HarvardX Capstone: MovieLens

Somosree Banerjee

11/03/2020

Contents

I. Introduction

This report is part of the capstone project of the EdX course "HarvardX: PH125.9x Data Science: Capstone". The goal is to challenge and demonstrate how the knowledge acquired through the different topics covered in "HarvardX: PH125.9x Data Science" can be applied in solving real world problems.

II. Summary

For the MovieLens project, the data set provided is generated by the GroupLens research lab. The aim to create a recommendation system using the "prediction version of problem". The report is split in three sections: 1. Data Loading and Data Wrangling for further Analysis 2. Data Exploration and Analysis to understand the structure of the data set. 3. A machine learning algorithm to calculate RMSE. "Penalized Least Square Approach" has been considered to calculate the RMSE.

III. Data Loading and Data Wrangling

Libraries Loaded

library(tidyverse) library(caret) library(data.table) library(splitstackshape) library(DT) library(lubridate)

The data set 'movielens' gets split into a training-test set called 'edx' and a set for validation purposes called 'validation'.

```r
################################
# Create edx set, validation set
################################
#Memory
memory.limit()
```

```
## [1] 8031
```

```r
memory.limit(size=56000)
```

```
## [1] 56000
```

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

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

```
## -- Attaching packages --------------------------- tidyverse 1.3.0 --
```

```
## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

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

```r
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
```

```r
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
```

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

```
## Loading required package: splitstackshape
```

```
## Warning: package 'splitstackshape' was built under R version 3.6.3
```

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

```
## Loading required package: DT
```

```
## Warning: package 'DT' was built under R version 3.6.3
```

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

```
## Loading required package: lubridate
```

```
## Warning: package 'lubridate' was built under R version 3.6.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(tidyverse)
library(caret)
library(data.table)
library(splitstackshape)
library(DT)
library(lubridate)
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
```

```r
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.cha
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
```

```r
# 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")
```

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

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

III. Data Exploration & Analysis

```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
##
##
##
```

#Quantitative:Rating Analysis

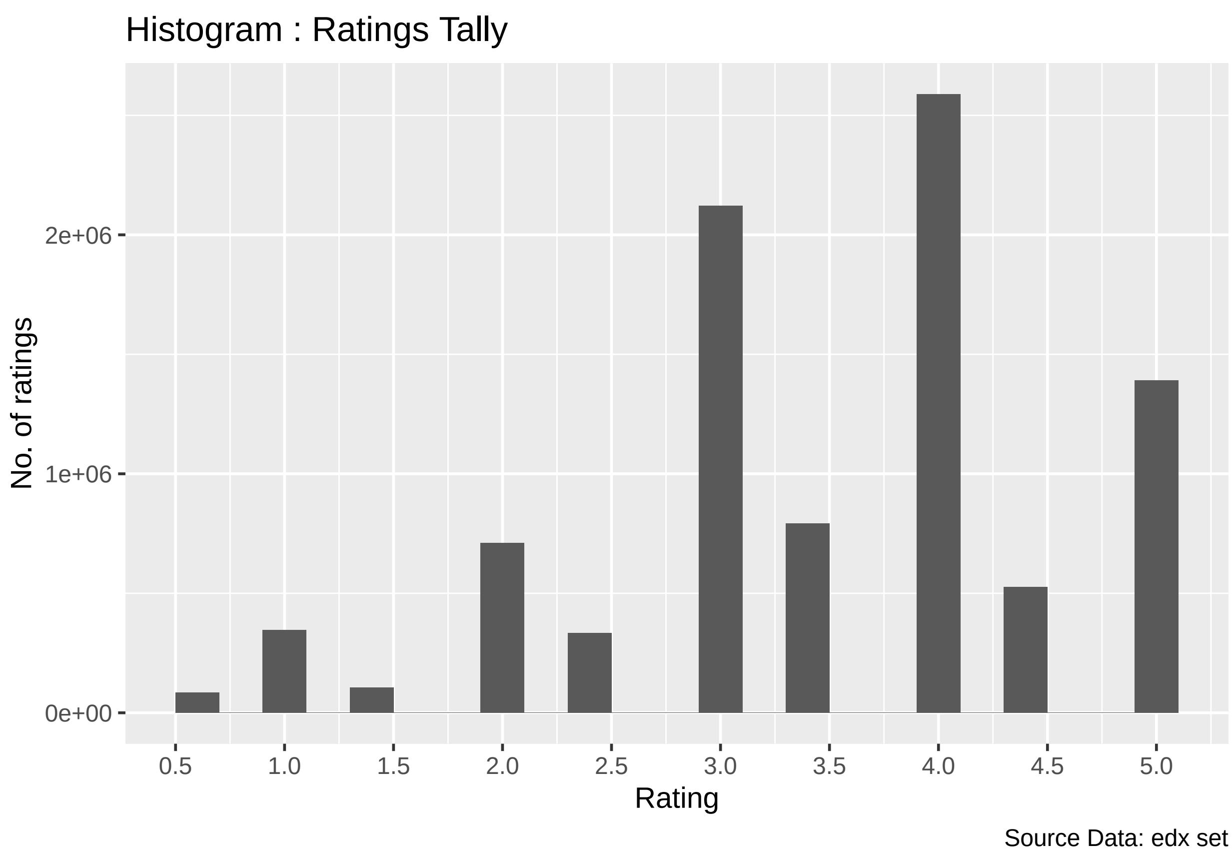
Table Showing the Rating Count

