Capstone_MovieLens

Puja

01/13/2022

INTRODUCTION/OVERVIEW/EXECUTIVE SUMMARY

This is a data analysis report prompted by the 9th and final section, Capstone, of the edx program: HarvardX-DataScience. The aim of this project is to test its students' overall capabilities by allowing them to apply all the skills they've learnt thus far in an intriguing project surrounding the world of movies! This project expects its students to take a deep dive into the MovieLens data set and familiarize themselves with different components and the different relationships shared between these MovieLens components. They will then have to implement machine learning skills and construct a body of code (algorithm) using a training set in order to try and predict movie ratings. The goal of this project is to ultimately apply this constructed code onto a test set to see whether it is able to correctly predict movie ratings. We will do this by observing the Root Mean Square Error (RMSE) result.

GENERATE GIVEN DATA

The following code shown below is provided by the "HarvardEDX:Data Science - Capstone" course to its students. This body of code allows you to download the "MovieLens" data, as well as create the training and validation sets. Students are required to create a algorithm using the given 'edx' data set and apply it to the given 'validation' data set.

```
##########################################################
# Create edx set, validation set (final hold-out test 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 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- 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
```

```
library(tidyverse)
library(caret)
library(data.table)

# 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")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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") # if using R 3.5 or earlier, use `set.seed(1)`
```

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

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

```r
# 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")
```

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

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

METHODS/ANALYSIS

DATA SURVEILLANCE

I shall now conduct a thorough Data Surveillance of the given data set in order to familiarize myself with it. My observations will be divided into 2 parts; namely: **Exploration** and **Visualization**. As I carry out my surveillance, I shall also attempt to apply data **cleaning** methods where it may see fit to do so.

EXPLORATION

The purpose of this section is to broaden our understanding of the data. This will be done by carrying out numerous data exploration techniques used throughout the course. It will also include the code used in order to answer the quiz "MovieLens_Dataset", as I found it to be extremely helpful in familiarizing myself to the contents of the data.

Overall summary of edx dataset.

```r
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
##
##
##
```

```r
glimpse(edx)
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
```

```
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

Number of different users, movies, ratings, timestamps, titles and genres in edx data set.

```
Distinct_Numbers <- edx %>%
  summarize(D_userId = n_distinct(edx$userId),
            D_movieId = n_distinct(edx$movieId),
            D_rating = n_distinct(edx$rating),
            D_timestamp = n_distinct(edx$timestamp),
            D_title = n_distinct(edx$title),
            D_genres = n_distinct(edx$genres))
Distinct_Numbers
```

```
##   D_userId D_movieId D_rating D_timestamp D_title D_genres
## 1    69878     10677       10     6519590   10676      797
```

Number of movie ratings made for each of the genres in the edx data set.

```
Genre_Ratings <- edx %>%
  summarize(Drama_Ratings = sapply("Drama", function(d) {
    sum(str_detect(edx$genres, d))}),
    Comedy_Ratings = sapply("Comedy", function(c) {
      sum(str_detect(edx$genres, c))}),
    Thriller_Ratings = sapply("Thriller", function(t) {
      sum(str_detect(edx$genres, t))}),
    Romance_Ratings = sapply("Romance", function(r) {
      sum(str_detect(edx$genres, r))})
    )
Genre_Ratings
```

```
##   Drama_Ratings Comedy_Ratings Thriller_Ratings Romance_Ratings
## 1       3910127        3540930          2325899         1712100
```

Movies with the most amount of cumulative ratings

```
Movie_Ratings <- edx %>%
  group_by(title, genres) %>%
  summarize(cumulative_ratings = n()) %>%
  arrange(desc(cumulative_ratings))
```

```
## `summarise()` has grouped output by 'title'. You can override using the `.groups` argument.
```

```
Movie_Ratings
```

```
## # A tibble: 10,677 x 3
## # Groups:   title [10,676]
##    title                                genres                 cumulative_ratin~
##    <chr>                                <chr>                              <int>
##  1 Pulp Fiction (1994)                  Comedy|Crime|Drama                 31362
##  2 Forrest Gump (1994)                  Comedy|Drama|Romance~              31079
##  3 Silence of the Lambs, The (1991)     Crime|Horror|Thriller              30382
##  4 Jurassic Park (1993)                 Action|Adventure|Sci~              29360
##  5 Shawshank Redemption, The (1994)     Drama                              28015
##  6 Braveheart (1995)                    Action|Drama|War                   26212
```

```
##  7 Fugitive, The (1993)                 Thriller                          25998
##  8 Terminator 2: Judgment Day (1991)    Action|Sci-Fi                     25984
##  9 Star Wars: Episode IV - A New Hope (~ Action|Adventure|Sci~            25672
## 10 Apollo 13 (1995)                     Adventure|Drama                   24284
## # ... with 10,667 more rows
```

The most popular ratings used

```
Most_Used_Ratings <- edx %>%
  group_by(rating) %>%
  summarize(times_used = n()) %>%
  top_n(10) %>%
  arrange(desc(times_used))
```

```
## Selecting by times_used
```

```
Most_Used_Ratings
```

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

Genres with most amount of cumulative ratings

