HarvardX PH125.9x

Data Science: Capstone

MovieLens Project Submission

Riaz Mohamed

12/07/2021

Table of Contents

Introduction

The goal of this project is to build a movie recommendation system using machine learning.
In this project, we will explore and visualize the MovieLens dataset that consists of over 10 million film ratings. We will develop an ML model by creating both training and test sets to predict movie ratings on a validation set that results in an RMSE or Root Mean Square Error of less than . 86481.
The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed

Dataset

GroupLens Research has collected and made available rating data sets from the MovieLens website (https://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. titled "MovieLens," was developed by researchers at the University of Minnesota and was designed to generate personalized film recommendations. For this project, the 10M version will be used. It contains 10 million ratings on 10,000 movies by 72,000 users. It was released in 2009.

Steps of Analysis

- Data Preparation - We will load the data, split the dataset into edx and validation sets. We will then create a train and test set.
- Data Exploration - We will explore the data, plot histograms, create tables and graphs to observe what attributes of the dataset that affect ratings.

- Modeling - Once we determine the biases, we will apply these biases to our models to determine the RMSE we expect to achieve.
- Validation - Validate our model against the validation set

Data Preparation

Install all needed libraries

To install the required packages, we use if(!require and load the packages from http://cran.us.r-project.org

```
# Install all needed libraries

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")
if(!require(forcats)) install.packages("forcats", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra" , repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(plotly)) install.packages("plotly", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
```

Load Libraries

```
# Load required libraries

library(tidyverse)
library(caret)
library(kableExtra)
library(data.table)
library(dplyr)
library(tidyverse)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)
library(plotly)
library(ggthemes)
```

Load the MovieLens dataset

```
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)`
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)
```

View the edx Dataset

```
edx %>% as_tibble()
```

```
> edx %>% as_tibble()
# A tibble: 9,000,055 x 6
   userId movieId rating timestamp title                                  genres
    <int>   <dbl>  <dbl>     <int> <chr>                                  <chr>
 1      1     122      5 838985046 Boomerang (1992)                       Comedy|Romance
 2      1     185      5 838983525 Net, The (1995)                        Action|Crime|Thriller
 3      1     292      5 838983421 Outbreak (1995)                        Action|Drama|Sci-Fi|Thriller
 4      1     316      5 838983392 Stargate (1994)                        Action|Adventure|Sci-Fi
 5      1     329      5 838983392 Star Trek: Generations (1994)          Action|Adventure|Drama|Sci-Fi
 6      1     355      5 838984474 Flintstones, The (1994)                Children|Comedy|Fantasy
 7      1     356      5 838983653 Forrest Gump (1994)                    Comedy|Drama|Romance|War
 8      1     362      5 838984885 Jungle Book, The (1994)                Adventure|Children|Romance
 9      1     364      5 838983707 Lion King, The (1994)                  Adventure|Animation|Children|Drama|Music…
10      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (199… Action|Comedy
# … with 9,000,045 more rows
```

Train and Test Sets

We have split the original dataset into the edx and validation sets. We will now split the edx set into two (test and train). These sets will be used to both build and test our algorithm. We will test the accuracy of the results with a validation set

```
set.seed(1, sample.kind="Rounding")

# create a test & train set

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

# Matching userId and movieId in both train and test sets
test <- temp %>%
  semi_join(train, by = "movieId") %>%
  semi_join(train, by = "userId")

# Add rows baqck into the train set
removed <- anti_join(temp, test)
train <- rbind(train, removed)

rm(removed, temp, test_index)

# Table shows userId, movieId , ratings, timestamp, title and gnres
train %>% as_tibble()
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

View the train Dataset

```
train %>% as_tibble()
```

```
> train %>% as_tibble()
# A tibble: 8,100,065 x 6
   userId movieId rating timestamp title                                    genres
    <int>   <dbl>  <dbl>     <int> <chr>                                    <chr>
 1      1     122      5 838985046 Boomerang (1992)                         Comedy|Romance
 2      1     292      5 838983421 Outbreak (1995)                          Action|Drama|Sci-Fi|Thriller
 3      1     316      5 838983392 Stargate (1994)                          Action|Adventure|Sci-Fi
 4      1     329      5 838983392 Star Trek: Generations (1994)            Action|Adventure|Drama|Sci-Fi
 5      1     355      5 838984474 Flintstones, The (1994)                  Children|Comedy|Fantasy
 6      1     356      5 838983653 Forrest Gump (1994)                      Comedy|Drama|Romance|War
 7      1     362      5 838984885 Jungle Book, The (1994)                  Adventure|Children|Romance
 8      1     364      5 838983707 Lion King, The (1994)                    Adventure|Animation|Children|Drama|Music…
 9      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (199… Action|Comedy
10      1     377      5 838983834 Speed (1994)                             Action|Romance|Thriller
# … with 8,100,055 more rows
```

Data Exploration

User Effect

- Let us explore the train data set

