HarvardX Data Science Professional Certificate PH125.9x Capstone Course Movie Rating Prediction Using a Machine Learning Algorithm

Willie Costa

2020-04-21

Contents

1 Introduction

This project forms a part of the completion requirements for the HarvardX course PH125.9x, which is the capstone course for the Data Science Professional Certificate curriculum. The methods presented herein are based on those outlined in the book *Introduction to Data Science: Data Analysis and Prediction Algorithms with R* by Rafael Irizarry, as summarized in the PH125.8x: Machine Learning course. The dataset used for the present effort was supplied as part of the HarvardX course.

In its simplest form, a machine learning algorithm is one that is capable of making predictions with no *a priori* knowledge of the outcome, with the aim being to process disparate data into an informative and intuitive solution. For the present case, this involves the ability to correctly predict a movie rating within the test set based on the known ratings provided in the training set.

The goal of this project is to develop and train a machine learning algorithm capable of predicting the ratings given by a set of users to a particular set of films. The dataset used for this project is the MovieLens "10M version," which can be considered an open-source, functionally equivalent (and likely far smaller) version of the dataset used by Netflix for the "Netflix Challenge" of the early 2000s. The residual mean squared error (RMSE) is the sole metric by which the accuracy of the prediction algorithm is measured. Per the project charter, the prediction algorithm is graded on a sliding scale, with maximum points given to a prediction algorithm that produces an RMSE of less than 0.86490.

This report presents an overview of the dataset, exploratory data analysis, the creation of the prediction model, and the performance of the model when evaluated against the test set.

1.1 Dataset

The dataset is imported as per the instructions given in the project charter using the following code. The `cache=TRUE` modifier is used so that the code chunk compiles *once*, and subsequent uses of `knit` do not require the import code to execute.

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

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\willi\AppData\Local\Temp\RtmpO2GM9B\downloaded_packages
```

```r
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")

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

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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
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 `edx` object contains all of the training data, and the `validation` object contains all of the test data. As a matter of courtesy, the code used for data import removes unnecessary objects once the training and test sets have been created.

The datasets are modified with the `mutate` command to incorporate the release year of each movie. First, the `timestamp` variable of each dataset is converted to a human-readable release date using the `lubridate` package and January 1, 1970 as the epoch start date. The most important release variable is the year of release, which is used for the subsequent recommendation algorithm. The code to perform these functions is given as follows:

```
# Convert timestamp column into human-readable data for both datasets.
# Release year is most critical.
library(lubridate)
edx <-  mutate(edx,release_date = as.Date(as.POSIXct(edx$timestamp,
          origin="1970-01-01")))
edx <-  mutate(edx,release_year = year(release_date))
validation <-  mutate(validation,release_date = as.Date(as.POSIXct(validation$timestamp,
         origin="1970-01-01")))
validation <-  mutate(validation,release_year = year(release_date))
```

1.2 Data Overview

The data are split 90-10 between training and test sets, respectively:

```
dim(edx)                # 9,000,055 observations of 8 variables
```

```
## [1] 9000055       8
```

```r
dim(validation)          # 999,999 observations of 8 variables
```

```
## [1] 999999      8
```

Note that the original datasets only had six variables, and the `release_date` and `release_year` variables were added after the datasets were imported. An overview of the training set is given below:

```r
head(edx)                # View header of training file
```

```
##   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 release_date release_year
## 1                Comedy|Romance   1996-08-02         1996
## 2         Action|Crime|Thriller   1996-08-02         1996
## 3  Action|Drama|Sci-Fi|Thriller   1996-08-02         1996
## 4       Action|Adventure|Sci-Fi   1996-08-02         1996
## 5 Action|Adventure|Drama|Sci-Fi   1996-08-02         1996
## 6       Children|Comedy|Fantasy   1996-08-02         1996
```

```r
summary(edx)             # Summary statistics
```

```
##      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           release_date         release_year
##  Length:9000055     Length:9000055     Min.   :1995-01-09   Min.   :1995
##  Class :character   Class :character   1st Qu.:2000-01-01   1st Qu.:2000
##  Mode  :character   Mode  :character   Median :2002-10-24   Median :2002
##                                        Mean   :2002-09-21   Mean   :2002
##                                        3rd Qu.:2005-09-15   3rd Qu.:2005
##                                        Max.   :2009-01-05   Max.   :2009
```

```r
mu <- mean(edx$rating)   # Mean rating of all movies across all genres in original dataset
```

The training set is tidy, with no missing values or "N/A" fields. Note that many of the summary statistics are not inherently useful (e.g. median and quantiles for `timestamp`), but the `rating` variable - and, specifically, its mean (`mu`) - is of primary importance for the prediction algorithm.

Examination of the dataset shows that there are a total of 10,677 unique movies, of 797 unique genres, that have been reviewed by 69,878 users:

```r
unique_movies <- edx %>% summarize(n_distinct(movieId))
unique_users <- edx %>% summarize(n_distinct(userId))
unique_genres <- edx %>% summarize(n_distinct(genres))
unique_movies
```

```
##   n_distinct(movieId)
## 1               10677
```

```r
unique_users
```

```
##   n_distinct(userId)
## 1              69878
```

```r
unique_genres
```

```
##   n_distinct(genres)
## 1                797
```

2 Exploratory Data Analysis

One of the fundamental characteristics regarding uncontrolled datasets (viz. those that do not require directed output from users or participants) is that data will be "lumpy." Specifically, there is no guarantee that all movies will receive the same number of reviews, nor that all users will review all movies. In fact, the ten movies with the most reviews are heavily biased toward action and drama; by contrast, the ten movies with the fewest ratings are biased toward foreign, independent, and "art house" films:

```r
# Most ratings
edx %>% group_by(title) %>% summarize(n_ratings=n()) %>% arrange(desc(n_ratings))
```

```
## # A tibble: 10,676 x 2
##    title                                                        n_ratings
##    <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
```

```r
# Fewest ratings
edx %>% group_by(title) %>% summarize(n_ratings=n()) %>% arrange((n_ratings))
```