```
Genre_Ratings <- edx %>%
  group_by(genres) %>%
  summarize(cumulative_ratings = n()) %>%
  arrange(desc(cumulative_ratings))
Genre_Ratings
```

```
## # A tibble: 797 x 2
##    genres                       cumulative_ratings
##    <chr>                                     <int>
##  1 Drama                                    733296
##  2 Comedy                                   700889
##  3 Comedy|Romance                           365468
##  4 Comedy|Drama                             323637
##  5 Comedy|Drama|Romance                     261425
##  6 Drama|Romance                            259355
##  7 Action|Adventure|Sci-Fi                  219938
##  8 Action|Adventure|Thriller                149091
##  9 Drama|Thriller                           145373
## 10 Crime|Drama                              137387
## # ... with 787 more rows
```

Extracting the year from the edx column 'title' in order to have a separate column called 'titleyear'.

```r
library(stringr)
year_from_title <- '\\d{4}(?=\\))'
titleyear = str_extract(edx$title, year_from_title)
```

New edx dataframe with added column 'titleyear'.

```r
new_edx <- edx %>%
  add_column(titleyear)
new_edx
```

```
##          userId movieId rating  timestamp                         title
##       1:      1     122    5.0  838985046              Boomerang (1992)
##       2:      1     185    5.0  838983525               Net, The (1995)
##       3:      1     292    5.0  838983421               Outbreak (1995)
##       4:      1     316    5.0  838983392               Stargate (1994)
##       5:      1     329    5.0  838983392 Star Trek: Generations (1994)
##      ---
## 9000051:  32620   33140    3.5 1173562747         Down and Derby (2005)
## 9000052:  40976   61913    3.0 1227767528           Africa addio (1966)
## 9000053:  59269   63141    2.0 1226443318 Rockin' in the Rockies (1945)
## 9000054:  60713    4820    2.0 1119156754  Won't Anybody Listen? (2000)
## 9000055:  64621   39429    2.5 1201248182                Confess (2005)
##                                  genres titleyear
##       1:                 Comedy|Romance      1992
##       2:          Action|Crime|Thriller      1995
##       3:  Action|Drama|Sci-Fi|Thriller      1995
##       4:        Action|Adventure|Sci-Fi      1994
##       5: Action|Adventure|Drama|Sci-Fi      1994
##      ---
## 9000051:                Children|Comedy      2005
## 9000052:                    Documentary      1966
## 9000053:         Comedy|Musical|Western      1945
## 9000054:                    Documentary      2000
## 9000055:                 Drama|Thriller      2005
```

```r
head(new_edx)
```

```
##    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 titleyear
## 1:                Comedy|Romance      1992
## 2:         Action|Crime|Thriller      1995
## 3:  Action|Drama|Sci-Fi|Thriller      1995
## 4:       Action|Adventure|Sci-Fi      1994
## 5: Action|Adventure|Drama|Sci-Fi      1994
## 6:       Children|Comedy|Fantasy      1994
```

Count how many times each year appears

```r
new_edx %>% count(titleyear)
```

```
##     titleyear     n
##  1:      1915   180
##  2:      1916    84
##  3:      1917    32
##  4:      1918    73
##  5:      1919   158
##  6:      1920   575
##  7:      1921   406
##  8:      1922  1825
##  9:      1923   316
## 10:      1924   457
## 11:      1925  2654
## 12:      1926   383
## 13:      1927  4133
## 14:      1928  1336
## 15:      1929   573
## 16:      1930  2472
## 17:      1931  7669
## 18:      1932  3263
## 19:      1933  7675
## 20:      1934  5954
## 21:      1935  6234
## 22:      1936  4120
## 23:      1937 13456
## 24:      1938  7775
## 25:      1939 27422
## 26:      1940 26708
## 27:      1941 23883
## 28:      1942 20051
## 29:      1943  3563
## 30:      1944 11918
## 31:      1945  5838
## 32:      1946 16882
## 33:      1947  6558
## 34:      1948  9878
## 35:      1949  7862
## 36:      1950 17377
## 37:      1951 21789
## 38:      1952 11540
## 39:      1953 18740
## 40:      1954 30089
## 41:      1955 21673
## 42:      1956 16053
## 43:      1957 24557
## 44:      1958 23336
## 45:      1959 30843
## 46:      1960 29637
## 47:      1961 26004
## 48:      1962 33622
## 49:      1963 31489
## 50:      1964 40477
## 51:      1965 23475
## 52:      1966 18284
## 53:      1967 39745
```