```
> train %>% as_tibble()
# A tibble: 8,100,065 x 6
   userId movieId rating timestamp title                                    genres
    <int>   <dbl>  <dbl>     <int> <chr>                                    <chr>
 1      1     122      5 838985046 Boomerang (1992)                         Comedy|Romance
 2      1     292      5 838983421 Outbreak (1995)                          Action|Drama|Sci-Fi|Thriller
 3      1     316      5 838983392 Stargate (1994)                          Action|Adventure|Sci-Fi
 4      1     329      5 838983392 Star Trek: Generations (1994)            Action|Adventure|Drama|Sci-Fi
 5      1     355      5 838984474 Flintstones, The (1994)                  Children|Comedy|Fantasy
 6      1     356      5 838983653 Forrest Gump (1994)                      Comedy|Drama|Romance|War
 7      1     362      5 838984885 Jungle Book, The (1994)                  Adventure|Children|Romance
 8      1     364      5 838983707 Lion King, The (1994)                    Adventure|Animation|Children|Drama|Music…
 9      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (199… Action|Comedy
10      1     377      5 838983834 Speed (1994)                             Action|Romance|Thriller
# … with 8,100,055 more rows
```

Conclusion: We see from the table that the same user has rated multiple movies.

- Let us check the unique set from the train set

\

```
train %>% summarize(
  users=n_distinct(userId),
  movies=n_distinct(movieId),
  minRating=min(rating),
  maxRating=max(rating)
)
```

```
> # Lets us check the unique set
> train %>% summarize(
+   users=n_distinct(userId),
+   movies=n_distinct(movieId),
+   minRating=min(rating),
+   maxRating=max(rating)
+ )
  users movies minRating maxRating
1 69878  10677       0.5         5
> |
```

Conclusion: We see 69878 users and 10677 unique movies with a rating between 0.5 to 5. We also know that the same users have rated many movies, hence there may be movies that users haven’t rated and users who have not rated any movies. From this, we understand that this set contains many NA values as well.

- RATINGS: Let us check the top 10 movie ratings by the number of ratings and an average rating

```
edx %>% group_by(title) %>%
  summarize(avgRating = mean(rating),numberOfRatings = n()) %>%
  arrange(desc(avgRating)) %>%
  top_n(10, wt=avgRating)
```

```
+   top_n(10, wt=avgRating)
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 10 x 3
   title                                                                        avgRating numberOfRatings
   <chr>                                                                            <dbl>           <int>
 1 Blue Light, The (Das Blaue Licht) (1932)                                         5                   1
 2 Fighting Elegy (Kenka erejii) (1966)                                             5                   1
 3 Hellhounds on My Trail (1999)                                                    5                   1
 4 Satan's Tango (Sátántangó) (1994)                                                5                   2
 5 Shadows of Forgotten Ancestors (1964)                                            5                   1
 6 Sun Alley (Sonnenallee) (1999)                                                   5                   1
 7 Constantine's Sword (2007)                                                       4.75                2
 8 Human Condition II, The (Ningen no joken II) (1959)                              4.75                4
 9 Human Condition III, The (Ningen no joken III) (1961)                            4.75                4
10 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980) 4.75                4
```

Conclusion: Are these movies the highly rated movies? The top 5 have only 1 rating. To determine the highest rated movies, we need at an average a higher number of polls for the movie itself

- Let us take the top 10 Rated movies which have at the least 500 ratings

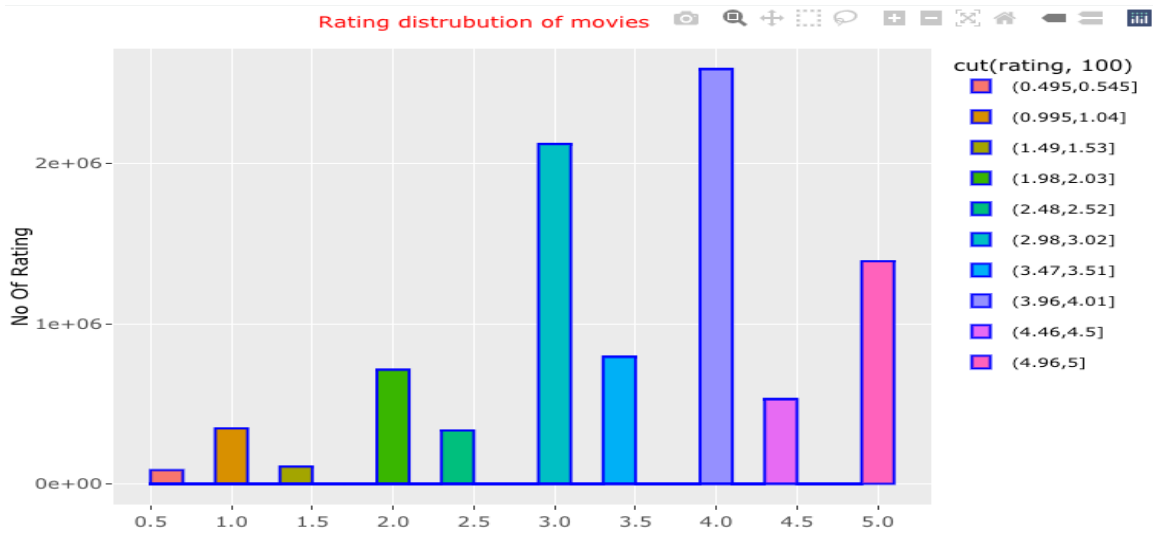
```
edx %>% group_by(title) %>%
  summarize(avgRating = mean(rating),numberOfRatings = n()) %>%
  arrange(desc(avgRating)) %>%
  top_n(10, wt=avgRating)
```