```
head(sort(table(edx$rating)),10)
```

```
##
##     0.5     1.5     2.5       1     4.5       2     3.5       5       3       4
##   85374  106426  333010  345679  526736  711422  791624 1390114 2121240 2588430
```

#Plot Histogram (Graphical Representation)

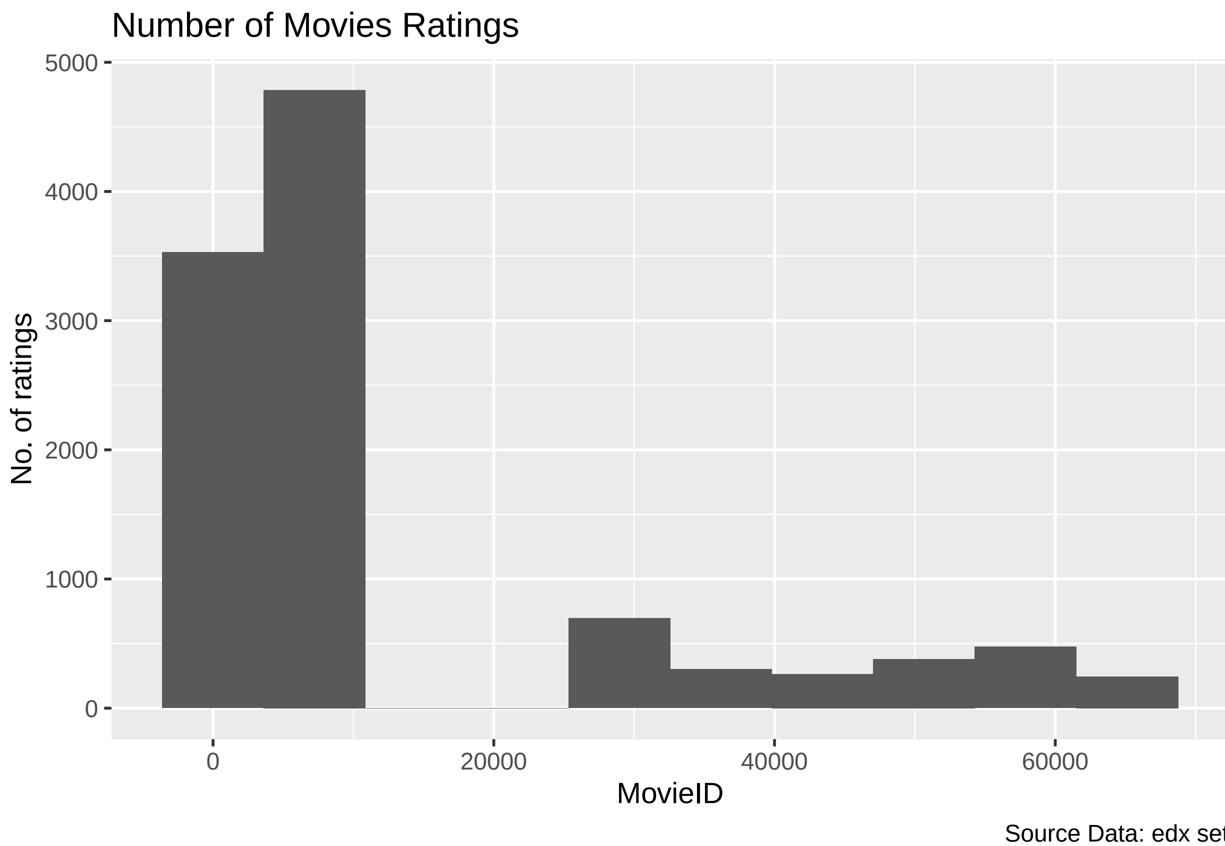
```
ggplot(edx, aes(x= edx$rating)) +
  geom_histogram( binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
