MovieLens Capstone Project

Shilpa Susan Thomas

29 April 2020

Contents

1 Introduction

This project is a requirement for the course HarvardX: PH125.9x Data Science: Capstone. The purpose of this project is to provide a movie recommendation system using the MovieLens dataset. The 10M version of the MovieLens dataset will be used to make the computation easier.

A machine learning algorithm will be developed using a subset of this dataset. To test the final version of the algorithm, it will be used to predict user ratings for movies in the validation set as if they were unknown.

The test value that will be used to evaluate the best machine learning algorithm is the Root Mean Square Error (RMSE). RMSE measures how much error there is between two data sets. In other words, it compares a predicted value and an observed or known value. The smaller an RMSE value, the closer predicted and observed values are. So, the aim of this project is to develop a machine learning algorithm that will predict user ratings for movies and have the lowest RMSE value as possible.

The key steps that are performed included data cleaning and dividing the data into a training subset and validation set, data exploration and visualisation and find the best possible model that can be used to predict movie ratings.

2 Data Preparation

The 10M version of the MovieLens dataset was compiled by GroupLens. It is downloaded and split into a training set called edx and a validation set which was 10% of the the MovieLens data. In order to do this project, a few R packages are installed and loaded as well.

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

# 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 <- 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)
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)
```

3 Data Exploration and Analysis

The training and validation datasets have been defined. Now, we can look at the basic structure of the data and calculate some statistics.

```r
# Data Exploration

#the training set
# intial 7 rows with header
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```r
# basic summary statistics
nrow(edx)
```

```
## [1] 9000055
```

```r
ncol(edx)
```

```
## [1] 6
```

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

We can see that the edx (training set) data has 9000055 rows and 6 columns which represent the variables. These are userID, movieID, rating, timestamp, title and genres. Each row represents one rating by one user for a single movie.

```r
edx %>% summarize(n_movies = n_distinct(movieId))
```

```
##   n_movies
## 1    10677
```

```r
edx %>% summarize(n_users = n_distinct(userId))
```

```
##   n_users
## 1   69878
```

```r
drama <- edx %>% filter(str_detect(genres,"Drama"))
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
romance <- edx %>% filter(str_detect(genres,"Romance"))
nrow(drama)
```

```
## [1] 3910127
```

```r
nrow(comedy)
```

```
## [1] 3540930
```

```r
nrow(thriller)
```

```
## [1] 2325899
```

```r
nrow(romance)
```

```
## [1] 1712100
```

```r
edx %>% group_by(title) %>% summarise(number = n()) %>%
  arrange(desc(number))
```

