movielens_capstone_final

Irene Siemen

5/20/2020

Introduction:

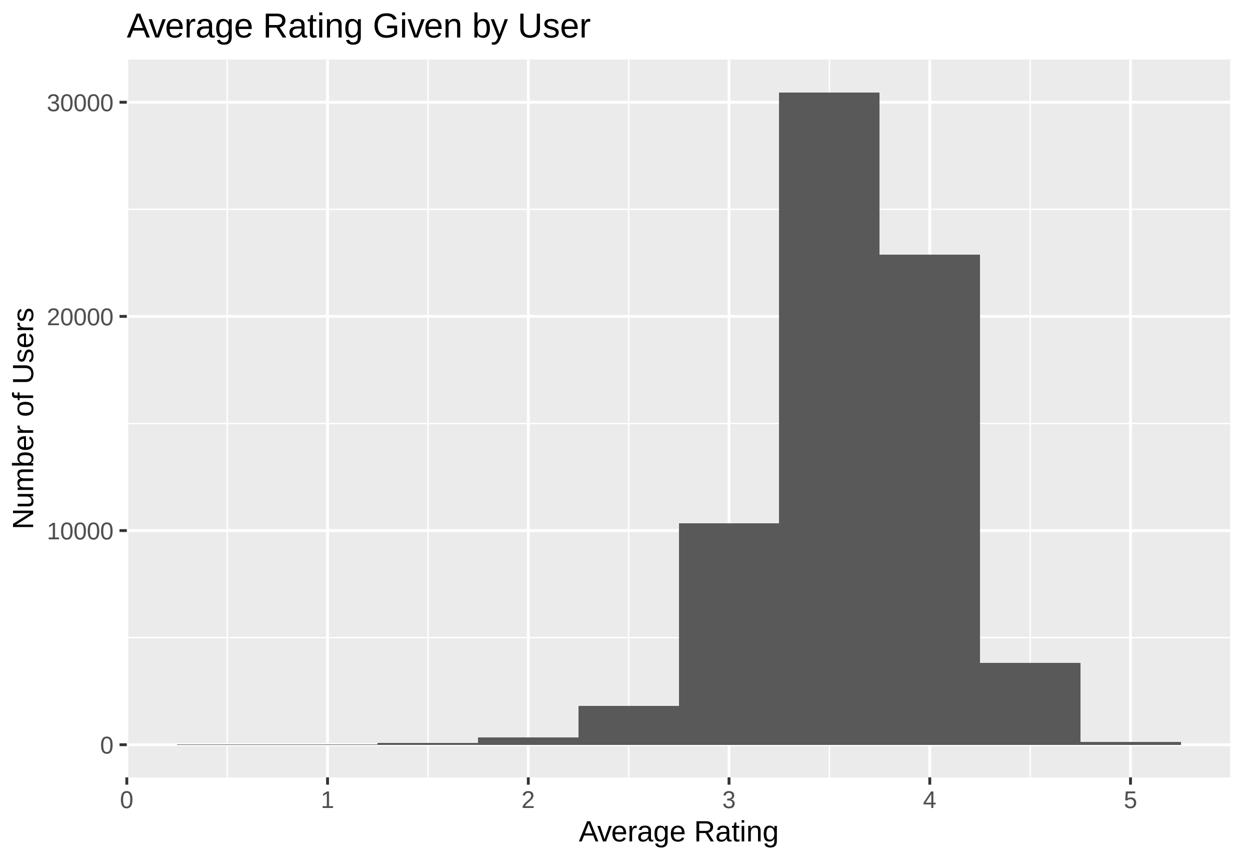
The MovieLens dataset contains thousands of movie ratings that were given by specific users. The dataset includes the user ID, movie ID, movie title, rating on scale of 0 to 5, movie genre, year and the timestamp that review was created. The goal of this project was to predict the rating that a user will give a movie. The predictions were generated by creating and training a machine learning algorithm using a subset of the data, then creating predictions on the validation data set. The success of the machine learning algorithm was evaluated using the root mean square error (RMSE).

