Movie Lense Project

Roja Kasula

March 01,2022

Contents

1 Executive Summary :

The Movie Lense recommendation system is built on the MovieLens data set that was provided as part of the edx course.The goal of the project is to create and train a recommendation machine learning algorithm to predict a movie ratings given by a users in the data set.RMSE is The root mean square error (RMSE) allows us to measure how far predicted values are from observed values in a regression analysis. It will be used to evaluate the accuracy of the algorithm, The required criteria for the projects is a RMSE lower than 0.8775.

1.1 Load Dataset

Initial analysis of the data set is performed below to understand the data . Data set is downloaded as per instruction from the MovieLens 10M dataset.

```
# Create edx set, validation set

# Install Packages
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")
if(!require(hexbin)) install.packages("hexbin", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",repos = "http://cran.us.r-project.org")
if(!require(rmarkdown)) install.packages("rmarkdown",repos = "http://cran.us.r-project.org")

# Load Libraries
library(tidyverse)
library(caret)
library(data.table)
library(hexbin)
library(ggplot2)
library(rmarkdown)


# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

# Check for download
datafile <- "MovieLens.RData"
if(!file.exists("MovieLens.RData"))
{
  print("Download")
  dl <- tempfile()
  download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

  ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                        col.names = c("userId", "movieId", "rating", "timestamp"))

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

  # Validation set will be 10% of MovieLens data

  set.seed(1, sample.kind = "Rounding") # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
  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)

  save(edx, validation, file = datafile)

} else {
  load(datafile)
}
```

Edx dataset contains rows corresponding to an users rating of a movie. The set contains the vaiables: "userId", "movieId", "rating", "timestamp", "title", "genres".

```
# Data Analysis
# Summary of the  Data

head(edx, 6)
```

```
##   userId movieId rating timestamp title genres
## 1      1     122      5 838985046  <NA>   <NA>
## 2      1     185      5 838983525  <NA>   <NA>
## 4      1     292      5 838983421  <NA>   <NA>
## 5      1     316      5 838983392  <NA>   <NA>
## 6      1     329      5 838983392  <NA>   <NA>
## 7      1     355      5 838984474  <NA>   <NA>
```

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

Summary of the data set displays structure of edx with no missing values. Movies are rated between 0.5 and 5.0, and total no.of rows = 9000055 .

```
# Lists movies, users and genres on edx data set
```

The dataset contains ~10,700 unique movies, ~70,000 unique movies, and ~800 unique combinations of genres, and a mean movie rating of ~3.5 out of 5.

```
edx %>% summarise(
  uniq_movies = n_distinct(movieId),
  uniq_users =  n_distinct(userId),
  uniq_genres = n_distinct(genres))
```

```
##   uniq_movies uniq_users uniq_genres
## 1       10677      69878           1
```

