Moviedex Report

Ciaran Boyle

10/21/2020

Contents

1 - Introduction

1.1 - Description of the Movielens Dataset

The dataset being observed is the "MovieLens 10M Dataset". It was compiled by "GroupLens", a research lab in the University of Minnesota.[1]

It contains approximately ten million movie ratings, given by 72,000 different users. The movies consist of 20 individual genres, with a total of 797 different genre combinations. The earliest user rating is dated at August 1996, and the latest user rating is dated at January 2008.[2]

1.2 - Goal of the Project

The main goal of this project was to use the Movielens dataset to develop a machine learning algorithm. Various parameters including: User ID, Movie ID, Genre, and date on which the movie was rated, were all used as inputs to predict movie ratings.

The accuracy of the algorithm was assessed through the residual-mean-square error (RMSE) between the predicted ratings and the true ratings of the validation set.

1.3 - Key steps taken

Following the generation of the training, test, and validation datasets, visualization techniques were performed to understand the nature of the different features contained within the datasets. This analysis helped determine the starting point for the development of the prediction model.

Development of the prediction model followed a step by step procedure, beginning with a simple model and adding more and more parameters in an effort to take account the important biases.

[1] GroupLens (2009), What is GroupLens? (https://grouplens.org/about/what-is-grouplens/)
[2] GroupLens (2009), MovieLens 10M Dataset (https://grouplens.org/datasets/movielens/10m/)

2. Methods/Analysis

2.1 - Data Preparation

Before downloading the data, it is crucial that the proper packages are installed and loaded.

```
################################
# Install and load rpackages
################################
if (!require("tidyverse")) {
  install.packages("tidyverse")
  library(tidyverse)
}
if (!require("caret")) {
  install.packages("caret")
  library(caret)
}
if (!require("data.table")) {
  install.packages("data.table")
  library(data.table)
}
if (!require("dplyr")) {
  install.packages("dplyr")
  library(dplyr)
}
```

Once the appropriate packages have been loaded, the downloading of the Movielens dataset can begin. The data was downloaded and prepared using the code below:

```
################################
# Generate EDX Dataset
################################
# 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")
```

```
# if using R 3.6 or earlier:
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)`
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)
```

The code above also generated a validation set, which will be used to help determine the accuracy of the prediction models developed during this assignment. The downloaded data was stored in the data table named **edx**, and split into two separate sets of data: **training_edx** and **test_edx**. **Training_edx** was used to develop various prediction models, which were used to calculate the RMSE score of the test set **test_edx**.

```
################################
# Generate Training and Test Datasets
################################
#generate training and test sets
set.seed(28, sample.kind="Rounding")
partition_index <- createDataPartition(y = edx$rating, times = 1, p = 0.7, list=FALSE)
training_edx <- edx[partition_index,]
temp <- edx[-partition_index,]
validation_edx <- validation

# Make sure userId and movieId in train set are also in test set
test_edx <- temp %>%
  semi_join(training_edx, by = "movieId") %>%
  semi_join(training_edx, by = "userId")
```

```
# Add rows removed from test set back into train set
removed <- anti_join(temp, test_edx)
training_edx <- rbind(training_edx, removed)
rm(partition_index, temp, removed)
```

2.2 - Data Exploration and Visualization

Once the training and test sets were generated, the `edx` dataset was examined more thoroughly. The first thing to do was to examine the different features of each observation.

```
str(edx, vec.len=2)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 ...
##  $ rating   : num  5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Using the code above, six distinct features were identified. A more indepth outline of the different features is located below.

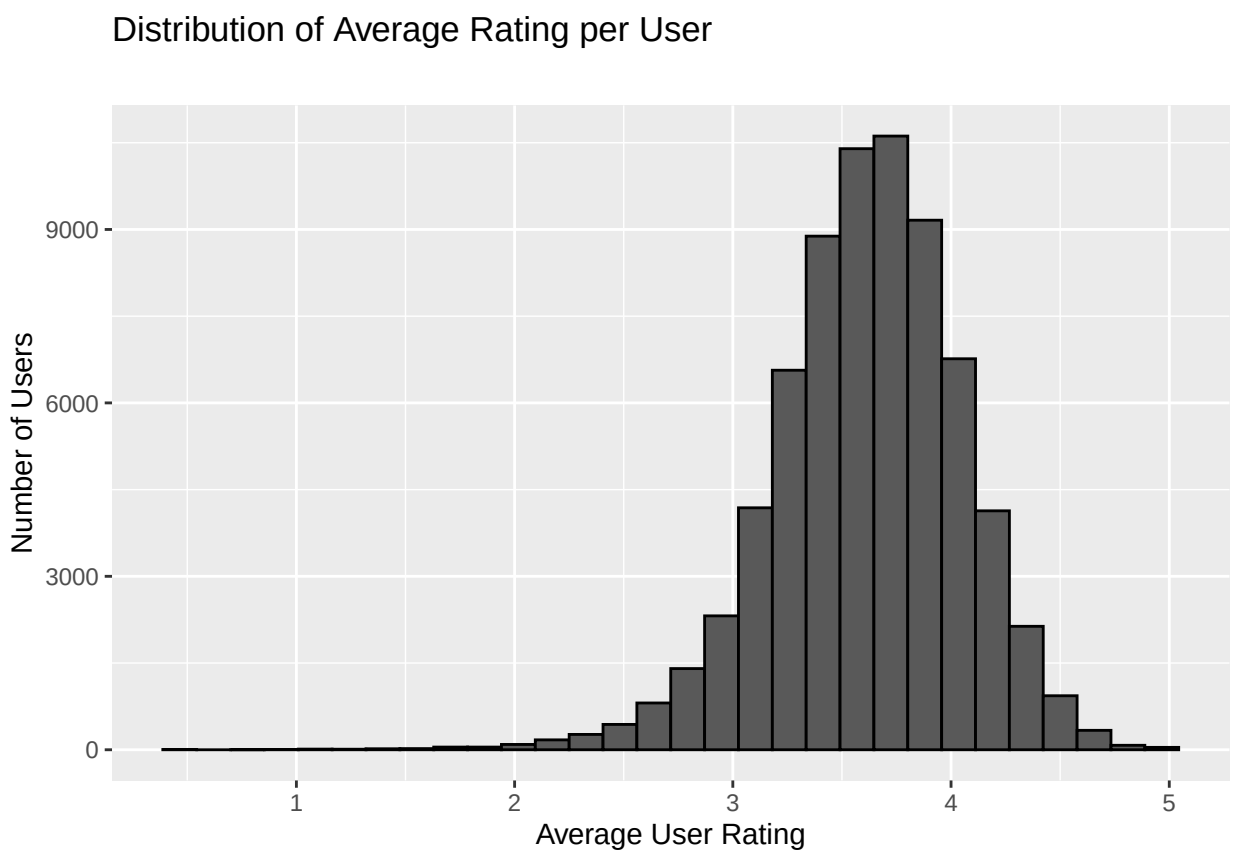
Feature Name	Object Class	Short Description
userId	integer	Unique ID of the user
movieId	numeric	Unique ID of the movie rated by the user
rating	numeric	Rating given to the movie, by the user
timestamp	integer	Timestamp indicating when the rating was given
title	character	Name of the movie rated by the user
genres	character	Genre/s of the movie rated by the user

2.2.1 - Distribution of Average User Rating

One of the first things to examine within the dataset was the average rating per user. There will always be users who rate movies more lightly, while others who rate movies more harshly. Thus, understanding the distribution of the average rating per user was crucial to understanding the users themselves.

