HarvardX PH125.9 - Data Science: Capstone Project - Movielens

Reema Sajnani

2024-05-11

Contents

1 EXECUTIVE SUMMARY

This project aims to produce a movie recommendation system, based on the 10M version of the *Movielens* data set from the *dslabs* package. The data set contains ratings of movies given by individuals over time. Using the code provided in the **HarvardX PH125.9 - Data Science: Capstone** course as a starting point, the data set is split into the **edx** partition (used to create the prediction algorithm) and the **final_holdout_test** partition (used to the evaluate the aforementioned algorithm results). We first explore the data and analyze each input to notice some trends and patterns. We then use the inputs to evaluate their influence on our predictions. We then put our prediction models to test and choose the one that minimizes the RMSE (Root Mean Squared Error).

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

The best resulting code achieved in the project delivers an RMSE of **0.8641**, which was validated against a target of **0.8649**.

2 STARTING CODE

2.1 Load Required Packages & Libraries

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

## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.3.3

##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

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

library(tidyverse)

library(caret)

library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.3.3
```

```r
#MovieLens 10M dataset:

#https://grouplens.org/datasets/movielens/10m/

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

```r
options(timeout = 120)

dl <- "ml-10M100K.zip"

if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

```r
# Final hold-out test set will be 10% of MovieLens data

set.seed(123, sample.kind="Rounding")
```

```
## Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
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 final hold-out test set are also in edx set

final_holdout_test <- temp %>% semi_join(edx, by = "movieId") %>% semi_join(edx, by = "userId")
```

```r
# Add rows removed from final hold-out test set back into edx set

removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

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

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

3 DATA EXPLORATION & VISUALIZATION

3.1 Data Class

After the data is split into the training set, named **edx**, the first step is to understand the data and each of the 6 vectors.

- **userId** `<integer>` unique identifier per person
- **movieId** `<numeric>` unique identifier per movie
- **rating** `<numeric>` a rating on the scale of 0 to 5 per movie per user
- **timestamp** `<integer>` exact time of rating
- **title** `<character>` name and year of movie release
- **genres** `<character>` genre tags for each movie

Each of the nearly 9 million rows captures a rating given by one of the 69,878 unique users to one of the 10,677 unique movies in the library.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                          genres
## 1                Comedy|Romance
## 3                        Comedy
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
## 8      Comedy|Drama|Romance|War
```