```
## # A tibble: 10,676 x 2
##    title                                                        number
##    <chr>                                                         <int>
##  1 Pulp Fiction (1994)                                           31362
##  2 Forrest Gump (1994)                                           31079
##  3 Silence of the Lambs, The (1991)                              30382
##  4 Jurassic Park (1993)                                          29360
##  5 Shawshank Redemption, The (1994)                              28015
##  6 Braveheart (1995)                                             26212
##  7 Fugitive, The (1993)                                          25998
##  8 Terminator 2: Judgment Day (1991)                             25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)  25672
## 10 Apollo 13 (1995)                                              24284
## # ... with 10,666 more rows
```

There are 10677 different movies for user ratings that have been given and 69878 distinct users. There are 3910127 dramas, 3540930 comedys, 2325899 thrillers and 1712100 romantic movies. There are other genres as well. The movie Pulp Fiction has the greatest number of ratings.

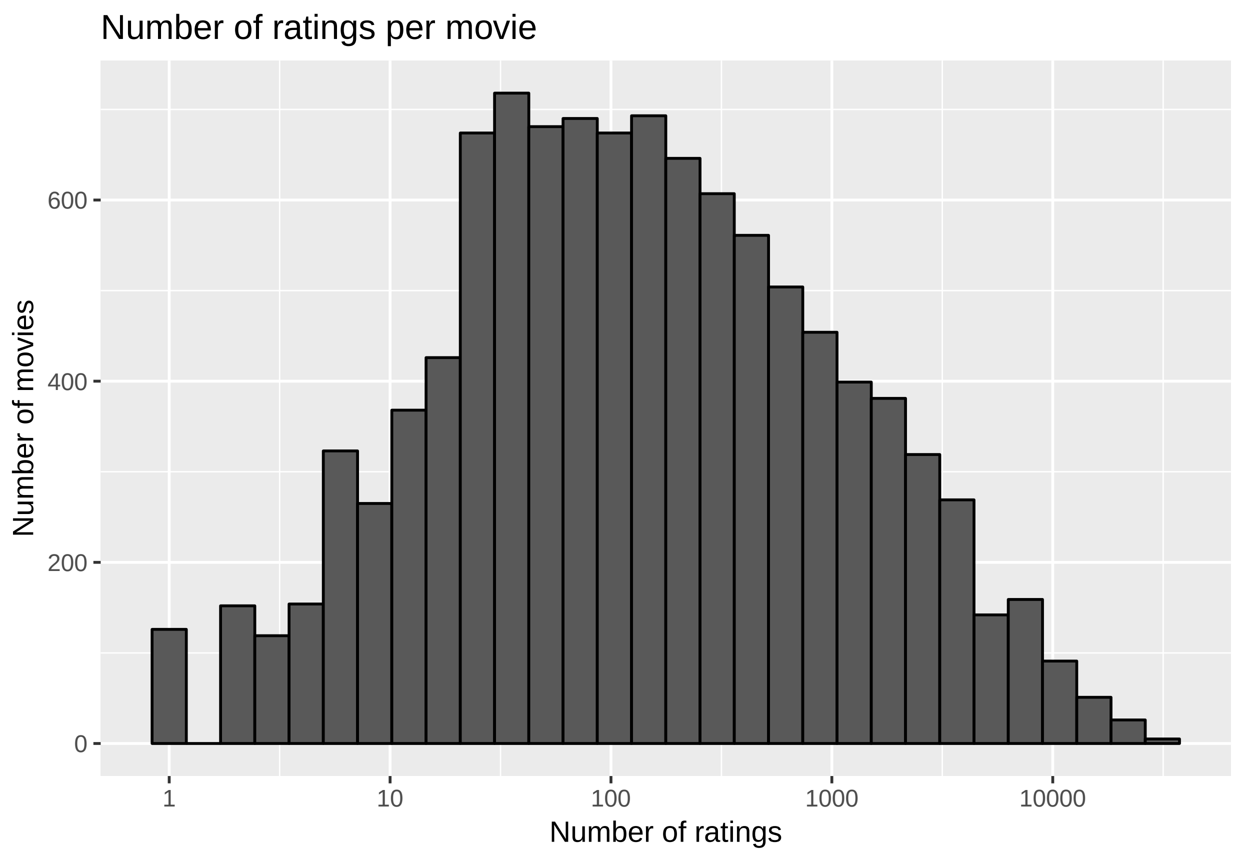
```r
#modify datasets to make the year of the movie as a separate column
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
#check if year has been added correctly
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                          genres year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4       Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6       Children|Comedy|Fantasy 1994
```

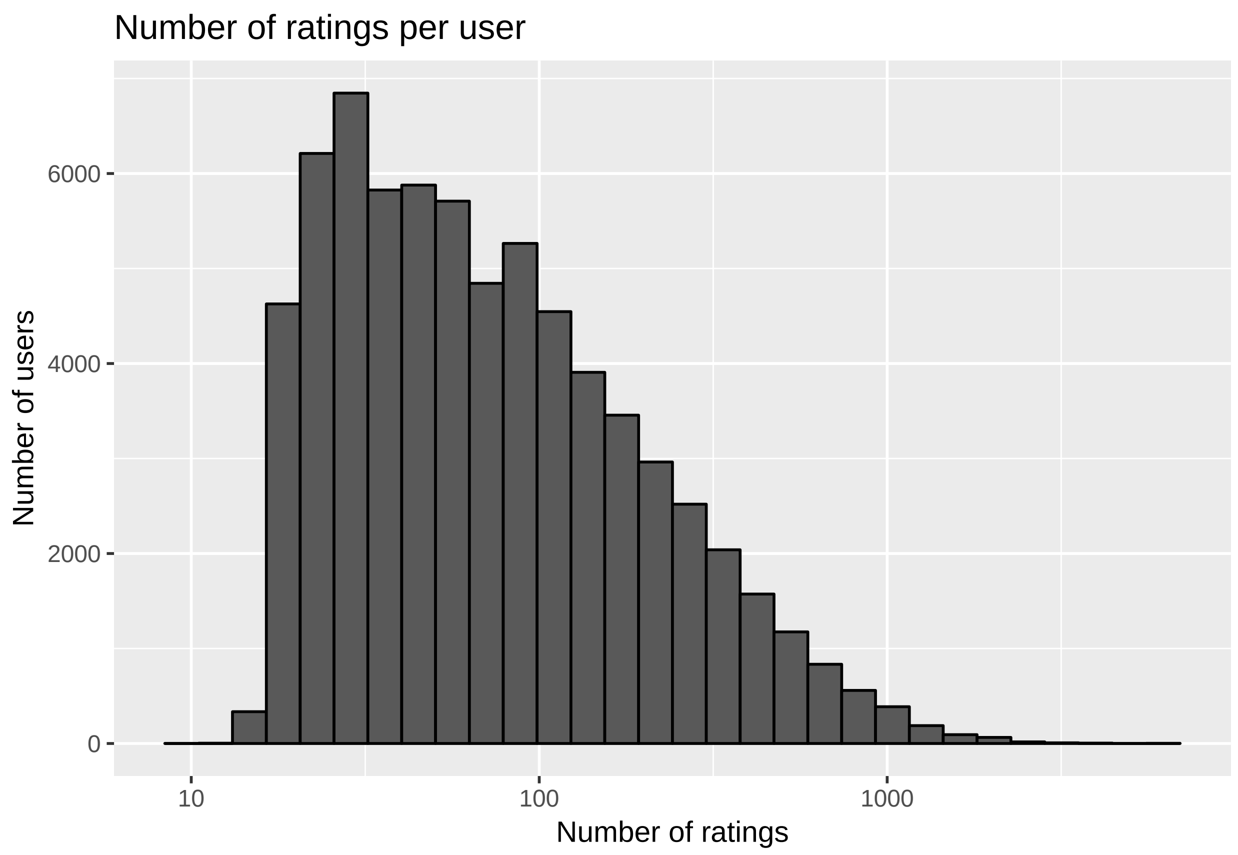
The year that each movie was released has been extracted to make a new column so that its effect could be analysed further.

3.1 Data Visualisations