```
## 54:      1968  48313
## 55:      1969  25702
## 56:      1970  27660
## 57:      1971  59092
## 58:      1972  39512
## 59:      1973  50625
## 60:      1974  47460
## 61:      1975  67996
## 62:      1976  45298
## 63:      1977  60035
## 64:      1978  55020
## 65:      1979  84452
## 66:      1980 104842
## 67:      1981  96947
## 68:      1982 121258
## 69:      1983  94372
## 70:      1984 163168
## 71:      1985 139778
## 72:      1986 175548
## 73:      1987 171738
## 74:      1988 171628
## 75:      1989 228426
## 76:      1990 230409
## 77:      1991 196681
## 78:      1992 236834
## 79:      1993 481184
## 80:      1994 671298
## 81:      1995 786762
## 82:      1996 593518
## 83:      1997 429751
## 84:      1998 402187
## 85:      1999 489537
## 86:      2000 382823
## 87:      2001 305705
## 88:      2002 272180
## 89:      2003 211397
## 90:      2004 204811
## 91:      2005 128613
## 92:      2006 103870
## 93:      2007  75788
## 94:      2008  26741
##     titleyear      n
```

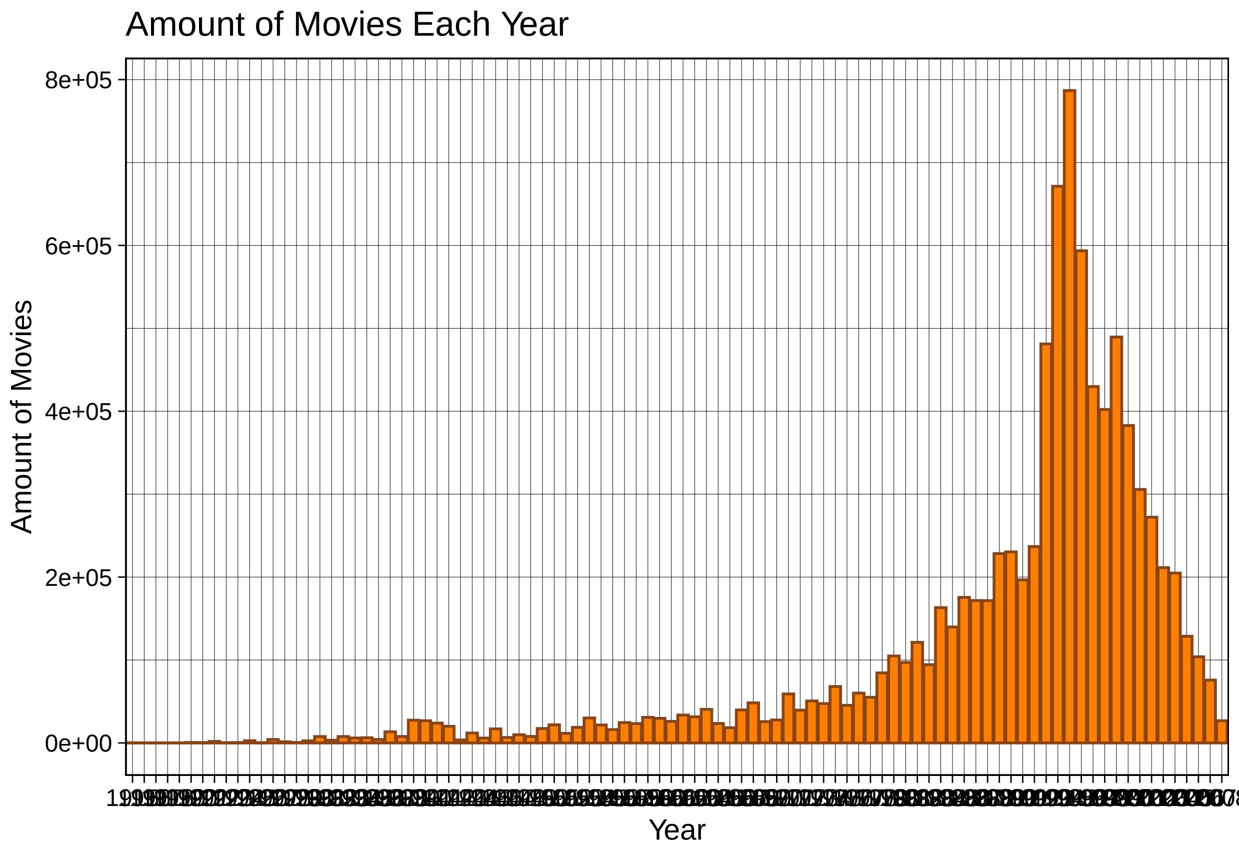
VISUALIZATION

In this section, we will build up on what we've learnt thus far using visual aids. I personally find data visualization to be extremely accommodating when it comes to working with great amounts of data as they help in seeing the bigger picture more clearly (pun intended).

Bar Graph of Amount of Movies Each Year

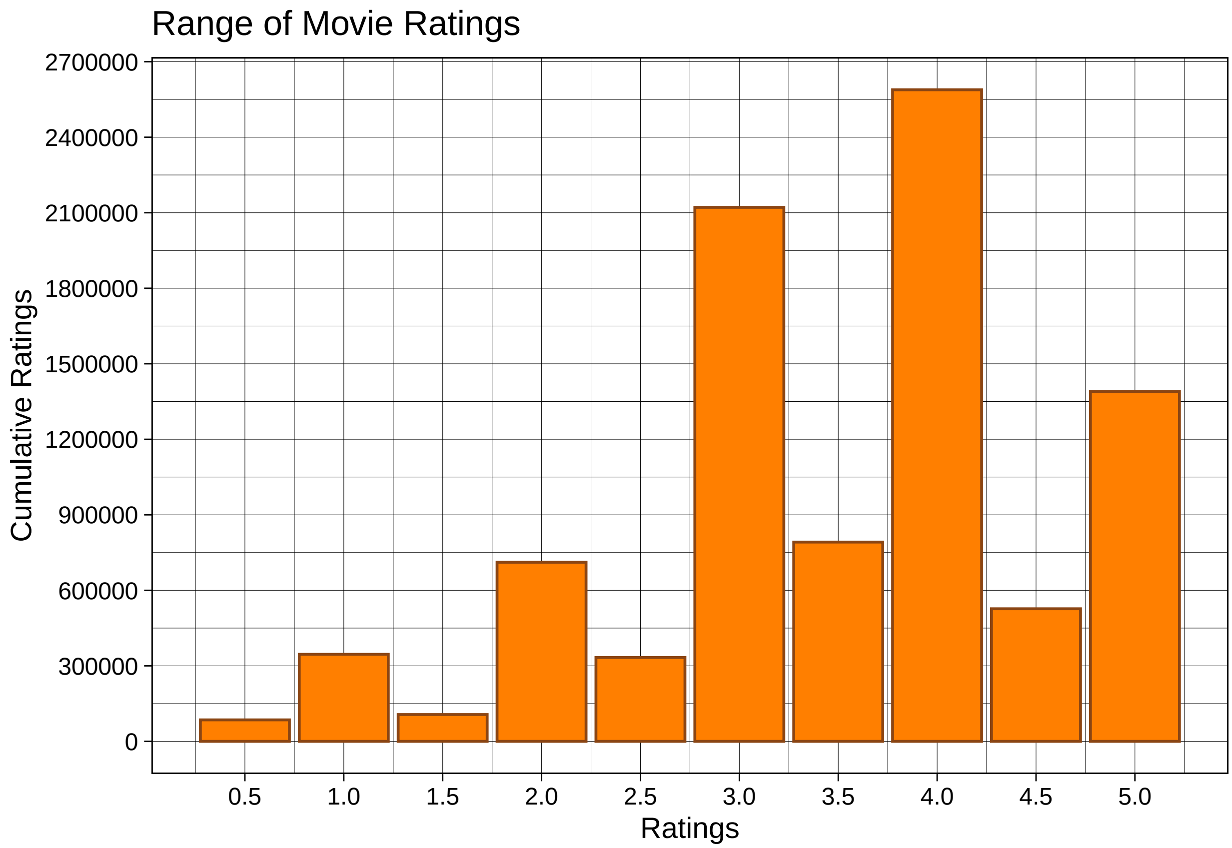
```
Amount_of_Movies_Each_Year <- new_edx %>%
  ggplot(aes(titleyear)) +
  geom_bar(color = "chocolate4",  fill = "darkorange1") +
```

