MoveLens_Project

Scott Boersma

11/16/2020

Introduction

This project is to see an application of machine learning with recommendation system. These systems are used to make suggestions for movies, products,and even services. It is the box that pops up to say "customers also bought" to give you something you did not even think about. A great example of this is when a book called "Touching the Void" was recommended to people who bought "Into Thin Air." "Touching the Void" became a best seller after being recommended to people who bought "Into Thin Air" as it was a best seller. The goal of a recommendation system is to be able to predict the rating someone would give and recommend the items that are predicted a high rating for that product, movie, or content. A successful recommendation system can increase user retention, satisfaction, and loyalty leading to higher profits and increased growth.

User recommendation systems are typically based on a 1 to 5 rating system. In nearly all instances, 1 is low (bad) and five is high (best). Often times there are other indicators depending on the type of content being analyzed. Some recommendation systems have access to users comments, likes, or other reactions. These indicators can be used recommend or show more content that gets a certain emotion. Most recommendation systems work off the principle of giving a person more of what they want and this want is often a single want or type of content. For instance using YouTube for music can get more music play lists in your feed where as TicTok attempts to bring you everything you want as well. These indicators are outside of the scope of this project.

All recommendation systems need a way to measure the accuracy of its predictions. In this project, the error is measured by the root mean squared error. The goal of this project is to account for enough of the effects and biases in the ratings to predict ratings that produce an error of less than 0.86490.

The document consists of an introduction, an overview, a summary, method and analysis.

Overview

The prompt of the project comes from the Netflix Challenge. The goal of the challenge was to improve Netflix's in house algorithm by 10% and win 1 Million dollars. This project uses the MovieLens data set from a research lab in the University of Minnesota called GroupLens.

The complete MovieLens data set consists of 27 million ratings, 58,000 movies, and 280,00 users. The data set used in this paper is a subset of the original with 10 million ratings, 10,000 movies and 72,000 users. The subset being roughly 10% of the original data set.

The goal of the project is to create a movie recommendation system from the subset of the original MovieLens data. The algorithm will be tested using the root mean squared error. The error is calculated after each identified effect/bias. In this exploration, any improvement in the RMSE from the bias was kept regardless of how small. The error goal of this project is an error less than 0.86490.

1. Data preparation: download, parse, import, and prepare the data to be processed and analyzed.

2. Data exploration: explore the data to understand the variables, relationships between them, and where possible predictors lie.
3. Data analysis and modeling: creating the model based on insights from the exploration of the data set.
4. Results
5. Conclusion

Executive Summary

As stated earlier this project is motivated by the Netflix movie challenge. First the data was downloaded and cleaned into tidy format. It was then split into a training (edx set) set with 90 percent of the data and a test (validation set) set with 10 percent of the data. The train set was split again 90/10. This was done to preserve the original test set (validation) for the final test of the algorithm. Then each set of data was then mutated to add columns for the year the movie was released, the date of the rating, and the day of the week of the rating. Exploration was done. This exploration is conducted to get an understanding of the data. For example, the edx data set has over nine million observations and six variable and the validation set had under one million observations with six variables. After initially exploring the data set, there is some probability that the year the movie was released, date, length of time from the first rating by each user, and the day of the week could have an effect on the data. Thus additional columns were made to clearly look at these other possible indications. All of which, were deemed possible predictors.

Method

There were two approaches to reach the goal error value. The first was looking at the different variables and seeing if they had an effect or bias as to help predict the rating a particular user would give. After each predictor, an error score was taken and stored in the table called result. After the predictors were gone through, the process of regularization commenced. This was done to give more credit to movies with lots of rating and less credit to movies with few ratings.

The second approach was to use matrix factorization. The assumption here being the more similar a two movies are the closer the rating. There rating is correlated positively. The other end of this idea is that movies dissimilar to one another have an inverse relationship. An example of these is if a user highly rates a gangster movie they are likely to rate a gangster movie similarly. The same user also most likely does not like romantic comedy and hence rates them lower. This is because they are very different to one another. This methodology is used with the RecoSystem library.

Data Preparation - Prompt Code in this section the data is downloaded and made into tidy format. Then the data is split into a train set called edx and a test set called validation. The edx set contains 90% of the total data downloaded and the validation contains the remaining 10%. These sets were kept in their original format for purity and any modifications were assigned to a new variable name.

```r
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")

library(tidyverse)
library(caret)
library(data.table)

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

```r
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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)
```

```r
# Adding all additional libraries used in this project
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")


library(lubridate)
library(ggplot2)
library(ggthemes)
library(scales)
```

```r
head(edx)
```

Initial exploration

```
##    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
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

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

Edx contains 6 variables. The userId and movieId's are numeric. The timestamp column is in epoch time and the year the movie is released is in parentheses in the title column. Finally it is learned, there are nearly 70 thousand users rating 10.6 thousand movies.

```
# Number of years ratings were given
year(as_datetime(max(edx$timestamp))) - year(as_datetime(min(edx$timestamp)))
```