```
##   n_users n_movies
## 1   69878    10677
```

3.2 Missing Values

A summary view of the data set also confirms no missing values.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18127   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35750   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35873   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53609   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000062     Length:9000062
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

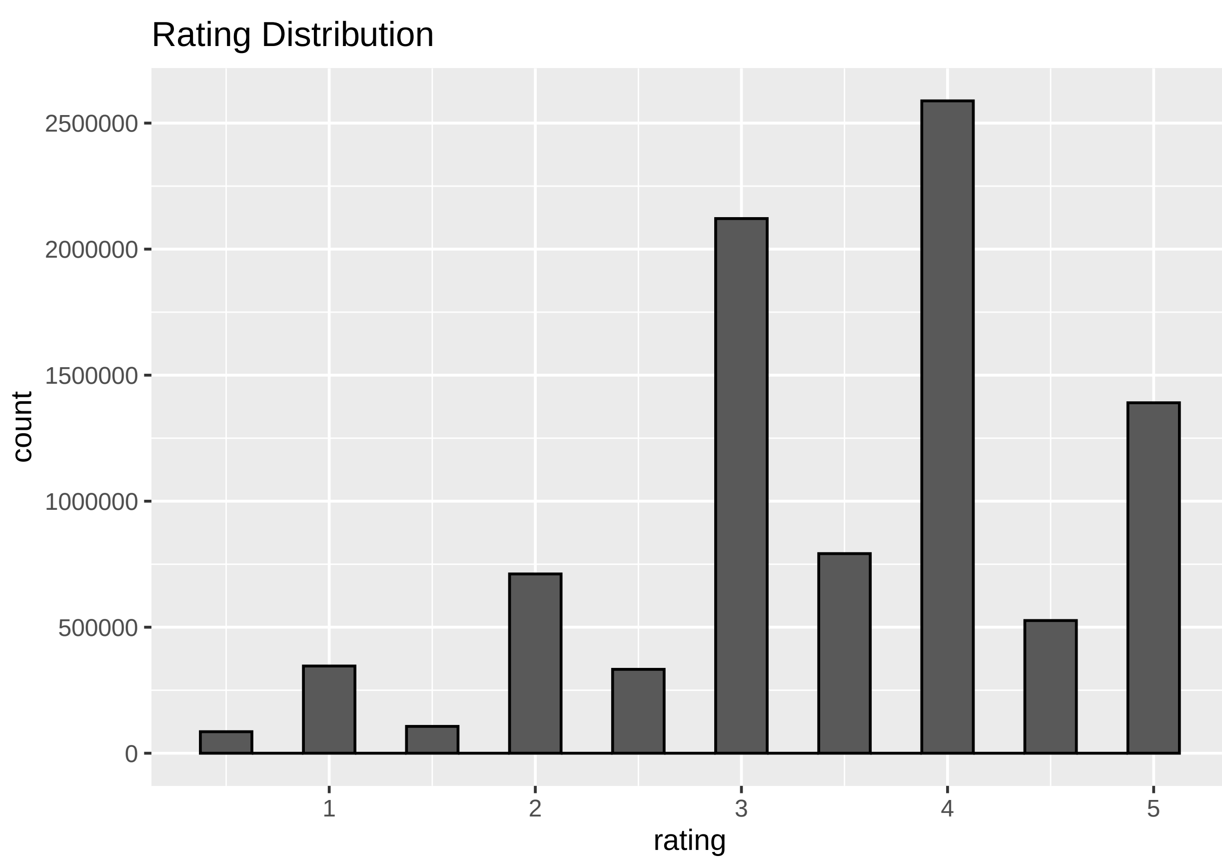
3.3 Movies' Ratings

The most reviewed movies seem to be popular movie names from the 90s, hinting that re-known films are probably likely to be rated more often than lesser heard of movies.

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

## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                      count
##      <int> <chr>                                                      <int>
##  1     296 Pulp Fiction (1994)                                        31471
##  2     356 Forrest Gump (1994)                                        30948
##  3     593 Silence of the Lambs, The (1991)                           30336
##  4     480 Jurassic Park (1993)                                       29332
##  5     318 Shawshank Redemption, The (1994)                           27982
##  6     110 Braveheart (1995)                                          26140
##  7     457 Fugitive, The (1993)                                       26105
##  8     589 Terminator 2: Judgment Day (1991)                          26079
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25715
## 10     150 Apollo 13 (1995)                                           24347
## # i 10,667 more rows
```

The data also shows that lower ratings are less common and so are half star ratings. The most common ratings are 4,3 and 5 respectively



On closer examination, we will also notice that some movies are not rated as often as others, in fact, 126 movies are rated only once. More so, the least frequently rated movies titles and their average ratings appear to be obscure. This calls for regularization and a penalty to be applied in the calculation of the model.