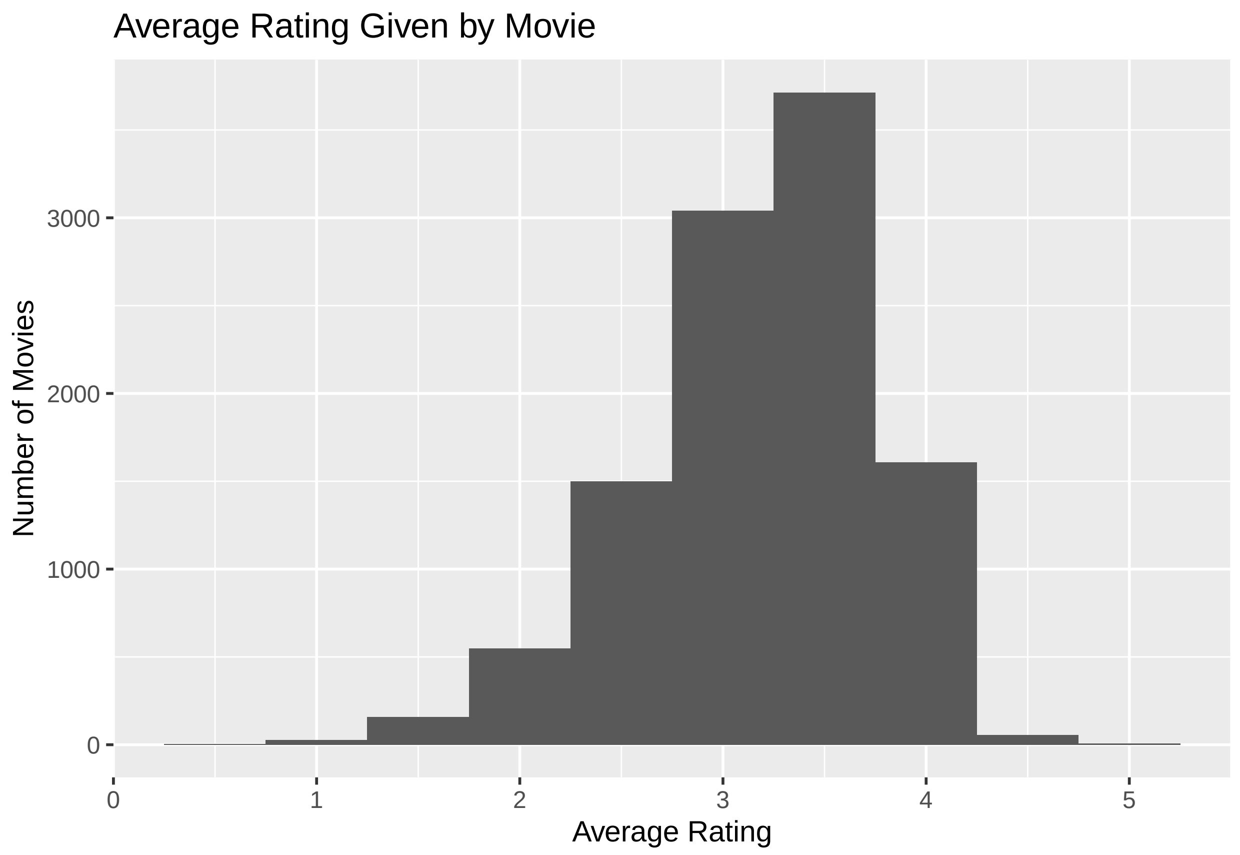
The first step was to complete exploratory data analysis (EDA) on the training data set to review the data. The EDA included reviewing the characteristics of the variables, checking for NaNs, performing filters and counts, and graphing the relationships between variables. The EDA revealed trends in the data, such as some users prefer specific genres, and these trends were used to establish baseline bias values. Once the genres were split and the bias were calculated, then a random forest model was created to determine the components that most affect the rating. The fitted random forest model was used to create predictions on the training and test data and returned a desirable RMSE value. Finally, the random forest model was used to create predictions for the validation data set and calculate the final RMSE value.

Methods and Analysis:

Data Wrangling The first step was to import the data, then partition out test and validation data sets from the training data. Then, some basic data wrangling was completed on the training data. This included extracting the movie's release year from the title column, converting the timestamp variable to a date format and extracting the year, and removing any NaN values.

EDA The next step was to complete exploratory data analysis (EDA) on the training data set to review the data. The EDA included reviewing the characteristics of the variables, checking for NaNs, performing filters and counts, and graphing the relationships between variables. The EDA revealed trends in the data, such as some movies are higher rated than others, and these trends were used to establish baseline bias values. The first graphs created were two histograms that show the distribution of ratings; one histogram is grouped by users and the other histogram is grouped by movies. THe distribution is similiar between the two groupings.