```
+   top_n(10, wt=avgRating)
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 10 x 3
   title                                         numberOfRatings avgRating
   <chr>                                                   <int>     <dbl>
 1 Shawshank Redemption, The (1994)                        28015      4.46
 2 Godfather, The (1972)                                   17747      4.42
 3 Usual Suspects, The (1995)                              21648      4.37
 4 Schindler's List (1993)                                 23193      4.36
 5 Casablanca (1942)                                       11232      4.32
 6 Rear Window (1954)                                       7935      4.32
 7 Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)            2922      4.32
 8 Third Man, The (1949)                                    2967      4.31
 9 Double Indemnity (1944)                                  2154      4.31
10 Paths of Glory (1957)                                    1571      4.31
```

Conclusion: The most popular movies have higher number of ratings

- We will now analyze the distribution of ratings vs number of ratings

```
# Distribution of Ratings
ratingDist <- ggplot(edx, aes(rating, fill = cut(rating, 100))) +
  geom_histogram(color = "blue", binwidth = 0.2) + scale_x_continuous(breaks = seq(0.5, 5, 0.5)) +
  labs(title = "Rating distrubution of movies", x = "Ratings", y = "No Of Rating") +
  theme(axis.text = element_text(size = 10),
  plot.title = element_text(size = 11, color = "red", hjust = 0.25))
```



Conclusion: The histogram confirms most viewers tend to rate a movie on an average at 3 or above.

Genre Effect

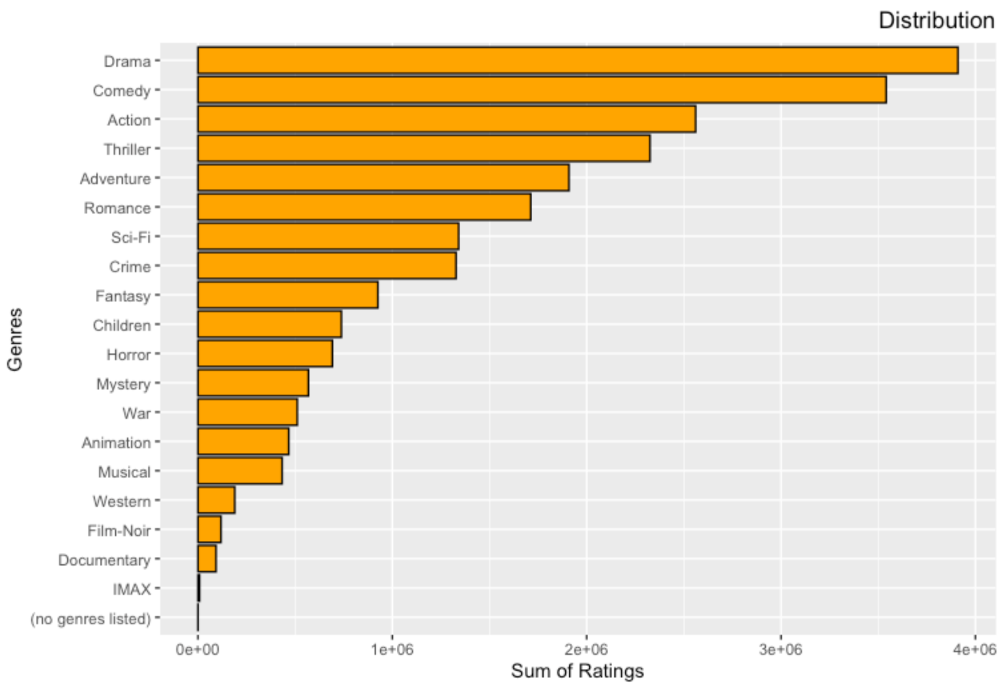
- We will now analyze if the genre of a movie line up with ratings and the number of ratings being rated

```
# THE GENRE EFFECT

edx_genres <-edx %>% separate_rows(genres, sep = "\\|")

edx_genres %>%as_tibble()

edx_genres %>%
  group_by(genres) %>% summarize(Ratings_Sum = n(), Average_Rating = mean(rating)) %>%
  arrange(-Ratings_Sum)


edx_genres %>%as_tibble()


# Arrange the Genres by Mean Rating
edx_genres %>%
  group_by(genres) %>% summarize(Ratings_Sum = n(), avgRating = mean(rating)) %>%
  arrange(-avgRating)
```