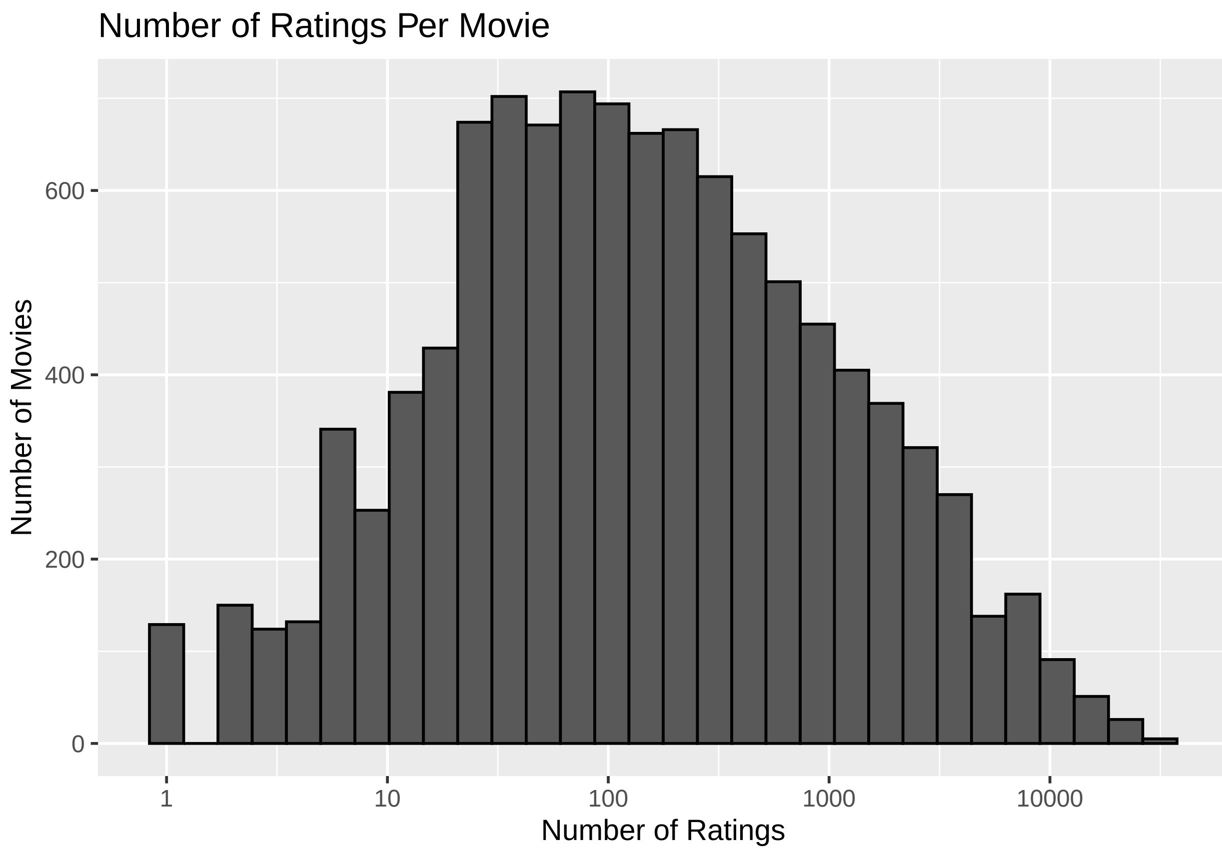