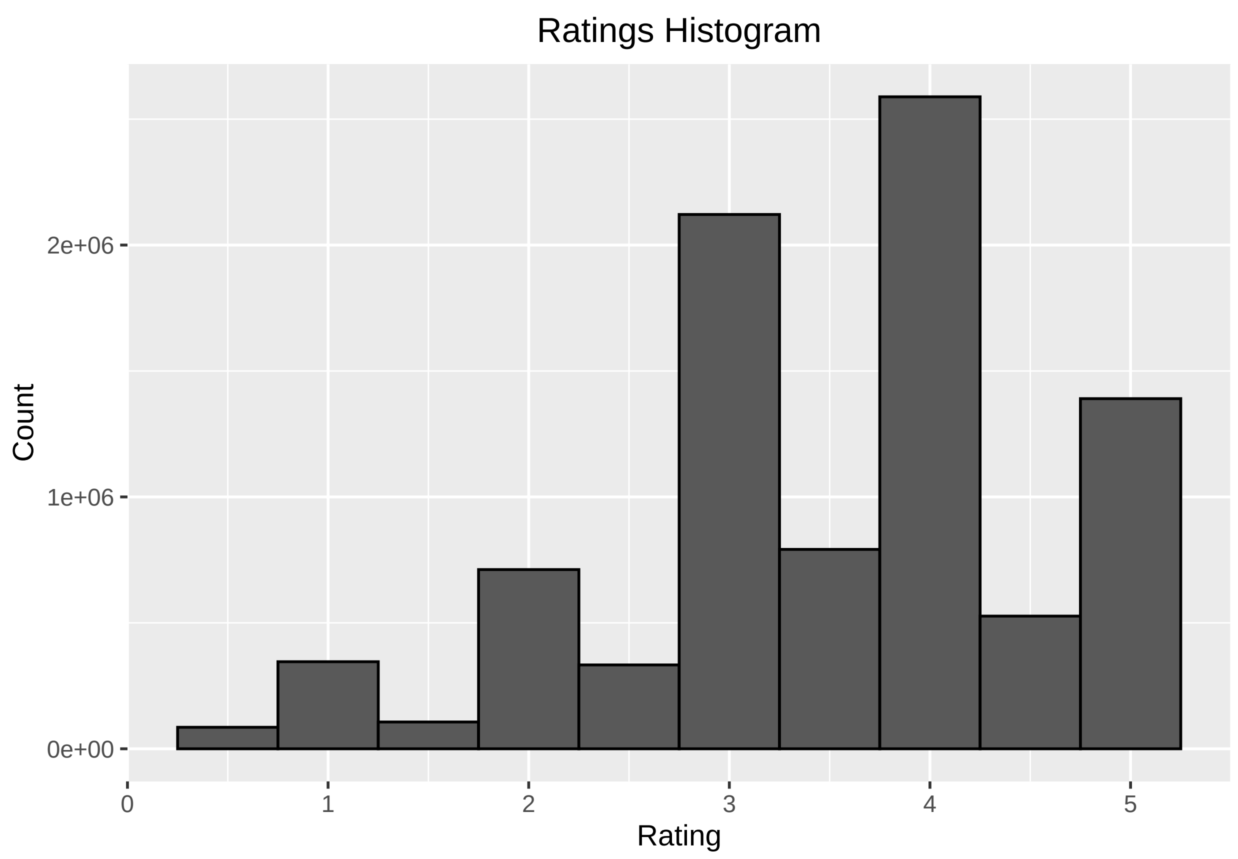
```
## # A tibble: 10,676 x 2
##    title                                                          n_ratings
##    <chr>                                                               <int>
##  1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)                           1
##  2 100 Feet (2008)                                                         1
##  3 4 (2005)                                                                1
##  4 Accused (Anklaget) (2005)                                               1
##  5 Ace of Hearts (2008)                                                    1
##  6 Ace of Hearts, The (1921)                                               1
##  7 Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figl~          1
##  8 Africa addio (1966)                                                     1
##  9 Aleksandra (2007)                                                       1
## 10 Bad Blood (Mauvais sang) (1986)                                         1
## # ... with 10,666 more rows
```

```
# Movies rated only once
edx %>% group_by(title) %>% summarize(n_ratings=n()) %>% filter(n_ratings == 1) %>%
  count() %>% pull()
```

```
## [1] 126
```

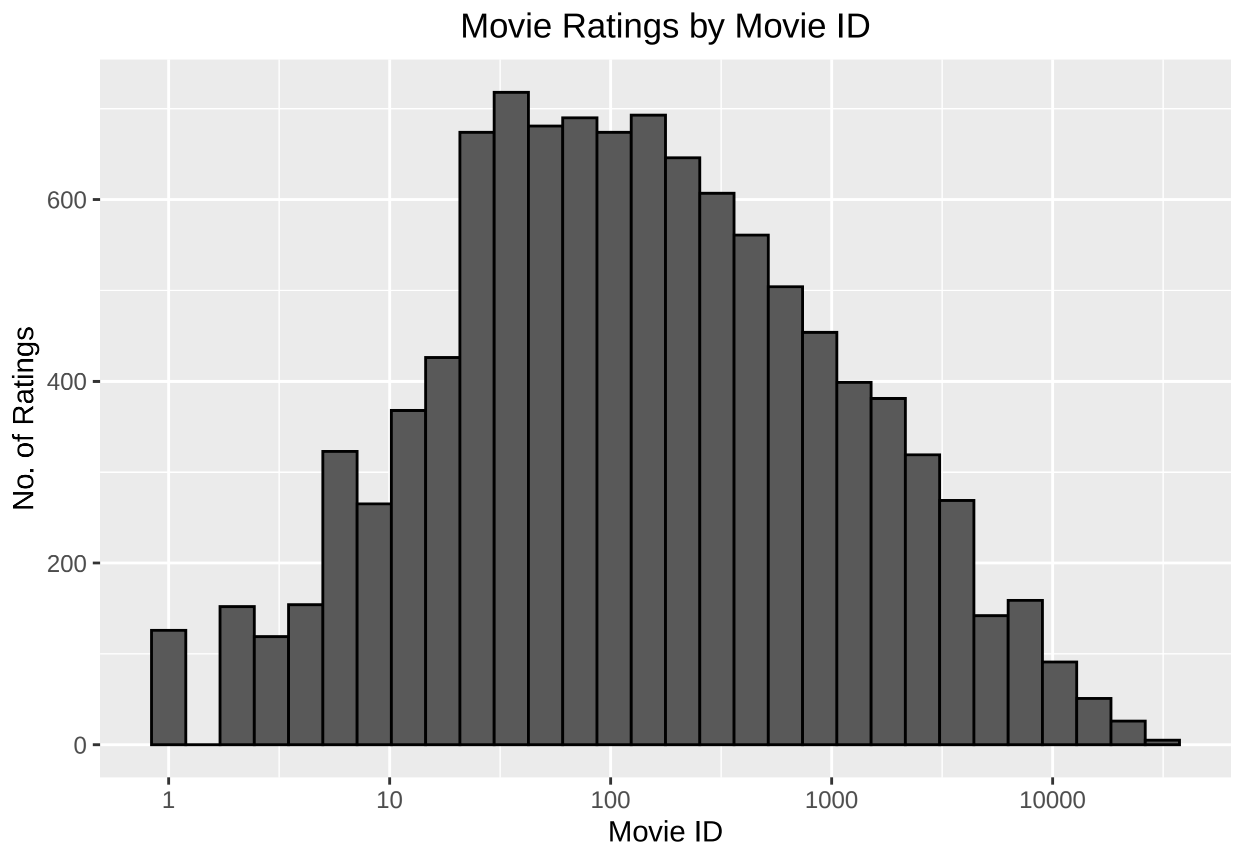
Intuitively, not all movies will receive the same rating from all users, else there would be no need for a prediction algorithm. A histogram of the ratings from the training set shows that ratings are predominantly left-skewed, with a mean near 3.5, which was confirmed previously via `summary(edx)`. It should be noted that users are far more likely to give movies "full" instead of "half" ratings (viz. a 3 instead of a 3.5).