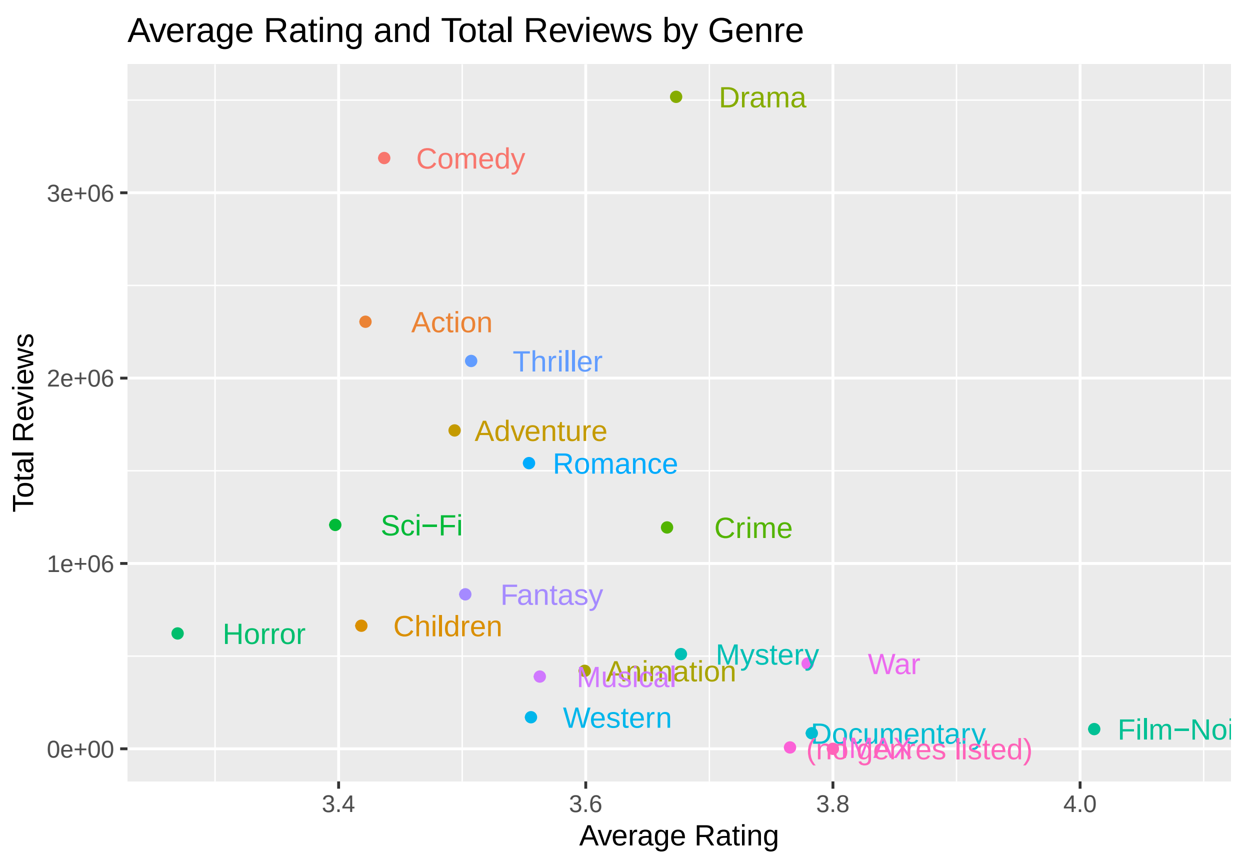

Average Rating Given by User
Number of Users
30000
20000
10000
0
0
1
2
3
4
5
Average Rating



The next step of the EDA was to see if genre appeared to affect ratings. Each movie could be associated with one or more genres. A list of distinct individual genres was created with a summary of count and average. Then, a scatterplot showed the average rating of all movies that listed each genre. It appeared that genre marginally affected the movie's rating, although difference in average rating decrees as the number of ratings increases. The genre column was separated into individual binary columns for each specific genre to allow for easier computations.