```
  labs(x = "Year", y = "Amount of Movies") +
  ggtitle("Amount of Movies Each Year") +
  theme_linedraw()
Amount_of_Movies_Each_Year
```



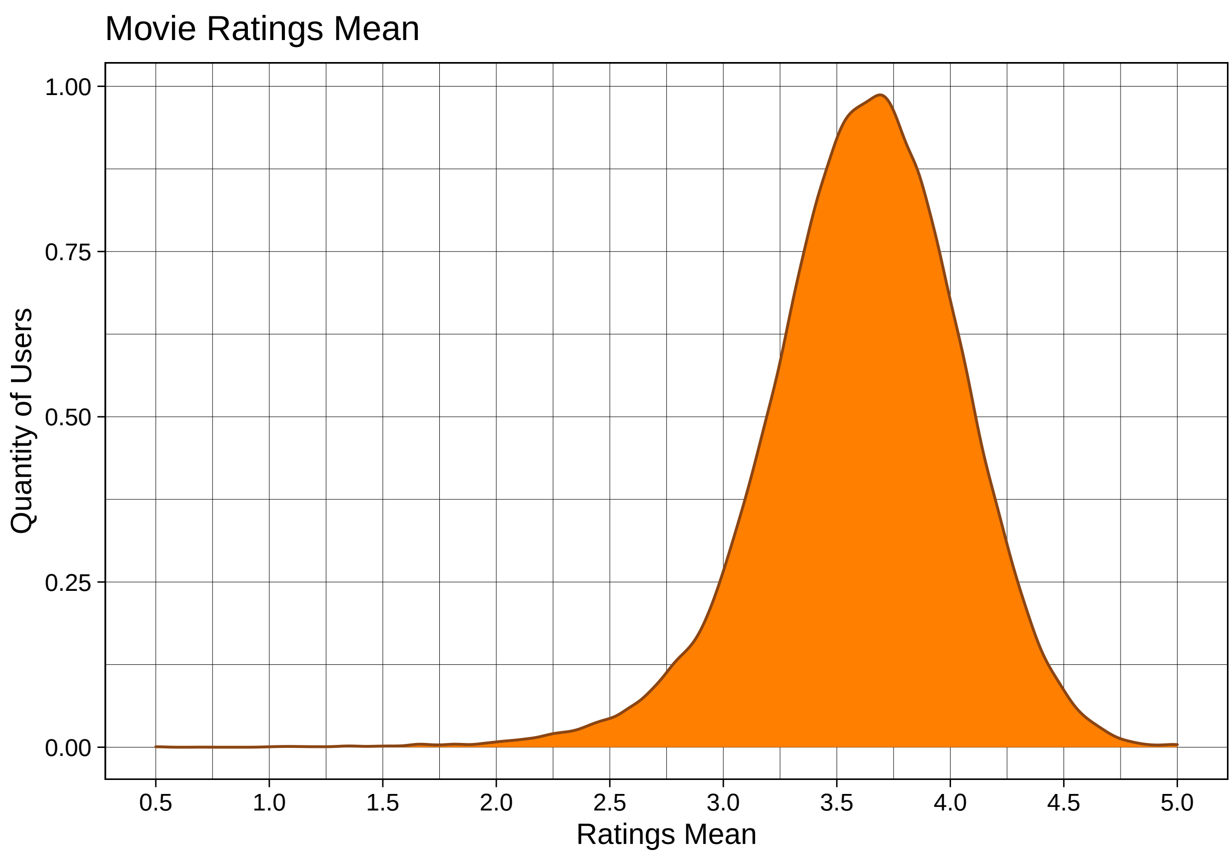
Bar Graph of Range of Movie Ratings