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

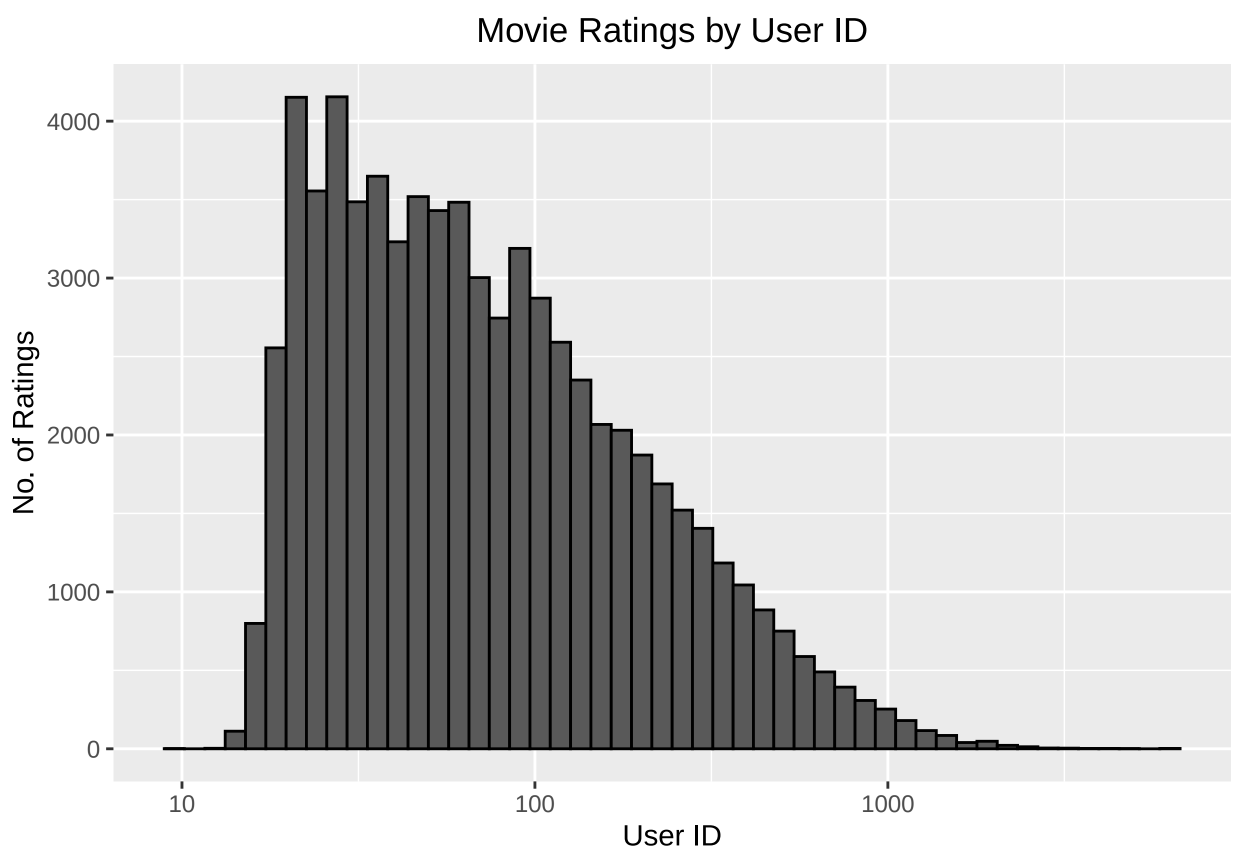


The data also show that not all users are equally active in rating movies, nor did all movies receive the same amount of attention from users (the latter of which was previously demonstrated). The majority of users seem to have rated between 20 and 100 movies, and most movies received between 300-600 ratings.

```
# Histogram of number of ratings by movie
edx %>% dplyr::count(movieId) %>%
  ggplot(aes(n)) + geom_histogram(bins =30, color="black") +
  scale_x_log10() +
  ggtitle("Movie Ratings by Movie ID") +
  xlab("Movie ID") + ylab("No. of Ratings") +
  theme(plot.title = element_text(hjust = 0.5))
```

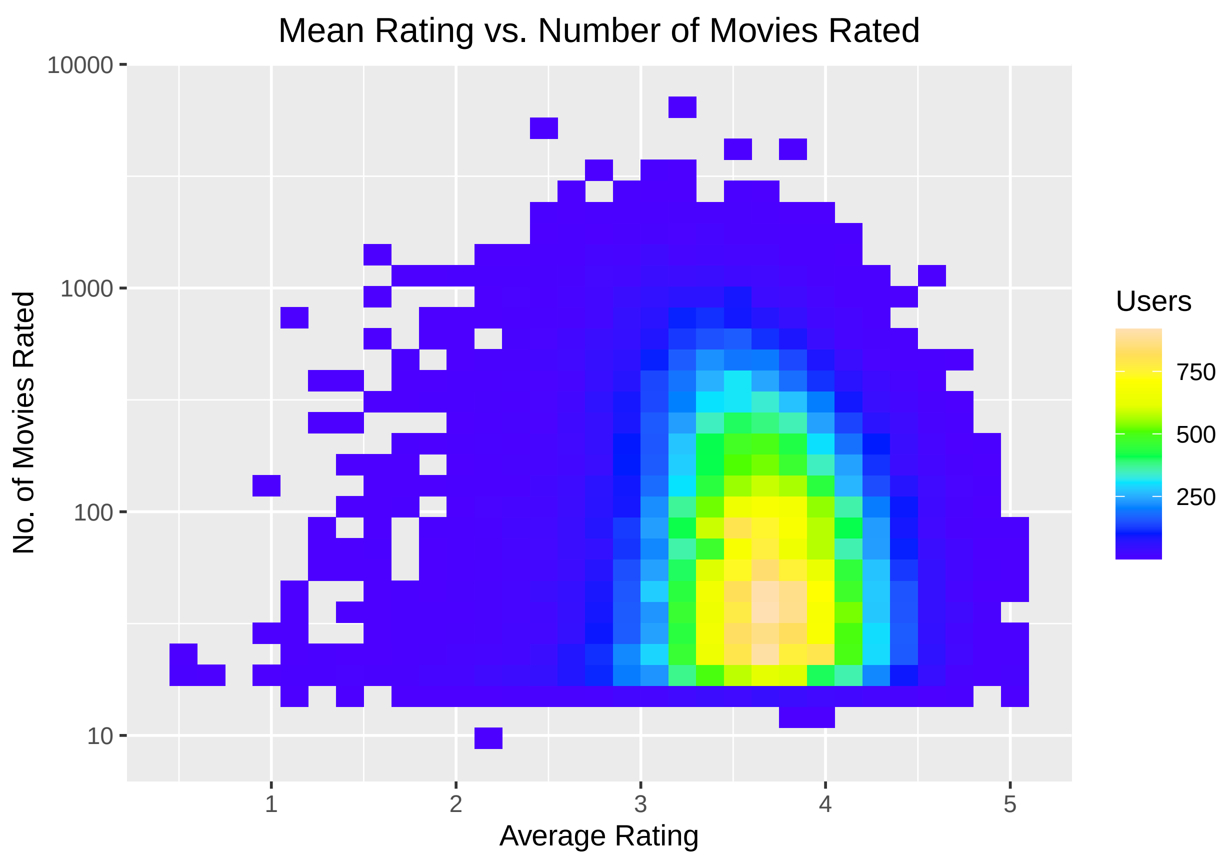


```r
# Histogram of number of ratings by user
edx %>% dplyr::count(userId) %>% ggplot(aes(n)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10() +
  ggtitle("Movie Ratings by User ID") +
  xlab("User ID") + ylab("No. of Ratings") +
  theme(plot.title = element_text(hjust = 0.5))
```



