas vs rmses
qplot(lambdas, rmses)
```



The best lambda value that we can use for our prediction is

```
#which lambdas results in the smallest rmses?
best_lambdas <- lambdas[which.min(rmses)]
best_lambdas
```

```
## [1] 0.5
```

Now let's apply our predictioin on the validation set and see how reasonable our prediction is.

```
# I am renaming the variable to _val for my own sanity..

mu_val <- mean(validation$rating)
l_val <- best_lambdas

b_i_val <- validation %>%
  group_by(movieId) %>%
  summarize(b_i_val = sum(rating - mu_val)/(n() + l_val))

b_u_val <- validation %>%
  left_join(b_i_val, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u_val = sum(rating - b_i_val - mu_val)/(n() +l_val))

predicted_ratings <- validation %>%
  left_join(b_i_val, by = "movieId") %>%
  left_join(b_u_val, by = "userId") %>%
  mutate(pred = mu_val + b_i_val +  b_u_val) %>% pull(pred)
```

Lets see what is the RMSE of our predicted rating on the validation set.

```
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8258487
```

6. Conclusion

I used movieId and userId to calculate the RMSE and was able to achieve reasonable RMSE. This was a good project that helped me reinforce the techniques that I learned in the Harvard Exd Data Science Courses.Thanks!