```
# A tibble: 20 x 3
   genres             Ratings_Sum avgRating
   <chr>                    <int>     <dbl>
 1 Film-Noir               118541      4.01
 2 Documentary              93066      3.78
 3 War                     511147      3.78
 4 IMAX                      8181      3.77
 5 Mystery                 568332      3.68
 6 Drama                  3910127      3.67
 7 Crime                  1327715      3.67
 8 (no genres listed)           7      3.64
 9 Animation               467168      3.60
10 Musical                 433080      3.56
11 Western                 189394      3.56
12 Romance                1712100      3.55
13 Thriller               2325899      3.51
14 Fantasy                 925637      3.50
15 Adventure              1908892      3.49
16 Comedy                 3540930      3.44
17 Action                 2560545      3.42
18 Children                737994      3.42
19 Sci-Fi                 1341183      3.40
20 Horror                  691485      3.27
```

- Visual Representation

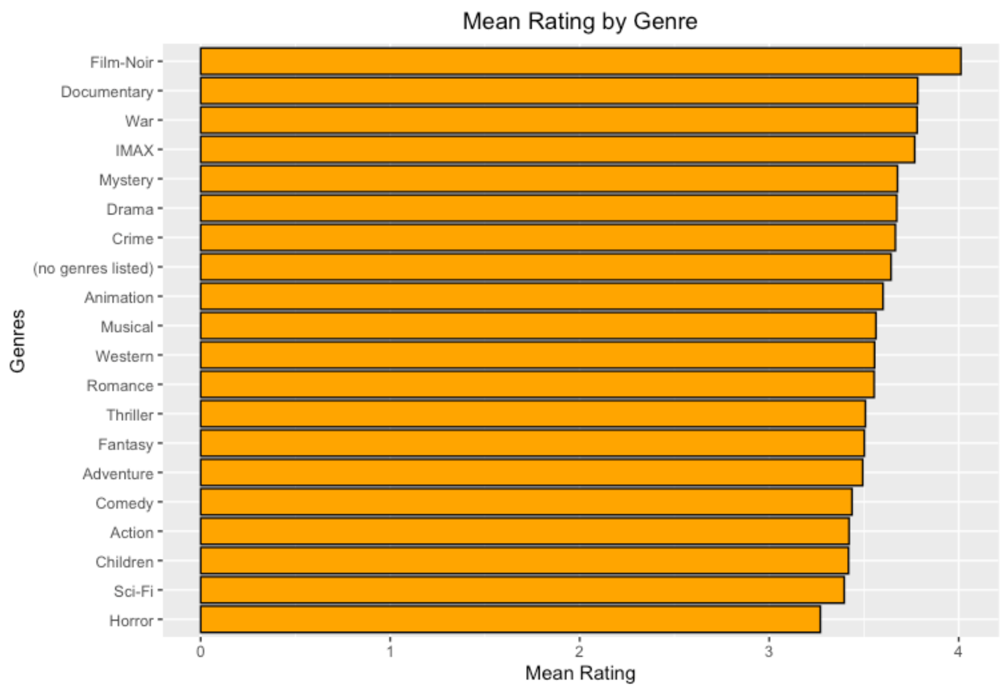
```
# Visual representation
# Coerce genres from characters to factors
edx$genres <-as.factor(edx$genres)
edx_genres$genres <-as.factor(edx_genres$genres)


# Sum of Movie Ratings per Genre
genres_ratings_edx <-edx_genres %>% group_by(genres) %>% summarize(Ratings_Sum = n())
genres_ratings_edx %>% ggplot(aes(x = Ratings_Sum , y = reorder(genres, Ratings_Sum)))+
  ggtitle("Distribution")+
  xlab("Sum of Ratings")+
  ylab("Genres")+
  geom_bar(stat = "identity",  fill = "orange", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```



Conclusion: The number of ratings for the first four-five highly rated genres has fewer ratings and hence some of these can skew data as well. We also have some false Genres

- Let us analyze the mean Rating per genre

```
# Visual representation
# Coerce genres from characters to factors
edx$genres <-as.factor(edx$genres)
edx_genres$genres <-as.factor(edx_genres$genres)


# Sum of Movie Ratings per Genre
genres_ratings_edx <-edx_genres %>% group_by(genres) %>% summarize(Ratings_Sum = n())
genres_ratings_edx %>% ggplot(aes(x = Ratings_Sum , y = reorder(genres, Ratings_Sum)))+
  ggtitle("Distribution")+
  xlab("Sum of Ratings")+
  ylab("Genres")+
  geom_bar(stat = "identity",  fill = "orange", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```



Conclusion: Average ratings for genres are between 3 and 4. Genre doesn't affect ratings.

Age of Movie Effect

To understand the effects of time (age of a movie) on ratings, we will require to convert the timestamps to years and then look at the affect

- Convert timestamps to the year of rating, we will do the same conversions for all our datasets (edx, validation , train, and test)

```
edx <- edx %>% mutate(timestamp = format(as.POSIXct(timestamp, origin = "1970-01-01",
                              tz = Sys.getenv("TZ")),"%Y"))
names(edx)[names(edx) == "timestamp"] <- "RatingYear"
head(edx) edx <- edx %>% mutate(timestamp = format(as.POSIXct(timestamp, origin = "1970-01-01",
                              tz = Sys.getenv("TZ")),"%Y"))
names(edx)[names(edx) == "timestamp"] <- "RatingYear"
head(edx)
```

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

- To get a sense of the time frame of the ratings we can use the range function

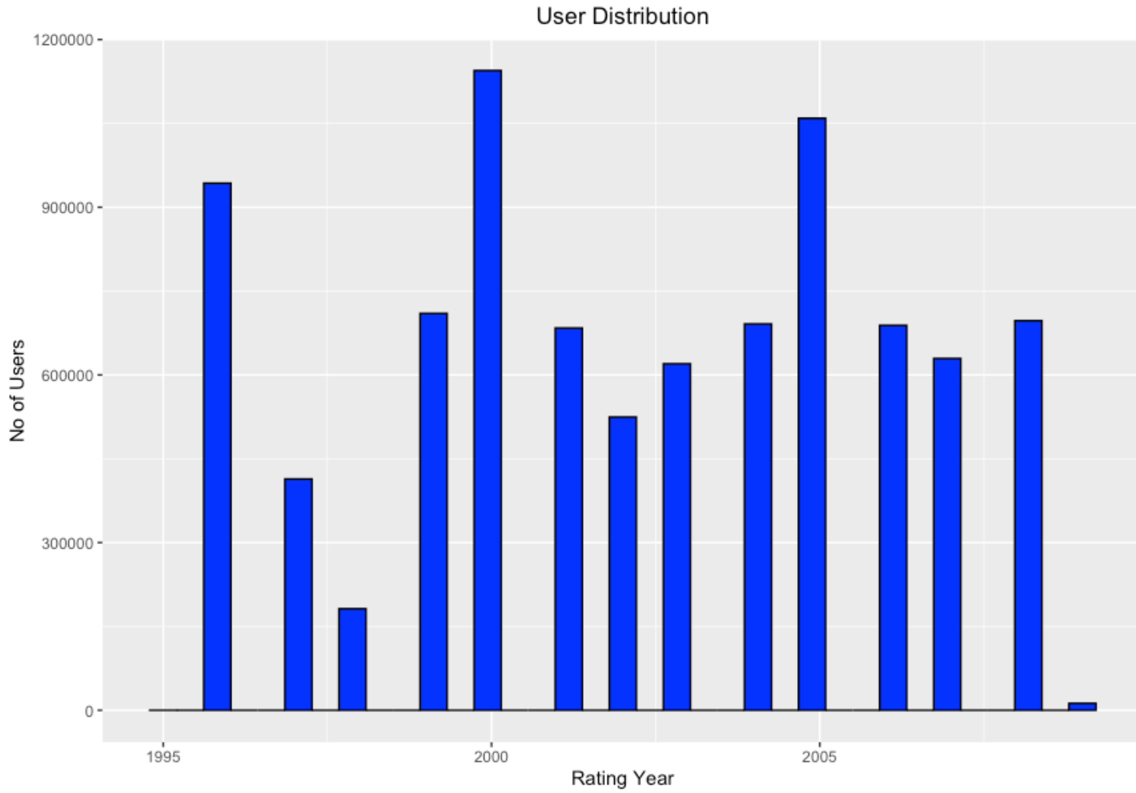
```
range(edx$RatingYear)
```