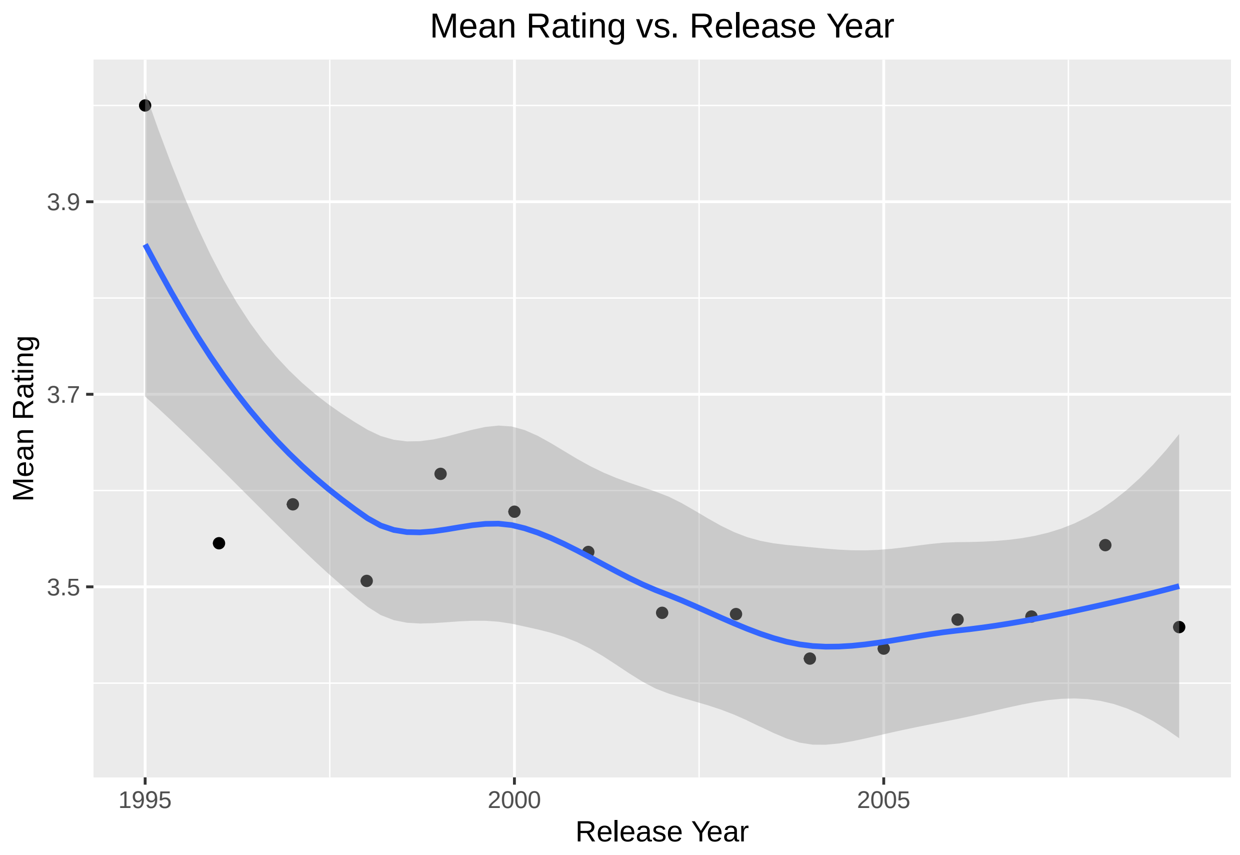
There is a clear correlation between the average rating assigned by a user and the total number of movies that a given user will watch, which is most easily seen via heatmap. Most of the users gave a rating between 3.0 and 4.0 and watched fewer than 100 movies.

```
# Heat map of average rating versus number of movies rated
edx %>% group_by(userId) %>% summarize(mu_user = mean(rating), number = n()) %>%
  ggplot(aes(x = mu_user, y = number)) + geom_bin2d() +
  scale_fill_gradientn(colors = topo.colors(10)) +
  labs(fill="Users") + scale_y_log10() +
  ggtitle("Mean Rating vs. Number of Movies Rated") +
  xlab("Average Rating") + ylab("No. of Movies Rated") +
  theme(plot.title = element_text(hjust = 0.5))
```



Similarly, there is a clear correlation between a movie's release date and the average ratings for all movies released that year:

```
# Mean rating versus release year; geom_smooth() using
# method 'loess' and formula 'y~x'
edx %>% group_by(release_year) %>% summarize(rating = mean(rating)) %>%
  ggplot(aes(release_year, rating)) +
  geom_point() + geom_smooth() +
  ggtitle("Mean Rating vs. Release Year") +
  xlab("Release Year") + ylab("Mean Rating") +
  theme(plot.title = element_text(hjust = 0.5))
```



No specific reasons are given as to why these phenomena exist, but it should be noted that the clustering of ratings for a relatively low number of movies watched will skew the prediction unless it is taken into account. Similarly, the difference in mean rating by release year must be taken into account to prevent skewing of the predictive model. The users in the training set may have elected to rate movies that they thought would be entertaining or had seen before, or perhaps they chose to only review the typical big-budget Hollywood blockbusters. Perhaps the users had a certain affinity for movies produced and released during certain time periods (e.g. "the good old days") or were simply unable to give films from certain time periods due credit; it should also be mentioned that the users were merely *rating* movies, and the datasets provide no way of verifying that the users actually *watched* the movies they were rating. The exact reasons behind the skewed distributions of number of ratings per movie, the user ratings count, or the mean rating by release year are all unknown (and unknowable), but these effects must still be accounted for in the prediction model. These effects are termed, respectively, the "movie effect," the "user effect," and the "release date effect," and are detailed in the following section.

3 Predictive Methodology and Data Analysis

The predictive model must overcome several biases to be of use. As previously mentioned, there exist three primary biases that can skew the prediction algorithm:

- Movie bias: blockbusters, on average, are rated higher and more frequently than foreign and independent films
- User bias: personal preferences can skew ratings regardless of the "inherent quality" of a given movie
- Time bias: user tastes and mindsets may evolve over time based on unknown externalities, complicating the ability of users to remain objective when rating movies

These effects are incorporated as penalties within the prediction algorithm.