```
Range_of_Movie_Ratings <- edx %>%
  ggplot(aes(rating)) +
  geom_bar(color = "chocolate4",  fill = "darkorange1") +
  scale_x_continuous(breaks = c(0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)) +
  scale_y_continuous(breaks = c(0, 300000, 600000, 900000, 1200000, 1500000,
                                1800000, 2100000, 2400000, 2700000)) +
  labs(x = "Ratings", y = "Cumulative Ratings") +
  ggtitle("Range of Movie Ratings") +
  theme_linedraw()
Range_of_Movie_Ratings
```



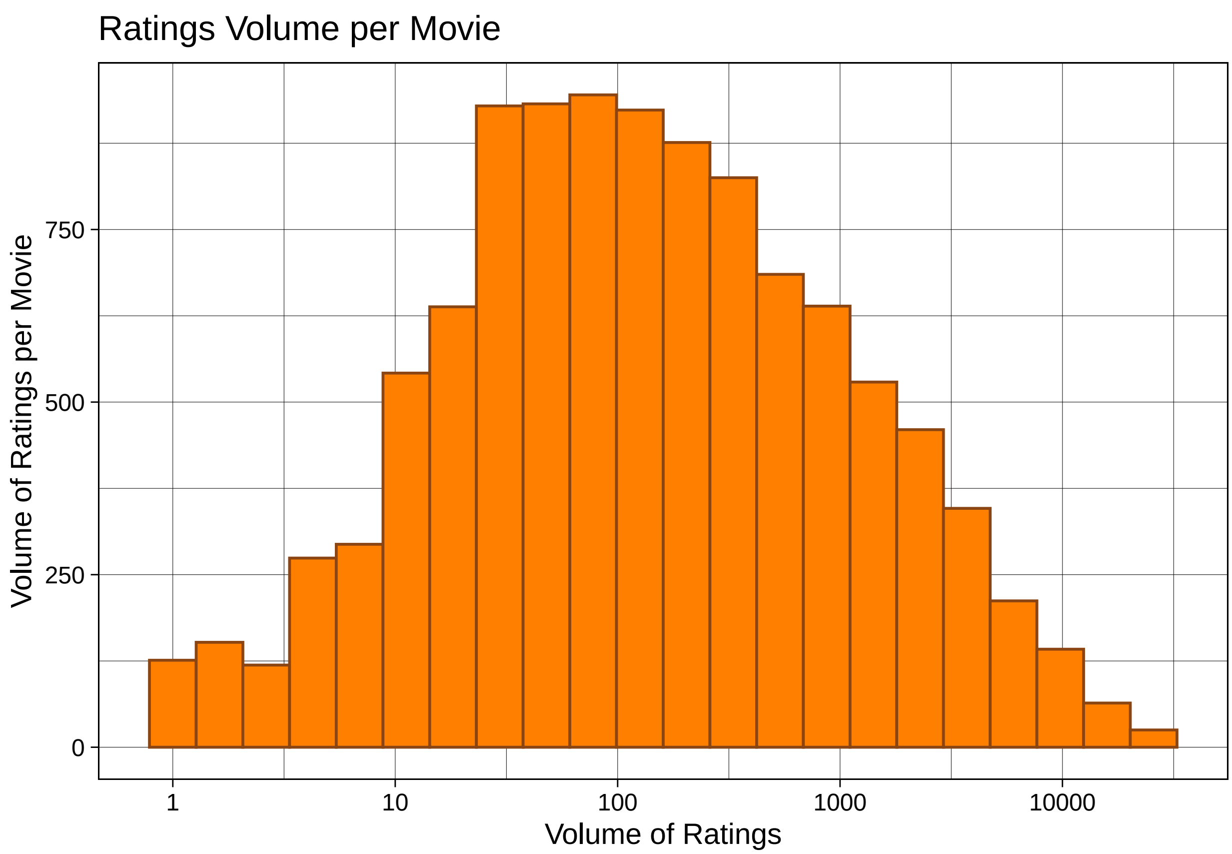
Density Graph of the Movie Ratings' Mean