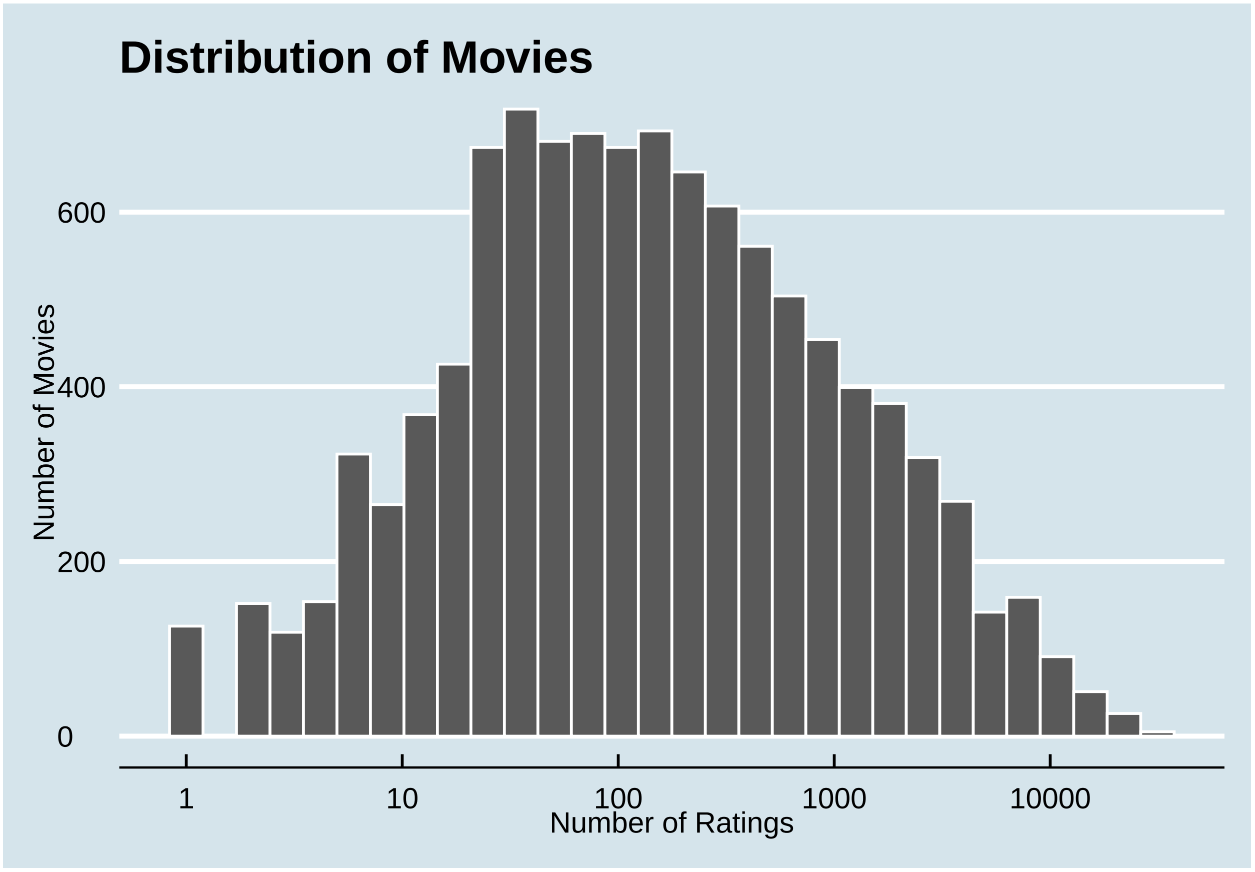
```
## [1] 14
```

The ratings were taken over a 14 year period.

```
edx %>%
  group_by(genres) %>%
  summarise(n=n()) %>%
  head()
```

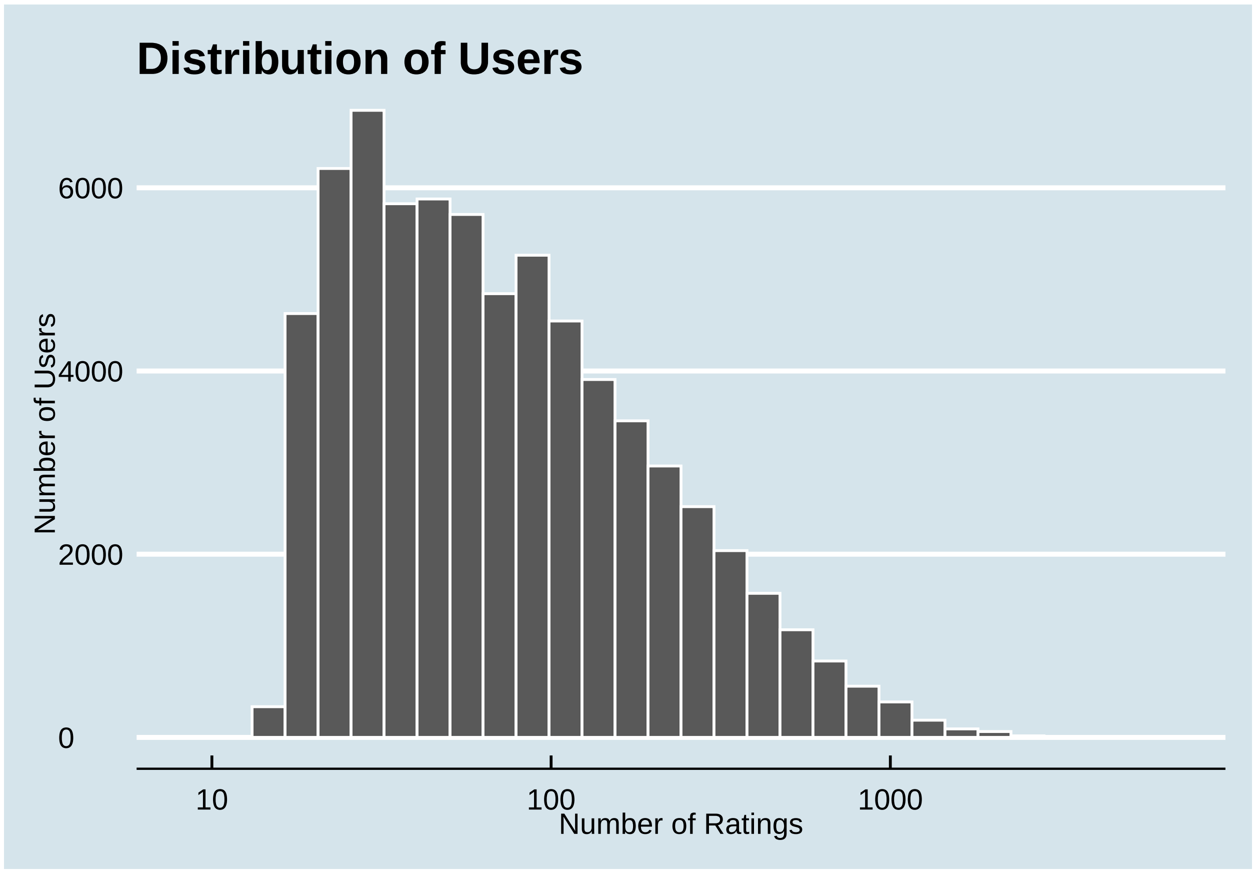
```
## # A tibble: 6 x 2
##   genres                                               n
##   <chr>                                            <int>
## 1 (no genres listed)                                   7
## 2 Action                                           24482
## 3 Action|Adventure                                 68688
## 4 Action|Adventure|Animation|Children|Comedy        7467
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy   187
## 6 Action|Adventure|Animation|Children|Comedy|IMAX       66
```

```
# movie distribution
edx %>%
  group_by(movieId) %>%
  summarize(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "white")+
  scale_x_log10()+
  ggtitle("Distribution of Movies")+
  xlab("Number of Ratings")+
  ylab("Number of Movies")+
  theme_economist()
```