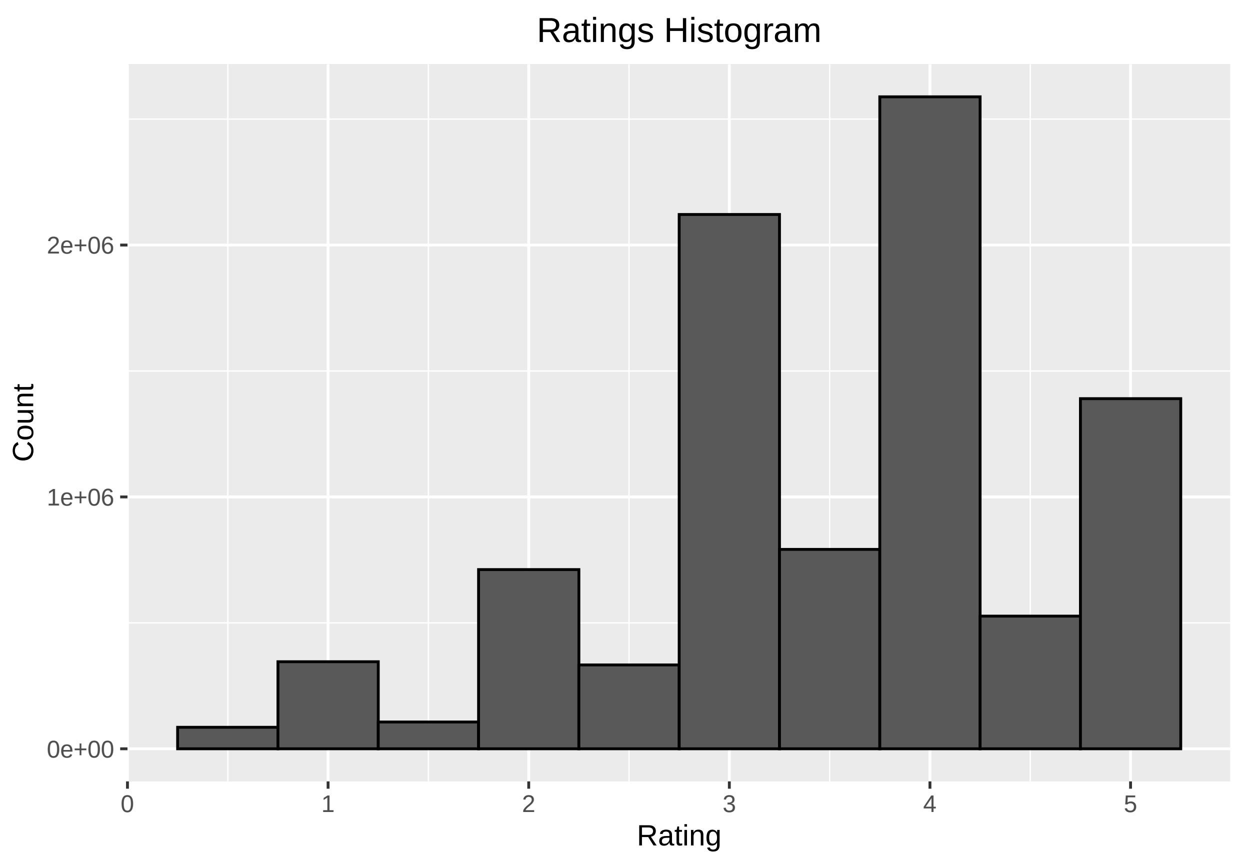
```
# Ratings Mean on edx data set
mean(edx$rating)
```

```
## [1] 3.512465
```

```
# Ratings Histogram on edx data set.
```