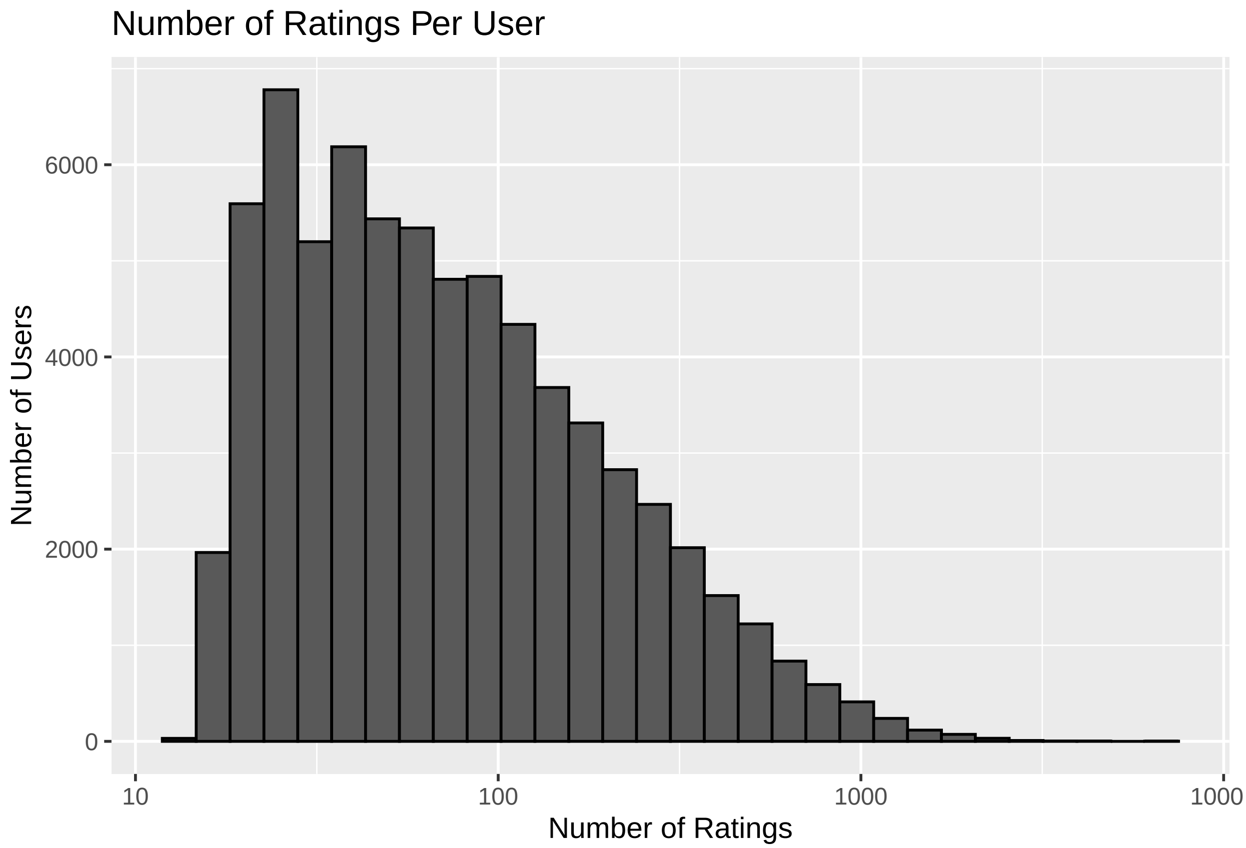