As previously noted, the RMSE of the prediction algorithm is the sole performance parameter of interest, and is the mean squared error between the i^{th} prediction for user u, $\hat{y}_{u,i}$, and the actual rating for a given movie by a given user, $y_{u,i}$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2} \tag{3.1}$$

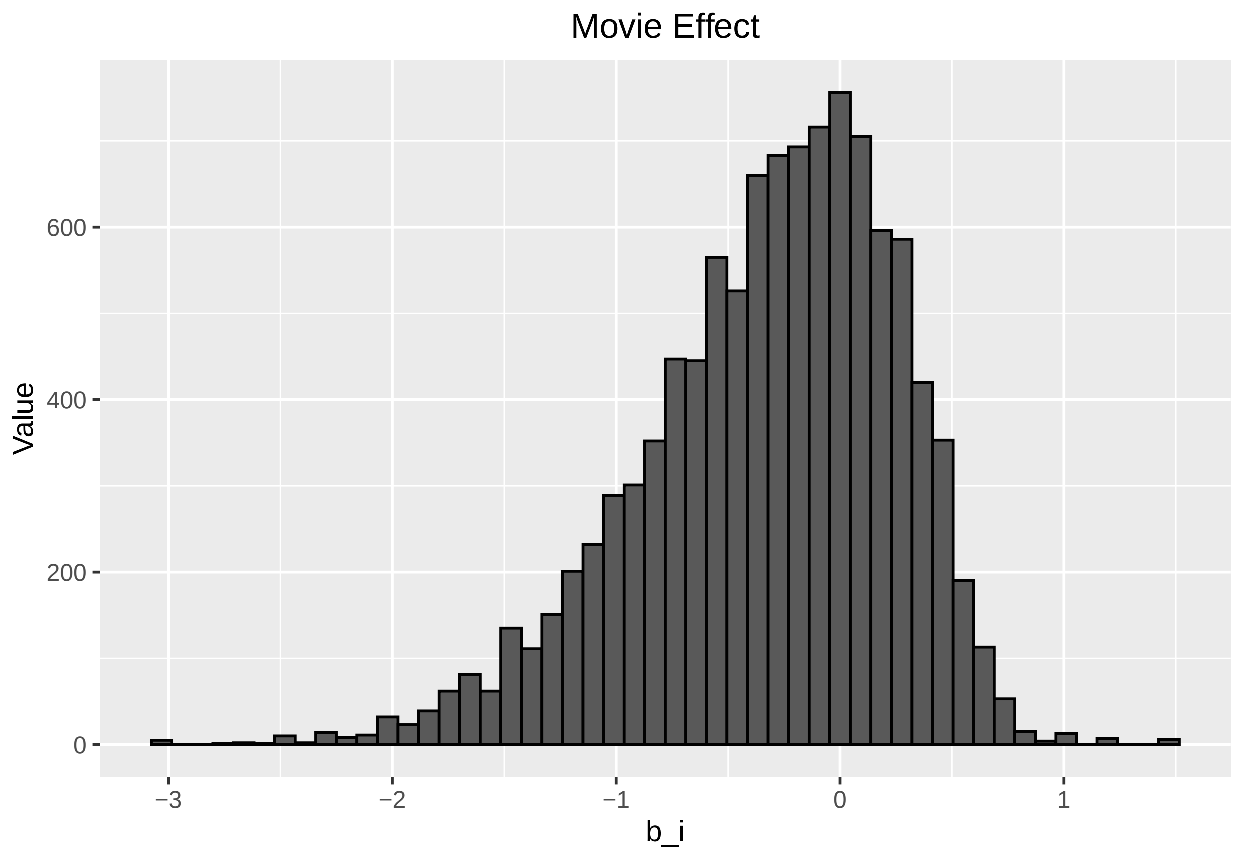
where N is the total number of users. A buildup approach to the prediction algorithm begins with a naïve prediction and increases in complexity (viz. incorporating penalties) as necessary until the target RMSE is reached. The RMSE function is defined as follows:

```r
# Create RMSE loss function
RMSE <- function(pred_rating, true_rating){
  sqrt(mean((pred_rating - true_rating)^2))}
```

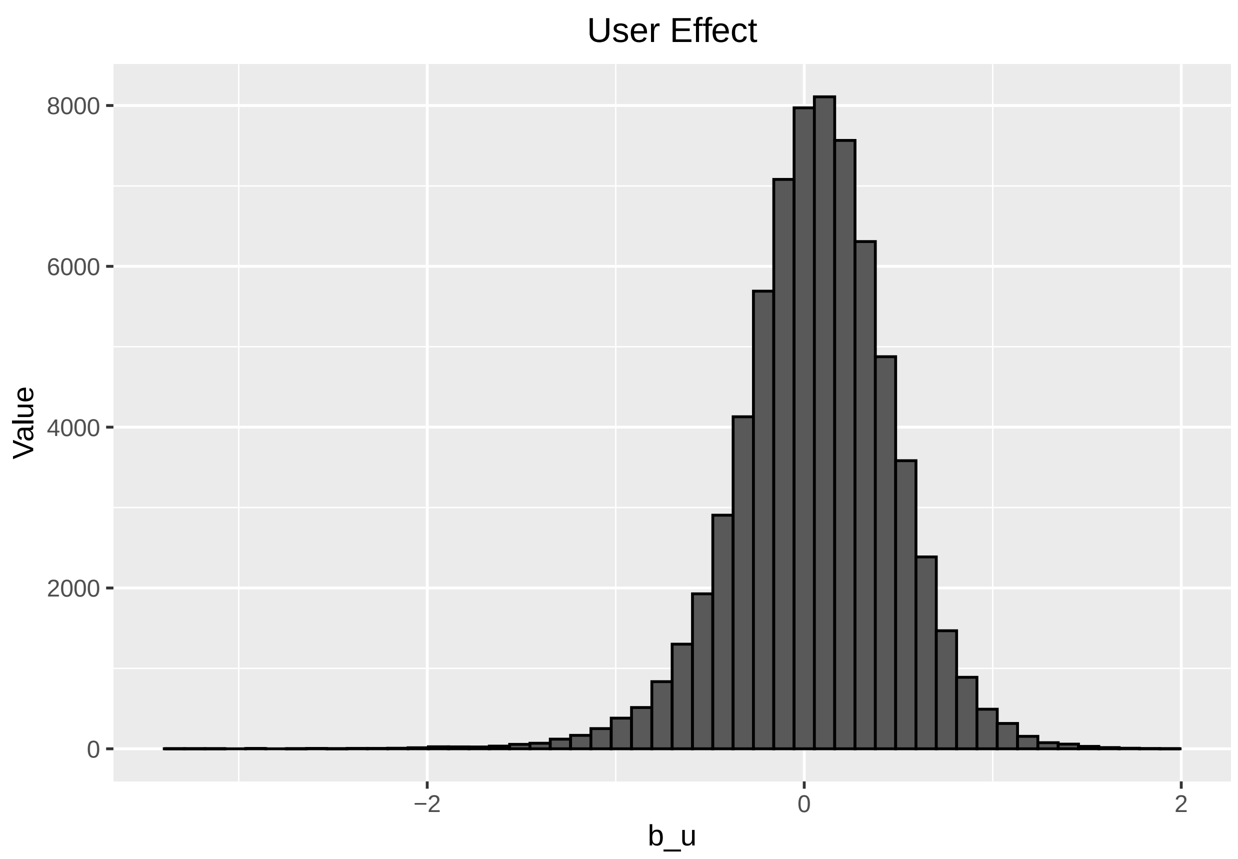
where `true_rating` is the actual rating given to a particular movie in the validation set.