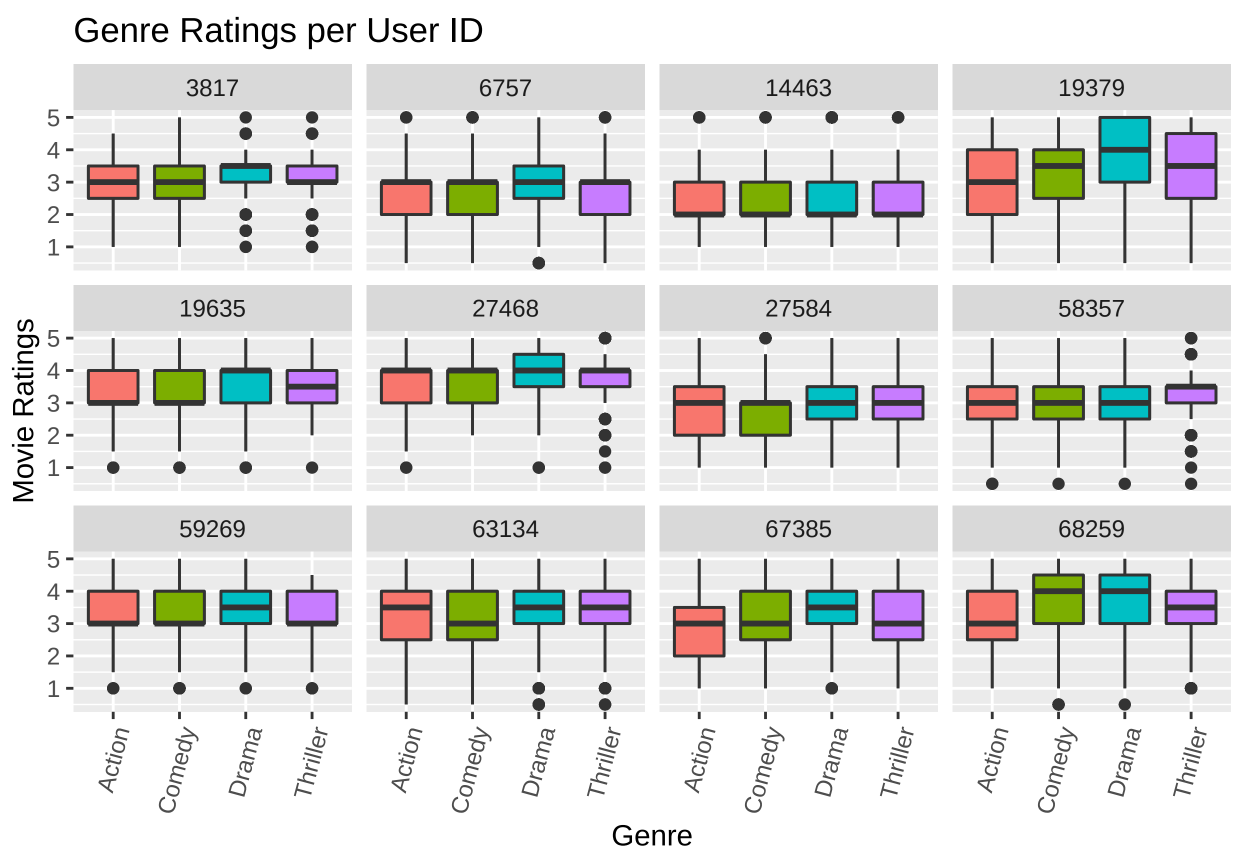
```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



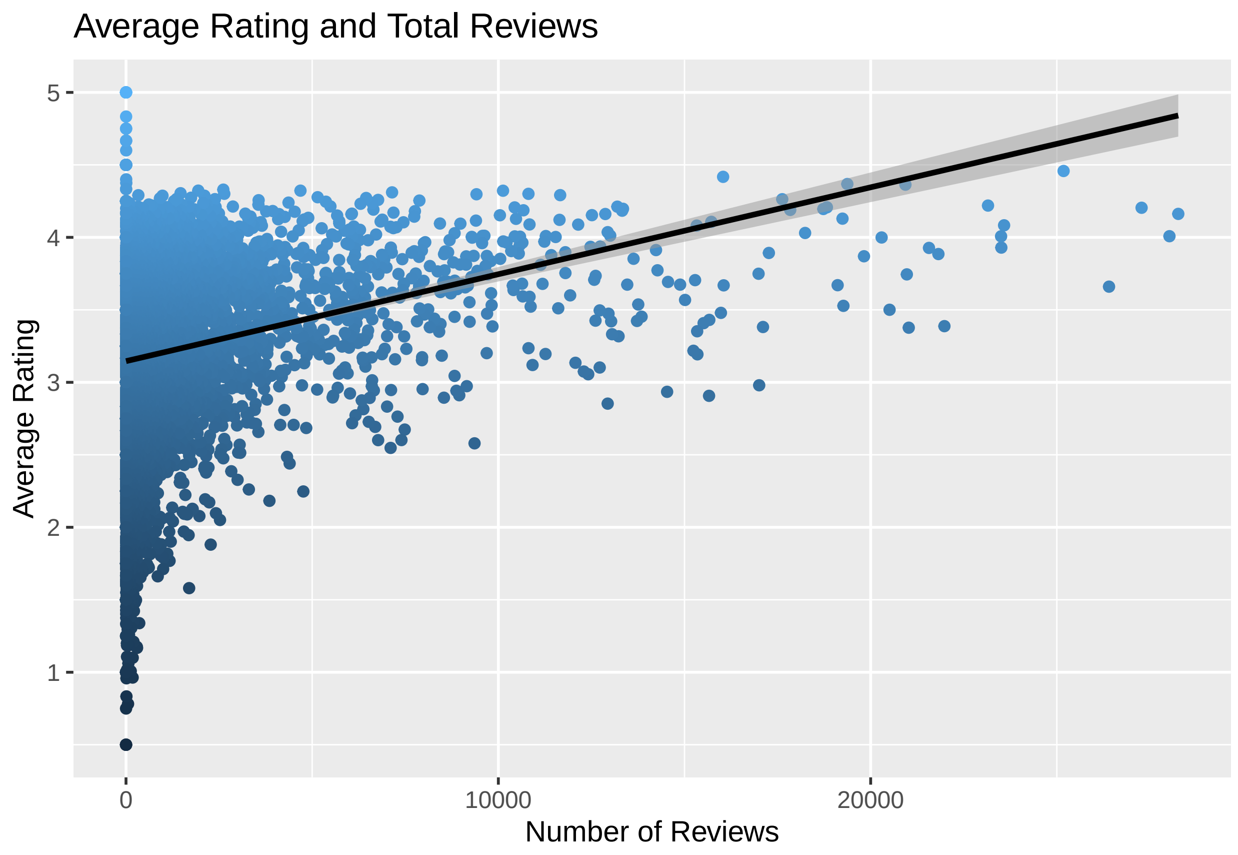
Since some variation in the average rating for each genre, it was decided to look at genre variation among users and see if different users prefer different genres. The top 4 most reviewed genres are drama, comedy, thriller and action. These genres are associated with 94% of the movies in the training data. Boxplots were created showing the rating distributions for these genres faceted by the 12 users who had left the most reviews. The chart shows that some users rate all genres the same and other users have distinct genre preferences.