    labs(x="Rating", y="No. of ratings", caption = "Source Data: edx set") +
  ggtitle("Histogram : Ratings Tally")
```



#Conclusions: The above graphical representation concludes the following facts: 1. No user had given 0 as rating 2. The top 5 ratings from most to least are : 4, 3, 5, 3.5 and 3. The half star ratings are less common than whole star ratings.

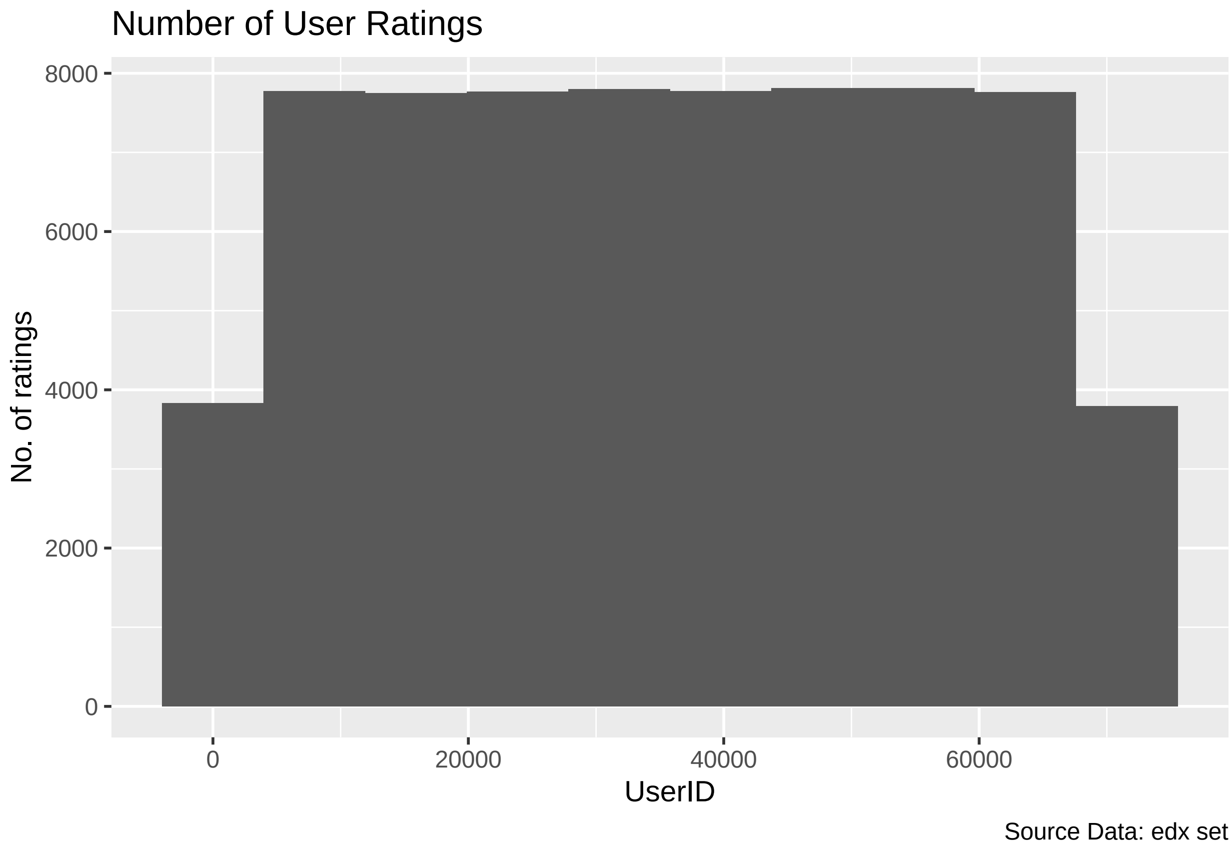
#Quantitative:Movie Id Vs Ratings Analysis #Plot Histogram (Graphical Representation)