```
edx %>% group_by(userId) %>% summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(color="black", size=0.5) +
  labs(title = "Distribution of Average Rating per User", subtitle = "") +
```

```
  xlab("Average User Rating") +
  ylab("Number of Users")
```



From this graph, it can be concluded that on on average, a large majority of users have an average rating of 3 or above. Significantly fewer users have an average rating of 2 or lower. The exact proportions were calculated by using the code below.

```
temp <- edx %>% group_by(userId) %>%
  summarize(avg_rating = mean(rating)) #temp storage for summarized average ratings

mean(temp$avg_rating>=3) #user ratings that are greater than or equal to 3
```

```
## [1] 0.9281748
```

```
mean(temp$avg_rating<=2) #user ratings that are less than or equal to 2
```

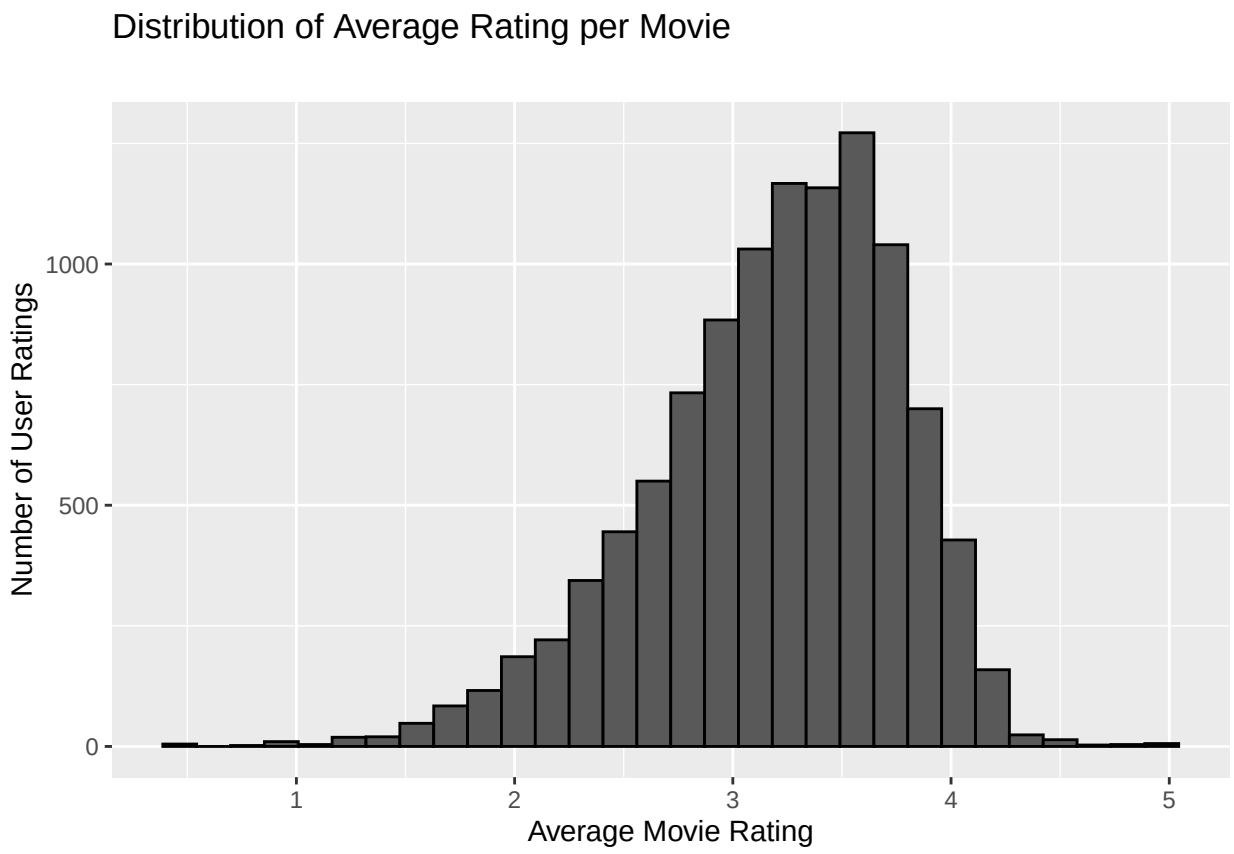
```
## [1] 0.002776267
```

Thus, approximately 92.8% of users have an average rating of 3 or above, while approximately 0.3% of users have an average rating of 2 or less.

2.2.2 - Distribution of Average Movie Rating

In addition to examining the distribution of the average rating per user, the distribution of the average rating per movie was also examined. Critically acclaimed movies by reviewers and users alike will be expected to have a higher average rating, while movie flops will be expected to have a lower average rating.

```
edx %>% group_by(movieId) %>% summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) + geom_histogram(color="black", size=0.5) +
  labs(title = "Distribution of Average Rating per Movie", subtitle = "") +
  xlab("Average Movie Rating") +
  ylab("Number of User Ratings")
```



From this graph, it can be concluded that on average, most movies have an average rating somewhere between 3 and 4. The exact proportions were calculated by using the code below.

```
temp <- edx %>% group_by(movieId) %>%
  summarize(avg_rating = mean(rating)) #temp storage for summarized average ratings

