HarvardX PH125.9x
Course name: Data Science Capstone
Project: MovieLens Project

Name: Amrutha Killada

1 Executive Summary

This project is for educational purposes as a submission for the course HarvardX: PH125.9x Data Science Capstone. The objective of the project is to train a machine learning algorithm that could predict movie ratings.

1.1 Dataset

The dataset used for this project is MovieLens 10M dataset from MovieLens website.

https://grouplens.org/datasets/movielens/10m/
http://files.grouplens.org/datasets/movielens/ml-10m.zip

Code for loading the dataset and creating subsets of the data for training and testing:

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

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos =
"http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", 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 <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-
10M100K/ratings.dat"))),
```

```
                 col.names = c("userId", "movieId", "rating",
"timestamp"))

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

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# 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)
 edx <- rbind(edx, removed)

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

We use ‘edx’ data subset to train the algorithm and ‘validation’ subset to test movie ratings from the dataset.

1.2 Data set

```
#first 6 rows of the dataset
head(edx)
```

```
  userId movieId rating timestamp                         title                        genres
1      1     122      5 838985046              Boomerang (1992)                Comedy|Romance
2      1     185      5 838983525               Net, The (1995)         Action|Crime|Thriller
3      1     231      5 838983392          Dumb & Dumber (1994)                        Comedy
4      1     292      5 838983421               Outbreak (1995)  Action|Drama|Sci-Fi|Thriller
5      1     316      5 838983392               Stargate (1994)       Action|Adventure|Sci-Fi
6      1     329      5 838983392 Star Trek: Generations (1994) Action|Adventure|Drama|Sci-Fi
```

Edx dataset contains “userId”, “movieId”, ”rating”, “timestamp”, ”title”,” genres”.

```
#summary statistics
summary (edx)
```

```
     userId         movieId          rating        timestamp              title              genres         
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08   Length:9000061     Length:9000061    
 1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08   Class :character   Class :character  
 Median :35743   Median : 1834   Median :4.000   Median :1.035e+09   Mode  :character   Mode  :character  
 Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09                                         
 3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09                                         
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09                                         
```

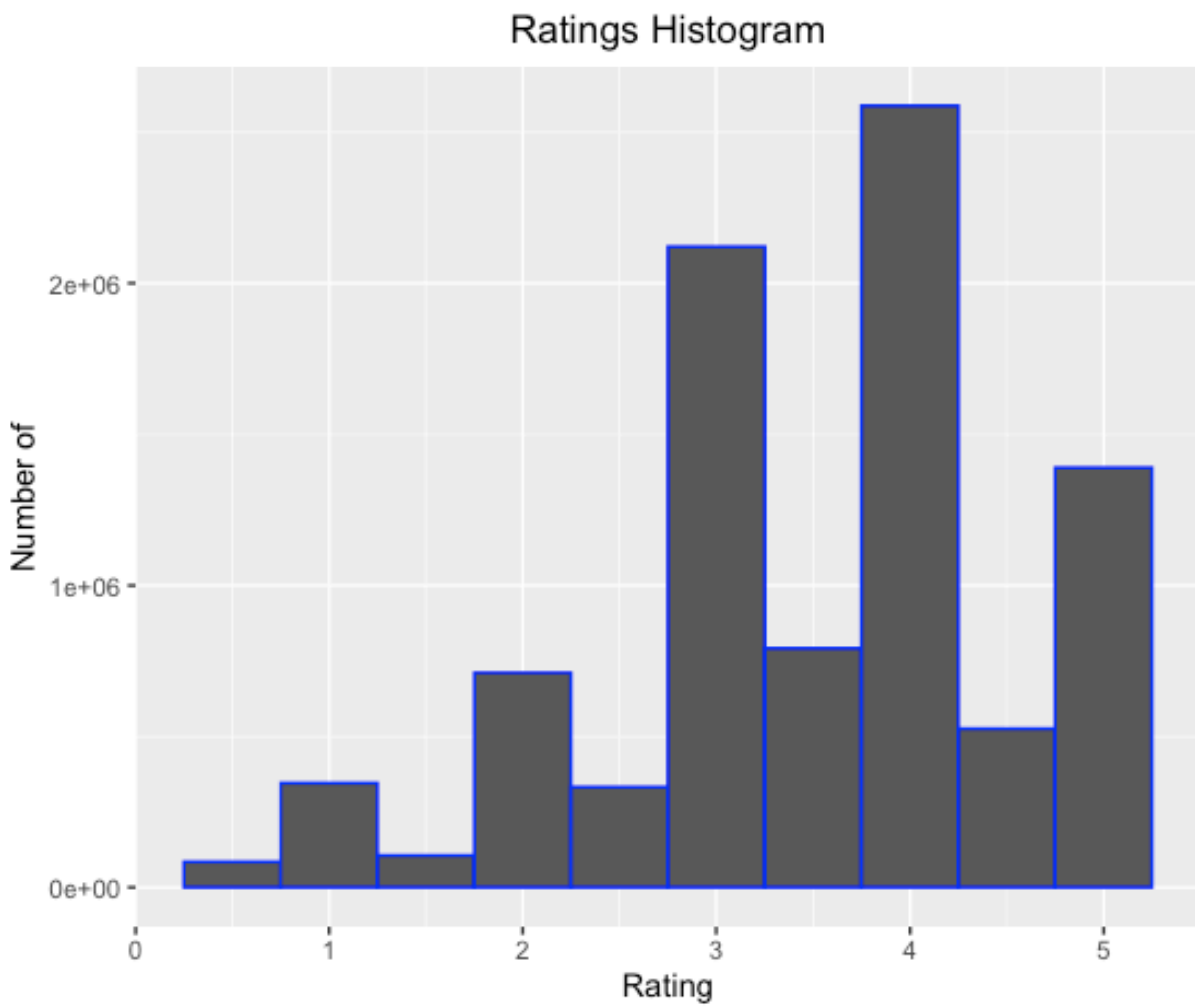
We learn that the movies are rated between 0.5 to 5.0 and mean movie rating is 3.512.

```
n_distinct(edx$movieId)
n_distinct(edx$userId)
n_distinct(edx$genres)
```