```
Movie_Ratings_Mean <- edx %>%
  group_by(userId) %>%
  summarize(average = mean(rating)) %>%
  ggplot(aes(average)) +
  geom_density(color = "chocolate4",  fill = "darkorange1") +
  ggtitle("Movie Ratings Mean") +
  scale_x_continuous(breaks = c(seq(0.5,5,0.5))) +
  labs(x = "Ratings Mean", y = "Quantity of Users") +
  theme_linedraw()
Movie_Ratings_Mean
```



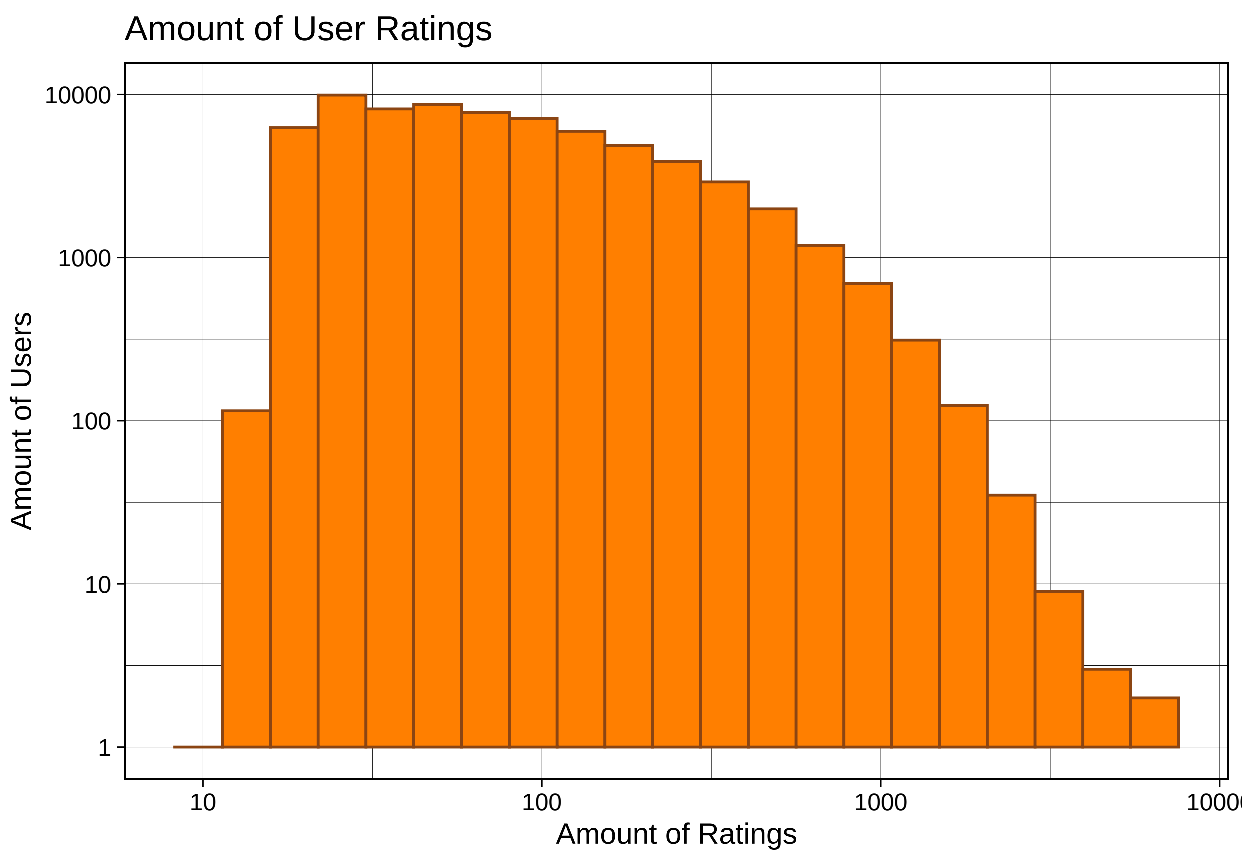
Histogram for Ratings Volume per Movie

```
Ratings_Volume_per_Movie <- edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 0.21, color = "chocolate4", fill = "darkorange1") +
  scale_x_log10() +
  scale_y_continuous() +
  labs(x = "Volume of Ratings", y = "Volume of Ratings per Movie") +
  ggtitle("Ratings Volume per Movie") +
  theme_linedraw()
Ratings_Volume_per_Movie
```



Histogram of the Amount of User Ratings