```
edx %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(movieId)) + geom_histogram(bins = 10) +
  labs(x="MovieID", y="No. of ratings", caption = "Source Data: edx set") +
  ggtitle("Number of Movies Ratings")
```



#Conclusion: The above graphical representation for MovieID depicts that movies with few ratings tend to have more volatile ratings than movies which are rated more. #Quantitative:UserId Vs Ratings Analysis #Plot Histogram (Graphical Representation)

```
edx %>% group_by(userId) %>% summarize(n = n()) %>%
  ggplot(aes(userId)) + geom_histogram(bins = 10) +
  labs(x="UserID", y="No. of ratings", caption = "Source Data: edx set") +
  ggtitle("Number of User Ratings")
```



#Conclusion: The above graphical representation for User ID depicts that users who rate just a few movies tend to have more volatile ratings than users who rate lots of movies #Qualitative:Genres vs Ratings Analysis 1. Segregate and extract the Genres from the combination of Genres

```
edx_Splitted <- cSplit(edx, "genres", sep = "|" ,  direction = "long")
```

2. Calculate the Rating Count per Genre

```
edx_Genre_rating <- edx_Splitted %>%
  group_by(genres) %>%
  summarize(RatingCount = n()) %>%
  arrange(desc(RatingCount))
#Tabular Representation
datatable(edx_Genre_rating, rownames = FALSE, filter="top", options = list(pageLength = 50, scrollX=T) )
  formatRound('RatingCount',digits=0, interval = 3, mark = ",")
```

Show 50 entries

Search:

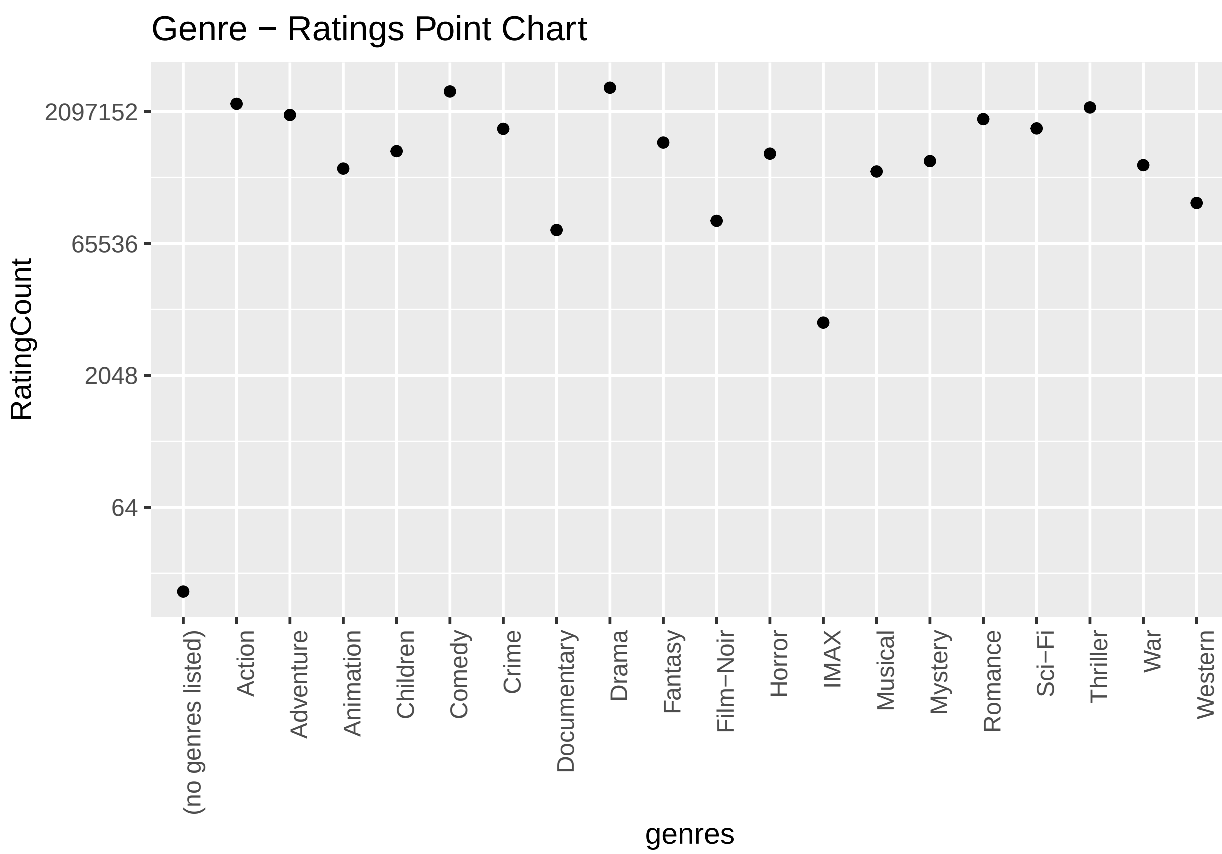
genres	RatingCount
All	All
Drama	3,910,127
Comedy	3,540,930
Action	2,560,545
Thriller	2,325,899
Adventure	1,908,892
Romance	1,712,100
Sci-Fi	1,341,183
Crime	1,327,715
Fantasy	925,637
Children	737,994
Horror	691,485
Mystery	568,332
War	511,147
Animation	467,168
Musical	433,080
Western	189,394
Film-Noir	118,541
Documentary	93,066
IMAX	8,181
(no genres listed)	7

Showing 1 to 20 of 20 entries

Previous 1 Next

#Graphical representation (Point Chart)

```
ggplot(edx_Genre_rating, aes(x= genres, y=RatingCount)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90,hjust = 1))+
  scale_y_continuous(trans = "log2")+
  ggtitle("Genre - Ratings Point Chart")
```



#Conclusion: The above tabular and graphical representation concludes that three Genres with the highest Rating count are: 1. Drama 2. Comedy 3. Action Movies with "no genre" have least movie ratings (7).

#Qualitative:Movie Titles vs Ratings Analysis Calculate the Rating Count per Movie