Table 1: Least Frequently Rated Movies

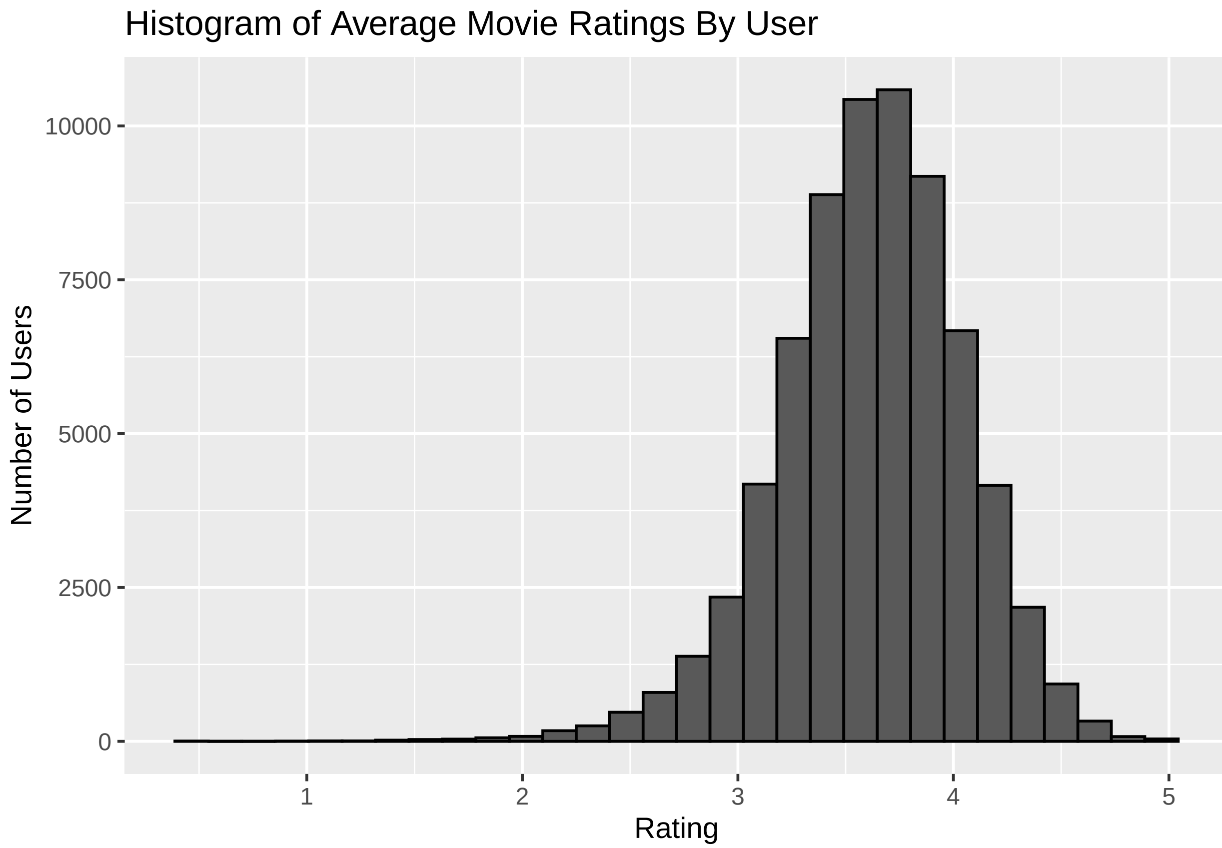
title	count	rating
100 Feet (2008)	1	2.0
4 (2005)	1	2.5
Accused (Anklaget) (2005)	1	0.5
Ace of Hearts (2008)	1	2.0
Ace of Hearts, The (1921)	1	3.5
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di…) (1971)	1	1.5
Africa addio (1966)	1	3.0
Bad Blood (Mauvais sang) (1986)	1	4.5
Battle of Russia, The (Why We Fight, 5) (1943)	1	3.5
Besotted (2001)	1	0.5

3.4 Users' Ratings

The graph showing how often users rate movies is skewed to the right, implying that some users rarely rate movies, while some others rate thousands of movies.



We can also infer that some users are more critical in their ratings and some of them might have not rated too many movies. This phenomenon, as well, calls for a penalty to be applied in the calculation of the model.



3.5 Genre Ratings

In the data set, we have 797 unique combinations of genres. When looking into the highest rated genre combinations, we see that most of the top rated ones are also the more frequently rated ones. Genre combinations with lower average ratings seem to be rated less often.

Table 2: Genres that Have Received the Highest Average Ratings

genres	count	rating
Animation|IMAX|Sci-Fi	7	4.714286
Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi	3	4.333333
Drama|Film-Noir|Romance	3040	4.313651
Action|Crime|Drama|IMAX	2358	4.292197
Animation|Children|Comedy|Crime	7122	4.279837
Film-Noir|Mystery	6014	4.235367
Crime|Film-Noir|Mystery	4010	4.227681
Film-Noir|Romance|Thriller	2431	4.218840
Crime|Film-Noir|Thriller	4860	4.207407
Crime|Mystery|Thriller	26945	4.199146

Table 3: Genres that Have Received the Lowest Average Ratings

<table>
<tr><th>genres</th><th>count</th><th>rating</th></tr>
<tr><td>Documentary|Horror</td><td>638</td><td>1.464733</td></tr>
<tr><td>Action|Animation|Comedy|Horror</td><td>2</td><td>1.500000</td></tr>
<tr><td>Action|Horror|Mystery|Thriller</td><td>320</td><td>1.581250</td></tr>
<tr><td>Action|Drama|Horror|Sci-Fi</td><td>5</td><td>1.600000</td></tr>
<tr><td>Comedy|Film-Noir|Thriller</td><td>24</td><td>1.729167</td></tr>
<tr><td>Adventure|Drama|Horror|Sci-Fi|Thriller</td><td>220</td><td>1.777273</td></tr>
<tr><td>Action|Horror|Mystery|Sci-Fi</td><td>19</td><td>1.868421</td></tr>
<tr><td>Action|Adventure|Drama|Fantasy|Sci-Fi</td><td>58</td><td>1.879310</td></tr>
<tr><td>Action|Children|Comedy</td><td>524</td><td>1.887405</td></tr>
<tr><td>Children|Fantasy|Sci-Fi</td><td>57</td><td>1.903509</td></tr>
</table>