Movie effects were found to be unevenly distributed, with a left skew indicating that some small values are driving the mean rating below the median. Conversely, the user effects were found to be much more centered: thus, although there exists a disproportionate number of ratings given per total movies rated, the b_u "grumpy user" penalty is small. Histograms of the movie and user effects are given as follows:

```r
# There will be movie effects (b_i, some movies are higher rated than others) and
# user effects (b_u, for 'cranky users'). Must account for these in RMSE model.
movie_effect <- edx %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
movie_effect %>% qplot(b_i, geom ="histogram", bins = 50, data = ., color = I("black")) +
  ggtitle("Movie Effect") +
  xlab("b_i") + ylab("Value") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
user_effect <- edx %>% left_join(movie_effect, by='movieId') %>% group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_effect %>% qplot(b_u, geom ="histogram", bins = 50, data = ., color = I("black")) +
  ggtitle("User Effect") +
  xlab("b_u") + ylab("Value") +
  theme(plot.title = element_text(hjust = 0.5))
```



3.1 Prediction Method 1: Naïve Prediction by Mean

The simplest prediction model is one that assumes that a constant rating is given by all users across all movies, with any differences between predicted and actual ratings due solely to random error. For this model, all predictions for all movies, $Y_{u,i}$, are assumed to be equal to the mean movie rating, μ:

$$Y_{u,i} = \mu + \epsilon_{u,i} \tag{3.2}$$

The mean movie rating is therefore taken to be the "true" rating for all movies. The RMSE for this method is calculated as:

```r
# Prediction method 1: mean rating (naive prediction)
rmse_naive <- RMSE(mu, validation$rating)
rmse_naive <- format(round(rmse_naive,6), nsmall = 6)

# Create tibble to store results
rmse_results = tibble(Method = "Naive analysis", RMSE = rmse_naive)
```

The resulting RMSE using this method is 1.061202, which is clearly an insufficient level of performance.

3.2 Prediction Method 2: Incorporation of Movie Effects

The incorporation of the movie effects penalty accounts for the inherent bias that big-budget blockbusters appear more likely to attract a higher number of ratings than other types of movies. This model incorporates a bias term, b_i, for each movie based on the mean rating for that movie and the overall mean rating for all movies:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i} \tag{3.3}$$

The movie bias is incorporated into the predictor as follows:

```r
library(tibble)
# Prediction method 2: incorporate movie effects
pred_rating <- mu + validation %>%
  left_join(movie_effect,by="movieId") %>% pull(b_i)
rmse_b_i <- RMSE(pred_rating,validation$rating)      # 0.9439087
rmse_b_i <- format(round(rmse_b_i,6), nsmall = 6)
rmse_results <- rbind(rmse_results, tibble(Method="Movie effects model", RMSE = rmse_b_i))
```

The resulting RMSE using this method is 0.943909, an improvement of 11.05% over the naïve predictor.

3.3 Prediction Method 3: Incorporation of Movie and User Effects

The exploratory analyses showed that there is clear user bias in the dataset, not only in the number of movies rated by each user but also in the average rating given by each user. The accuracy of the prediction model can be enhanced by incorporating a user bias penalty, b_u, into the prediction:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i} \tag{3.4}$$

The predictor is then updated as follows:

```r
# Prediction method 3: incorporate movie effects and user effects
pred_rating2 <- validation %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>% mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
rmse_b_u <- RMSE(pred_rating2, validation$rating)
rmse_b_u <- format(round(rmse_b_u,6), nsmall = 6)
rmse_results <- rbind(rmse_results, tibble(Method="User effects model", RMSE = rmse_b_u))
```

The resulting RMSE using this method is 0.865349, an improvement of 8.32% over the movie effects model; however, this is still far short of the target accuracy, and therefore a different approach is necessary.

3.4 Prediction Method 4: Regularization

From the exploratory analysis, it is obvious that rating activity is not evenly distributed: some users rated very few movies (fewer than 50), and some movies are rated very few times - several, in fact, were only rated once. The RMSE method is sensitive to large errors, of precisely the type caused by this sort of uneven distribution; therefore, these results only serve to increase the noise of the model and should be discarded.

Matrix regularization constrains the total variability of the effect sizes by penalizing large outcomes based on small sample sizes. For example, if $b_i >> 0$ this would be an indication of fewer users rating the i^{th} movie, and the effect of this coefficient should be minimized. This is done via the parameter λ:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u,i} (Y_{u,i} - \hat{\mu}) \tag{3.5}$$

where n_i is the number of ratings, b, for movie i. The function to be minimized is