```
Amount_of_User_Ratings <- edx %>%
  group_by(userId) %>%
  summarize(count = n()) %>%
  ggplot(aes(count)) +
  geom_histogram(bins = 21, color = "chocolate4",  fill = "darkorange1") +
  scale_x_log10() +
  scale_y_log10() +
  labs(x = "Amount of Ratings", y = "Amount of Users") +
  ggtitle("Amount of User Ratings") +
  theme_linedraw()
Amount_of_User_Ratings
```



MODELING APPROACHES AND INSIGHTS GAINED

We shall now start with the process of modelling our algorithms. Thanks to the various methods of **Data Surveillance** above, we now have an extremely clear idea of the data set that we're working with and it's content. We shall now use the knowledge gained to help construct the algorithm. The accuracy of our work will be determined by the utilization of the Residual Mean Squared Error (RMSE) Formula.

The RMSE formula helps us to calculate the difference (i.e. error) between the predicted and the observed values. Our aim is to obtain an error value of less than 0.8649. We shall apply this RMSE formula to the models below in order to generate their error quantities and see whether we are able to get it below 0.8649.

RMSE FORMULA

```
RMSE_Formula <- function(observed, predicted){
  sqrt(mean((observed - predicted)^2))}
```

I shall now represent "validation$rating" as "a" in order to save myself from typing it out whenever I have to calculate the RMSE.

```
a <- validation$rating
```

1ST MODEL

The first model which I will be applying to the data set will be the most straightforward application model. It will be implemented by using the mean/average of all the recorded movie ratings in the data set.

Compute the mean/average of all recorded ratings belonging to the edx data set.

```
mu_cap <- mean(edx$rating)
```

RMSE of Model 1

```
RMSE_MODEL_1 <- RMSE_Formula(a, mu_cap)
RMSE_MODEL_1
```

```
## [1] 1.061202
```

INSIGHTS GAINED FROM MODEL 1:

As predicted, our RMSE result for Model 1 is not nearly as close to our desired reading. This may be because the ratings alone is not sufficient enough to help our prediction.

2ND MODEL

The second model which I will be applying to the data set, in order to try an d reduce my error value, will be an upgrade to my first model. I shall now examine how the users who decide what ratings to give to the movies will impact the RMSE value.

Compute the impact of all recorded users belonging to the edx data set. Compute penalty term b1; associated with users

```
user_rating <- edx %>%
  group_by(userId) %>%
  summarise(b1 = mean(rating - mu_cap))
user_rating_forecast <- mu_cap + validation %>%
  left_join(user_rating, by = 'userId') %>%
  .$b1
```

RMSE of Model 2

```
RMSE_MODEL_2 <- RMSE_Formula(a, user_rating_forecast)
RMSE_MODEL_2
```

```
## [1] 0.978336
```

INSIGHTS GAINED FROM MODEL 2:

I am pleased to see that the RMSE has decreased slightly after introducing the userId. This is because we are fed more information into our model, which helped to make a slightly more accurate prediction compared to Model 1.

3RD MODEL

After running the second model with the addition of users to our 1ST MODEL, we can see that the RMSE has decreased. This is what we want. For this 3RD MODEL, I shall observe the effect of including the 'movieId' to the model.

Compute the impact of all recorded "movieId's" belonging to the edx data set. Compute penalty term b2; associated with movieId's.

```
movie_user_rating <- edx %>%
  left_join(user_rating, by = 'userId') %>%
  group_by(movieId) %>%
  summarize(b2 = mean(rating-mu_cap-b1))
movie_user_rating_forecast <- validation %>%
  left_join(user_rating, by = 'userId') %>%
```

```
  left_join(movie_user_rating, by = 'movieId') %>%
  mutate(forecast = mu_cap + b1 + b2) %>%
  pull(forecast)
```

RMSE OF MODEL 3

```
RMSE_MODEL_3 <- RMSE_Formula(a, movie_user_rating_forecast)
RMSE_MODEL_3
```

```
## [1] 0.8816096
```

INSIGHTS GAINED FROM MODEL 3:

Our error value has decreased compared to our previous model, but unfortunately not as much as I'd hoped for. I have also run out of variables o keep adding to my model. This means that I am going to have to turn to alternative methods in order to decrease my RMSE value to the desired result.

4TH MODEL

The third model, which includes ratings, userID & moiveID, managed to obtain a lower RMSE reading compared to the first and second model, which is extremely reasuring that we're on the right path. I shall now implement the 'regulization' method to the data set in the hopes of obtaining an even lower RMSE reading than previously recorded.

Regulization

```
Regulization_Lamda <- seq(0, 50, 0.5)

Regulization_RMSE <- sapply(Regulization_Lamda, function(lam){
    Regulization_b1 <- edx %>%
      group_by(userId) %>%
      summarize(Regulization_b1 = sum(rating - mu_cap)/(n() + lam))

    Regulization_b2 <- edx %>%
      left_join(Regulization_b1, by = "userId") %>%
      group_by(movieId) %>%
      summarize(Regulization_b2 = sum(rating - Regulization_b1 - mu_cap)/(n() + lam))

    Regulization_b1_b2_mu_Forecast <- validation %>%
      left_join(Regulization_b1, by = "userId") %>%
      left_join(Regulization_b2, by = "movieId") %>%
      mutate(forecast2 = mu_cap + Regulization_b1 + Regulization_b2) %>%
      pull(forecast2)
    return(RMSE_Formula(a, Regulization_b1_b2_mu_Forecast))
  })
```

RMSE OF MODEL 4