mean(temp$avg_rating>=3 & temp$avg_rating<=4) #greater or equal to 3 & less or equal to 4
```

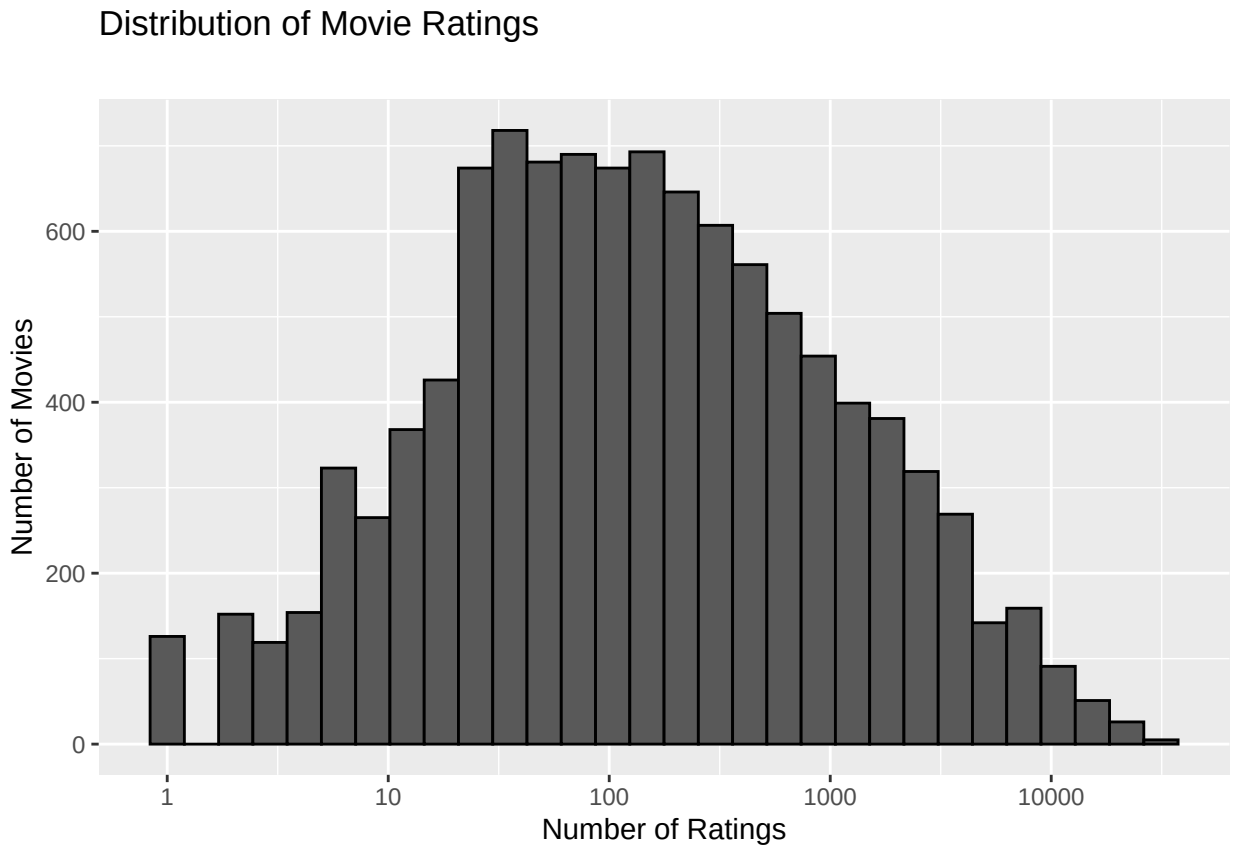
```
## [1] 0.6372577
```

Thus, approximately 63.7% of movies have an average rating of between 3 and 4.

2.2.3 - Distribution of the Number of Movie Ratings

In addition to examining the distribution of the average user rating, the distribution of the movie ratings was also examined. Examining the distribution of the movie ratings is useful, as the implication is that blockbusters and cult classics will be more likely to have a larger number of ratings, while niche movies are less likely to have been rated as much.

```
edx %>% group_by(movieId) %>% summarize(count = n()) %>% ggplot(aes(count)) +
                        geom_histogram(color="black", size=0.5) +
                        scale_x_log10() +
                        labs(title = "Distribution of Movie Ratings", subtitle = "") +
                        xlab("Number of Ratings") +
                        ylab("Number of Movies")
```



From this graph, it can be concluded that most movies have been rated by a large number of users. Relatively few movies have less than 10 ratings, while more than half have been rated at least 100 times.

```
temp <- edx %>% group_by(movieId) %>%
  summarize(count = n()) #temp summarized average ratings

mean(temp$count<=10) #movies with 10 ratings or less
```

```
## [1] 0.1066779
```

```
mean(temp$count>=100) #movies with at least 100 ratings
```

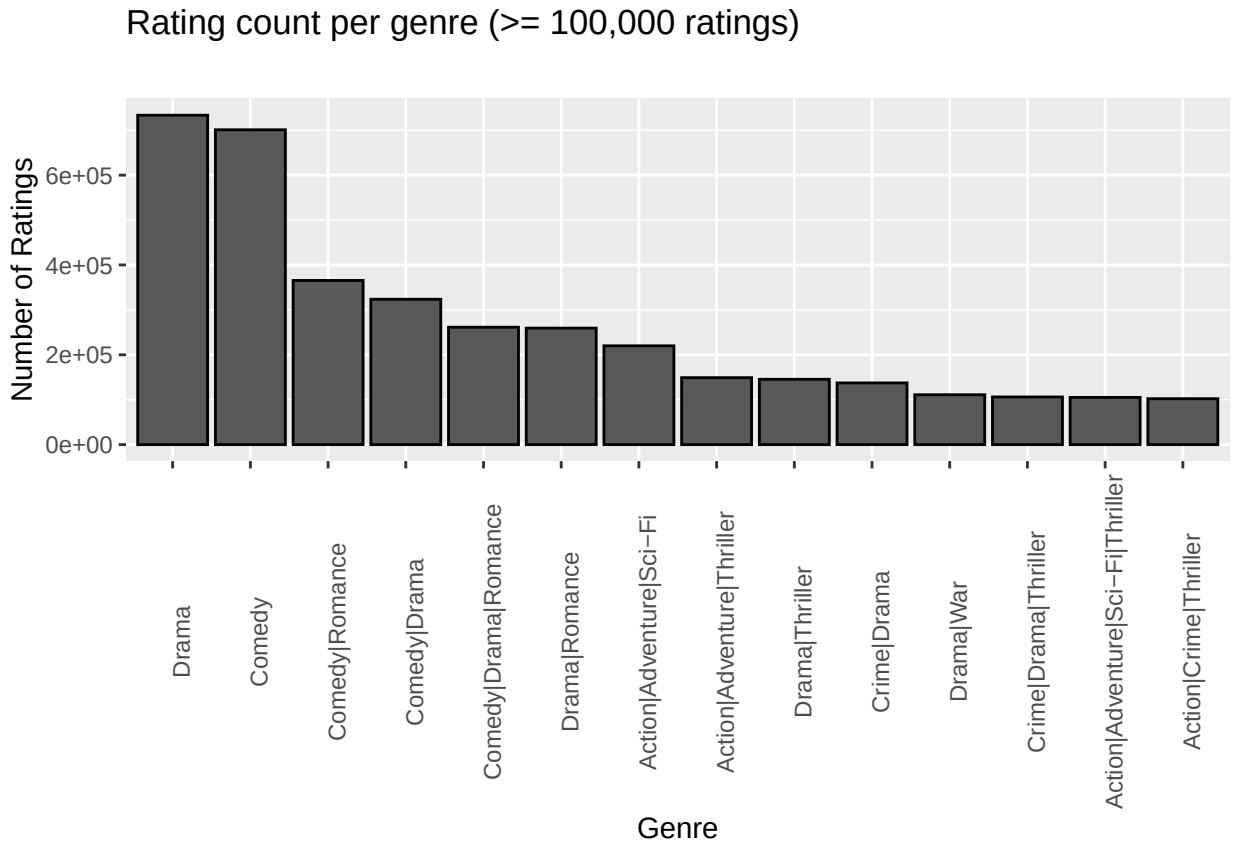
```
## [1] 0.5348881
```

Thus, approximately 9.8% of movies have 10 ratings or less, and approximately 53.49% of movies have 100 ratings or more.

2.2.4 - Distribution of Genre Ratings

While examining the distribution of movie ratings is helpful, it does not give any insight into the types of movies the users prefer. To answer this question, the number of ratings per genre was examined in order to determine the popularity of certain genres compared to others. As there are a large variety of different genres, only genres with at least 100,000 ratings were included in the chart.