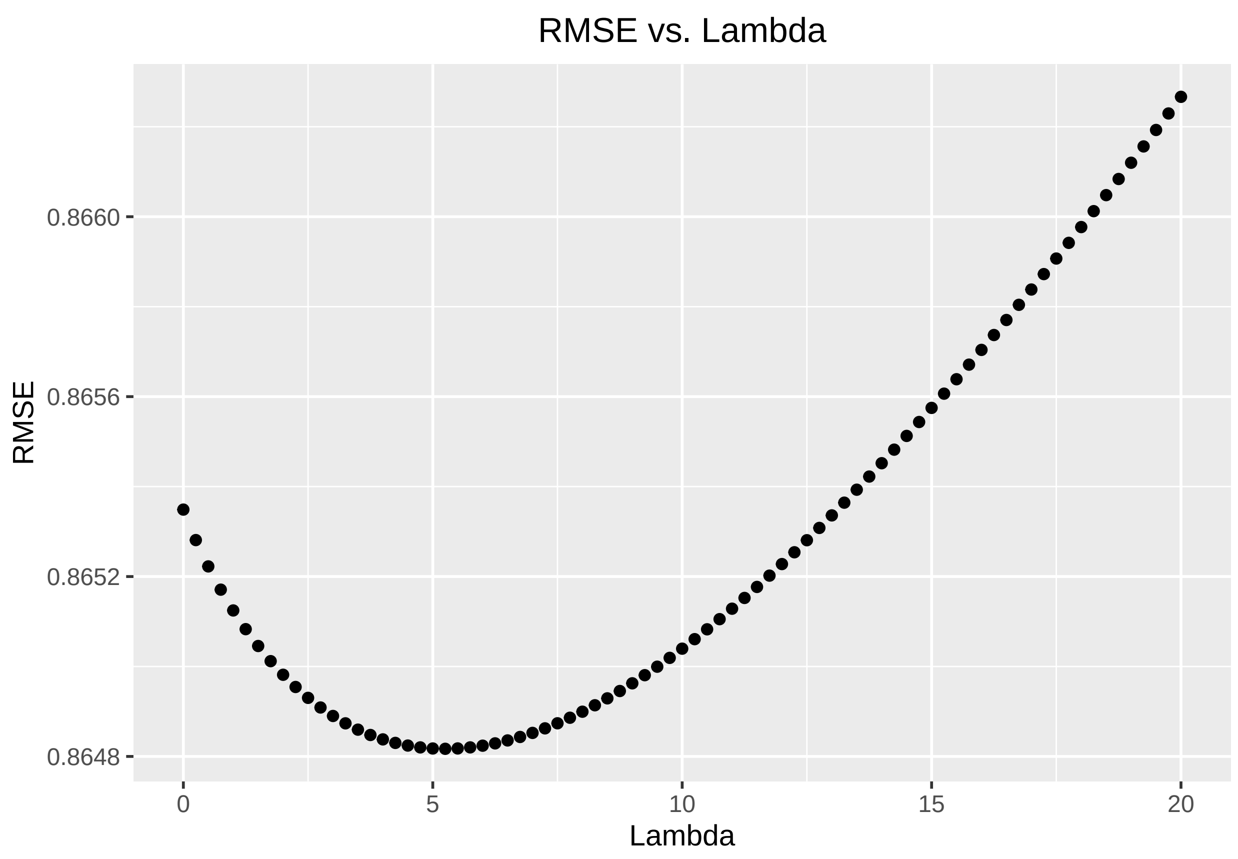
$$f(\lambda) = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda\sum_i b_i^2 \tag{3.6}$$

and, similarly, for incorporating user, (b_u), and release year, b_y, effects,

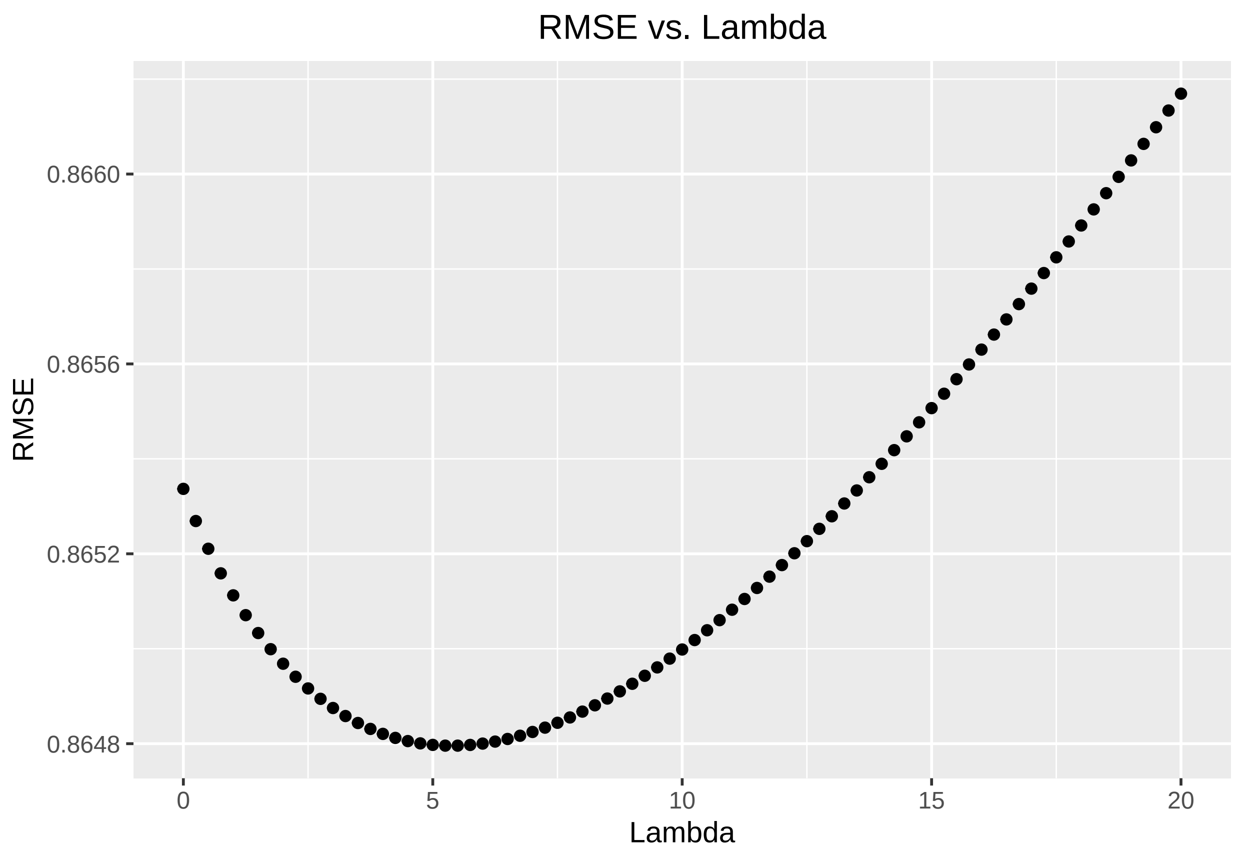
$$f(\lambda) = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u - b_y)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_y b_y^2\right) \tag{3.7}$$

Note that λ is a tuning parameter: the larger the value of λ, the more each effect shrinks. Cross-validation of the training set is therefore used to select λ. Regularization is incorporated into the RMSE calculation, incorporating movie, user, and release year effects, as follows:

```
# Regularization - because the previous results were terrible.
lambdas <- seq(0, 20, 0.25)
rmses <- sapply(lambdas, function(l){
  b_i <- edx %>%                                  # Add movie effect
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() +l))

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

  pred_rating <- validation %>%                   # Prediction
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>% pull(pred)

  return(RMSE(pred_rating, validation$rating))})

rmse_reg <- min(rmses)  # 0.864817
rmse_reg <- format(round(rmse_reg,6), nsmall = 6)


# Plot RMSE against Lambdas to find optimal lambda
qplot(lambdas, rmses) +
  ggtitle("RMSE vs. Lambda") +
  xlab(paste("Lambda")) + ylab("RMSE") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
lambda <- lambdas[which.min(rmses)]    # 5.25

rmse_results <- rbind(rmse_results, tibble(Method="Regularization", RMSE = rmse_reg))


# Include regularization plus release year
rmses <- sapply(lambdas, function(l){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() +l))

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

  b_y <- edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(release_year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+1), n_y = n())

  pred_rating <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_y, by = "release_year") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>% pull(pred)

  return(RMSE(pred_rating, validation$rating))})

rmse_reg <- min(rmses)
rmse_reg <- format(round(rmse_reg,6), nsmall = 6)


# Plot RMSE against Lambdas to find optimal lambda
qplot(lambdas, rmses) +
  ggtitle("RMSE vs. Lambda") +
  xlab(paste("Lambda")) + ylab("RMSE") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
lambda <- lambdas[which.min(rmses)]    # 5.50

rmse_results <- rbind(rmse_results,
                      tibble(Method="Regularization plus release year", RMSE = rmse_reg))
```