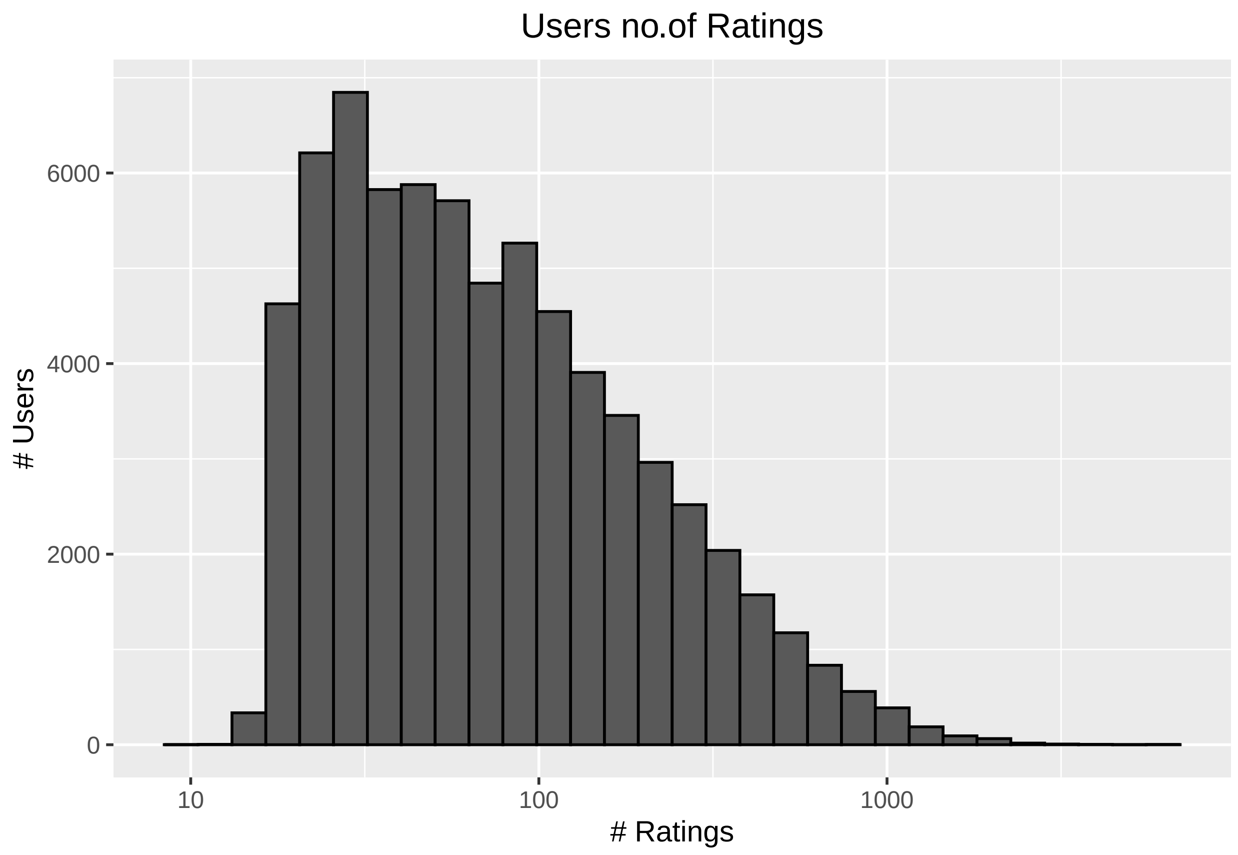
A histogram of the data set mapping ratings and counts are below. The highest rating users give to movies are, in decreasing order 4.0, 3.0, 5.0. Overall users rate movies higher in the ratings scale. Similarly, users are more likely to rate full ratings rather than half ratings on the scale.

```
# Ratings Histogram
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.5, color = "black",bins=30) +
  xlab("Rating") +
  ylab("Count") +
  ggtitle("Ratings Histogram") +
  theme(plot.title = element_text(hjust = 0.5))
```



The majority of users rate between 10 and 100 movies, while some may rate over 1,000. Including a variable in the model to consider for no. of ratings should be discussed.