```
## Selecting by n
```



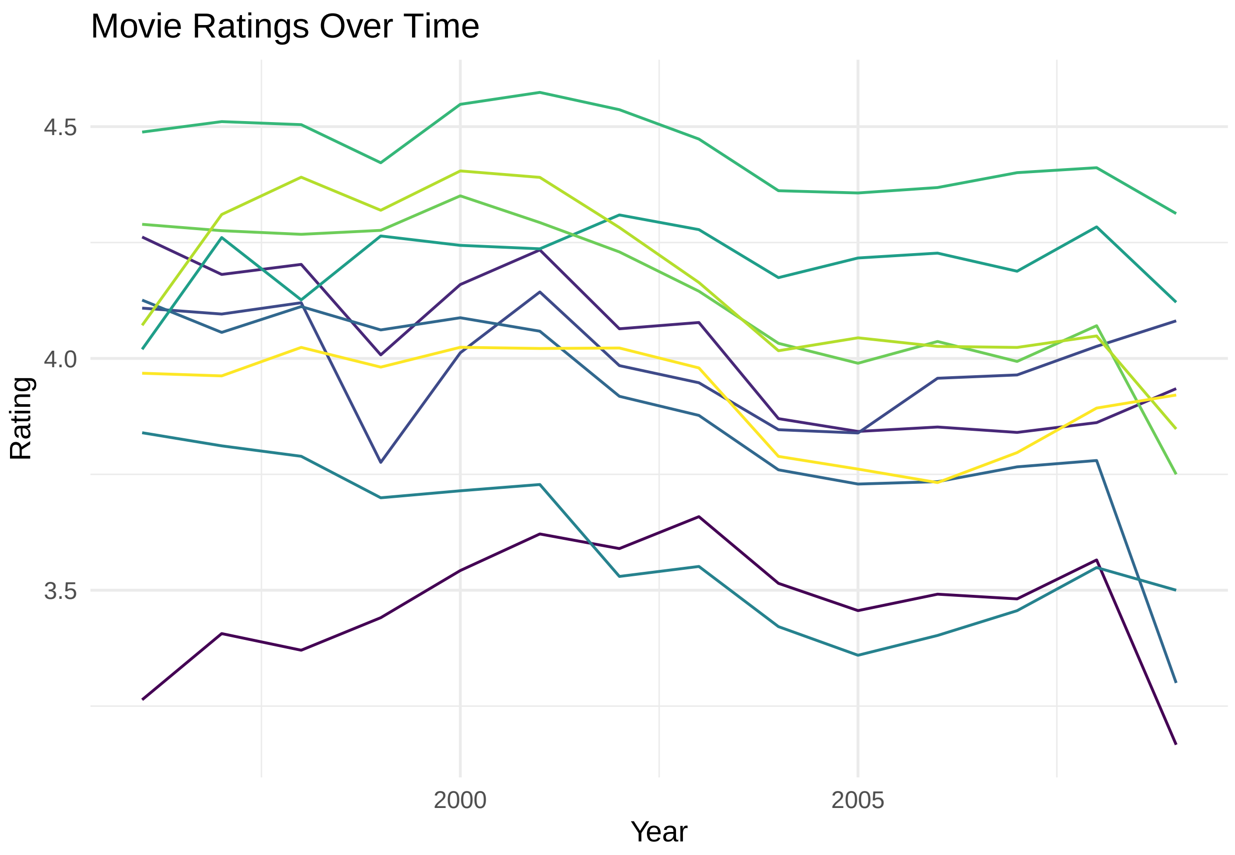
Another scatterplot was created to compare the number of reviews per movie to the average rating. The plot illustrated that movies with fewer reviews have a wider range of ratings and that average rating increased as the number of ratings increased. This makes sense because popular movies tend to have better ratings.