```
edx %>% group_by(genres) %>% summarize(count = n()) %>% filter(count>=100000) %>%
  transform(., genres=reorder(genres,-count)) %>%
  ggplot(aes(x=factor(genres), y=count)) +
  geom_bar(stat = "identity", color="black", size=0.5) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Rating count per genre (>= 100,000 ratings)", subtitle = "") +
  xlab("Genre") +
  ylab("Number of Ratings")
```



From this graph, it appears that Comedy and Drama are the most popular genres. Together, they make up over 1.4 million user ratings. The next 4 most popular genres have Drama and/or Comedy make up part/all of the genre combination.

3. Results

3.1 - Developing the machine learning algorithm

Once the data was examined, modelling & optimization of the model began. The RMSE score was used to determine the accuracy of each model. To do this, a function was written to calculate the RMSE between a set of predicted ratings generated by the model and the actual ratings.

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

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,m}^{N}(\hat{R}_{u,m} - R_{u,m})^2}$$

The RMSE is calculated by taking the difference between each predicted rating ($\hat{R}_{u,m}$) and the true rating ($R_{u,m}$), squaring it, and adding up the sum of the squares together. This is done with accordance to the number of observations (N) present in the dataset. The sum of the squares is then divided by the number of observations (N) within the dataset, and then "square rooted". The result is known as the residual mean squared error, or the $RMSE$.

Before calculating the RMSE of the validation set, the RMSE will be calculated between the true ratings of the test set, and the predicted ratings generated by prediction model. The model that generates the lowest RMSE score will be used to calculate the RMSE of the final validation set.

3.2 - Prediction Models

A large variety of different prediction models was created. The very first of which is simple in nature, with the subsequent models becoming more complex in nature, and taking multiple different features into account.

3.2.1 - User Average

The simplest model simply involves calculating the average rating (μ), and to use the average as the predictor score.

$$R_{u,m} = \mu + \epsilon_{u,m}$$

$\epsilon_{u,m}$ simply refers to the difference between the predicted rating and the true rating for a particular movie that a user has rated. This is also known as the error. As the prediction models develop and become more complex, more and more biases are added into the formula in an attempt to reduce the error. If done correctly, the error between each predicted rating and the true rating (and subsequently the RMSE) should become smaller.

```
################################
# Compute the user average and calculate the RMSE
################################
average_user_rating <- mean(training_edx$rating)
RMSE(average_user_rating,test_edx$rating)
```

```
## [1] 1.060948
```

This model gives us an RMSE score of approximately 1.061. There are still various parameters that can be incorporated into future prediction models to further reduce the RMSE score.

3.2.2 - Movie Bias

As seen in the chart for the average ratings per movie, the average rating can vary from movie to movie. It is thus logical to assume that for any given movie, there is likely to be an inherent movie bias (β_m) that has an influence on the rating the user gives.

$$R_{u,m} = \mu + \beta_m + \epsilon_{u,m}$$

Beginning with this prediction model, various different biases were added in an attempt to more accurately predict the rating a user would give to a particular movie. The equation to calculate any given bias is as follows:

```
################################
# Compute the movie bias and calculate the RMSE
################################
average_movie_rating <- training_edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - average_user_rating))

predicted_ratings <- test_edx %>%
  left_join(average_movie_rating, by="movieId") %>%
  mutate(pred = average_user_rating + b_m) %>%
  pull(pred)
RMSE(predicted_ratings, test_edx$rating)
```

```
## [1] 0.9443302
```

This model gives us an RMSE score of approximately 0.944. This is a lot lower than the previous RMSE score of 1.061.

3.2.3 - User Bias

As seen in the chart for the average ratings per user, some users are more critical when giving out ratings compared to other users. A user's own personal bias (β_u) is thus also likely to have an effect on the rating they give.

$$R_{u,m} = \mu + \beta_m + \beta_u + \epsilon_{u,m}$$

```
################################
# Compute the user bias and calculate the RMSE
################################
user_bias <- training_edx %>%
  left_join(average_movie_rating, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - average_user_rating - b_m))

##test RMSE
predicted_ratings <- test_edx %>%
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  mutate(pred = average_user_rating + b_m + b_u) %>%
  pull(pred)
RMSE(predicted_ratings, test_edx$rating)
```

```
## [1] 0.8676968
```

This model gives an RMSE score of approximately 0.868. Again, a lot lower than the previous RMSE score of 0.944.

3.2.4 - Movie-Year Bias

It has already been established that the movie title and a user's own personal bias has an effect on how they rate a movie. It would be interesting to see if the average rating for any given movie varies from year to year. This can be defined as the movie-year bias ($\beta_{m,y}$).

$$R_{u,m} = \mu + \beta_m + \beta_u + \beta_{m,y} + \epsilon_{u,m}$$

```
################################
# Compute the movie-year bias and calculate the RMSE
################################
year_bias <- training_edx %>%
  mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
  #used to convert timestamp into year and extract

  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  group_by(movieId, year_rated) %>% #group entries by movieId and year in which it was rated
  summarize(b_y = mean(rating - average_user_rating - b_m - b_u))
```

```
##test RMSE
predicted_ratings <- test_edx %>%
  mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  left_join(year_bias, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  mutate(pred = average_user_rating + b_m + b_u + b_y) %>%
  pull(pred)
RMSE(predicted_ratings, test_edx$rating)
```

```
## [1] 0.8654799
```

This model gives an RMSE score of approximately 0.865. Compared to the previous score of 0.867, the implication is that there is some bias (albeit very little) regarding the year on which a user rates a movie. However, this bias could also be explained due to other various biases rather than a direct relationship between review date and the movie title.

3.2.5 - User-Genre Bias

It is a given that some users will prefer certain genres over others. For instance, one user may give more favourable ratings towards Comedy films, while giving a lower score to Romance films. Thus, this user-genre bias ($\beta_{u,g}$) should be taken into account when attempting to predict the rating a user would give towards a particular movie. To remove the effect of genres that have a very low number of ratings, only genres that have 1000 or more ratings are considered in the user-genre bias.

$$R_{u,m} = \mu + \beta_m + \beta_u + \beta_{m,y} + \beta_{u,g} + \epsilon_{u,m}$$

```
################################
# Compute the user-genre bias and calculate the RMSE
################################
#index to filter out genres with less than 1000 ratings when calcualting the usergenre bias
usergenre_bias_index <- training_edx %>%
  mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  left_join(year_bias, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  group_by(genres) %>%
  summarize(n=n(), index=1) %>% #counts observations per genre
  filter(n>=1000) %>% #filters out genres with less than 1000 ratings
  arrange(desc(n)) %>%
  select(-n)
```