```
RMSE_MODEL_4 = min(Regulization_RMSE)
RMSE_MODEL_4
```

```
## [1] 0.8794441
```

INSIGHTS GAINED FROM MODEL 4:

On the bright side, the RMSE has decreased compared to our previous model. However, the error value is still not as low as I need it to be. This is extremely unfortunate as I was expecting to obtain the required

RMSE amount after implementing regulization. I will now have to apply another method now in the hopes of obtaining the required RMSE amount.

5TH MODEL

Seeing that our fourth Model (which used regulization) regrettably was unable to achieve and RMSE equal to or less than 0.8649; I have no choice but to try an alternative method to obtain this amount. I am now going to attempt the **Matrix Factorization** method in the hopes of obtaining the required RMSE value.

MATRIX FACTORIZATION

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

```
## Loading required package: recosystem
```

```r
library(recosystem)
set.seed(1)
matfac_recosys <- Reco()
valid_matfac <- with(validation, data_memory(rating = rating, user_index = userId,
                                              item_index = movieId))
edx_matfac <- with(edx, data_memory(rating = rating, user_index = userId, item_index = movieId))

matfac_parameters <- matfac_recosys$tune(edx_matfac, opts = list(niter = 30, nthread = 3,
                                                                  dim = c(10, 30)))
matfac_recosys$train(edx_matfac, opts = c(matfac_parameters$min, niter = 30, nthread = 3))
```

```
## iter      tr_rmse          obj
##    0       0.9736   1.2040e+07
##    1       0.8726   9.9005e+06
##    2       0.8384   9.1694e+06
##    3       0.8166   8.7475e+06
##    4       0.8018   8.4744e+06
##    5       0.7903   8.2802e+06
##    6       0.7803   8.1266e+06
##    7       0.7721   8.0078e+06
##    8       0.7650   7.9102e+06
##    9       0.7589   7.8261e+06
##   10       0.7536   7.7576e+06
##   11       0.7489   7.6993e+06
##   12       0.7448   7.6484e+06
##   13       0.7410   7.6053e+06
##   14       0.7376   7.5664e+06
##   15       0.7344   7.5335e+06
##   16       0.7315   7.5000e+06
##   17       0.7288   7.4731e+06
##   18       0.7264   7.4469e+06
##   19       0.7241   7.4243e+06
##   20       0.7220   7.4054e+06
##   21       0.7200   7.3859e+06
##   22       0.7182   7.3669e+06
##   23       0.7164   7.3500e+06
##   24       0.7148   7.3352e+06
##   25       0.7134   7.3215e+06
##   26       0.7120   7.3092e+06
##   27       0.7106   7.2958e+06
```

```
##   28       0.7094   7.2861e+06
##   29       0.7081   7.2739e+06
```

```
matfac_solution <- matfac_recosys$predict(valid_matfac, out_memory())
```

RMSE OF MODEL 5

```
RMSE_MODEL_5 <- RMSE_Formula(a, matfac_solution)
RMSE_MODEL_5
```

```
## [1] 0.7808162
```

INSIGHTS GAINED FROM MODEL 5:

FINALLY! After running the data sets through Model 5, which uses the Matrix Factorization method, I have finally managed to obtain an RMSE value which is (equal to or) below 0.864999. The RMSE has decreased considerably compared to our previous Model 4. This method is able to generate a strong RMSE of 0.781.

RESULTS

I shall now create a 'results' table in which I shall record all of the RMSE Model recordings.

```
tab <- matrix(c(RMSE_MODEL_1, RMSE_MODEL_2, RMSE_MODEL_3, RMSE_MODEL_4, RMSE_MODEL_5), ncol=1,
              byrow = FALSE)
colnames(tab) <- c('RMSE Results')
rownames(tab) <- c('RMSE MODEL 1', 'RMSE MODEL 2', 'RMSE MODEL 3', 'RMSE MODEL 4', 'RMSE MODEL 5')
tab <- as.table(tab)
tab
```

```
##              RMSE Results
## RMSE MODEL 1    1.0612018
## RMSE MODEL 2    0.9783360
## RMSE MODEL 3    0.8816096
## RMSE MODEL 4    0.8794441
## RMSE MODEL 5    0.7808162
```

After completing 5 Model attempts with the sole purpose of obtaining an RMSE of less than 0.864999; I can happily say that I have managed to attain an RMSE result of 0.78065. Models 1-3 comprise of me adding more variables each time with hopes of decreasing the RMSE to the desired amount. Despite managing to get lower RMSE values each time as I progressed through these models, they still weren't low enough. I then had to turn to alternative methods such as regulization and ultimately to matrix factorization, which finally allowed me to obtain an RMSE below 0.864999.

CONCLUSION

This Capstone Project proved to be extremely joyful and insightful. It managed to test its students on numerous methods taught throughout this entire Data Science course. Moreover, it was able to encourage its students to go beyond what they've learned during this course and carry out additional research and learn new techniques and skills in order to complete this project. The chosen data set, MovieLens, was a fun and relatable topic which kept its students engaged and interested (it definitely convinced me to watch a few movies which I've never seen before).