```
## `geom_smooth()` using formula 'y ~ x'
```

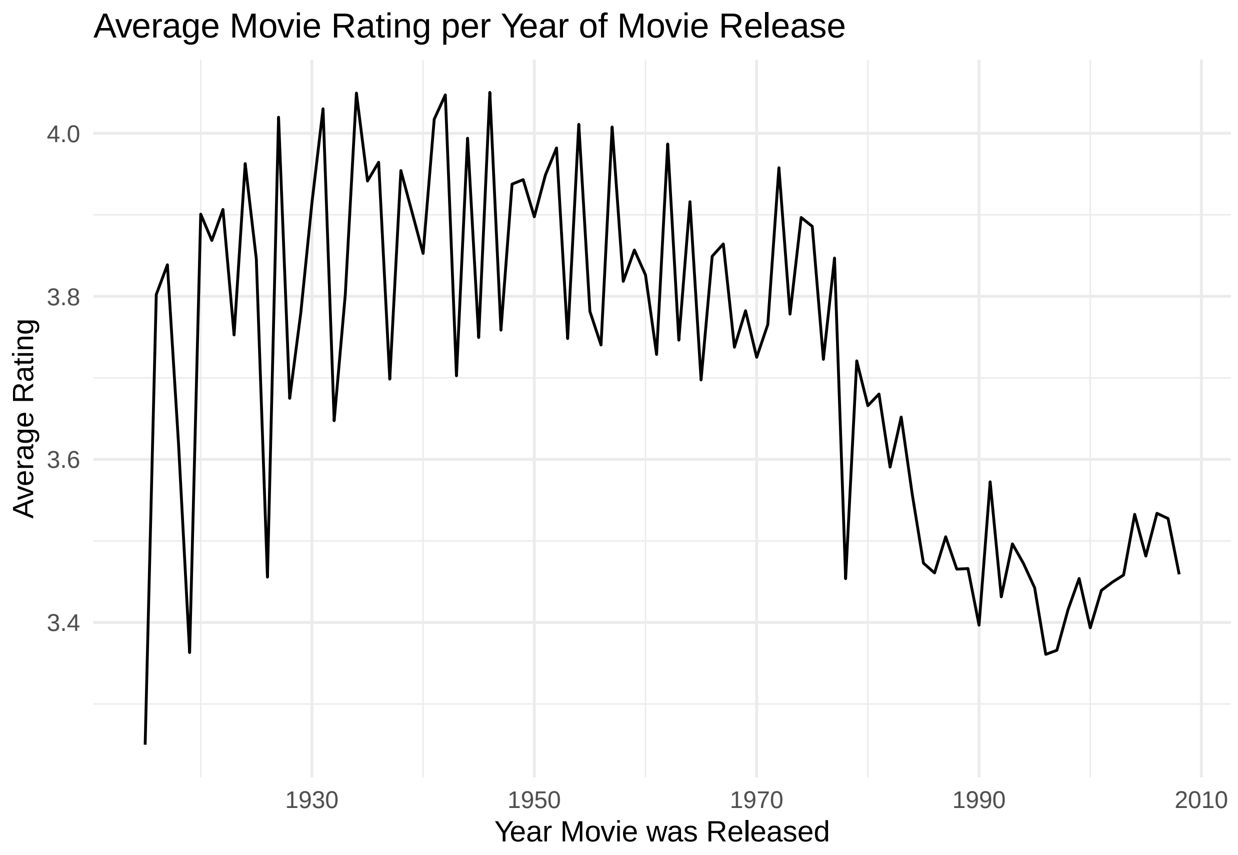


Timeseries plots were created to see if a movie's or user's average rating changed over time. One plot showed the average rating over time for the ten most rated movies and the other for the ten users who left the most reviews. A movie's ratings appear to decrease over the years. However, there is not a strong trend in how users' ratings change over the years.

```
## Selecting by n
```



Finally, a line graph was created to see if movies released in certain years tend to have higher rated movies.



####BIAS PREDICTORS The EDA revealed trends in the data such as some movies are higher rated than others, certain users tend to leave higher reviews than other, movie ratings tend to decrease over time, and users prefer specific genres. Four columns were added: Drama, Comedy, Action, and Thriller. These genres are associated with 94% of the movies in the training data. If one or more of these genres were associated with the movie, then the rating was copied to the appropriate column.