```
usergenre_bias <- training_edx %>%
  mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  left_join(year_bias, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  left_join(usergenre_bias_index, by="genres") %>%
  filter(index==1) %>% #filters out genres not in index (i.e.: less than 1000 ratings)
  group_by(userId,genres) %>%
  summarize(b_ug = mean(rating - average_user_rating - b_m - b_u - b_y))

##test RMSE
predicted_ratings <- test_edx %>%
  mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias, by="userId") %>%
  left_join(year_bias, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  left_join(usergenre_bias, by=c("userId","genres")) %>%
  replace(is.na(.), 0) %>%
  mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
  pull(pred)
RMSE(predicted_ratings, test_edx$rating)
```

```
## [1] 0.92131
```

This model gives an RMSE score of approximately 0.921. While this is higher than the previous two models, regularization can be used in an effort to help lower the score.

3.2.6 - Model Summaries

All of the previous prediction models can be summarized in the table below.

Model	RMSE Score
User Average	1.06095
User Average + Movie Bias	0.94433
User Average + Movie & User Bias	0.86770
User Average + Movie, User & Movie-Year Bias	0.86548
User Average + Movie, User, Movie-Year, & User-Genre Bias	0.92131

The most basic prediction model had an RMSE score of approximately 1.061. The model with the lowest RMSE score of 0.865 takes a lot of other features and parameters into account. This improvement in the RMSE score can be considered

quite good, with an almost 20% reduction from the intial RMSE to the lowest RMSE.

3.3 - Regularized Prediction Models

While the RMSE scores that have been calculated thus far are decent, regularization may help further reduce the scores. Regularization is the process of penalizing large estimates formed using sample size, to constrain the total variability of the effect sizes.[3] The equation to calculate any given regularized bias is as follows:

$$\beta_{(\lambda)} = \frac{1}{\lambda + N} \sum (R_{u,m} - \mu)$$

The regulated bias is calculated by taking the difference between the true rating, the average rating, and all other biases, adding them together, and then dividing by the number of observations within that group, plus an arbitrary number. This arbitrary number is the lambda.
This new regulated bias replaces the original bias in the equation, and is then used to calculate the new RMSE score. The optimal lambda is considered to be the number that produces the minimum RMSE score out of all other lambdas.

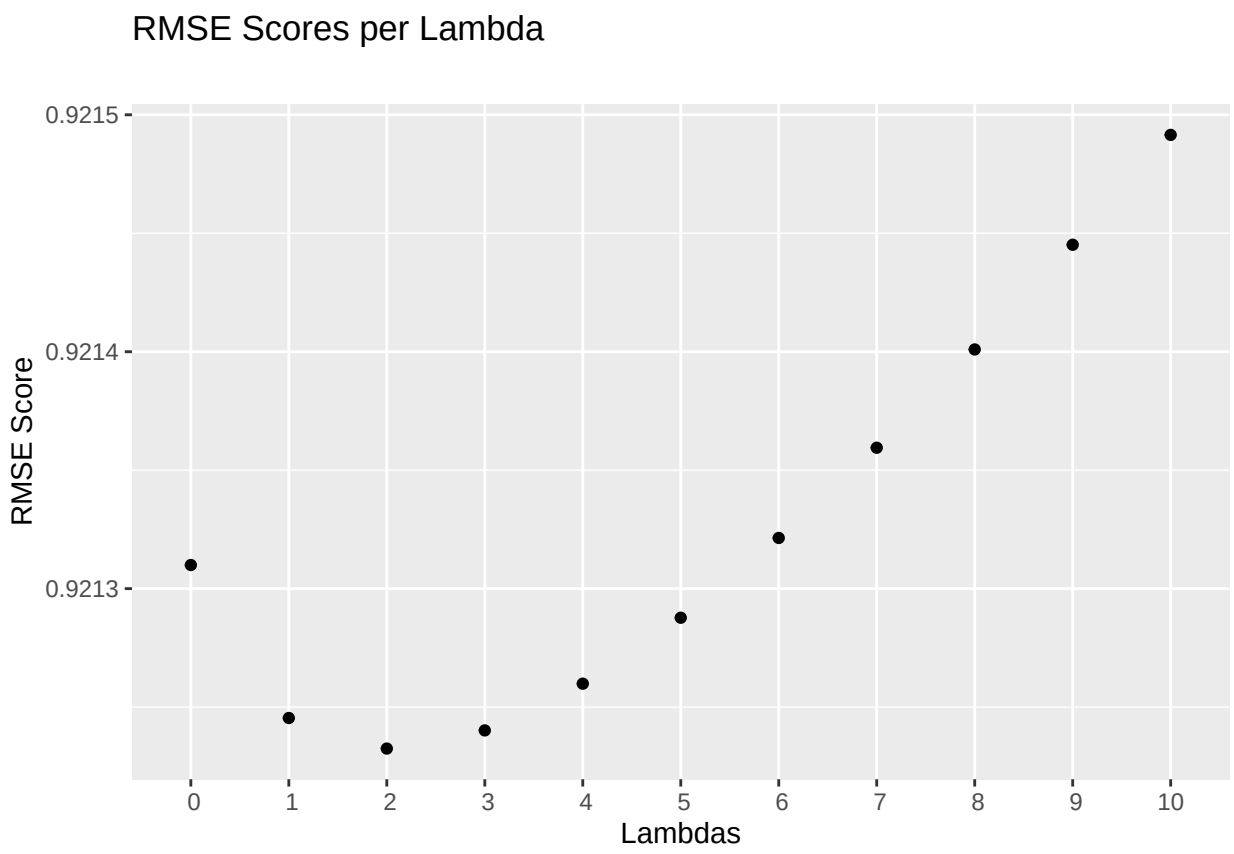
3.3.1 - Movie Bias (Regularized)

$$R_{u,m} = \mu + \beta_{m(\lambda_m)} + \beta_u + \beta_{m,y} + \beta_{u,g} + \epsilon_{u,m}$$

```
#################################
# Regularize the movie bias and calculate the RMSE
#################################
##generate test lambdas
lambdas <- seq(0,10,1)

##calculate rmses for different lambdas
moviebias_rmses <- sapply(lambdas, function(l){

  average_movie_rating_reg <- training_edx %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - average_user_rating)/(n()+l))

  predicted_ratings <- test_edx %>%
    mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias, by="userId") %>%
    left_join(year_bias, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    left_join(usergenre_bias, by=c("userId","genres")) %>%
    replace(is.na(.), 0) %>%
    mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
```

[3] Rafael A. Irizarry (2020), Chapter 33.9.2 - Penalized Least Squares (https://rafalab.github.io/dsbook/large-datasets.html#penalized-least-squares)

```
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})

#generate graph
movie_bias_chart <-
  ggplot(as.data.frame(cbind(lambdas, moviebias_rmses)), aes(lambdas, moviebias_rmses)) +
  geom_point() +
  labs(title = "RMSE Scores per Lambda", subtitle = "") +
  xlab("Lambdas") + scale_x_discrete(limits=c(0:10)) +
  ylab("RMSE Score")
movie_bias_chart
```



The optimal lambda can be examined through the use of the code below.