```
edx_Movie_Ratings <- edx %>%
  group_by(title, genres) %>%
  summarize(RatingCount = n()) %>%
  arrange(desc(RatingCount))
#Tabular Representation
datatable(edx_Movie_Ratings, rownames = FALSE, filter="top", options = list(pageLength = 10, scrollX=T)
  formatRound('RatingCount',digits=0, interval = 3, mark = ",")
```

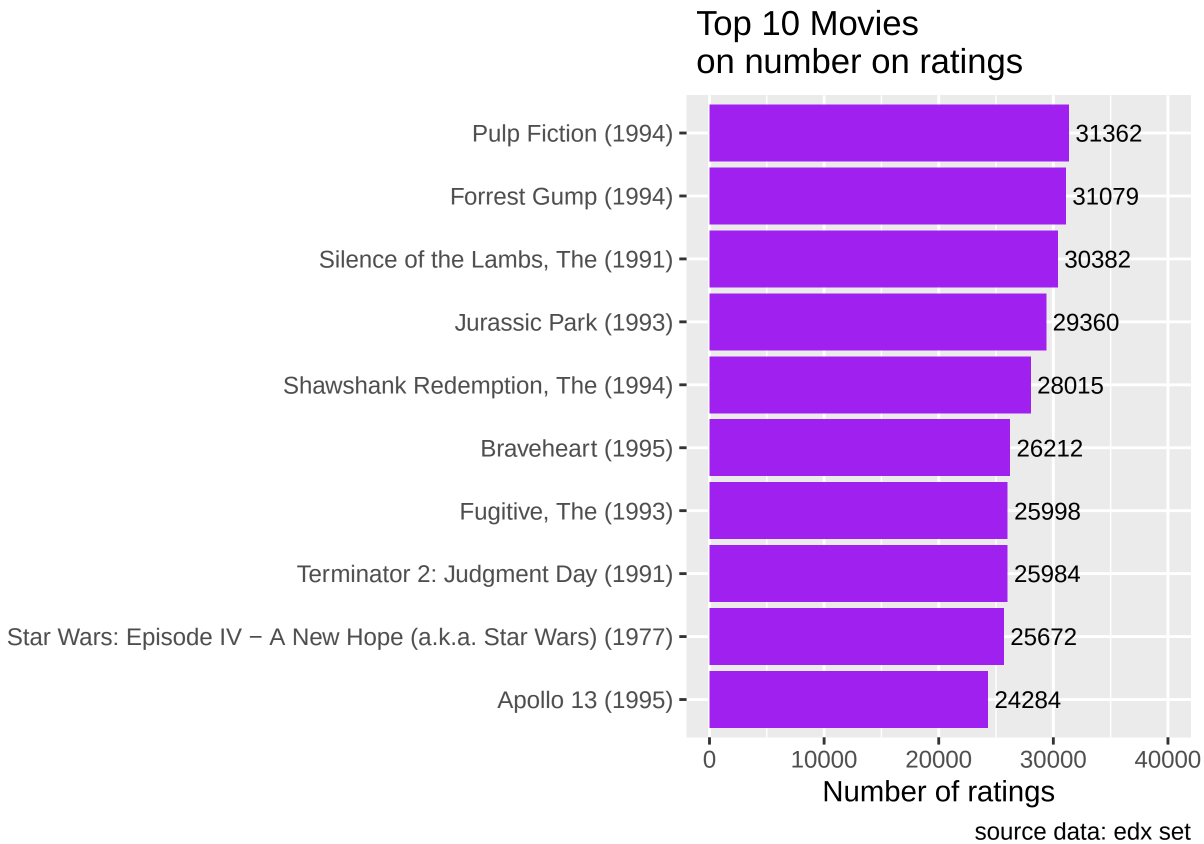
Show 10 entries Search:

title	genres	RatingCount
All	All	All
Pulp Fiction (1994)	Comedy|Crime|Drama	31,362
Forrest Gump (1994)	Comedy|Drama|Romance|War	31,079
Silence of the Lambs, The (1991)	Crime|Horror|Thriller	30,382
Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	29,360
Shawshank Redemption, The (1994)	Drama	28,015
Braveheart (1995)	Action|Drama|War	26,212
Fugitive, The (1993)	Thriller	25,998
Terminator 2: Judgment Day (1991)	Action|Sci-Fi	25,984
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	Action|Adventure|Sci-Fi	25,672
Apollo 13 (1995)	Adventure|Drama	24,284

Showing 1 to 10 of 10,677 entries Previous 1 2 3 4 5 … 1068 Next

#Graphical representation (Bar Chart)

```
edx %>% group_by(title) %>% summarise(count = n()) %>% top_n(10,count) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x=reorder(title, count), y=count)) + coord_flip(y=c(0, 40000)) +
  geom_bar(stat='identity', fill="purple") +
  labs(x="", y="Number of ratings") +
  geom_text(aes(label= count), hjust=-0.1, size=3) +
  labs(title=" Top 10 Movies \n on number on ratings" , caption = "source data: edx set")
```



#Conclusion: The above tabular and graphical representation of "Movie Title" confirms previous analysis. The movies which have the highest number of ratings are in the top genres categories : movies like Pulp fiction (1994), Forrest Ump(1994) or Jurrasic Park(1993) which are in the top 5 of movie's ratings number , are part of the Drama, Comedy or Action genres.

#Movie Age vs Movie Ratings Extract Premier year from Movie Title

```
#RegEx could have been used as well, but there are movie titles with number in it resulting in wrong de
PremierYear <- as.numeric(substr(as.character(edx$title),nchar(as.character(edx$title))-4,nchar(as.chara
```

Modify the Data Frame with Premier Year and also validate the Premier Year