This reveals that there are 10677 unique movies, 69878 unique users, and 797 unique genres.

1.3 Data visualization

A plot of dataset mapping ratings and number of each unique ratings is attached below. The majority movie ratings are 4, 3 and 5 respectively.

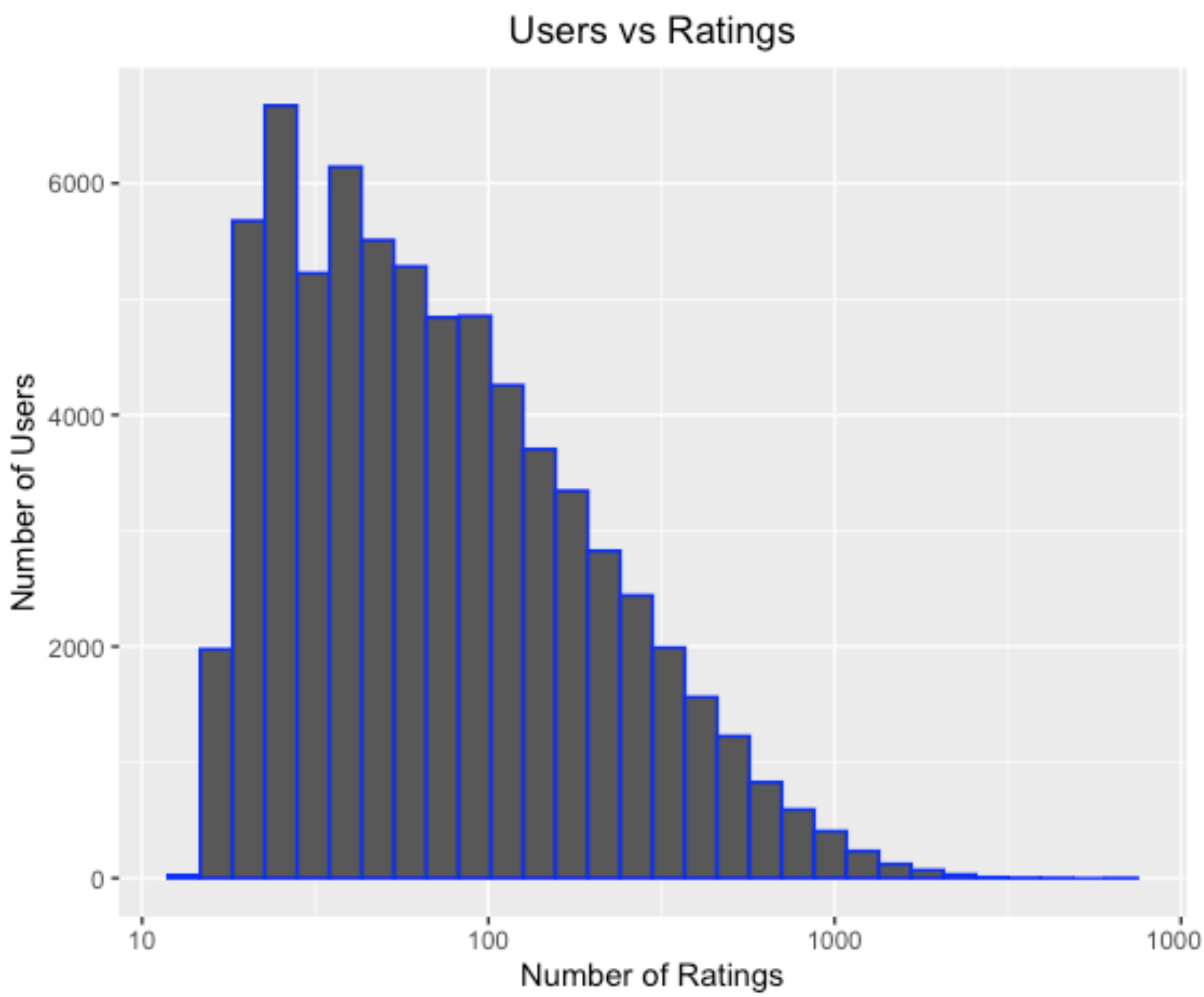



```
#Top 10 popular genres

pop_genres <- edx %>%
  group_by(genres) %>%
  summarise(number = n()) %>%
  arrange(desc(number))
knitr::kable(head(pop_genres,10))
```

```
|genres                              | number|
|:-----------------------------------|------:|
|Drama                               | 733353|
|Comedy                              | 700883|
|Comedy&#124;Romance                 | 365894|
|Comedy&#124;Drama                   | 323518|
|Comedy&#124;Drama&#124;Romance      | 261098|
|Drama&#124;Romance                  | 259735|
|Action&#124;Adventure&#124;Sci-Fi   | 220363|
|Action&#124;Adventure&#124;Thriller | 148933|
|Drama&#124;Thriller                 | 145359|
|Crime&#124;Drama                    | 137424|
```