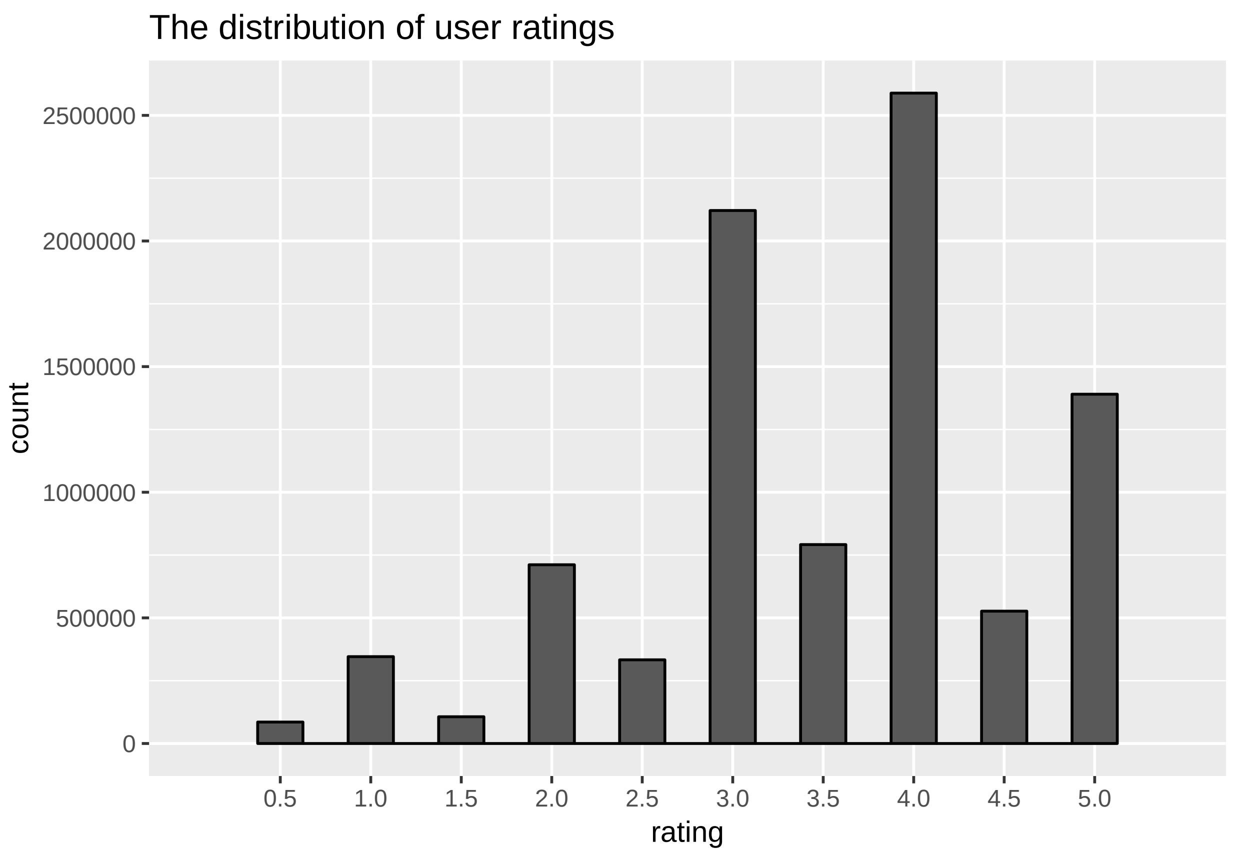
The following plot shows that some movies are rated more than others. This shows that some movies are more popular than others. Some movies have been given very few ratings or even one. So we will have to include a movie bias in our final model.



The following plot shows that some users give more ratings than others. So the more frequent users' ratings may have a bias when predicting ratings. We will have to take this into account when the model is built.

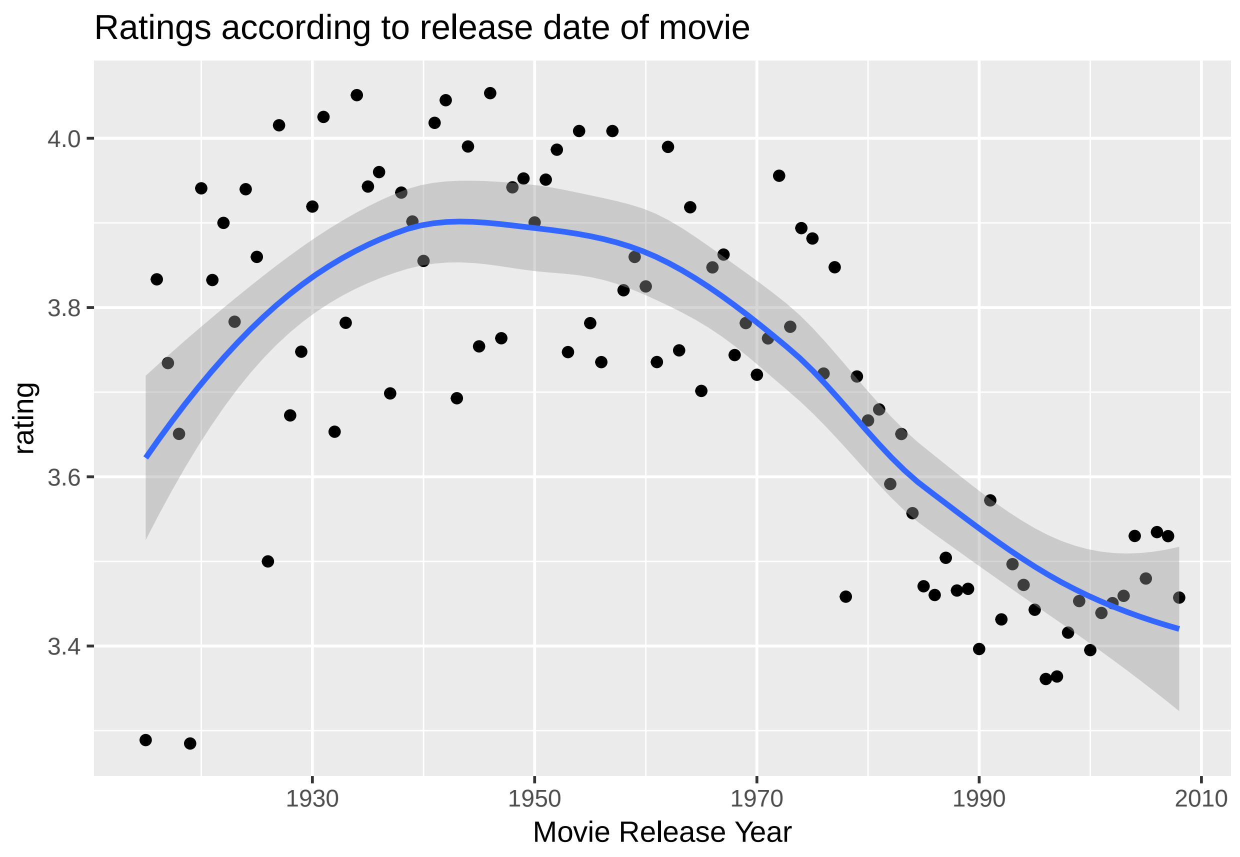


The following plot shows that users tend to give a 3.0 and 4.0 rating for a movie more than lower ratings. In general, half ratings like 2.5 and 3.5 are less common than whole ratings.



We can see from the following plot that the average rating given by users do change over time. More specifically, users give lower ratings for movies released in the recent years than those released before 1960. This could also reflect the personalities of users. Some may prefer to watch old movies and give more positive ratings than those who watch new movies.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



4 Modelling Process

4.1 RMSE

The Root Mean Squared Error (RMSE) can be defined by the following formula:

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

with N being the number of user/movie combinations and the sum occurring over all these combinations.

The written function in R to compute the RMSE for the user ratings and their predictions is:

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

As said before, the aim is to find a model that will have the lowest possible RMSE.

4.2 Partitioning of the edx dataset

The edx dataset is further partitioned into separate training and test sets to design and test the machine learning algorithm. This will allow the validation dataset to be only used to test the final model.