```
#check optimal lambda
lambda_moviebias <- lambdas[which.min(moviebias_rmses)]
lambda_moviebias
```

```
## [1] 2
```

The optimal lambda for the movie bias is “2”. This can be incorporated into the prediction model as the regulated movie bias, and be used to calculate the new RMSE.

```
##incorporate optimal lambda rating into movie bias
average_movie_rating_reg <- training_edx %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - average_user_rating)/(n()+2))

#check RMSE score for optimal lambda
moviebias_rmses[which.min(moviebias_rmses)]
```

```
## [1] 0.9212325
```

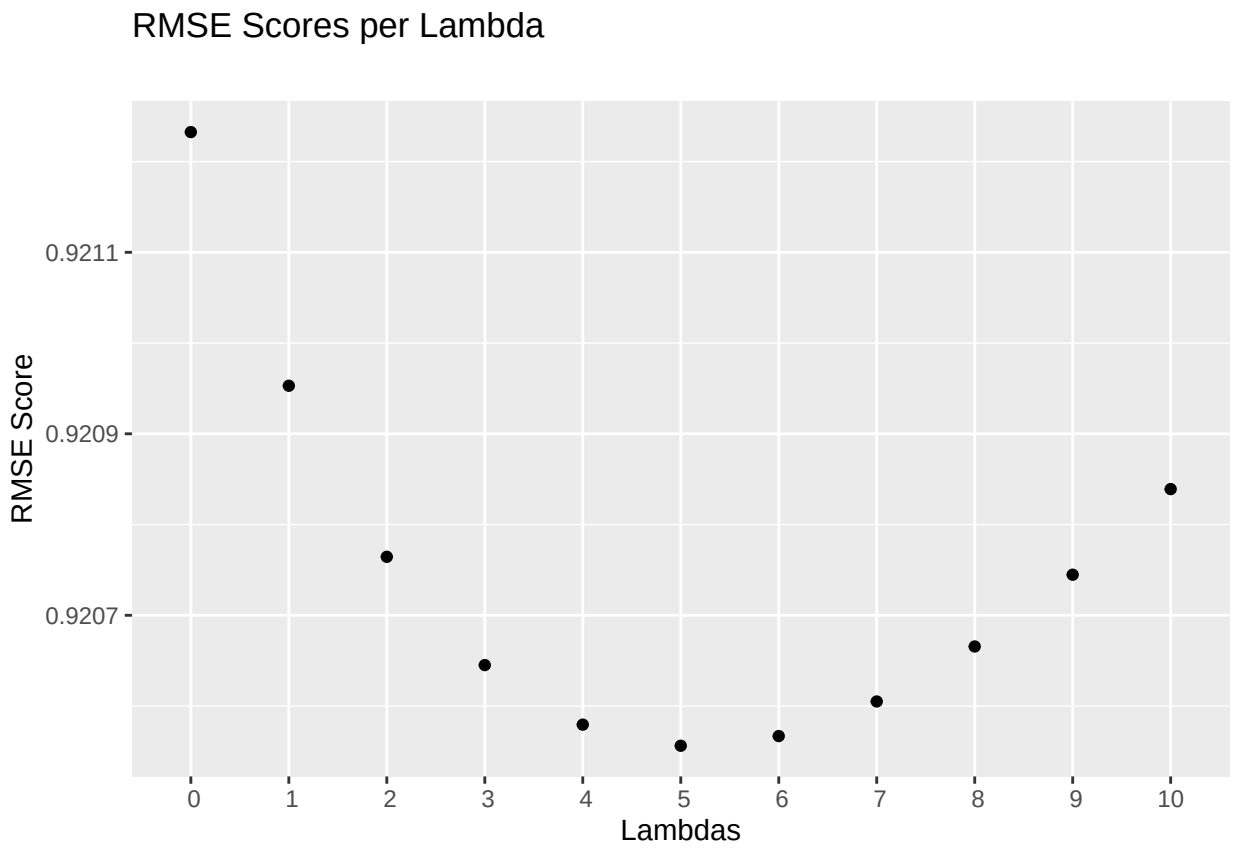
Compared to the previous RMSE score of 0.92131, this RMSE score of 0.92123 is not significantly lower.

3.3.2 - User Bias (Regularized)

$$R_{u,m} = \mu + \beta_{m(\lambda_m)} + \beta_{u(\lambda_u)} + \beta_{m,y} + \beta_{u,g} + \epsilon_{u,m}$$

```
##################################
# Regularize the user bias and calculate the RMSE
##################################
##generate test lambdas
lambdas <- seq(0,10,1)

##calculate rmses for different lambdas
userbias_rmses <- sapply(lambdas, function(l){

  user_bias_reg <- training_edx %>%
    left_join(average_movie_rating, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - average_user_rating - b_m)/(n()+l))

  predicted_ratings <- test_edx %>%
    mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    left_join(year_bias, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    left_join(usergenre_bias, by=c("userId","genres")) %>%
    replace(is.na(.), 0) %>%
    mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})

#generate graph
```

```
user_bias_chart <-
  ggplot(as.data.frame(cbind(lambdas,userbias_rmses)), aes(lambdas, userbias_rmses)) +
  geom_point() +
  labs(title = "RMSE Scores per Lambda", subtitle = "") +
  xlab("Lambdas") + scale_x_discrete(limits=c(0:10)) +
  ylab("RMSE Score")
user_bias_chart
```



The optimal lambda can be examined through the use of the code below.

```
#check optimal lambda
lambda_userbias <- lambdas[which.min(userbias_rmses)]
lambda_userbias
```

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

The optimal lambda for the movie bias is “5”. This can be incorporated into the prediction model as the regulated user bias, and be used to calculate the new RMSE.

```
##incorporate optimal lambda rating into user bias
user_bias_reg <- training_edx %>%
  left_join(average_movie_rating, by = "movieId") %>%
```

```
  group_by(userId) %>%
  summarize(b_u = sum(rating - average_user_rating - b_m)/(n()+5))

#check RMSE score for optimal lambda
userbias_rmses[which.min(userbias_rmses)]
```