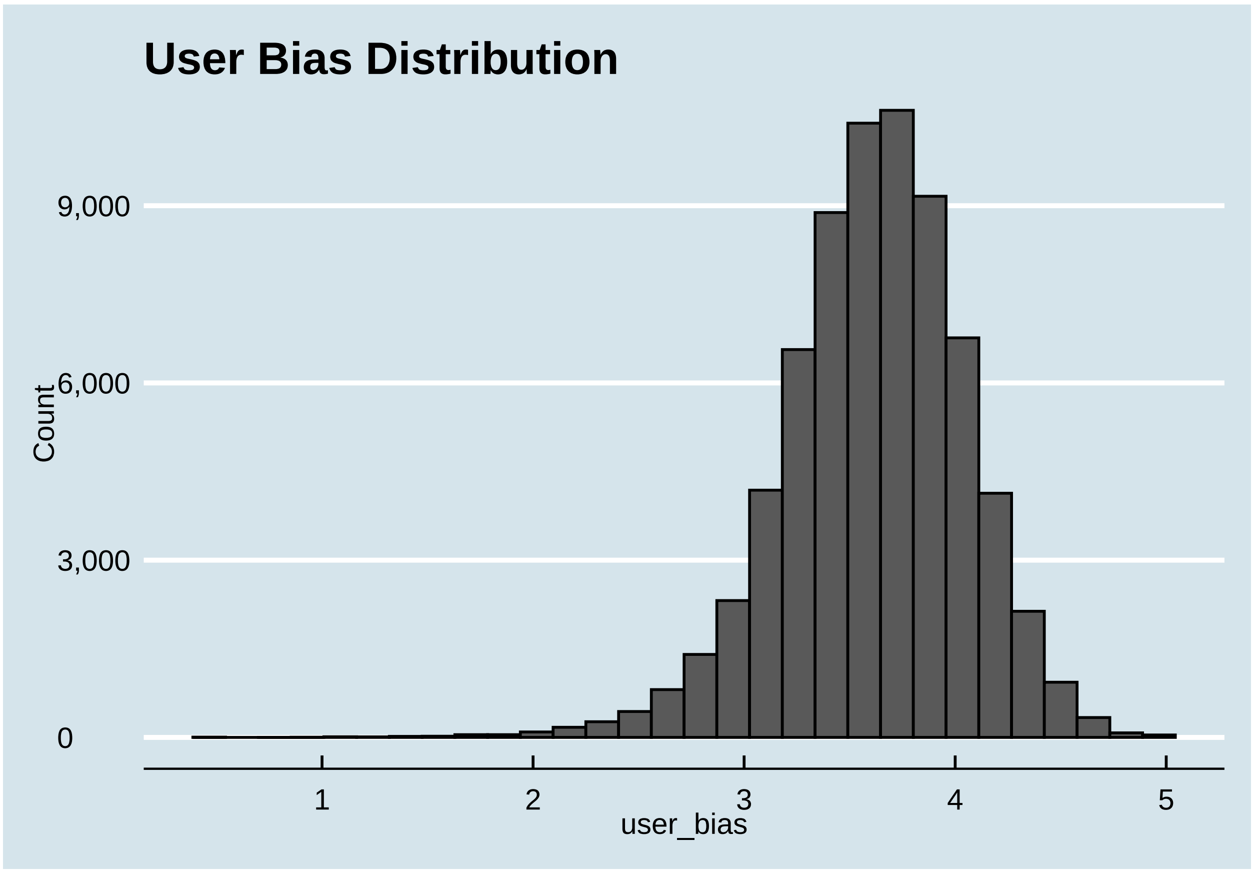
The movie distribution is assumed to be normally distributed since people tend to avoid negative experiences.

```
# user distribution
edx %>%
  group_by(userId) %>%
  summarize(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "white")+
  scale_x_log10()+
  ggtitle("Distribution of Users")+
  xlab("Number of Ratings")+
  ylab("Number of Users") +
  theme_economist()
```



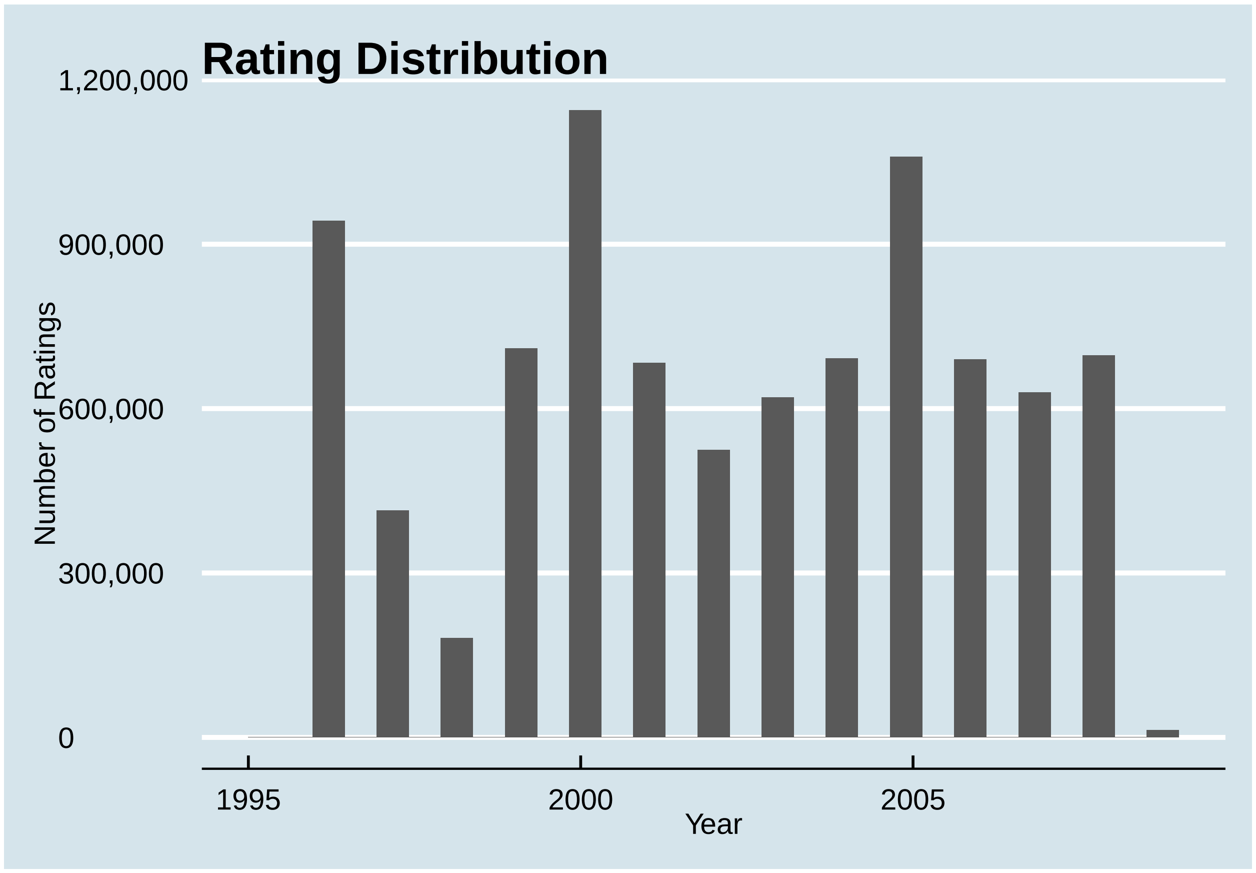
This graph shows some users rating very little and other users rating a very frequently.

```
# User Rating Distribution
edx %>%
  group_by(userId) %>%
  summarize(user_b = mean(rating)) %>%
  filter(n() >= 100) %>%
  ggplot(aes(user_b)) +
  geom_histogram(color = "black") +
  ggtitle("User Bias Distribution")+
  xlab("user_bias")+
  ylab("Count") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```



The ratings are normally distributed among users around 3.5 approximately.

```r
# Year Distribution
edx %>%
  mutate(year = year(as_datetime(timestamp))) %>%
  ggplot(aes(x = year))+
  geom_histogram() +
  ggtitle("Rating Distribution")+
  xlab("Year") +
  ylab("Number of Ratings") +
  scale_y_continuous(labels = comma)+
  theme_economist()
```



Many years float around the average per number of ratings while some years have a spike in ratings and other years drop significantly in ratings.