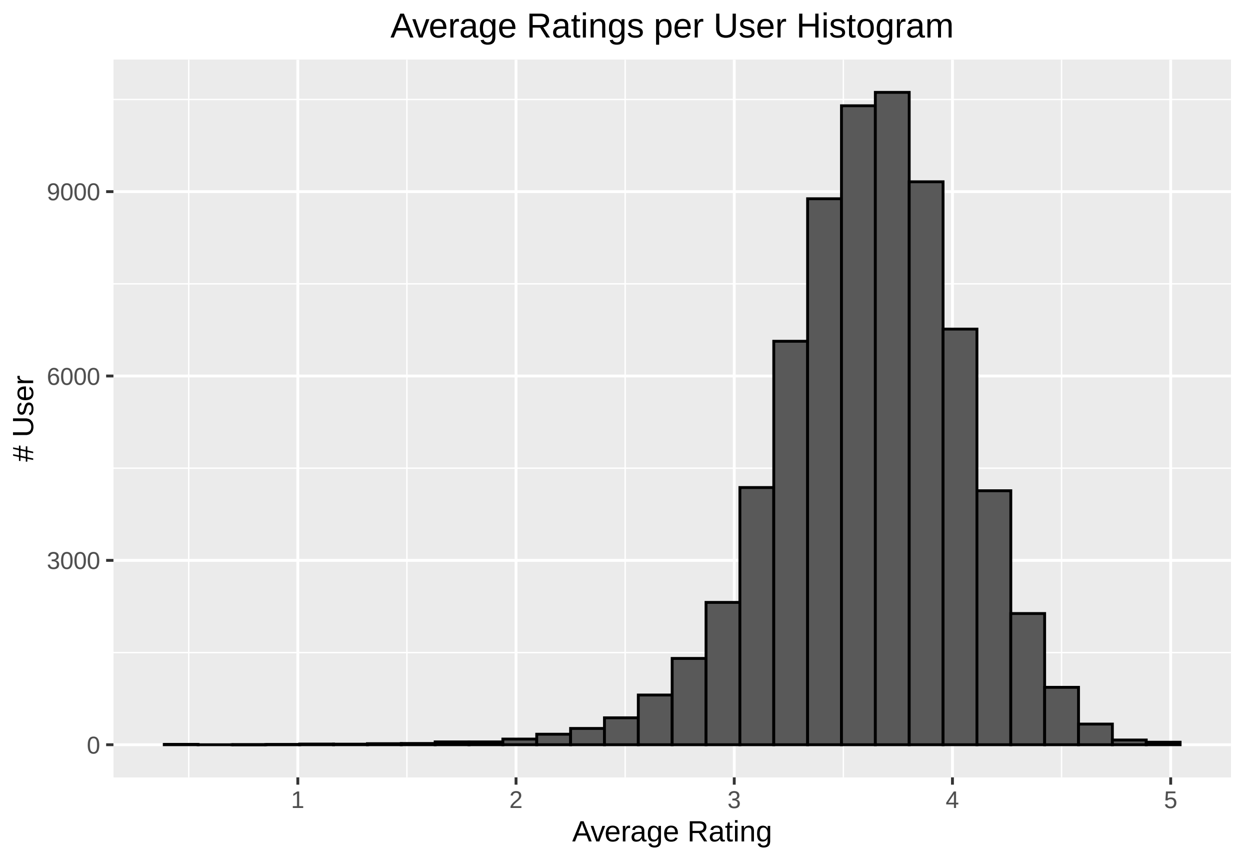
```
# Ratings Users - No. of Ratings on edx data set
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "black",bins=30) +
  scale_x_log10() +
  xlab("# Ratings") +
  ylab("# Users") +
  ggtitle("Users no.of Ratings") +
  theme(plot.title = element_text(hjust = 0.5))
```



A heatmap showing average movie rating against number of movies rated is plotted. The most common movies ratings and number of movies rated are high lighted. This occurs for a rating between 3.5 and 4.0 and between 10 and 100 movies.