```
## [1] 0.9205562
```

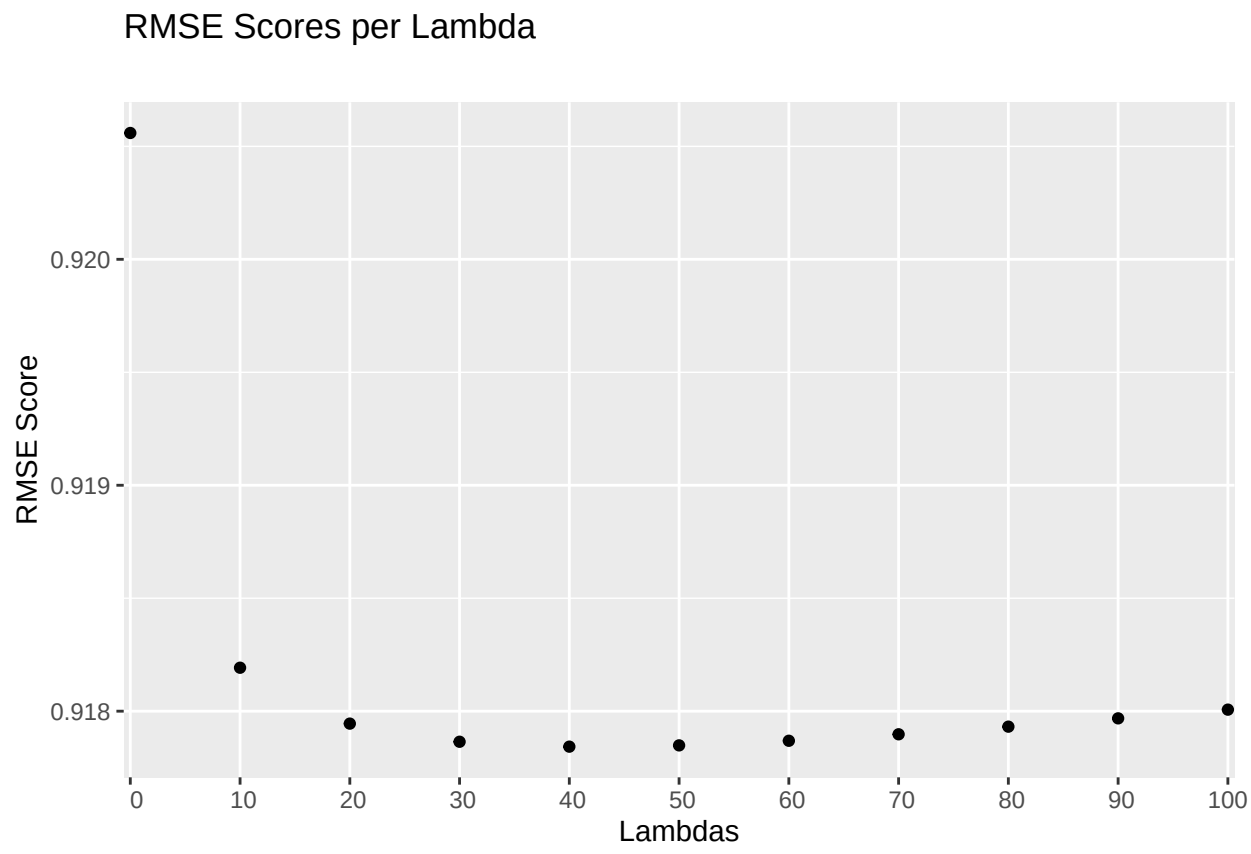
Compared to the previous RMSE score of 0.92123, this RMSE score of 0.92056 is also not significantly lower.

3.3.3 - Movie-Year Bias (Regularized)

$$R_{u,m} = \mu + \beta_{m(\lambda_m)} + \beta_{u(\lambda_u)} + \beta_{m,y(\lambda_{m,y})} + \beta_{u,g} + \epsilon_{u,m}$$

```
##################################
# Regularize the movie-year bias and calculate the RMSE
##################################
##generate test lambdas
lambdas <- seq(0,100,10)

##calculate rmses for different lambdas
movieyear_rmses <- sapply(lambdas, function(l){

  year_bias_reg <- training_edx %>%
    mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    group_by(movieId, year_rated) %>%
    summarize(b_y = sum(rating - average_user_rating - b_m - b_u)/(n()+l))

  predicted_ratings <- test_edx %>%
    mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    left_join(usergenre_bias, by=c("userId","genres")) %>%
    replace(is.na(.), 0) %>%
    mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})

#generate graph
```

```
movieyear_chart <-
  ggplot(as.data.frame(cbind(lambdas,movieyear_rmses)), aes(lambdas, movieyear_rmses)) +
  geom_point() +
  labs(title = "RMSE Scores per Lambda", subtitle = "") +
  xlab("Lambdas") + scale_x_discrete(limits=seq(0,100,10)) +
  ylab("RMSE Score")
movieyear_chart
```



The optimal lambda can be examined through the use of the code below.

```
#check optimal lambda
lambda_movieyear <- lambdas[which.min(movieyear_rmses)]
lambda_movieyear
```

```
## [1] 40
```

The optimal lambda for the movie bias is “40”. This can be incorporated into the prediction model as the regulated movie-year bias, and be used to calculate the new RMSE.

```
##incorporate optimal lambda rating into movie-year bias
year_bias_reg <- training_edx %>%
  mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
```

```
  left_join(average_movie_rating, by="movieId") %>%
  left_join(user_bias_reg, by="userId") %>%
  group_by(movieId, year_rated) %>%
  summarize(b_y = sum(rating - average_user_rating - b_m - b_u)/(n()+40))

#check RMSE score for optimal lambda
movieyear_rmses[which.min(movieyear_rmses)]
```

```
## [1] 0.917843
```

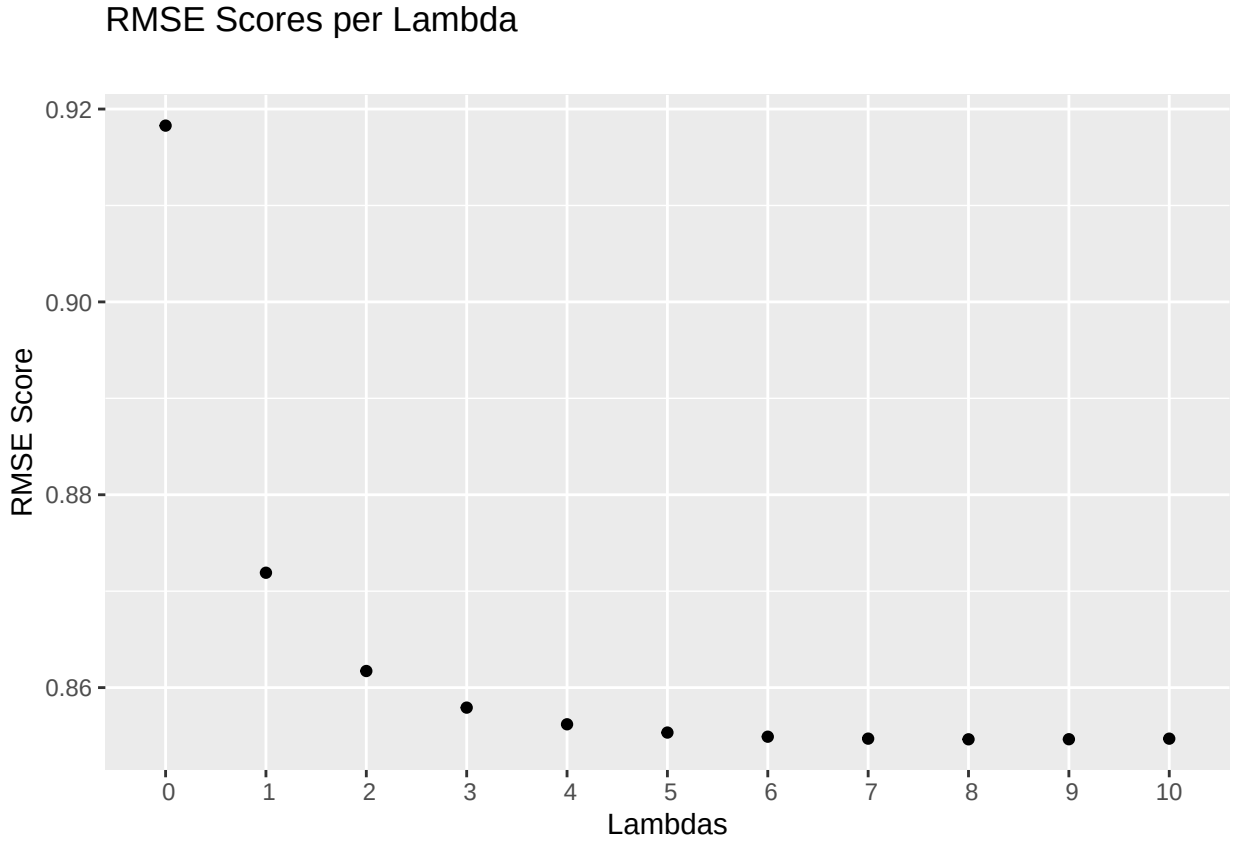
Compared to the previous RMSE score of 0.92056, this RMSE score of 0.91784 is also not significantly lower.

3.3.4 - User-Genre Bias (Regularized)

$$R_{u,m} = \mu + \beta_{m(\lambda_m)} + \beta_{u(\lambda_u)} + \beta_{m,y(\lambda_{m,y})} + \beta_{u,g(\lambda_{u,g})} + \epsilon_{u,m}$$