```
edx %>%
  group_by(rating) %>%
  summarize(count=n())
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```

From this table it can be seen whole numbers are used more often than half numbers and higher ratings are used more than lower ratings. The top five ratings given in order from most to least is 4, 3, 5, 3.5, 2.

```r
set.seed(31, sample.kind = "Rounding")
test_index2 <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-test_index2]
temp2 <- edx[test_index2]

#ensure userId and movieId are in both train and test sets
edx_test <- temp2 %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

removed2 <- anti_join(temp2, edx_test)
edx_train <- rbind(edx_train, removed2)

rm(test_index2, temp2, removed2)
```

```r
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")

library(lubridate)
library(ggplot2)
library(ggthemes)
library(scales)
```

Data Preparation continued Before the analysis can begin, each of the data sets was modified to include a column for the year a movie was released, the date of the rating, the day of the week the rating happened, and the age in days of the user life span of ratings. In order to successfully create the age column a start date column was created. The age column was created from taking the date the rating was made and subtracting it from the start date or the date of the first rating that user made.

```r
# First extracting the year released for each moving (####).
# Then extracting the year from the parentheses as some movies have four digits in their title.
# Finally converting the epoch time found in time stamp column to YYYY-MM-DD

edx2 <- edx %>%
  mutate(year = str_extract((str_extract(edx$title, "\\(\\d{4}\\)$")), "\\d{4}"),
         date = date(as_datetime(edx$timestamp)),
         day_of_week = weekdays(as_datetime((edx$timestamp)))) %>%
  group_by(userId) %>%
  mutate(start_date = min(date)) %>%
  ungroup() %>%
  mutate(age = date - start_date)

validation2 <- validation %>%
  mutate(year = str_extract((str_extract(validation$title, "\\(\\d{4}\\)$")), "\\d{4}"),
         date = date(as_datetime(validation$timestamp)),
         day_of_week = weekdays(as_datetime((validation$timestamp)))) %>%
  group_by(userId) %>%
  mutate(start_date = min(date)) %>%
  ungroup() %>%
  mutate(age = date - start_date)

edx_train2 <- edx_train %>%
  mutate(year = str_extract((str_extract(edx_train$title, "\\(\\d{4}\\)$")), "\\d{4}"),
         date = date(as_datetime(edx_train$timestamp)),
         day_of_week = weekdays(as_datetime((edx_train$timestamp)))) %>%
  group_by(userId) %>%
  mutate(start_date = min(date)) %>%
  ungroup() %>%
  mutate(age = date - start_date)

edx_test2 <- edx_test %>%
  mutate(year = str_extract((str_extract(edx_test$title, "\\(\\d{4}\\)$")), "\\d{4}"),
         date = date(as_datetime(edx_test$timestamp)),
         day_of_week = weekdays(as_datetime((edx_test$timestamp)))) %>%
  group_by(userId) %>%
  mutate(start_date = min(date)) %>%
  ungroup() %>%
  mutate(age = date - start_date)
```

```r
edx_test2 <- edx_test2 %>%
  semi_join(edx_train2, by = "age")

validation2 <- validation2 %>%
  semi_join(edx2, by = "age")

# Error calculated with RMSE (also found in caret package)
RMSE1 <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2)
  )}
```

Analysis

In this section, is how the algorithm is produced by attempting to account for people's biases and other effects.

Simplest Possible Prediction First attempt is to come up with the simplest possible prediction. In this case, it is the mean of all ratings. Guessing the average means a guess will be closer to the actual vs using another number.

```r
# first predict by the average rating
mu <- mean(edx_train2$rating)

# create a table to check progress
result <- tibble(Method = "Avg", RMSE = RMSE(mu, edx_test2$rating))
result
```

```
## # A tibble: 1 x 2
##   Method  RMSE
##   <chr>  <dbl>
## 1 Avg     1.06
```

Account for the Movie Effect Every movie ever produced are not all the same quality. Great movies are seen by many people while independent or low quality movies are often neglected by users. People tend to see what other people are doing and watch what they are watching. This means that independent movies are not seen as much because they do not have the same marketing budget. When it comes to low quality movies, people tend to avoid bad experiences. Also when a friend tells a friend a movie is bad and not to see it, they will often follow this and thus low quality movies are not seen and have lower numbers of ratings.

```r
m_bias <- edx_train2 %>%
  group_by(movieId) %>%
  summarize(m_bias = mean(rating - mu))


m_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  mutate(pred = mu + m_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "Movie bias",
                                   RMSE = RMSE(m_bias_pred, edx_test2$rating)))
result
```

```
## # A tibble: 2 x 2
##   Method      RMSE
##   <chr>      <dbl>
## 1 Avg        1.06
## 2 Movie bias 0.944
```

User Bias Users have their own experiences that make up their set of preferences and attitudes towards ratings. Some users will rate everything with a high rating, while others will rate negatively when there is a strong dislike for the movie.

```r
u_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(u_bias = mean(rating - mu - m_bias))

u_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId") %>%
  mutate(pred = mu + m_bias + u_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "User bias",
                                   RMSE = RMSE(u_bias_pred, edx_test2$rating)))
result
```

```
## # A tibble: 3 x 2
##   Method      RMSE
##   <chr>      <dbl>
## 1 Avg        1.06
## 2 Movie bias 0.944
## 3 User bias  0.866
```

Genre Bias People have their own preferences in genre.

```r
g_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias,  by = "userId") %>%
  group_by(genres) %>%
  summarise(g_bias = mean(rating - mu - m_bias - u_bias))

g_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  mutate(pred = mu + m_bias + u_bias + g_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "Genre bias",
                                   RMSE = RMSE(g_bias_pred, edx_test2$rating)))

result
```

```
## # A tibble: 4 x 2
##   Method      RMSE
##   <chr>      <dbl>
## 1 Avg        1.06
## 2 Movie bias 0.944
## 3 User bias  0.866
## 4 Genre bias 0.865
```

Year Released Technology changes from year to year. This technology improves the overall experience. Year released is used as a proxy for superficial looks of movies.

```r
y_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  group_by(year) %>%
  summarise(y_bias = mean(rating - mu - m_bias - u_bias - g_bias))


y_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year") %>%
  mutate(pred = mu + m_bias + u_bias + g_bias + y_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "Year bias",
                                   RMSE = RMSE(y_bias_pred, edx_test2$rating)))

result
```

```
## # A tibble: 5 x 2
##   Method      RMSE
##   <chr>      <dbl>
## 1 Avg        1.06
## 2 Movie bias 0.944
## 3 User bias  0.866
## 4 Genre bias 0.865
## 5 Year bias  0.865
```

Date Bias The date bias takes into account when the rating happened. Movies can get rated differently depending on when it was watched relative to other movies. It can create a greater contrast and cause a skew in a user's rating.

```r
d_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  group_by(date) %>%
  summarise(d_bias = mean(rating - mu - m_bias - u_bias - g_bias - y_bias))

d_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  left_join(d_bias, by = "date")    %>%
  mutate(pred = mu + m_bias + u_bias + g_bias + y_bias + d_bias) %>%
  pull(pred)


result <- bind_rows(result, tibble(Method = "Date bias",
                                   RMSE = RMSE(d_bias_pred, edx_test2$rating)))
```

Time Bias Time bias is an attempt to take into account that users change over time. Often it happens the more movies one sees the harsher the critic you become.

```r
t_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  left_join(d_bias, by = "date")    %>%
  group_by(age) %>%
  summarise(t_bias = mean(rating - mu - m_bias - u_bias - g_bias - y_bias - d_bias))

t_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  left_join(d_bias, by = "date")    %>%
  left_join(t_bias, by = "age")   %>%
  mutate(pred = mu + m_bias + u_bias + g_bias + y_bias + d_bias + t_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "Time bias",
                                   RMSE = RMSE(t_bias_pred, edx_test2$rating)))
```