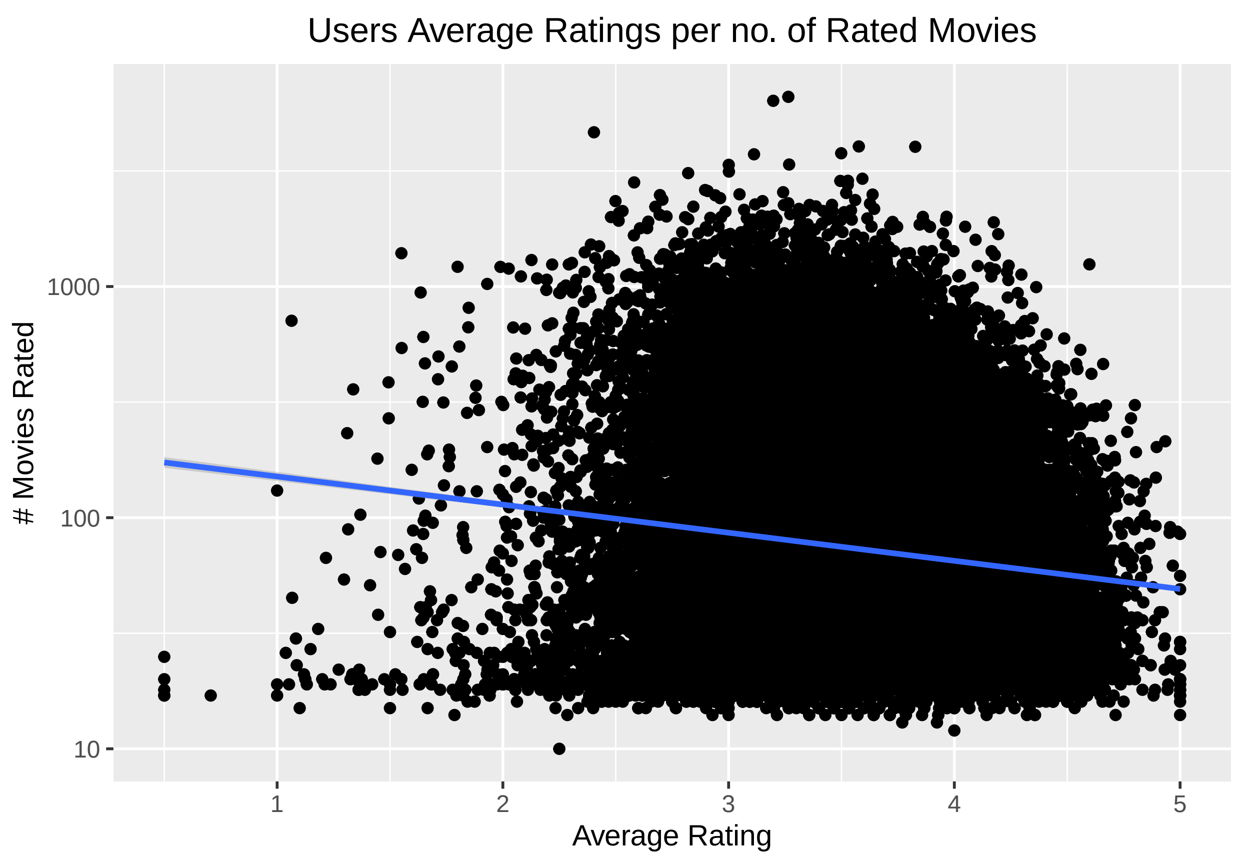
A linear curve was fitted to the data to show the overall trajectory of the ratings with number of movies rated. . The more ratings a user gives results in a lower mean rating.

```
# Ratings Users - Mean on edx data set
edx %>%
  group_by(userId) %>%
  summarise(mu_user = mean(rating)) %>%
  ggplot(aes(mu_user)) +
  geom_histogram(color = "black",bins=30) +
  ggtitle("Average Ratings per User Histogram") +
  xlab("Average Rating") +
  ylab("# User") +
  theme(plot.title = element_text(hjust = 0.5))
```