```
edx_Movie_Aging_Details <- edx %>% mutate(Rated_Year = year(as_datetime(timestamp)), Premier_Year = Prem
head(edx_Movie_Aging_Details)
```

```
##   userId movieId rating                         title
## 1      1     122      5                Boomerang (1992)
## 2      1     185      5                 Net, The (1995)
## 3      1     292      5                 Outbreak (1995)
## 4      1     316      5                 Stargate (1994)
## 5      1     329      5 Star Trek: Generations (1994)
## 6      1     355      5         Flintstones, The (1994)
##                          genres Rated_Year Premier_Year
## 1                Comedy|Romance       1996         1992
## 2         Action|Crime|Thriller       1996         1995
## 3  Action|Drama|Sci-Fi|Thriller       1996         1995
```

```
## 4       Action|Adventure|Sci-Fi       1996         1994
## 5 Action|Adventure|Drama|Sci-Fi       1996         1994
## 6       Children|Comedy|Fantasy       1996         1994
```

```
edx_Movie_Aging_Details %>% filter(Premier_Year < 1900 || Premier_Year > 2018) %>% group_by(movieId, ti
```

```
## # A tibble: 0 x 5
## # Groups:   movieId, title, Premier_Year [0]
## # … with 5 variables: movieId <dbl>, title <chr>, Premier_Year <dbl>,
## #   Rated_Year <dbl>, n <int>
```

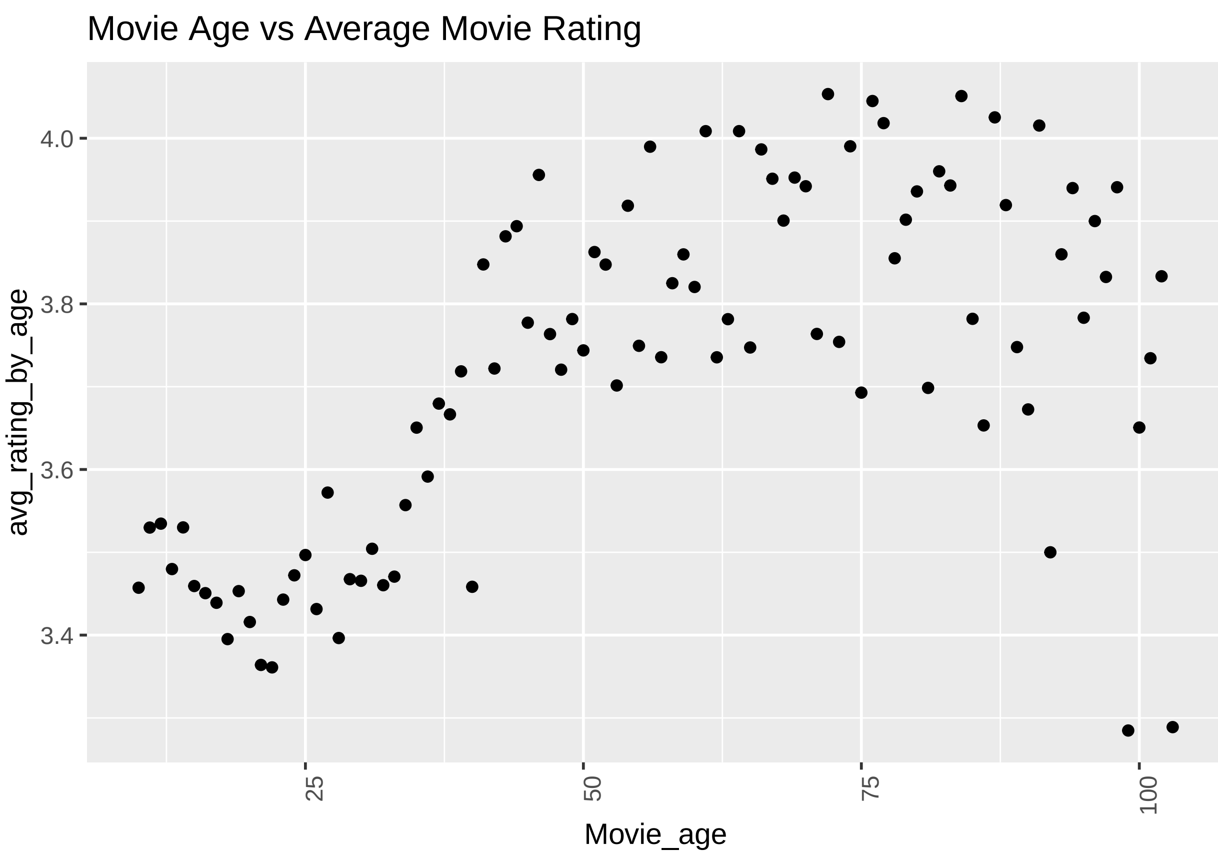
Calculate Movie Age and Average Rating