```r
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

4.2.1 Average Model

This model does not take any other features into consideration. Only the average of the user ratings are calculated. So, the model will only consist of a μ term and any random differences represented by a ϵ term.

```r
#Average model
mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating, mu)
rmse_results <- data_frame(method = "Just the Average Model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
rmse_results %>% knitr::kable() %>% kable_styling(position = 'center')%>%
  column_spec(1,border_left = T)%>%column_spec(2,border_right = T)%>%
  row_spec(0,bold=T)
```

method	**RMSE**
Just the Average Model	1.060054

We will be building the results table with the different RMSE's for each of the models we develop. This first RMSE will be used as a guideline to see how much we can improve it further using other features.

4.2.2 Movie Effect Model

From the exploratory analysis, we saw that some movies have more ratings than others as well as higher ratings. So, we will introduce a term b_i which is the bias for each movie i.

```r
#Movie Effect Model
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings_movie <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings_movie, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                    RMSE = model_1_rmse ))
rmse_results %>% knitr::kable() %>% kable_styling(position = 'center')%>%
  column_spec(1,border_left = T)%>%column_spec(2,border_right = T)%>%
  row_spec(0,bold=T)
```

method	**RMSE**
Just the Average Model	1.0600537
Movie Effect Model	0.9429615

By just adding the movie bias, we can see a significant decrease in the value of the RMSE.

4.2.3 Movie + User Effects Model

We also saw that ratings are affected by the users. One user may give a positive review for one movie but another user will give the same one a negative review. So we will introduce a further term b_u which is the bias for each user u.

```r
#Movie + User Effect Model
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings_user <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_ratings_user, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))

rmse_results %>% knitr::kable() %>% kable_styling(position = 'center')%>%
  column_spec(1,border_left = T)%>%column_spec(2,border_right = T)%>%
  row_spec(0,bold=T)
```

method	**RMSE**
Just the Average Model	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646844

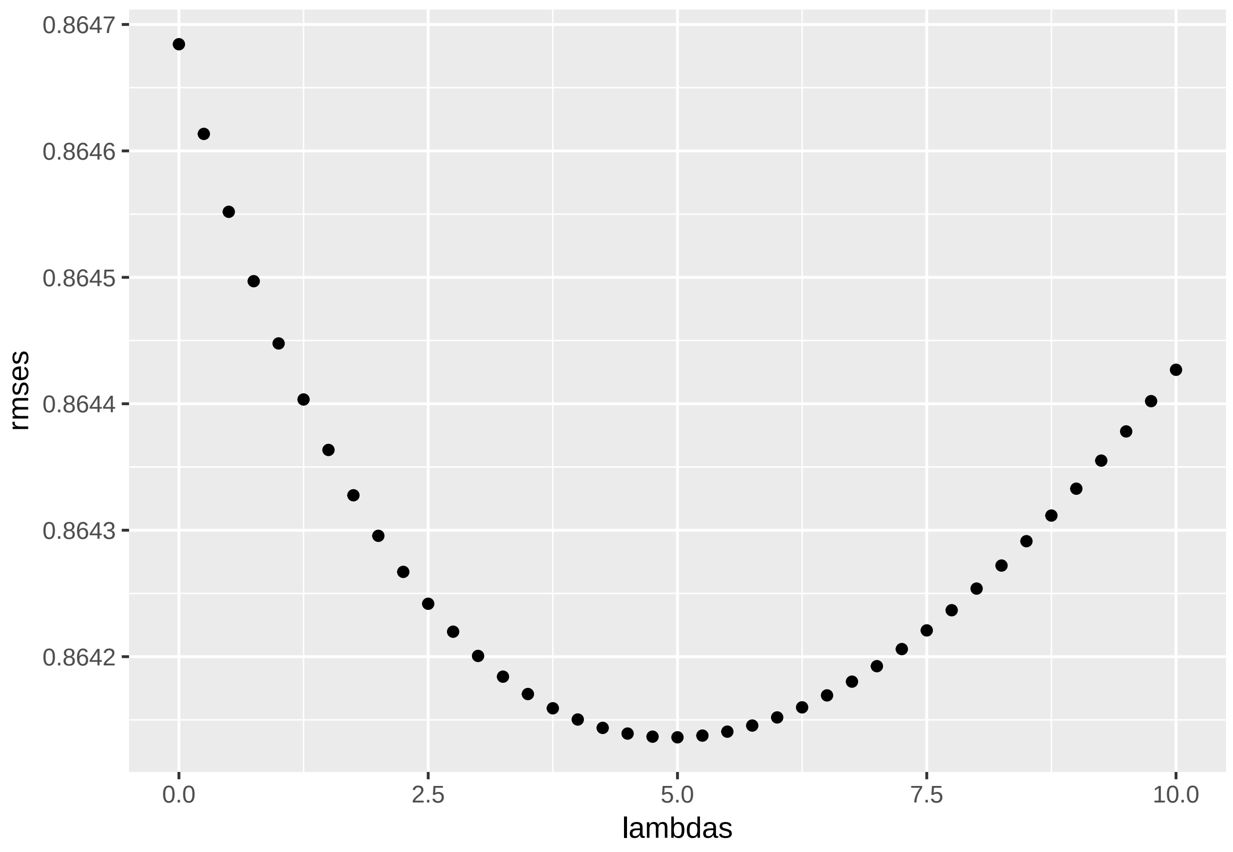
We can see that the RMSE has reduced further.