4 DATA MODELLING

The model relies on the statistical concept of RMSE (Root Mean Squared Error) and Regularization.

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

RMSE represents the error made while predicting a movie rating. Our goal is to model an algorithm such that RMSE is minimized as much as possible, while being under the goal of **0.8649**.

4.1 Basic Model

This is the simplest approach that predicts the same rating for all movies and is based on the average of ratings in the training data set.

The formula used for this model is:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

where $\hat{\mu}$ is the mean of all ratings (3.5125) and $\varepsilon_{i,u}$ is the random errors sampled from the same 0 centered distribution.

```
##         Model     RMSE
## 1 Basic Model 1.060057
```

The RMSE on the `final_holdout_test` data set is **1.06**. It is quite distant from the target of 0.8649 and indicates that this model doesn't produce ideal results.

4.2 Movie Effect Model

From exploration of the *edx* data set previously, we understood that some movies are not rated as often. We also noticed some obscurities in the least frequently rated movies.

The formula used for this model is:

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

where $\hat{\mu}$ is the mean of all ratings (3.5125) and $\varepsilon_{i,u}$ is the random errors sampled from the same 0 centered distribution. The b_i is a measure of the degree of popularity bias of each movie i.

```
##                Model      RMSE
## 1        Basic Model 1.0600565
## 2 Movie Effect Model 0.9431062
```

We used the impact of the movie ratings to calculate the RMSE of 0.9431, which is still much higher than the target of 0.8649

4.3 Movie & User Effect Model

In addition to the movie effect, we had also noticed an impact from users. Some users were rating less frequently and were also inclined to give movies lower ratings.

The formula used for this model is:

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

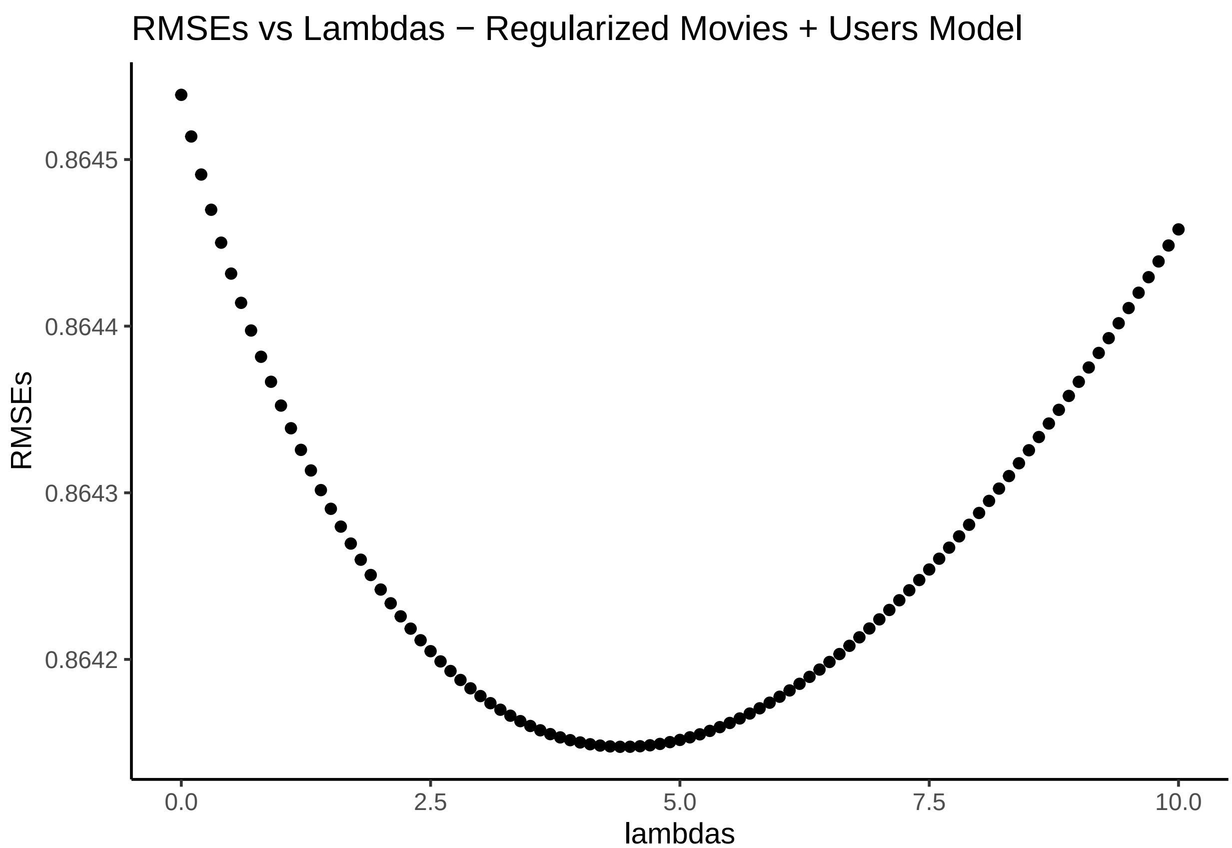
where $\hat{\mu}$ is the mean of all ratings (3.5125) and $\varepsilon_{i,u}$ is the random errors sampled from the same 0 centered distribution. The b_i is a measure of the degree of popularity bias of each movie i. The b_u is a measure of the degree of user bias of each movie u.

```
##                          Model      RMSE
## 1                  Basic Model 1.0600565
## 2           Movie Effect Model 0.9431062
## 3 Movie + User Effect Model 0.8645388
```

We used the impact of the movie and user ratings to calculate the RMSE of 0.8645, which is lower than the target of 0.8649, hence proving that this model is quite good. However, there might be room to better here.

4.4 Regularized Movie & User Effect Model

From the Movie and User Effects Models we have seen the influence these inputs have on our predictions. Therefore it becomes necessary to tune our model, calling for the use of Regularization. Essentially, regularization involves incorporating a penalty for high values of b_i and b_u into the sum of squares equation that we aim to minimize. The penalization or tuning parameter is called *Lambda*



The optimal Lambda turns out to be 5.25. At this value of Lambda, RMSE is minimized to **0.8641** which is even lower than our last model.

```
##                                   Model      RMSE
## 1                           Basic Model 1.0600565
## 2                    Movie Effect Model 0.9431062
## 3             Movie + User Effect Model 0.8645388
## 4 Regularized Movie + User Based Model 0.8641476
```

5 RESULTS & CONCLUSION

The RMSE values of all the represented models are the following:

```
##                                  Model      RMSE
## 1                          Basic Model 1.0600565
## 2                   Movie Effect Model 0.9431062
## 3            Movie + User Effect Model 0.8645388
## 4 Regularized Movie + User Based Model 0.8641476
```

We developed a machine learning algorithm utilizing the MovieLens data set to forecast movie ratings. The refined model, incorporating movie and user effects, demonstrates a notable decrease in RMSE, making it the preferred choice for our project. This optimal model exhibits an RMSE value of **0.8641**, surpassing the initial evaluation benchmark of 0.8649 set by our project's objectives.

6 LIMITATIONS

1. RMSE could be enhanced by integrating other factors such as genre, release year etc however, we limit the scope of the project since the goal has been achieved.
2. Due to lack of access to a professional laptop and tools, automated data wrangling formulas could not be used and RMSE was instead calculated manually.