```
##################################
# Regularize the user-genre bias and calculate the RMSE
##################################
##generate test lambdas
lambdas <- seq(0,10,1)

##calculate rmses for different lambdas
usergenre_rmses <- sapply(lambdas, function(l){

  usergenre_bias_index <- training_edx %>%
    mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    group_by(genres) %>%
    summarize(n=n(), index=1) %>%
    filter(n>=1000) %>%
    arrange(desc(n)) %>%
    select(-n)

  usergenre_bias_reg <- training_edx %>%
    mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    left_join(usergenre_bias_index, by="genres") %>%
```

```
    filter(index==1) %>%
    group_by(userId,genres) %>%
    summarize(b_ug = sum(rating - average_user_rating - b_m - b_u - b_y)/(n()+l))

  predicted_ratings <- test_edx %>%
    mutate(year_rated = year((as.POSIXct(test_edx$timestamp, origin="1970-01-01")))) %>%
    left_join(average_movie_rating_reg, by="movieId") %>%
    left_join(user_bias_reg, by="userId") %>%
    left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
    replace(is.na(.), 0) %>%
    left_join(usergenre_bias_reg, by=c("userId","genres")) %>%
    replace(is.na(.), 0) %>%
    mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})

#generate graph
usergenre_chart <-
  ggplot(as.data.frame(cbind(lambdas,usergenre_rmses)), aes(lambdas, usergenre_rmses)) +
  geom_point() +
  labs(title = "RMSE Scores per Lambda", subtitle = "") +
  xlab("Lambdas") + scale_x_discrete(limits=seq(0,10,1)) +
  ylab("RMSE Score")
usergenre_chart
```



The optimal lambda can be examined through the use of the code below.

```
#check optimal lambda
lambda_usergenre <- lambdas[which.min(usergenre_rmses)]
lambda_usergenre
```

```
## [1] 8
```

The optimal lambda for the movie bias is “8”. This can be incorporated into the prediction model as the regulated movie-year bias, and be used to calculate the new RMSE.

```
##incorporate optimal lambda rating into user-genre bias
usergenre_bias_reg <- training_edx %>%
  mutate(year_rated = year((as.POSIXct(training_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating_reg, by="movieId") %>%
  left_join(user_bias_reg, by="userId") %>%
  left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  left_join(usergenre_bias_index, by="genres") %>%
  filter(index==1) %>%
  group_by(userId,genres) %>%
  summarize(b_ug = sum(rating - average_user_rating - b_m - b_u - b_y)/(n()+8))
```

```
#check RMSE score for optimal lambda
usergenre_rmses[which.min(usergenre_rmses)]
```

```
## [1] 0.8546393
```

Compared to the previous RMSE score of 0.91784, this RMSE score of 0.85463 has been lowered significantly compared to the changes in the previous 3 prediction models. In fact, this RMSE score is currently the lowest out of all prediction models developed thus far.

3.3.5 - Regularized Model Summaries

Regularized Models	RMSE Score
User Average + Movie Bias	0.92123
User Average + Movie & User Bias	0.92056
User Average + Movie, User & Movie-Year Bias	0.91784
User Average + Movie, User, Movie-Year, & User-Genre Bias	0.85464

As seen in the table above, regularization of each of the different biases continued to reduce the RMSE score, with the regularization of the User-Genre bias having the greatest effect on the RMSE score.

3.4 - Calculating the RMSE for the Validation set

As a test, the final prediction model will be used to calculate the RMSE between the predicted scores and the true ratings of the validation set.

```
################################
# Calculate RMSE for validation set
################################
predicted_ratings <- validation_edx %>%
  mutate(year_rated = year((as.POSIXct(validation_edx$timestamp, origin="1970-01-01")))) %>%
  left_join(average_movie_rating_reg, by="movieId") %>%
  left_join(user_bias_reg, by="userId") %>%
  left_join(year_bias_reg, by=c("movieId", "year_rated")) %>%
  replace(is.na(.), 0) %>%
  left_join(usergenre_bias_reg, by=c("userId","genres")) %>%
  replace(is.na(.), 0) %>%
  mutate(pred = average_user_rating + b_m + b_u + b_y + b_ug) %>%
  pull(pred)
RMSE(predicted_ratings, validation_edx$rating)
```

```
## [1] 0.8537363
```

Not suprisingly, this RMSE score of approximately 0.85374 is very similar to the RMSE score between the test set and its predicted ratings.

4. Conclusion

4.1 - Brief Summary of Report

Visualization of the edx dataset provided insight into the structure and the distribution of the ratings, with accordance to various combinations of different features.

The prediction model that had the lowest RMSE score took multiple features into account, notably: the average user rating, average rating per movie, average rating per user, movie-year bias, and user-genre bias. In addition, the final model utilized regularization across all features (except the average user rating). This implies that there are multiple factors that influence the rating that a user chooses to give to a movie.

4.2 - Limitations

Due to the size of the dataset and the time it would take to compute the linear regressions accurately, estimations had to be used throughout the generation of the prediction models.

The dataset itself only contains a few features worth incorporating into the prediction models, namely: userId, movieId, and genre. Prediction models used by Netflix for instance, are likely to have incorporated numerous other factors into their prediction model.

4.3 - Future Work

The parameters examined within this report are only a few of the possible parameter combinations that can have an influence on the rating a user chooses to give a movie. For instance, to consider examining the relationship between a movie and its genre.

Basic linear regression was used to predict the movie ratings that a user would give to a particular movie. Exploration of other methods to predict user ratings would be another way to expand on this project.