4.2.4 Regularised Movie + User Effects Model

In the data exploration section, we saw that some users have given much more reviews than others. This can cause a greater effect in the prediction. In this model, we introduce a tuning parameter called lambda. We create a function to find the optimal lambda that will result in the lowest RMSE.

```r
#Regularised Movie + User Effect Model

lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)

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

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

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
```

From the following plot, we can see which lambda is the best to get the minimum RMSE.



In this model, the value of lambda is:

```
## [1] 5
```

Taking this value of lambda, we can now build the table with the RMSE values of our models.

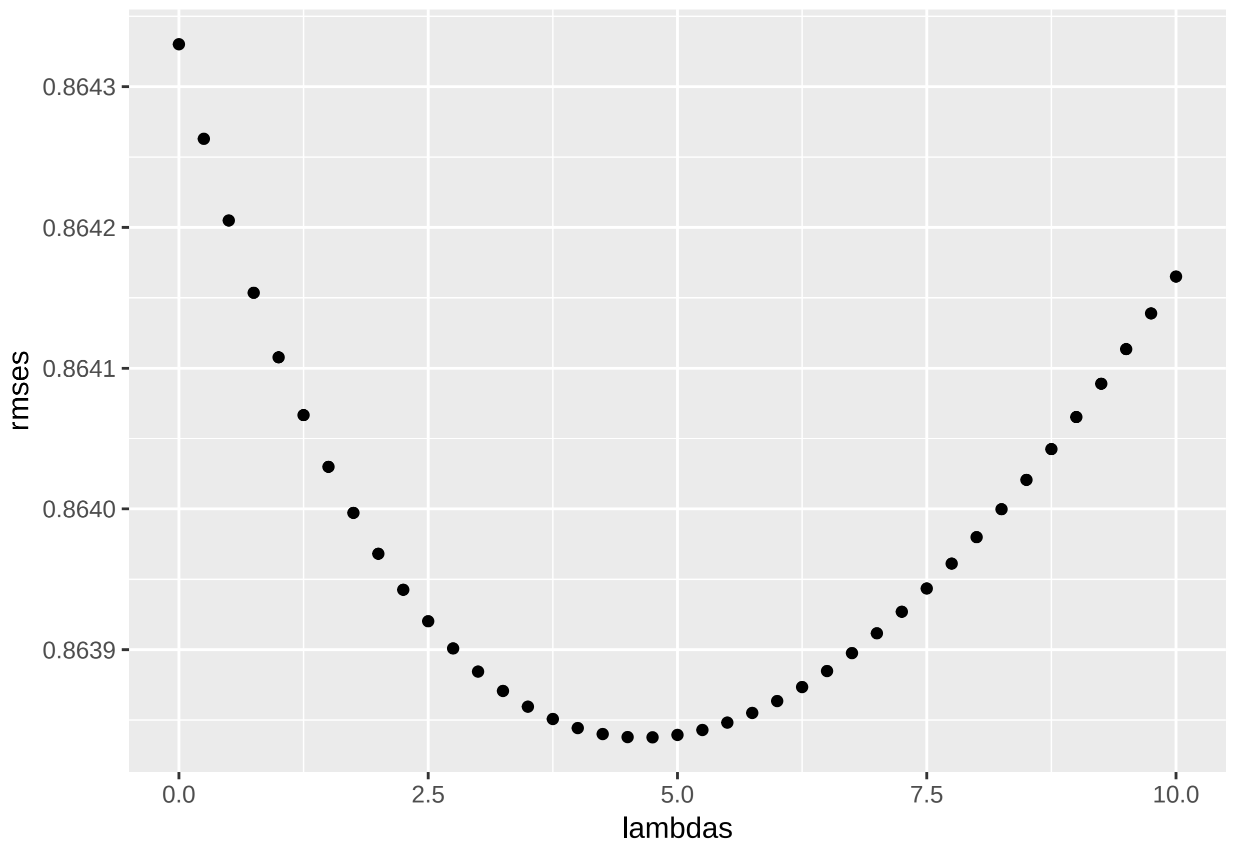
method	RMSE
Just the Average Model	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646844
Regularised Movie + User Effect Model	0.8641362