We see that Drama and Comedy are the most popular genres.

```
#ratings vs users

edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "blue", bin = 30) +
  scale_x_log10() +
  xlab("Number of Ratings") +
  ylab("Number of Users") +
  ggtitle("Users vs Ratings") +
  theme(plot.title = element_text(hjust = 0.5))
```



We can see that most of the users rate between 10 to 100 movies.

1.4 Analysis
Litreture Review :RMSE – The Residual Mean Square Error(RMSE) is the error function that will measure accuracy and quantify the typical error when predicting movie ratings in this assignment.

Basic Mean Analysis

This prediction model uses the mean of the dataset to predict the rating for movies. The model assumes that all differences are due to random error.

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

rmse_basic <- RMSE(validation$rating, mu)
rmse_basic
```

[1] 1.060651

```
#creating a table to display all the results of analysis

 rmse_table = tibble(Analysis = "Basic Analysis", RMSE = rmse_basic)
 rmse_table %>% knitr::kable()
```

```
|Analysis       |     RMSE|
|:--------------|--------:|
|Basic Analysis | 1.060651|
```

Movie effect model

This model calculates a bias term for each movie based on the difference between the movies mean rating and the overall mean rating.

```
#Introducing movie effect
movie_effect <- edx %>%
  group_by(movieId) %>%
  summarise(a = mean(rating - mu))
predict_ratings <- mu + validation %>%
  left_join(movie_effect, by = "movieId") %>%
  pull(a)
rmse_movie_effect <- RMSE(predict_ratings, validation$rating)
rmse_movie_effect
```

[1] 0.9437046

```
#Adding Movie effect result to the table
rmse_table <- bind_rows(rmse_table, tibble(Analysis = "Movie Effect
Analysis", RMSE = rmse_movie_effect))
rmse_table %>% knitr::kable()
```

```
|Analysis              |      RMSE|
|:---------------------|---------:|
|Basic Analysis        | 1.0606506|
|Movie Effect Analysis | 0.9437046|
```

User effect model

This model calculates a bias term for each user based on the difference between the users mean rating and the overall mean rating.