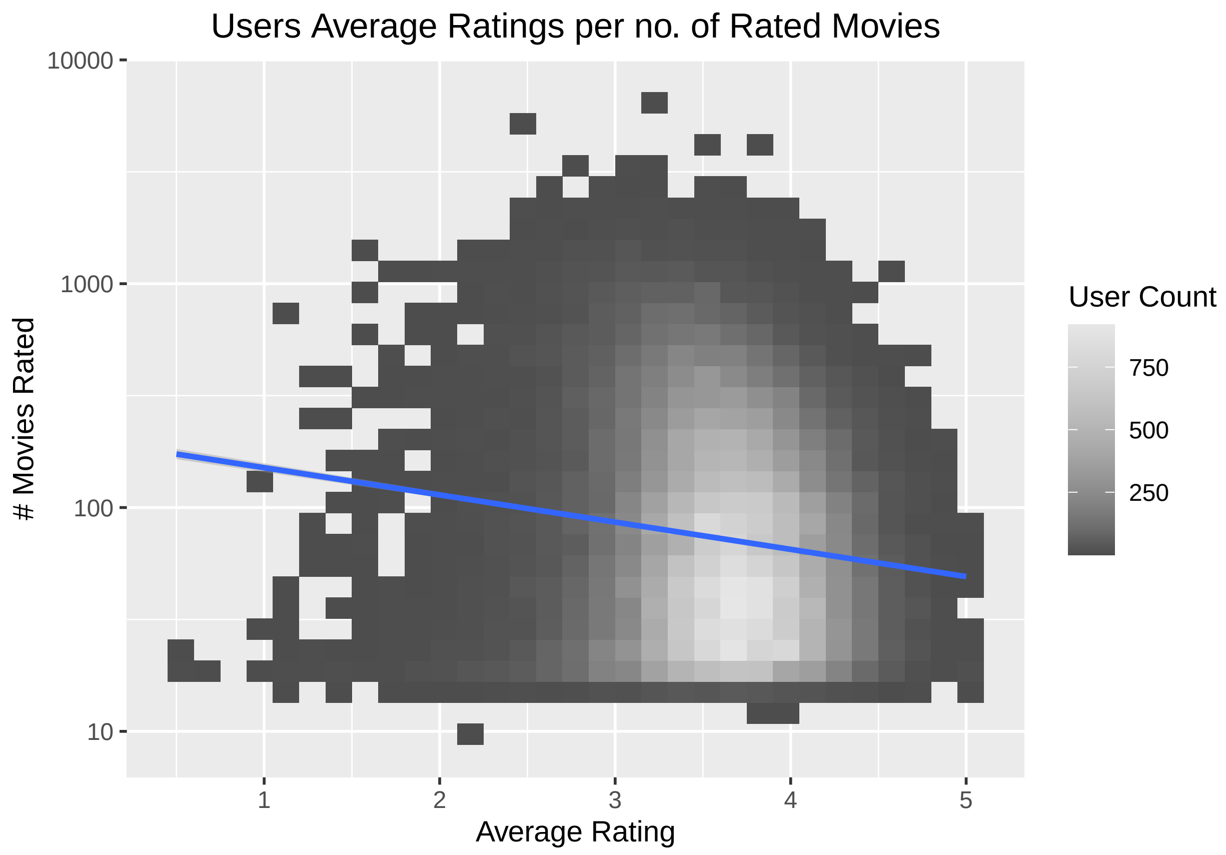
The number of ratings for each movie are shown below in the histogram. A number of outlier movies have been rated less than 10 times which will make predicting future ratings more difficult.

```
# Ratings Users - Mean by Number with Curve Fitted
edx %>%
  group_by(userId) %>%
  summarise(mu_user = mean(rating), number = n()) %>%
  ggplot(aes(x = mu_user, y = number)) +
  geom_point( ) +
  scale_y_log10() +
  geom_smooth(formula= y~x,method = "lm") +
  ggtitle("Users Average Ratings per no. of Rated Movies") +
  xlab("Average Rating") +
  ylab("# Movies Rated") +
  theme(plot.title = element_text(hjust = 0.5))
```



The number of ratings for each movie are shown below in the histogram. A number of outlier movies have been rated less than 10 times which will make predicting future ratings more difficult.

```r
# Ratings Movies - Number of Ratings

edx %>%
  group_by(userId) %>%
  summarise(mu_user = mean(rating), number = n()) %>%
  ggplot(aes(x = mu_user, y = number)) +
  geom_bin2d( ) +
  scale_fill_gradientn(colors = grey.colors(10)) +
  labs(fill="User Count") +
  scale_y_log10() +
  geom_smooth(formula= y~x,method = "lm") +
  ggtitle("Users Average Ratings per no. of Rated Movies") +
  xlab("Average Rating") +
  ylab("# Movies Rated") +
  theme(plot.title = element_text(hjust = 0.5))
```


Users Average Ratings per no. of Rated Movies
# Movies Rated
10000
1000
100
10
1
2
3
4
5
Average Rating
User Count
750
500
250

2 Methods and Analysis

The Residual Mean Square Error (RMSE) is the error function to that will measure accuracy and quantify the typical error we make when predicting the movie rating. An error larger than 0.8775, it means our typical error is larger than the required for this assignment almost a star, which is not good. RMSE defined as below.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

where; N is the number of users, movie ratings, and the sum incorporating the total combinations.