```
#Add genre rating for top 4 genres to the edx dataframe
edx<-edx%>%filter(str_count(genres,"Drama")==1 |
                    str_count(genres,"Comedy")==1 |
                    str_count(genres,"Action")==1 |
                    str_count(genres,"Thriller")==1)%>%
  mutate(Drama=case_when(str_count(genres,"Drama")==1 ~ rating, TRUE ~ 0),
         Comedy = case_when(str_count(genres,"Comedy")==1 ~ rating, TRUE ~ 0),
         Action=case_when(str_count(genres,"Action")==1 ~ rating, TRUE ~ 0),
         Thriller=case_when(str_count(genres,"Thriller")==1 ~ rating, TRUE ~ 0))
```

Nine bias predictors were calculated for the training data set: mean rating of a specific movie, mean rating of a specific user, mean rating for the year a movie was released, mean rating based on the number of reviews, and user's mean rating for the genres of drama, comedy, action and thriller. The value of each bias was calculated by subtracting the mean of movies (mu_rating) from mean of the specific bias group.

```
#Establish the baseline bias predictors for the rating. For example, the mean rating for all movies, et
#Baseline 1: the average rating for all movies
mu_rating<-mean(edx$rating)

#Basline 2: the movie bias. Calculate difference between a specific movie's rating and the average rati
```

```r
movie_avgs <- edx %>%  # calulate the movie bias
  group_by(movieId) %>%
  dplyr::summarize(movie_bias = mean(rating - mu_rating))

#Baseline 3: the user bias: calculate the difference between a specific user's average rating and the o
user_avgs <- edx %>%  # calulate the user bias
  group_by(userId) %>%
  dplyr::summarize(user_bias = mean(rating - mu_rating))


#Baseline 4: Number of Reviews bias: The number of reviews effects the average rating
n_ratings<-edx%>%
  group_by(movieId)%>%
  dplyr::summarize(n=n(),
                   avg_rating = mean(rating))

fit_lm_n_rating<-lm(avg_rating ~ n,data=n_ratings)
slope_n<-summary(fit_lm_n_rating)$coef[2,1]
intercept_n<-summary(fit_lm_n_rating)$coef[1,1]

n_ratings<-n_ratings%>%
  dplyr::mutate(y_hat_nratings = intercept_n+(n*slope_n),
                n_ratings_bias = mean(y_hat_nratings-mu_rating))

#Baseline 5: The year the movie was released effects the average rating
movie_year_avgs <- edx %>%
  group_by(movie_year) %>%
  dplyr::summarize(movie_year_bias = mean(rating - mu_rating))
movie_year_avgs
```

```
## # A tibble: 94 x 2
##    movie_year movie_year_bias
##         <int>           <dbl>
##  1       1915         -0.270
##  2       1916          0.282
##  3       1917          0.318
##  4       1918          0.0989
##  5       1919         -0.157
##  6       1920          0.382
##  7       1921          0.348
##  8       1922          0.107
##  9       1923          0.232
## 10       1924          0.443
## # ... with 84 more rows
```

```r
#Baseline 6: User-Genre Bias
drama_avgs <-edx%>%
  group_by(userId)%>%
  filter(Drama!=0)%>%
  dplyr::summarize(drama_bias = mean(Drama-mu_rating))

comedy_avgs <-edx%>%
  group_by(userId)%>%
  filter(Comedy!=0)%>%
  dplyr::summarize(comedy_bias = mean(Comedy-mu_rating))

action_avgs <-edx%>%
  group_by(userId)%>%
  filter(Action!=0)%>%
  dplyr::summarize(action_bias = mean(Action-mu_rating))

thriller_avgs <-edx%>%
  group_by(userId)%>%
  filter(Thriller!=0)%>%
  dplyr::summarize(thriller_bias = mean(Thriller-mu_rating),)

genre_avgs<-edx%>%
  left_join(drama_avgs, by = 'userId')%>%
  left_join(comedy_avgs, by = 'userId')%>%
  left_join(action_avgs, by = 'userId')%>%
  left_join(thriller_avgs, by = 'userId')%>%
  mutate(Drama = case_when(Drama == 0 ~ 0, TRUE ~ 1),
         Comedy = case_when(Comedy == 0 ~ 0, TRUE ~ 1),
         Action = case_when(Action == 0 ~ 0, TRUE ~ 1),
         Thriller = case_when(Thriller == 0 ~ 0, TRUE ~ 1),
         drama_bias = Drama*drama_bias,
         comedy_bias = Comedy*comedy_bias,
         action_bias = Action*action_bias,
         thriller_bias = Thriller*thriller_bias,
         genre_bias = (drama_bias + comedy_bias + action_bias + thriller_bias)/(Drama+Comedy+Action+Thri
  select(userId, movieId, drama_bias, comedy_bias,action_bias,thriller_bias, genre_bias)
```

All the biases were added to the mean rating of all movies to create the predicted rating.Predicted ratings were created on the training data set.