```
> range(edx$RatingYear)
[1] "1995" "2009"
```

Conclusion: A look at the range seems to point out that all movies have been rated between 1995-2009

- Visual Representation: A look at the range seems to point out that all movies have been rated between 1995-2009

```
# Visual representation
edx$RatingYear <-as.numeric(edx$RatingYear)
str(edx)

edx %>% ggplot(aes(RatingYear))+
  geom_histogram(fill = "blue", color = "black", bins = 35)+
  ggtitle("User Distribution")+
  xlab("Rating Year")+
  ylab("No of Users")+
  theme(plot.title = element_text(hjust = 0.5))
```



Conclusion: 1996, 2000 and 2005 have the highest user rating

- To find the age of a movie we will need to find the year it was released. The release year of the movie is part of the title

```
# A tibble: 9,000,055 x 6
   userId movieId rating RatingYear title                                    genres
    <int>   <dbl>  <dbl>      <dbl> <chr>                                    <fct>
 1      1     122      5       1996 Boomerang (1992)                         Comedy|Romance
 2      1     185      5       1996 Net, The (1995)                          Action|Crime|Thriller
 3      1     292      5       1996 Outbreak (1995)                          Action|Drama|Sci-Fi|Thriller
 4      1     316      5       1996 Stargate (1994)                          Action|Adventure|Sci-Fi
 5      1     329      5       1996 Star Trek: Generations (1994)            Action|Adventure|Drama|Sci-Fi
 6      1     355      5       1996 Flintstones, The (1994)                  Children|Comedy|Fantasy
 7      1     356      5       1996 Forrest Gump (1994)                      Comedy|Drama|Romance|War
 8      1     362      5       1996 Jungle Book, The (1994)                  Adventure|Children|Romance
 9      1     364      5       1996 Lion King, The (1994)                    Adventure|Animation|Children|Drama|Musical
10      1     370      5       1996 Naked Gun 33 1/3: The Final Insult (1994) Action|Comedy
# … with 9,000,045 more rows
```

```
premierDate <- stringi::stri_extract(edx$title, regex = "(\\d{4})", comments = TRUE ) %>% as.numeric()
edx <- edx %>% mutate(yearReleased = premierDate)
head(edx)
```

```
> premierDate <- stringi::stri_extract(edx$title, regex = "(\\d{4})", comments = TRUE ) %>% as.numeric()
> edx <- edx %>% mutate(yearReleased = premierDate)
> head(edx)
   userId movieId rating RatingYear                         title                        genres yearReleased
1:      1     122      5       1996              Boomerang (1992)                Comedy|Romance         1992
2:      1     185      5       1996               Net, The (1995)         Action|Crime|Thriller         1995
3:      1     292      5       1996               Outbreak (1995)  Action|Drama|Sci-Fi|Thriller         1995
4:      1     316      5       1996               Stargate (1994)       Action|Adventure|Sci-Fi         1994
5:      1     329      5       1996 Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi         1994
6:      1     355      5       1996       Flintstones, The (1994)       Children|Comedy|Fantasy         1994
>
```

- Visual Representation: Let us check if the age of a movie affects the movie ratings

```
edx %>% group_by(age_of_movie) %>% summarize(avg_movie_rating = mean(rating)) %>%
  ggplot(aes(age_of_movie, avg_movie_rating))+
  ggtitle("Age of Movie & Average Movie Rating")+
  xlab("Age of Movie")+
  ylab("Avg Movie Rating")+
  geom_smooth(method = "gam")+
  geom_point(color = "honeydew")+
  theme(plot.title = element_text(hjust = 0.5))
```

Conclusion: We see the longer the movie is there, there are more ratings as well as some of the highly-rated movies are there longer. This seems to be a positive effect.

Data Exploration Conclusion: We observe three factors that affect ratings of a movie , the movie itself (some movies are rated higher than others), users and the age of the movie. We will base our modeling on the three biases.

Modeling

From the above analysis, we know that the number of users, age of the movie seems to influence the overall ratings.

RMSE

The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed

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

Naive Baseline Model

The simplest model that can be built is a Naïve model that predicts the mean always. For the first model, we will predict the same rating for all movies. Here no bias is considered. This method confirms to a linear equation

Y u,i = Mu + E u,i .

Where E u,i = independent errors centered at 0