Day of the week Each day of the week has different feelings toward it. This is to account for each "feel" for each day.

```
w_bias <- edx_train2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  left_join(d_bias, by = "date")    %>%
  left_join(t_bias, by = "age")   %>%
  group_by(day_of_week) %>%
  summarise(w_bias = mean(rating - mu - m_bias - u_bias - g_bias - y_bias - d_bias - t_bias))

w_bias_pred <- edx_test2 %>%
  left_join(m_bias, by = "movieId") %>%
  left_join(u_bias, by = "userId")  %>%
  left_join(g_bias, by = "genres")  %>%
  left_join(y_bias, by = "year")    %>%
  left_join(d_bias, by = "date")    %>%
  left_join(t_bias, by = "age")   %>%
  left_join(w_bias, by = "day_of_week") %>%
  mutate(pred = mu + m_bias + u_bias + g_bias + y_bias + d_bias + t_bias + w_bias) %>%
  pull(pred)

result <- bind_rows(result, tibble(Method = "Week bias",
                                   RMSE = RMSE(w_bias_pred, edx_test2$rating)))
```

Checking Accuracy Here is an exploration of where the most error is happening. Through this exploration it is found that the most error is occurring with movies who only have a few users rate them. The exploration looks at the ten best and worst movies as by the movie bias.

```
titles <- edx_train2 %>%
  select(movieId, title) %>%
  distinct()

edx_train2 %>%
  count(movieId) %>%
  left_join(titles) %>%
  left_join(m_bias, by ="movieId") %>%
  arrange(-m_bias) %>%
  select(title, m_bias, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	m_bias	n
Hellhounds on My Trail (1999)	1.487564	1
Satan’s Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487564	2
Shadows of Forgotten Ancestors (1964)	1.487564	1
Fighting Elegy (Kenka erejii) (1966)	1.487564	1
Sun Alley (Sonnenallee) (1999)	1.487564	1
Blue Light, The (Das Blaue Licht) (1932)	1.487564	1
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237564	4
Human Condition II, The (Ningen no joken II) (1959)	1.237564	4
Human Condition III, The (Ningen no joken III) (1961)	1.237564	4
Money (Argent, L’) (1983)	1.237564	2

The 10 best movies

```r
m_bias %>%
  inner_join(titles, by = "movieId") %>%
  arrange(-m_bias) %>%
  select(title, m_bias) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	m_bias
Hellhounds on My Trail (1999)	1.487564
Satan’s Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487564
Shadows of Forgotten Ancestors (1964)	1.487564
Fighting Elegy (Kenka erejii) (1966)	1.487564
Sun Alley (Sonnenallee) (1999)	1.487564
Blue Light, The (Das Blaue Licht) (1932)	1.487564
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237564
Human Condition II, The (Ningen no joken II) (1959)	1.237564
Human Condition III, The (Ningen no joken III) (1961)	1.237564
Money (Argent, L’) (1983)	1.237564

The 10 worst movies

```r
m_bias %>%
  inner_join(titles, by = "movieId") %>%
  arrange(m_bias) %>%
  select(title, m_bias) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	m_bias
Besotted (2001)	-3.012436
Hi-Line, The (1999)	-3.012436
Crossover (2006)	-3.012436
Accused (Anklaget) (2005)	-3.012436
Confessions of a Superhero (2007)	-3.012436
War of the Worlds 2: The Next Wave (2008)	-3.012436

title	m_bias
SuperBabies: Baby Geniuses 2 (2004)	-2.718318
Hip Hop Witch, Da (2000)	-2.666282
Disaster Movie (2008)	-2.615884
From Justin to Kelly (2003)	-2.614775

The 10 best and worst are movies that are unknown. Further exploration into potentially why this is, questions the number of ratings these movies have by users.

Number of ratings for the worst.

```r
edx_train2 %>%
  left_join(m_bias, by ="movieId") %>%
  arrange(m_bias) %>%
  group_by(title) %>%
  summarise(n = n()) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	n
'burbs, The (1989)	1232
'night Mother (1986)	177
'Round Midnight (1986)	40
'Til There Was You (1997)	240
"Great Performances" Cats (1998)	2
batteries not included (1987)	397
...All the Marbles (a.k.a. The California Dolls) (1981)	19
...And God Created Woman (Et Dieu... crÃ©a la femme) (1956)	64
...And God Spoke (1993)	19
...And Justice for All (1979)	497

Number of ratings for the best movies.

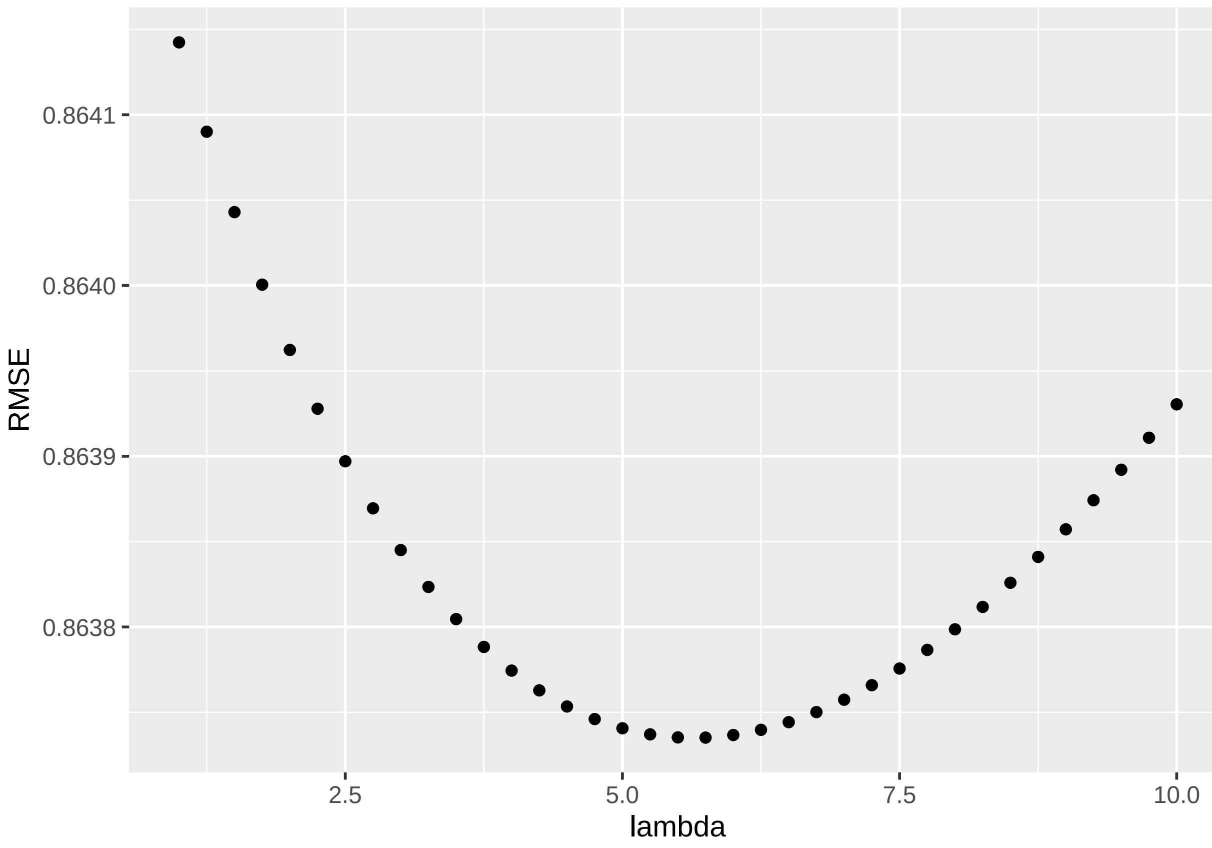
```r
edx_train2 %>%
  left_join(m_bias, by ="movieId") %>%
  arrange(-m_bias) %>%
  group_by(title) %>%
  summarise(n = n()) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	n
'burbs, The (1989)	1232
'night Mother (1986)	177
'Round Midnight (1986)	40
'Til There Was You (1997)	240
"Great Performances" Cats (1998)	2
batteries not included (1987)	397
...All the Marbles (a.k.a. The California Dolls) (1981)	19
...And God Created Woman (Et Dieu... crÃ©a la femme) (1956)	64

title	n
. . . And God Spoke (1993)	19
. . . And Justice for All (1979)	497

The movies have very few ratings. This is cause for implementation of the regularization technique. Where a value is added to penalize the total number of reviews. The term brings the movies with small ratings toward the mean.