2.1 Naive Baseline Model :

This simple prediction model uses the mean of the data set to predict the rating for all movies. The model assumes that all differences are due to a random error;

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, $\epsilon_{u,i}$ is the independent error, and μ the expected "true" rating for all movies. Predicting the mean gives the following naive RMSE.

```
# RMSE : The Root Mean Square Error
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

## Simple Prediction based on Mean Rating
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

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

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

```
## display output in Data Frame
rmse_output = data.frame(method = "Naive Baseline Mean Model", RMSE = rmse_naive)
rmse_output %>% knitr::kable()
```

method	RMSE
Naive Baseline Mean Model	1.061202

Investigating the data set allows for more advanced analysis and rating predictions with smaller error.

Investigating the data set allows for more advanced analysis and rating predictions with smaller error.

2.2 Movie Effects Model:

The Movie Effects Model calculates a bias term for each movie based on the difference between the movies mean rating and the overall mean rating.

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, $\epsilon_{u,i}$ is the independent error, and μ the mean rating for all movies, and b_i is the bias for each movie i.

```r
## Simple model taking into consideration the movie effects, b_i

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

```
## [1] 3.512465
```

```r
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

rmse_model_movie_effects <- RMSE(predicted_ratings, validation$rating)
rmse_model_movie_effects
```

```
## [1] 0.9439087
```

```r
rmse_output <- bind_rows(rmse_output,
                         data.frame(method="Movie Effects Model",
                                    RMSE = rmse_model_movie_effects))
```

The Movie Effect Model: predicting the movie rating with both bias, b_i, and mean, μ gives an improved prediction with a lower RMSE value.

The Movie Effect Model: predicting the movie rating with both bias, b_i, and mean, μ gives an improved prediction with a lower RMSE value.

2.3 Movie and User Effects Model:

The next step is to incorporate the individual User Effects, b_u, in to the model. Acknowledging each user inherent bias to mark all films higher or lower.

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, $\epsilon_{u,i}$ is the independent error, and μ the mean rating for all movies, b_i is the bias for each movie i, and b_u is the bias for each user u.

```
## Movie and User Effects Model
## Simple model taking into account the user effects, b_u
user_avgs <- edx %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

rmse_model_user_effects <- RMSE(predicted_ratings, validation$rating)
rmse_model_user_effects
```

```
## [1] 0.8653488
```

```
rmse_output <- bind_rows(rmse_output,data.frame(method="User Effects Model",RMSE = rmse_model_user_effe
rmse_output %>% knitr::kable()
```

method	RMSE
Naive Baseline Mean Model	1.0612018
Movie Effects Model	0.9439087
User Effects Model	0.8653488

Incorporating the user bias into the model resulted in a further reduced RMSE.