```
mean_train_mu <-mean(train$rating)
naivermse <- RMSE(test$rating, mean_train_mu)
naivermse
```

Conclusion: Naive RMSE : 1.060054

```
|method                  |      RMSE|
|:-----------------------|---------:|
|Naive Analysis by Mean  |  1.060054|
```

Median Model

Let us include the median or any other random number. The method confirms to a linear equation

Y u,i = Median + E u,i

where

Y u,i is the predicted rating

E u,i = independent errors centered at 0

```
mean_train_mu <-mean(train$rating)
median_train_mu <-median(train$rating)
medianrmse <-RMSE(test$rating, median_train_mu)
medianrmse
```

Conclusion: Median RMSE : 1.166756

```
|method                  |      RMSE|
|:-----------------------|---------:|
|Naive Analysis by Mean  |  1.060054|
|Analysis By Median      |  1.166756|
```

Movie Bias Model

From our data exploration, we know that some movies are rated more than others. So let us add another bias for movie effect

Y u,i = Mu + Movie Bias + E u,i

where

Y u,i is the predicted rating

E u,i = independent errors centered at 0

Mu = Average Rating

Create a prediction:

```
prediction_mov_eff <-mean_train_mu + test %>%
  left_join(movieBias, by = "movieId") %>% .$mean_rating_m
moviermse <-RMSE(test$rating, prediction_mov_eff)
moviermse
```

Conclusion: Movie Bias RMSE: 0.9429615. When we add movie bias to the equation the RMSE decreases but the model is not yet effective.

method	RMSE
Naive Analysis by Mean	1.0600537
Analysis By Median	1.1667562
Analysis By Movie Bias	0.9429615

Movie and User Bias Model

From our data exploration we know that some movies are rated more than others, users also affect the ratings of the movie. So let us add another bias to the above which is the user bias

Y u,i = Mu + Movie Bias + User Bias + E u,i

where

Y u,i is the predicted rating

E u,i = independent errors centered at 0

Mu = Average Rating

Create a prediction:

```
prediction_mov_usr_eff <-test %>% left_join(movieBias, by = "movieId") %>%
  left_join(movieUserBias, by = "userId") %>%
  mutate(predictions = mean_train_mu + mean_rating_m + mean_rating_m_u) %>% .$predictions
movie_usr_rmse <-RMSE(test$rating, prediction_mov_usr_eff)
movie_usr_rmse
```

Conclusion: Movie and User Bias RMSE: 0.8646843. When we add user bias to the equation the RMSE decreases and meets our target RMSE.

method	RMSE
Naive Analysis by Mean	1.0600537
Analysis By Median	1.1667562
Analysis By Movie Bias	0.9429615
Analysis By Movie & User Bias	0.8646843

Movie , User and Age of Movie Bias Model

Now let us introduce the movie age to the above. This is the movie age bias where the more the number of years the more the number of ratings

Y u,i = Mu + Movie Bias + User Bias + Movie Age Bias + E u,i
where
Y u,i is the predicted rating
E u,i = independent errors centered at 0
Mu = Average Rating

Create a prediction:

```
prediction_mov_usr_mage_eff <-test %>% left_join(movieBias, by = "movieId") %>%
  left_join(movieUserBias, by = "userId") %>% left_join(movieUserAgeBias, by = "age_of_movie") %>%
  mutate(predictions = mean_train_mu + mean_rating_m + mean_rating_m_u  + mean_rating_m_u_a)
%>% .$predictions
movie_usr_mage_rmse <-RMSE(test$rating, prediction_mov_usr_mage_eff)
movie_usr_mage_rmse
```

Conclusion: Movie User and Age of Movie Bias RMSE: 0.8643301. The movie age has a small difference in the RMSE.

method	RMSE
Naive Analysis by Mean	1.0600537
Analysis By Median	1.1667562
Analysis By Movie Bias	0.9429615
Analysis By Movie & User Bias	0.8646843
Analysis By Movie, User & Movie Age Bias	0.8643301

Conclusion: We have observed that there was a decrease in rmse when we had both movie and user bias when we added age to the equation there was very little difference.

Regularization

Regularization allows for reduced errors caused by movies with few ratings which can influence the prediction and skew the error metric. The method uses a tuning parameter, lambda, to minimize the RMSE. Modifying movie bias and user bias for movies with limited ratings.

Movie & User Bias Model with Regularization

- Adding a tuning parameter lambda to the model

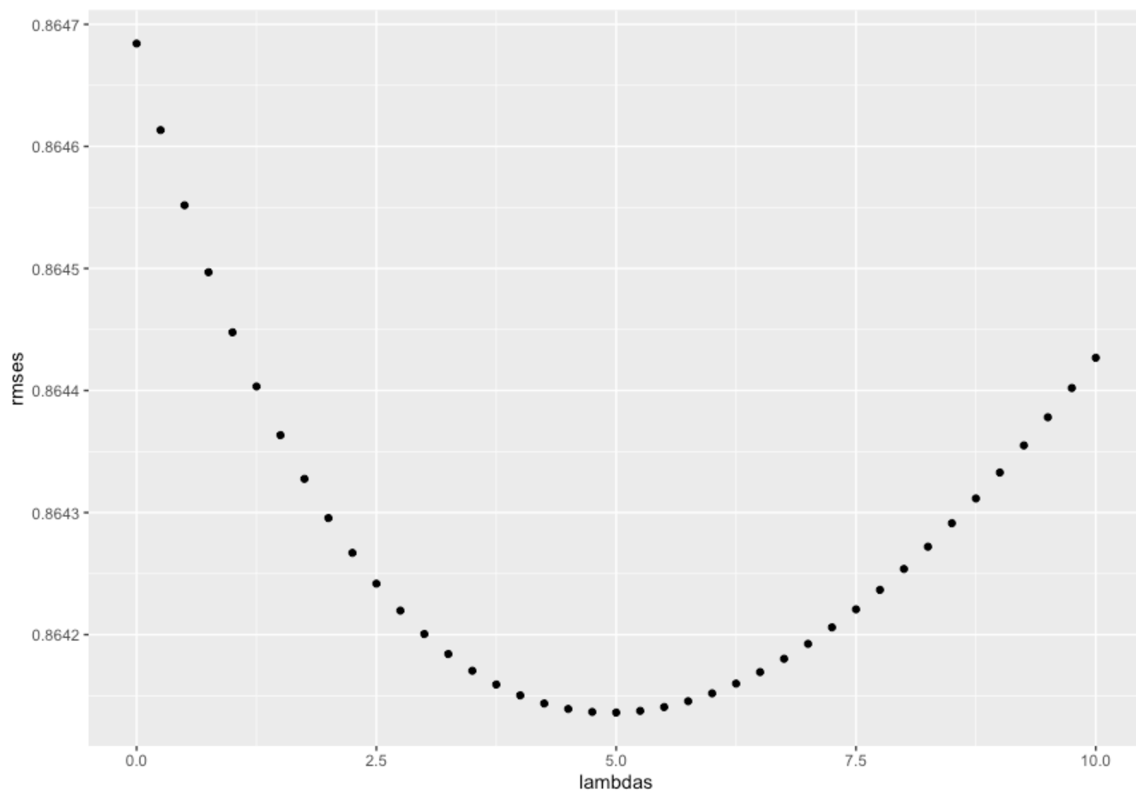
```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  train_mu <- mean(train$rating)

  movie_bias_i <- train %>%
    group_by(movieId) %>%
    summarise(movie_bias_i = sum(rating - train_mu)/(n()+l))

  user_bias_u <- train %>%
    left_join(movie_bias_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(user_bias_u = sum(rating - movie_bias_i - train_mu)/(n()+l))

  predicted_ratings <- test %>%
    left_join(movie_bias_i, by = "movieId") %>%
    left_join(user_bias_u, by = "userId") %>%
    mutate(predictions = train_mu + movie_bias_i + user_bias_u) %>%.$predictions


  return(RMSE(predicted_ratings, test$rating))
})


qplot(lambdas, rmses)
rmse_regularisation <- min(rmses)
rmse_regularisation


rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect with Regularization",
                                     RMSE = rmse_regularisation))
rmse_results %>% knitr::kable(., format = "pipe", padding = 2)
```

Conclusion: Movie and User Bias RMSE with regularization results is 0.8641362.

method	RMSE
Naive Analysis by Mean	1.0600537
Analysis By Median	1.1667562
Analysis By Movie Bias	0.9429615
Analysis By Movie & User Bias	0.8646843
Analysis By Movie, User & Movie Age Bias	0.8643301
Movie and User Effect with Regularization	0.8641362

- Visualization

```
qplot(lambdas, rmses)
```



Conclusion: The RMSE is minimum when the lambda is 5

Validation

Naive Baseline Model

The simplest model that can be built is a Naïve model that predicts the mean always. For the first model, we will predict the same rating for all movies. Here no bias is considered. This method confirms to a linear equation

Y u,i = Mu + E u,i .
Where E u,i = independent errors centered at 0

Validate our model against the validation set

```
mean_validation_mu <-mean(edx$rating)
naivermse_val <- RMSE(validation$rating, mean_validation_mu)
naivermse_val
```

Conclusion: Naive RMSE : 1.061202

```
|Method                   |   ValidationSet_RMSE|
|:------------------------|-------------------:|
|Naive Analysis by Mean   |            1.061202|
>
```

Median Model

Let's include the median or any other random number. The method confirms to a linear equation

Y u,i = Median + E u,i
where
Y u,i is the predicted rating
E u,i = independent errors centered at 0

Validate our model against the validation set

```
median_validation_mu <-median(edx$rating)
medianrmse_val <-RMSE(validation$rating, median_validation_mu)
medianrmse_val
```

Conclusion: Median RMSE: 1.168016

```
|Method                      |   ValidationSet_RMSE|
|:---------------------------|-------------------:|
|Naive Analysis by Mean      |            1.061202|
|Analysis By Median          |            1.168016|
>
```

Movie Bias Model

From our data exploration we know that some movies are rated more than others. So let us add another bias for movie effect

Y u,i = Mu + Movie Bias + E u,i

where

Y u,i is the predicted rating

E u,i = independent errors centered at 0

Mu = Average Rating

Validate our model against the validation set

Create a prediction:

```
prediction_mov_eff_val <-median_validation_mu + validation %>%
  left_join(movieBias_val, by = "movieId") %>% .$mean_rating_m_val
moviermse_val <-RMSE(validation$rating, prediction_mov_eff_val)
moviermse_val
```

Conclusion: Movie Bias RMSE: 0.9439087. When we add movie bias to the equation the RMSE decreases but the model is not yet effective.