4.2.5 Regularised Movie + User + Year Effects Model

We had also looked at the effect of the year when the movie was released. We had seen that movies released earlier had more positive ratings than newer movies. So, in this model, we introduce another term to represent this bias, b_y. Like in the previous model, we again find an optimal lambda.

```r
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)

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

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

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

  predicted_ratings <- test_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    .$pred

  return(RMSE(predicted_ratings, test_set$rating))
})
```

From the following plot, we can see which lambda is the best to get the minimum RMSE.



In this model, the value of lambda is:

```
## [1] 4.75
```

Taking this value of lambda, we can now build the table again with the RMSE values of our models.

method	**RMSE**
Just the Average Model	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646844
Regularised Movie + User Effect Model	0.8641362
Regularised Movie + User + Year Effect Model	0.8638377

5 Results

5.1 Final Model

From the table showing the RMSE's of the models we built, we can see that the Regularised Movie + User + Year Effect Model has the lowest value. So our final model can be represented by the following equation:

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

We can now use the validation ratings in our final model.

```r
# Final Model using whole edx dataset and validation dataset to predict

mu <- mean(edx$rating)

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda2), n_i = n())

b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda2), n_u = n())

b_y <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  group_by(year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+lambda2), n_y = n())

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

model_final_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Final Prediction Model using validation set",
                                     RMSE = model_final_rmse ))
rmse_results %>% knitr::kable() %>% kable_styling(position = 'center')%>%
  column_spec(1,border_left = T)%>%column_spec(2,border_right = T)%>%
  row_spec(0,bold=T)
```

method	**RMSE**
Just the Average Model	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646844
Regularised Movie + User Effect Model	0.8641362
Regularised Movie + User + Year Effect Model	0.8638377
Final Prediction Model using validation set	0.8645223

We can confirm that the final model using the validation ratings generates a RMSE of 0.86452. This model is useful since it takes into account any movie, user and year of movie release effects.

6 Conclusion

This report explains the process of developing a movie recommendation system based on user ratings using the 10M version of the MovieLens dataset. After dividing the dataset into training and validation sets, the training one was further divided into two so that it could be used to train the model, calculate parameters and for regularisation. In order to get more insights into the data, exploratory and analytic techniques were used. The final model took into consideration movie, user and year of movie release bias.

6.1 Limitations of the Model and Possible Improvements

The final model could not use the full version of the dataset because of hardware constraints (not enough RAM). In addition, there are much more features that could be analysed as well to improve the model, for example, the movie genres and the time when the rating was given. Other methods could also be employed to see if a lower RMSE can be achieved like Matrix Factorisation.