2.4 Regularization :

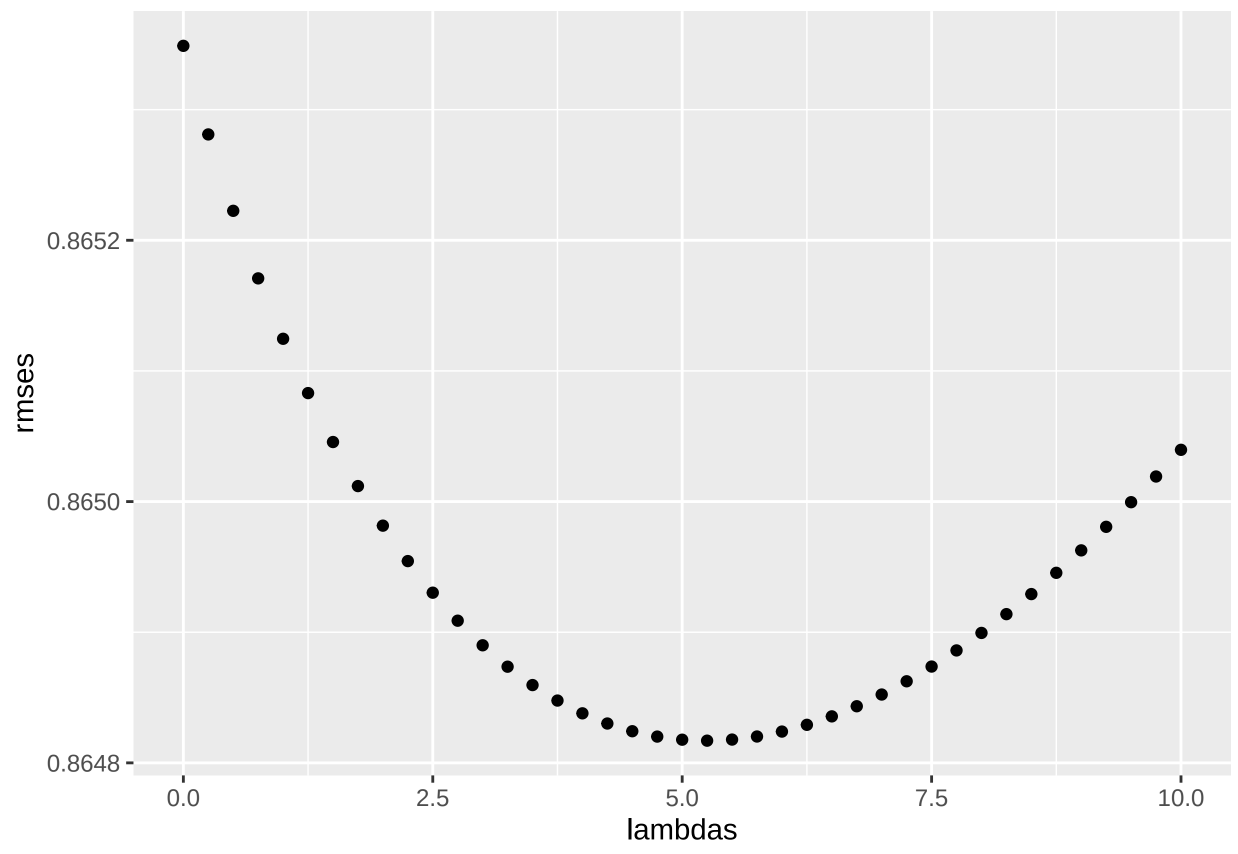
Regularization allows for reduced errors caused by movies with few ratings which can influence the prediction and skew the error metric. The method uses a tuning parameter, λ, to minimise the RMSE. Modifying b_i and b_u for movies with limited ratings.

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

```
# Predict via Regularization Model, movie and user effect model
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(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) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))

})
rmse_reg <- min(rmses)
rmse_reg
```

```
## [1] 0.864817
```

```
# Plot RMSE against Lambdas to find optimal lambda
qplot(lambdas, rmses)
```



```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```r
rmse_output <- bind_rows(rmse_output,
                         data.frame(method="Regularization Model",
                                    RMSE = rmse_model_user_effects))
rmse_output %>% knitr::kable()
```

method	RMSE
Naive Baseline Mean Model	1.0612018
Movie Effects Model	0.9439087
User Effects Model	0.8653488
Regularization Model	0.8653488

3 Results and Discussion :

The final output of the prediction models are shown below;

```
rmse_output %>% knitr::kable()
```

method	RMSE
Naive Baseline Mean Model	1.0612018
Movie Effects Model	0.9439087
User Effects Model	0.8653488
Regularization Model	0.8653488

The models from most accurate to least accurate are as follows; Regularised Movie and User Effects Model; Movie and User Effects Model; Movie Effects Model; and Simple Average Model.

The final model optimised for the prediction is the following;

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

The lowest value of RMSE predicted is 0.8648170.

4 Conclusion :

A machine learning algorithm to predict the ratings from the Movie Lens data set was constructed. The optimal model incorporated the effects of both user and movie bias were incorporated in to the model, and these values were regularized to incorporate movies will a low number of ratings.

The aim of the project was to develop an algorithm lower than 0.87750, which was achieved by the Movie and User Effects and Regularized Movie and User Effects model.

5 Appendix :

5.1 Environment

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          1.2
## year           2021
## month          11
## day            01
## svn rev        81115
## language       R
## version.string R version 4.1.2 (2021-11-01)
## nickname       Bird Hippie
```