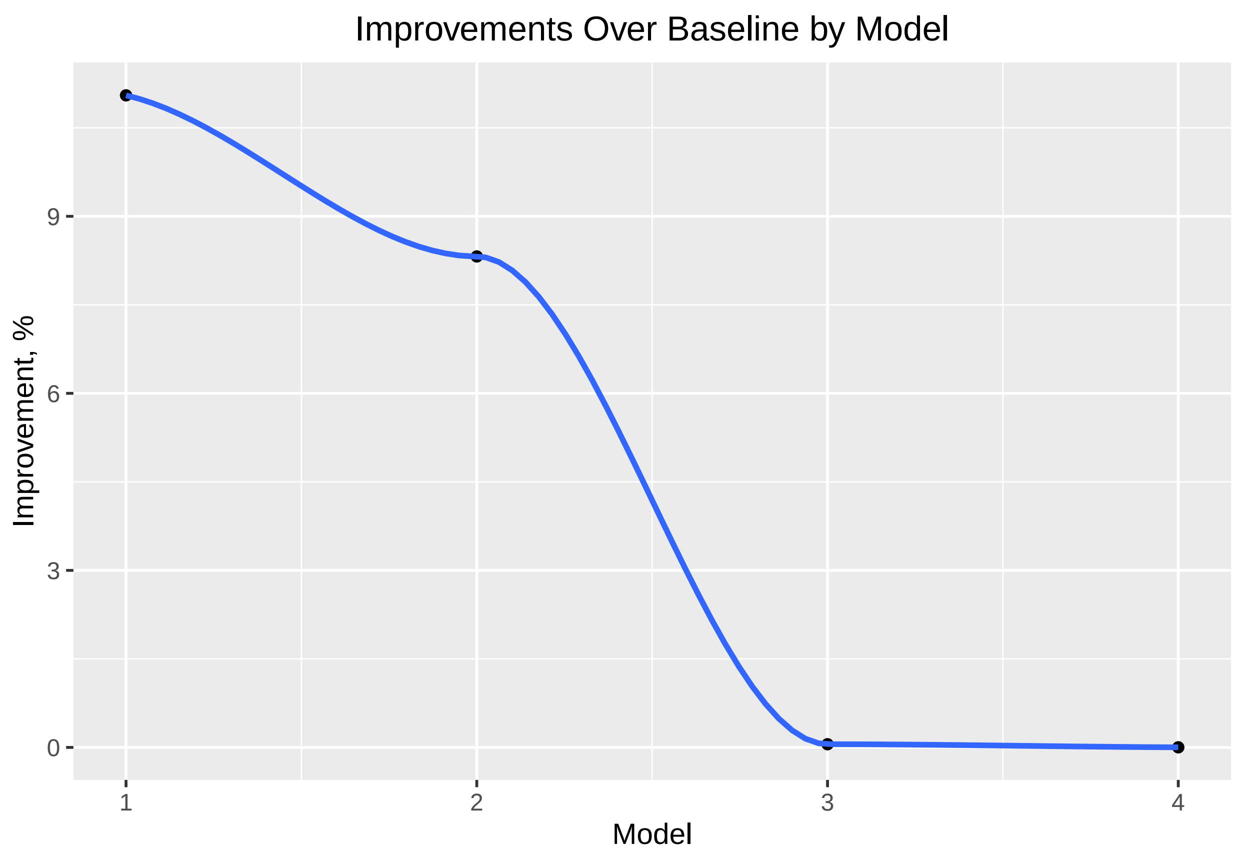
Note that the value of λ does not significantly change between the method that incorporates the release year effect and the method that does not (5.5 versus 5.25, respectively). RMSE results of the prediction model based on method are given as follows:

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

Method	RMSE
Naive analysis	1.061202
Movie effects model	0.943909
User effects model	0.865349
Regularization	0.864817
Regularization plus release year	0.864796

It must be noted that although each subsequent iteration of the prediction model improved the RMSE of the recommendations, the effect is highly nonlinear. While the incorporation of movie effects improved RMSE by over 11%, the first regularization model (not incorporating release year effects) improved on the combined movie-user effects model by only 0.0541%. The final regularization model, incorporating release year effects, only offered a further improvement of 0.002%. While this is a remarkable (18.51%) improvement over the naïve baseline, each subsequent refinement of model complexity resulted in a diminishing improvement over the previous iteration of the model:

```r
# Graph improvements with complexity
improvements <- c(11.05, 8.32, 0.0541, 0.002) # As percentages, not decimals
models <- c(1, 2, 3, 4)
qplot(models,improvements, geom = c("point","smooth")) +
  ggtitle("Improvements Over Baseline by Model") +
  xlab("Model") + ylab("Improvement, %") +
  theme(plot.title = element_text(hjust = 0.5))
```



This result has practical implications: while there are almost 800 unique genres identified in the movie set, a large majority of these are combinations of multiple genres (e.g. *Outbreak* is listed as **Action|Drama|Sci-Fi|Thriller**). Some of this is explainable by the fact that "pure-play" genre films tend to be more profitable, but the reality is that genres "[are not] defined in a rigid way [1]." A further refinement of the prediction model would require that all films within the `edx` dataset be separated into their "component genres" before repeating the regularization process. Unfortunately, this was physically impossible on the author's computer regardless of adjustments to virtual memory and tuning of system performance, and is left as a future work item; however, given the performance of the final prediction model, it is unlikely that the genre separation approach would yield significant reductions in RMSE at any rate.

The final prediction model is

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda}, + b_{y,n,\lambda} + \epsilon_{u,i} \tag{3.8}$$

and the lowest value of RMSE attained - 0.864796 - exceeds the maximum performance requirement specified by the project charter.

[1] Thompson, K.; Bordwell, D., *Film Art: An Introduction*, New York: McGraw-Hill, 2012.

4 Conclusions and Recommendations for Future Work

A machine learning movie recommendation algorithm was developed from the MovieLens 10M dataset with a 90-10 split between training and test data, respectively. Through the use of regularization and accounting for movie, user, and release year effects, the final prediction model is

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda}, + b_{y,n,\lambda} + \epsilon_{u,i} \tag{4.1}$$

and was found to perform to an RMSE of 0.864796, which is a higher accuracy than mandated in the project charter. The algorithm developed herein is therefore adequate for its intended use.

As a future work item, it is recommended that further investigation of RMSE be performed by splitting the entries in the dataset by genre, as a large proportion of the films were labeled as combined genres (e.g. **Comedy|Romance**) instead of as pure-play genres. System limitations prevented this from being performed for the present work, and its effects on RMSE are therefore unknown.