```r
lambda_values <- seq(1, 10, 0.25)
regularization <- sapply(lambda_values, function(lambda){

  mean_rating <- mean(edx_train2$rating)

  movie_bias <- edx_train2 %>%
    group_by(movieId) %>%
    summarize(movie_bias = sum(rating - mean_rating)/(n()+lambda))

  user_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    group_by(userId) %>%
    summarize(user_bias = sum(rating - mean_rating - movie_bias)/(n()+lambda))

  genre_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias,  by = "userId") %>%
    group_by(genres) %>%
    summarise(genre_bias = sum(rating - mean_rating - movie_bias - user_bias)/(n()+lambda))

  year_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias,  by = "userId")  %>%
    left_join(genre_bias, by = "genres")  %>%
    group_by(year) %>%
    summarise(year_bias = sum(rating - mean_rating - movie_bias -
                              user_bias - genre_bias)/(n()+lambda))

  date_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias, by = "userId")  %>%
    left_join(genre_bias, by = "genres")  %>%
    left_join(year_bias, by = "year")    %>%
    group_by(date)  %>%
    summarise(date_bias = sum(rating - mean_rating - movie_bias -
                              user_bias - genre_bias - year_bias)/(n()+lambda))

  time_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias, by = "userId")  %>%
    left_join(genre_bias, by = "genres")  %>%
    left_join(year_bias, by = "year")    %>%
    left_join(date_bias, by = "date")    %>%
    group_by(age) %>%
    summarise(time_bias = sum(rating - mean_rating - movie_bias - user_bias -
                              genre_bias - year_bias - date_bias)/(n()+lambda))

  week_bias <- edx_train2 %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias, by = "userId")   %>%
    left_join(genre_bias, by = "genres")  %>%
    left_join(year_bias, by = "year")     %>%
    left_join(date_bias, by = "date")     %>%
    left_join(time_bias, by = "age")    %>%
    group_by(day_of_week)  %>%
    summarise(week_bias = sum(rating - mean_rating - movie_bias - user_bias -
                              genre_bias - year_bias - date_bias - time_bias)/(n()+lambda))

  predict_ratings_edx <- edx_test2 %>%
    left_join(movie_bias, by = "movieId")%>%
    left_join(user_bias,  by = "userId") %>%
    left_join(genre_bias, by = "genres") %>%
    left_join(year_bias,  by = "year")   %>%
    left_join(date_bias,  by = "date")   %>%
    left_join(time_bias, by = "age")    %>%
    left_join(week_bias, by = "day_of_week") %>%
    mutate(pred = mean_rating + movie_bias + user_bias + genre_bias + year_bias + date_bias + time_bias
    pull(pred)

  RMSE(predict_ratings_edx, edx_test2$rating)
})

# Check range if have definite minimum
tibble(lambda = lambda_values, RMSE = regularization) %>%
  ggplot(aes(x = lambda, y = RMSE))+
  geom_point()
```



Regularization

```r
# Keep the lambda with lowest error for final test
lambda_best <- lambda_values[which.min(regularization)]

result <- bind_rows(result, tibble(Method = "Regularization",
                                   RMSE = min(regularization)))

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

Method	RMSE
Avg	1.0602653
Movie bias	0.9441673
User bias	0.8657613
Genre bias	0.8654331
Year bias	0.8652780
Date bias	0.8646833
Time bias	0.8644311
Week bias	0.8644308
Regularization	0.8637352

Checking Accuracy after Regularization Using the best lambda from the regularizing, the error is checked again.

```
reg <- edx_train2 %>%
  group_by(movieId) %>%
  summarize(m_bias = sum(rating - mu)/(n()+lambda_best), n = n())

edx_train2 %>%
  count(movieId) %>%
  left_join(reg) %>%
  left_join(titles, by = "movieId") %>%
  arrange(-m_bias) %>%
  select(title, m_bias, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	m_bias	n
Shawshank Redemption, The (1994)	0.9433182	25187
Godfather, The (1972)	0.9010500	15907
Usual Suspects, The (1995)	0.8514835	19484
Schindler's List (1993)	0.8498025	20825
Rear Window (1954)	0.8086621	7150
Casablanca (1942)	0.8074925	10094
Third Man, The (1949)	0.8000502	2707
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.7936123	2635
Double Indemnity (1944)	0.7929338	1963
Godfather: Part II, The (1974)	0.7910525	10679

Final Regularization This is the final test of the algorithm with the final test set (validation). This includes all biases and effects explored previously as well as Regularization.

```
lambda_best
```

```
## [1] 5.75
```

```
mean_edx <- mean(edx2$rating)

movie_bias <- edx2 %>%
  group_by(movieId) %>%
  summarize(movie_bias = sum(rating - mean_edx)/(n()+lambda_best))

user_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  group_by(userId) %>%
  summarize(user_bias = sum(rating - mean_edx - movie_bias)/(n()+lambda_best))

genre_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias,  by = "userId") %>%
  group_by(genres) %>%
  summarise(genre_bias = sum(rating - mean_edx - movie_bias -
                              user_bias)/(n()+lambda_best))
```

```r
year_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias,  by = "userId")  %>%
  left_join(genre_bias, by = "genres")  %>%
  group_by(year) %>%
  summarise(year_bias = sum(rating - mean_edx - movie_bias -
                            user_bias - genre_bias)/(n()+lambda_best))

date_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId")  %>%
  left_join(genre_bias, by = "genres")  %>%
  left_join(year_bias, by = "year")    %>%
  group_by(date) %>%
  summarise(date_bias = sum(rating - mean_edx - movie_bias -
                            user_bias - genre_bias -
                            year_bias)/(n()+lambda_best))

time_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId")  %>%
  left_join(genre_bias, by = "genres")  %>%
  left_join(year_bias, by = "year")    %>%
  left_join(date_bias, by = "date")    %>%
  group_by(age) %>%
  summarise(time_bias = sum(rating - mean_edx - movie_bias -
                            user_bias - genre_bias -
                            year_bias - date_bias)/(n()+lambda_best))

week_bias <- edx2 %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId")   %>%
  left_join(genre_bias, by = "genres")  %>%
  left_join(year_bias, by = "year")     %>%
  left_join(date_bias, by = "date")     %>%
  left_join(time_bias, by = "age")    %>%
  group_by(day_of_week)  %>%
  summarise(week_bias = sum(rating - mean_edx - movie_bias -
                            user_bias - genre_bias -
                            year_bias - date_bias -
                            time_bias)/(n()+lambda_best))

predict_ratings_edx <- validation2 %>%
  left_join(movie_bias, by = "movieId")%>%
  left_join(user_bias,  by = "userId") %>%
  left_join(genre_bias, by = "genres") %>%
  left_join(year_bias,  by = "year")   %>%
  left_join(date_bias,  by = "date")   %>%
  left_join(time_bias, by = "age")    %>%
  left_join(week_bias, by = "day_of_week") %>%
  mutate(pred = mean_edx + movie_bias + user_bias + genre_bias + year_bias + date_bias + time_bias + we
  pull(pred)
```