```
|Method                      |   ValidationSet_RMSE|
|:---------------------------|-------------------:|
|Naive Analysis by Mean      |           1.0612018|
|Analysis By Median          |           1.1680160|
|Analysis By Movie Bias      |           0.9439087|
```

Movie and User Bias Model

From our data exploration we know that some movies are rated more than others, users also affect ratings off the movie. So let us add another bias to the above which is the user Bias

Y u,i = Mu + Movie Bias + User Bias + E u,i
where
Y u,i is the predicted rating
E u,i = independent errors centered at 0
Mu = Average Rating

Create a prediction:

```
prediction_mov_usr_eff_val <-validation %>% left_join(movieBias_val, by = "movieId") %>%
  left_join(movieUserBias_val, by = "userId") %>%
  mutate(predictions = mean_validation_mu + mean_rating_m_val + mean_rating_m_u_val) %>%
.$predictions
movie_usr_rmse_val <-RMSE(validation$rating, prediction_mov_usr_eff_val)
movie_usr_rmse_val
```

Conclusion: Movie and User Bias RMSE : 0.8653488.

```
|Method                          |  ValidationSet_RMSE|
|:-------------------------------|-------------------:|
|Naive Analysis by Mean          |           1.0612018|
|Analysis By Median              |           1.1680160|
|Analysis By Movie Bias          |           0.9439087|
|Analysis By Movie  & User Bias  |           0.8653488|
>
```

Movie , User and Age off Movie Bias Model

Now let us introduce movie age to the above. This is the movie age bias where the more the number of years the more the number of ratings

Y u,i = Mu + Movie Bias + User Bias + Movie Age Bias + E u,i
where
Y u,i is the predicted rating
E u,i = independent errors centered at 0
Mu = Average Rating

Create a prediction:

```
prediction_mov_usr_mage_eff_val <-validation %>% left_join(movieBias_val, by = "movieId") %>%
  left_join(movieUserBias_val, by = "userId") %>% left_join(movieUserAgeBias_val, by =
"age_of_movie") %>%
  mutate(predictions = mean_validation_mu + mean_rating_m_val + mean_rating_m_u_val  +
mean_rating_m_u_a_val) %>% .$predictions
movie_usr_mage_rmse_val <-RMSE(validation$rating, prediction_mov_usr_mage_eff_val)
movie_usr_mage_rmse_val
```

Conclusion: Movie User and Age of Movie Bias RMSE: 0.8650043

```
|Method                                   |  ValidationSet_RMSE|
|:----------------------------------------|-------------------:|
|Naive Analysis by Mean                   |           1.0612018|
|Analysis By Median                       |           1.1680160|
|Analysis By Movie Bias                   |           0.9439087|
|Analysis By Movie  & User Bias           |           0.8653488|
|Analysis By Movie, User & Movie Age Bias |           0.8650043|
> |
```

Movie & User Bias Model with Regularization

- Adding a tuning parameter lambda to the model

```
lambdaval <- seq(0, 10, 0.25)
rmses <- sapply(lambdaval, function(l){

  edx_mu <- mean(edx$rating)

  movie_bias_i_val <- edx %>%
    group_by(movieId) %>%
    summarise(movie_bias_i_val = sum(rating - edx_mu)/(n()+l))

  user_bias_u_val <- edx %>%
    left_join(movie_bias_i_val, by="movieId") %>%
    group_by(userId) %>%
    summarise(user_bias_u_val = sum(rating - movie_bias_i_val - edx_mu)/(n()+l))

  predicted_ratings_val <- validation %>%
    left_join(movie_bias_i_val, by = "movieId") %>%
    left_join(user_bias_u_val, by = "userId") %>%
    mutate(predictions = edx_mu + movie_bias_i_val + user_bias_u_val) %>%.$predictions
```

```
  return(RMSE(predicted_ratings_val, validation$rating))
})


qplot(lambdaval, rmses)
rmse_regularisation_val <- min(rmses)
rmse_regularisation_val
```

Conclusion: Movie and User Bias RMSE with regularization results in : 0.864817

```
|Method                                   |  ValidationSet_RMSE|
|:----------------------------------------|-------------------:|
|Naive Analysis by Mean                   |           1.0612018|
|Analysis By Median                       |           1.1680160|
|Analysis By Movie Bias                   |           0.9439087|
|Analysis By Movie  & User Bias           |           0.8653488|
|Analysis By Movie, User & Movie Age Bias |           0.8650043|
|Movie and User Effect with Regularization |           0.8648170|
>
```

Results and Inference

Our final model has resulted in an RMSE of 0.86481 which is below the targeted RMSE of 0.8775. The final model we derived was the Movie and User Effect with Regularization. We have seen that the movie, users bias are factors that contribute to the ratings heavily. Age is also a factor, but we saw that the RMSE decreased but not by a lot.

Model	RMSE	ValidationSet_RMSE
Naïve Analysis by Mean	1.0600537	1.0612018
Analysis by Median	1.1667562	1.1680160
Analysis by Movie Bias	0.9429615	0.9439087
Analysis by Movie & User Bias	0.8646843	0.8653488
Analysis by Movie , User and Movie Age Bias	0.8643301	0.8650043
Movie and User Effect with Regularization	0.8641362	0.8648170

There will be other biases that could impact the ratings e.g., geography, actors, etc. We have not explored all the biases in the dataset but started off with a few. There will also be other methods of modeling that would help us achieve a lower RMSE.