```
edx_Movie_Aging_Details_Avg <- edx_Movie_Aging_Details %>%
  mutate(Movie_age = 2018 - Premier_Year) %>% group_by(Movie_age) %>% summarize(avg_rating_by_age = mea
head(edx_Movie_Aging_Details_Avg)
```

```
## # A tibble: 6 x 2
##   Movie_age avg_rating_by_age
##       <dbl>             <dbl>
## 1       103              3.29
## 2       102              3.83
## 3       101              3.73
## 4       100              3.65
## 5        99              3.28
## 6        98              3.94
```

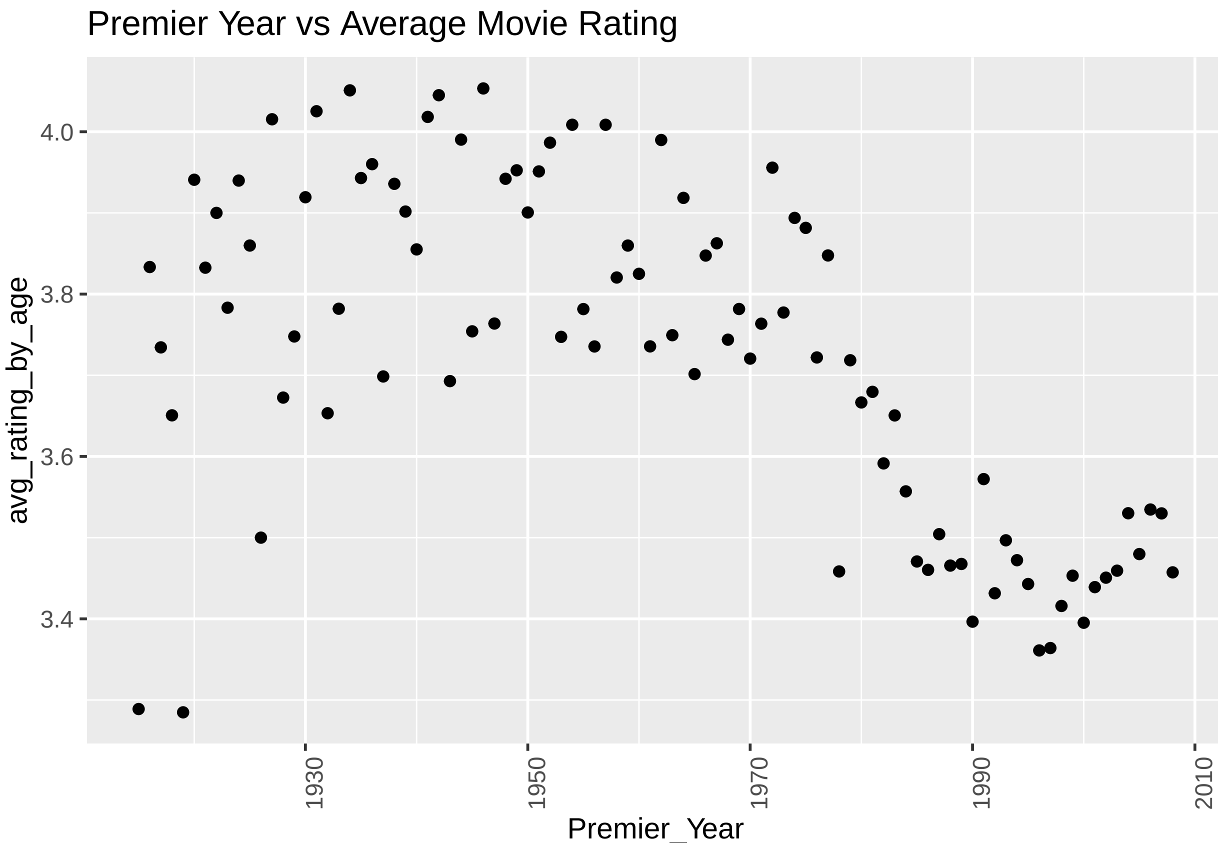
#Graphical Representation (Point Chart) : Age of movie vs average movie rating

```
edx_Movie_Aging_Details_Avg %>%
  ggplot(aes(Movie_age, avg_rating_by_age)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90,hjust = 1))+
  ggtitle("Movie Age vs Average Movie Rating")
```



#Graphical Representation (Point Chart) : Premier Year vs average movie rating