```
#Introducing user effect
user_effect <- edx %>%
  group_by(userId) %>%
  summarise(b = mean(rating - mu))
predict_ratings <- validation %>%
  left_join(user_effect, by = "userId") %>%
  pull(b)
rmse_user_effect <- RMSE(predict_ratings, validation$rating)
rmse_user_effect
```

[1] 3.645477

```
#Adding the User effect result to the table
rmse_table <- bind_rows(rmse_table, tibble(Analysis = "User Effect
Analysis", RMSE = rmse_user_effect))
rmse_table %>% knitr::kable()
```

```
|Analysis              |      RMSE|
|:---------------------|---------:|
|Basic Analysis        | 1.0606506|
|Movie Effect Analysis | 0.9437046|
|User Effect Analysis  | 3.6454765|
```

Genre effect model

This model calculates a bias term for each genre based on the difference between the genres mean rating and the overall mean rating.

```
#Introducing genre effect
genre_effect <- edx %>%
  group_by(genres) %>%
  summarise(c = mean(rating - mu))
predict_ratings <- validation %>%
  left_join(genre_effect, by = "genres") %>%
  pull(c)
rmse_genre_effect <- RMSE(predict_ratings, validation$rating)
rmse_genre_effect
```

[1] 3.656522

```
#Adding Genre effect result to the table
rmse_table <- bind_rows(rmse_table, tibble(Analysis = "Genre Effect
Analysis", RMSE = rmse_genre_effect))
rmse_table %>% knitr::kable()
```

```
|Analysis              |      RMSE|
|:---------------------|---------:|
|Basic Analysis        | 1.0606506|
|Movie Effect Analysis | 0.9437046|
|User Effect Analysis  | 3.6454765|
|Genre Effect Analysis | 3.6565224|
```

Movie and User effect model

In this model we incorporate user bias to the movie model.

```
#Introducing both movie and user effects
user_effect <- edx %>%
  left_join(movie_effect, by = "movieId") %>%
  group_by(userId) %>%
  summarise(a_u = mean(rating - mu - a))
predict_ratings <- validation %>%
  left_join(movie_effect, by = "movieId") %>%
  left_join(user_effect, by = "userId") %>%
  mutate(d = mu + a + a_u) %>%
  pull(d)
rmse_movie_user_effect <- RMSE(predict_ratings, validation$rating)
rmse_movie_user_effect
```

[1] 0.8655329

```
#Addind Movie-User Effect result to the table
rmse_table <- bind_rows(rmse_table, tibble(Analysis = "Movied & User
Effect Analysis", RMSE = rmse_movie_user_effect))
rmse_table %>% knitr::kable()
```

```
|Analysis                     |      RMSE|
|:----------------------------|---------:|
|Basic Analysis               | 1.0606506|
|Movie Effect Analysis        | 0.9437046|
|User Effect Analysis         | 3.6454765|
|Genre Effect Analysis        | 3.6565224|
|Movied & User Effect Analysis | 0.8655329|
```

Regularized Movie and User Effect

Regularization allows for reduced errors which are caused by movies which have fewer ratings.

```
#regularization
lambdas <- seq(0, 10, 20 , 1)
rmses <- sapply(lambdas, function(1) {
  mu <- mean(edx$rating)
  x <- edx %>%
    group_by(movieId) %>%
    summarise(x = sum(rating - mu) / (n()+1))
  y <- edx %>%
    left_join(x, by = "movieId") %>%
    group_by(userId) %>%
    summarise(y = sum(rating - x - mu) / (n()+1))

  predicted_ratings <- validation %>%
    left_join(x, by = "movieId") %>%
    left_join(y, by = "userId") %>%
    mutate(z = mu + x + y) %>%
    pull(z)
  return(RMSE(predicted_ratings, validation$rating))
})
rmse_regularization <- min(rmses)
rmse_regularization
```

[1] 0.8653141

```
#Adding the Regularization result to the table
rmse_table <- bind_rows(rmse_table, tibble(Analysis = "Regularized
Movie and User Effect", RMSE = rmse_regularization))
rmse_table %>% knitr::kable()
```

```
|Analysis                         |      RMSE|
|:--------------------------------|---------:|
|Basic Analysis                   | 1.0606506|
|Movie Effect Analysis            | 0.9437046|
|User Effect Analysis             | 3.6454765|
|Genre Effect Analysis            | 3.6565224|
|Movied & User Effect Analysis    | 0.8655329|
|Regularized Movie and User Effect | 0.8653141|
```

The RMSE is further reduced with regularization.

Results

Analysis	RMSE
Basic Analysis	1.0606506
Movie Effect Analysis	0.9437046
User Effect Analysis	3.6454765
Genre Effect Analysis	3.6565224
Movied & User Effect Analysis	0.8655329
Regularized Movie and User Effect	0.8653141

The lowest RMSE predicted is 0.8653141

In conclusion the best model is Regularized Movie and User Effect model to predict the Movie ratings.