```r
result <- bind_rows(result, tibble(Method = "Final Error",
                                   RMSE = RMSE(predict_ratings_edx, validation2$rating)))
result %>% knitr::kable()
```

Method	RMSE
Avg	1.0602653
Movie bias	0.9441673
User bias	0.8657613
Genre bias	0.8654331
Year bias	0.8652780
Date bias	0.8646833
Time bias	0.8644311
Week bias	0.8644308
Regularization	0.8637352
Final Error	0.8634359

```r
# Check Final RMSE meets desired goal
RMSE(predict_ratings_edx, validation2$rating) < 0.86490
```

```
## [1] TRUE
```

Recosystem test The RecoSystem package uses the method of Matrix Factorization to come up with an algorithm to predict ratings.

```r
# first is the train and test sets

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

# Set seed because it is a random function
set.seed(2020, sample.kind = "Rounding")
reco_train_data <- with(edx_train2, data_memory(user_index = userId,
                                               item_index = movieId,
                                               rating = rating))
reco_test_data <- with(edx_test2, data_memory(user_index = userId,
                                             item_index = movieId,
                                             rating = rating))
reco <- Reco()
tune <- reco$tune(reco_train_data)
reco_train <- reco$train(reco_train_data, opts = tune$min)
```

```
## iter      tr_rmse          obj
##    0       0.9747   1.0926e+07
##    1       0.8767   8.9911e+06
##    2       0.8446   8.3645e+06
##    3       0.8238   7.9922e+06
##    4       0.8090   7.7372e+06
##    5       0.7984   7.5589e+06
##    6       0.7902   7.4289e+06
##    7       0.7833   7.3255e+06
##    8       0.7771   7.2369e+06
##    9       0.7717   7.1661e+06
##   10       0.7671   7.1031e+06
##   11       0.7631   7.0528e+06
##   12       0.7595   7.0062e+06
##   13       0.7564   6.9689e+06
##   14       0.7536   6.9324e+06
##   15       0.7511   6.9018e+06
##   16       0.7488   6.8751e+06
##   17       0.7468   6.8500e+06
##   18       0.7450   6.8301e+06
##   19       0.7433   6.8108e+06
```

```
reco_predict <- reco$predict(reco_test_data, out_memory())
reco_rmse <-RMSE(reco_predict, edx_test2$rating)
result <- bind_rows(result, tibble(Method = "RecoSystem",
                                   RMSE = reco_rmse))

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

Method	RMSE
Avg	1.0602653
Movie bias	0.9441673
User bias	0.8657613
Genre bias	0.8654331
Year bias	0.8652780
Date bias	0.8646833
Time bias	0.8644311
Week bias	0.8644308
Regularization	0.8637352
Final Error	0.8634359
RecoSystem	0.7919203

The algorithm performs significantly better than even the regularized bias method.

```
# Uses the orginial, unmodified, test and train set as it does not use any mutated columns
set.seed(2020, sample.kind = "Rounding")
reco_train_data_edx <- with(edx, data_memory(user_index = userId,
                                             item_index = movieId,
                                             rating = rating))
reco_test_data_edx <- with(validation, data_memory(user_index = userId,
                                                   item_index = movieId,
                                                   rating = rating))

reco_edx <- Reco()
tune_edx <- reco_edx$tune(reco_train_data_edx)
reco_edx_train <- reco_edx$train(reco_train_data_edx, opts = tune_edx$min)
```

```
## iter      tr_rmse          obj
##    0       0.9658   1.1914e+07
##    1       0.8739   9.8940e+06
##    2       0.8410   9.1969e+06
##    3       0.8203   8.7951e+06
##    4       0.8063   8.5276e+06
##    5       0.7966   8.3446e+06
##    6       0.7889   8.2112e+06
##    7       0.7822   8.1030e+06
##    8       0.7764   8.0116e+06
##    9       0.7714   7.9383e+06
##   10       0.7671   7.8744e+06
##   11       0.7634   7.8233e+06
##   12       0.7602   7.7759e+06
##   13       0.7574   7.7387e+06
##   14       0.7549   7.7026e+06
##   15       0.7526   7.6722e+06
##   16       0.7506   7.6457e+06
##   17       0.7487   7.6208e+06
##   18       0.7471   7.6009e+06
##   19       0.7455   7.5818e+06
```

```r
reco_predict_edx <- reco_edx$predict(reco_test_data_edx, out_memory())
reco_rmse_edx <- RMSE(reco_predict_edx, validation$rating)
result <- bind_rows(result, tibble(Method = "Final Error: RecoSystem",
                                   RMSE = reco_rmse_edx))

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

Method	RMSE
Avg	1.0602653
Movie bias	0.9441673
User bias	0.8657613
Genre bias	0.8654331
Year bias	0.8652780
Date bias	0.8646833
Time bias	0.8644311
Week bias	0.8644308
Regularization	0.8637352
Final Error	0.8634359
RecoSystem	0.7919203
Final Error: RecoSystem	0.7887847

Results

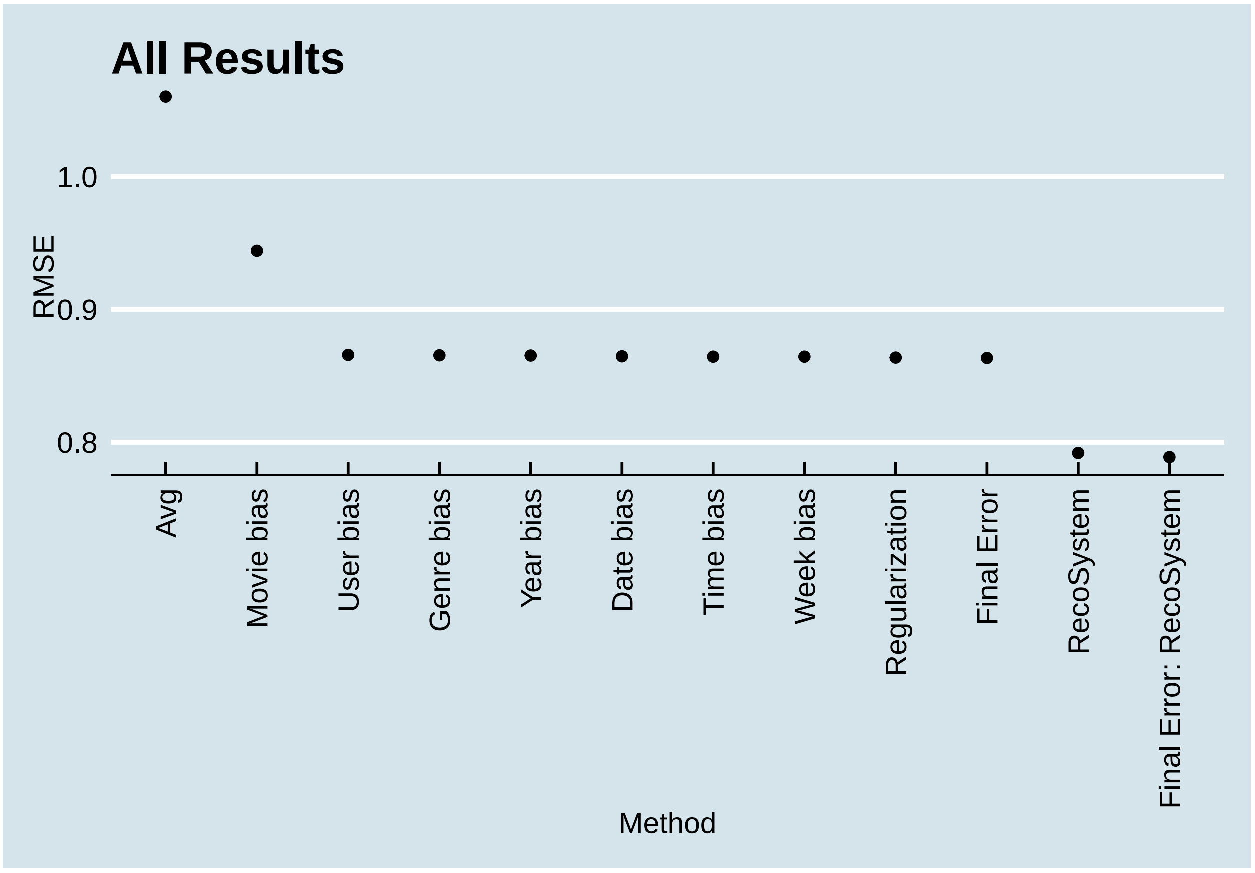
The Final Results are as follows:

```r
result %>% knitr::kable()
```

Method	RMSE
Avg	1.0602653

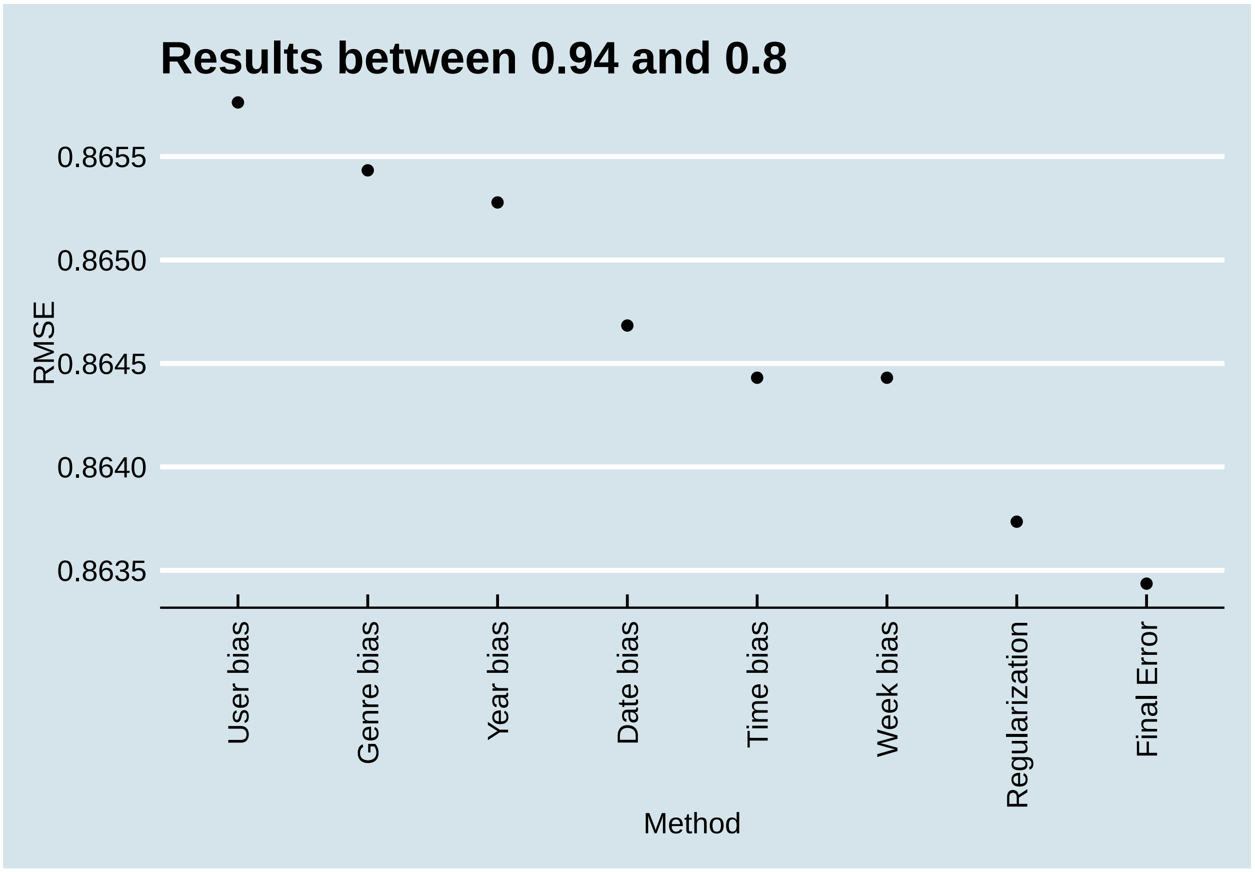
Method	RMSE
Movie bias	0.9441673
User bias	0.8657613
Genre bias	0.8654331
Year bias	0.8652780
Date bias	0.8646833
Time bias	0.8644311
Week bias	0.8644308
Regularization	0.8637352
Final Error	0.8634359
RecoSystem	0.7919203
Final Error: RecoSystem	0.7887847

```r
# Make results a factor to keep their order
result$Method <- factor(result$Method, levels = result$Method)

result %>%
  ggplot(aes(x = Method, y = RMSE)) +
  geom_point() +
  ggtitle("All Results") +
  xlab("Method") +
  ylab("RMSE") +
  theme_economist()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



From looking at the results table and the graph, it is clear movie bias and user bias had the greatest impact on the error score. The difference of 1.06 to 0.865.

```r
# results only for bias accounting
result %>%
  filter(RMSE < 0.94 & RMSE > 0.8) %>%
  ggplot(aes(x = Method, y = RMSE)) +
  geom_point() +
  ggtitle("Results between 0.94 and 0.8") +
  xlab("Method") +
  ylab("RMSE") +
  theme_economist() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Looking at the rest of the graph with out the average and the movie bias by filtering RMSE less than 0.9. It becomes clear which biases make the most difference. In this case date bias, makes the biggest impact. While looking at the day of the week effect makes a negligible difference. When looking at the table, the difference only shows up in the last decimal place. The genre effect came in second for impact in each bias. The last big improvement is made through regularization. Even with regularization the bias accounting method does not come even close to what the recosystem algorithm performance.

Conclusion

The goal of the project was achieved. Considering the two methods, each worked to achieve the desired error value. If creating a recommendation system from scratch it the best return on investment would be to use the recosystem and monitor it over time. The work done for this project was done on a personal computer. This limits the amount of computational resources to use to add in other machine learning algorithms. The recosystem algorithm took approximately two hours to run. Also the bias method computing time increases for each bias that is added. Any work beyond what is done here, would require more computational resources.

Other avenues with the current data set to explore, would be to adjust the time bias to years, rounded to the nearest year or another bin of time that may yield better results. There could also be an exploration to combine matrix factorization with bias accounting as well. Again all of these would require more computational resources beyond the average personal computer.