```
edx_avg_ratings <- edx_Movie_Aging_Details %>% group_by(Premier_Year) %>% summarise(avg_rating_by_age =
edx_avg_ratings %>% ggplot(aes(Premier_Year, avg_rating_by_age)) +   geom_point() +
  theme(axis.text.x = element_text(angle = 90,hjust = 1))+
  ggtitle("Premier Year vs Average Movie Rating")
```



#Conclusion: The above two graphical representations of "“Movie Age” and “Premier Year” against Average Movie Rating provide us with the following two facts: 1. Higher ratings the older a movies is up to 90 years old, then the ratings drop.In other words, Movies from earlier decades have more volatile ratings which has to be considered during accuracy calculation. 2. Recent movies get more ratings. Movies earlier than 1930 get few ratings, whereas newer movies, especially in the 90s get far more ratings.

IV. Result

MovieLens data set is a large data set (size 10M), hence an efficient method was needed to predict movie ratings. The PENALIZED LEAST SQUARE approach is based on the mean movie rating. This average is adjusted for user-effects and movie-effects in order to volatile ratings with respect to users and movies. To adjust for these effects, a penalty - LAMBDA - is taken into account.

#Determine Lambda

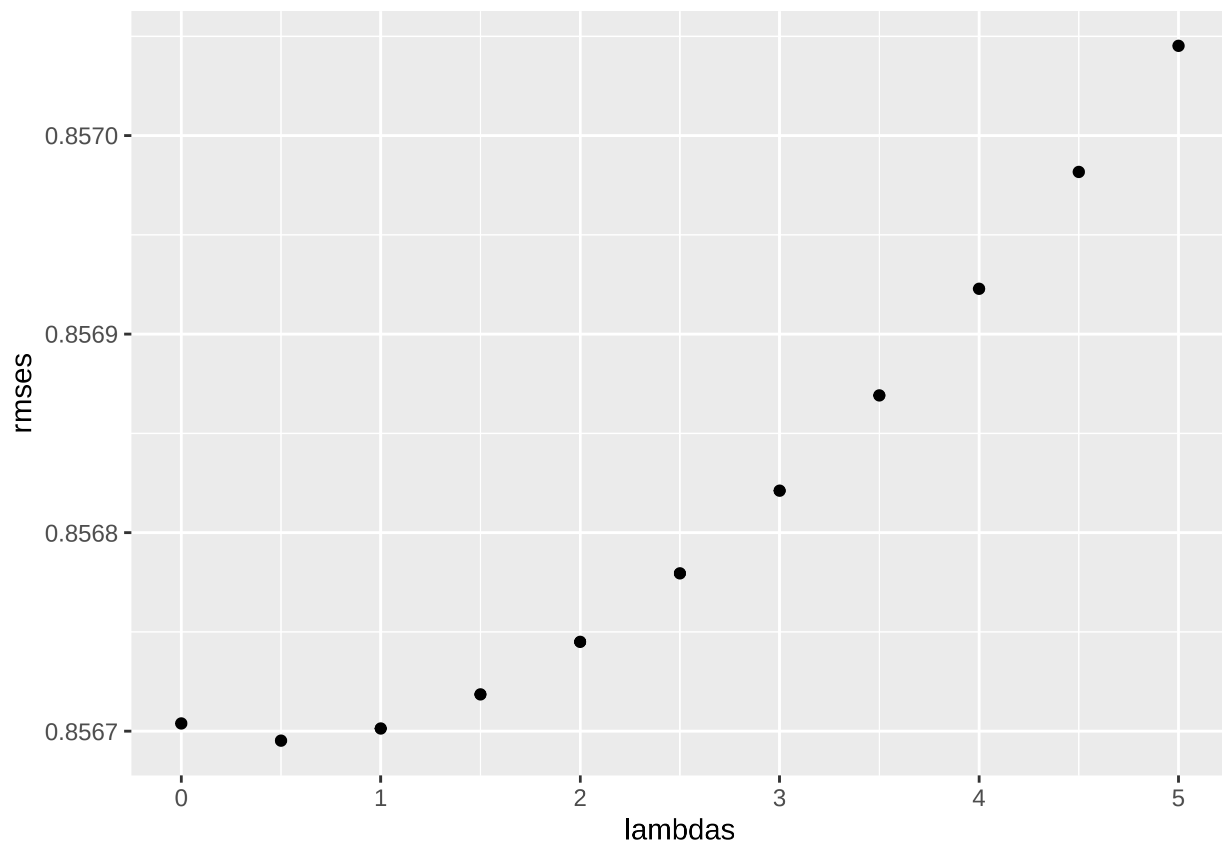
```
#RMSE Calculation
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
lambdas <- seq(0,5,.5)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_Movie_Aging_Details$rating)

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

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

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

  return(RMSE(predicted_ratings, edx_Movie_Aging_Details$rating))
})
#Graphical Representation of Lambda &rmses
qplot(lambdas, rmses)
```




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

```
## [1] "Optimal RMSE of 0.856695227644159 is achieved with Lambda 0.5"
```

Also, as per the above QPlot, the optimal RMSE is achieved with Lambda = 0.5 #Predicting the Validation Set using the optimal Lambda = 0.5

```r
mu <- mean(validation$rating)
l <- lambda
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + l))

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

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

RMSE(predicted_ratings, validation$rating)
```

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

After exploring the movies through graphical representations and calculating RMSE, it can be concluded that the best predictor for ratings was movieId, userId. The age of the movie didn't change the RMSE The RMSE calculated and validated against the Validation set - 0.8258487