```
predicted_ratings <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(movie_year_avgs, by='movie_year')%>%
  left_join(n_ratings, by='movieId') %>%
  left_join(genre_avgs, by = c('userId','movieId')) %>%
  select(rating, movieId, userId, movie_bias, user_bias, movie_year_bias, n_ratings_bias,
         genre_bias, drama_bias, comedy_bias,action_bias,thriller_bias, y_hat_nratings)%>%
  mutate(yhat_mu_rating = mu_rating,
         yhat_movie_bias = movie_bias+mu_rating,
         yhat_user_bias = user_bias+mu_rating,
         yhat_movie_year_bias = movie_year_bias + mu_rating,
         yhat_genre_bias = genre_bias + mu_rating,
         yhat = genre_bias + movie_year_bias+n_ratings_bias+user_bias+movie_bias+mu_rating,
         yhat_limits = case_when(yhat>=5 ~ 5, yhat<=0 ~ 0, TRUE ~yhat))

head(predicted_ratings)
```

```
##   rating movieId userId movie_bias user_bias movie_year_bias n_ratings_bias
## 1      5     122      1  -0.658417   1.47951      -0.0784592      -0.312592
## 2      5     185      1  -0.385783   1.47951      -0.0553389      -0.312592
## 3      5     231      1  -0.585958   1.47951      -0.0474453      -0.312592
## 4      5     292      1  -0.099688   1.47951      -0.0553389      -0.312592
## 5      5     316      1  -0.168270   1.47951      -0.0474453      -0.312592
## 6      5     329      1  -0.188774   1.47951      -0.0474453      -0.312592
##   genre_bias drama_bias comedy_bias action_bias thriller_bias y_hat_nratings
## 1    1.47951    0.00000     1.47951     0.00000       0.00000        3.27328
## 2    1.47951    0.00000     0.00000     1.47951       1.47951        3.84888
## 3    1.47951    0.00000     1.47951     0.00000       0.00000        3.98920
## 4    1.47951    1.47951     0.00000     1.47951       1.47951        3.90354
## 5    1.47951    0.00000     0.00000     1.47951       0.00000        4.03520
## 6    1.47951    1.47951     0.00000     1.47951       0.00000        3.90496
##   yhat_mu_rating yhat_movie_bias yhat_user_bias yhat_movie_year_bias
## 1        3.52049         2.86207              5              3.44203
## 2        3.52049         3.13470              5              3.46515
## 3        3.52049         2.93453              5              3.47304
## 4        3.52049         3.42080              5              3.46515
## 5        3.52049         3.35222              5              3.47304
## 6        3.52049         3.33171              5              3.47304
##   yhat_genre_bias    yhat yhat_limits
## 1               5 5.43005           5
## 2               5 5.72580           5
## 3               5 5.53352           5
## 4               5 6.01190           5
## 5               5 5.95121           5
## 6               5 5.93070           5
```

```
edx<-edx[complete.cases(predicted_ratings), ]
predicted_ratings<-predicted_ratings[complete.cases(predicted_ratings), ]
```

Then the root mean square error (RMSE) was used to evaluate the success of the predicted rating compared to the actual rating.

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

None of the predictions based on the biases returned a desirable RMSE value. In particular, the Cumulative Bias Model returned a value greater than 1 which is a very poor result. The model performed so poorly because the formula did not apply weights to the biases. Clearly, some of the biases such as the movie bias and user-genre bias have a stronger affect on the final rating.

```
#RMSE calculations for each yhat column
base_rmse <- RMSE(edx$rating,mu_rating)
movie_bias_rmse <- RMSE(edx$rating, predicted_ratings$yhat_movie_bias)
user_bias_rmse <- RMSE(edx$rating, predicted_ratings$yhat_user_bias)
nratings_bias_rmse <- RMSE(edx$rating, predicted_ratings$y_hat_nratings)
movie_year_bias_rmse <- RMSE(edx$rating, predicted_ratings$yhat_movie_year_bias)
genre_bias_rmse <-RMSE(edx$rating, predicted_ratings$yhat_genre_bias)
cumulative_bias_rmse <- RMSE(edx$rating, predicted_ratings$yhat_limits)

#store the RMSE results in a data frame
rmse_results <- bind_rows(data_frame(method = "Average Rating", RMSE = base_rmse),
                          data_frame(method="Movie Bias Model", RMSE = movie_bias_rmse ),
                          data_frame(method="User Bias Model",RMSE = user_bias_rmse ),
                          data_frame(method="Number of Ratings Bias Model",RMSE = nratings_bias_rmse ),
                          data_frame(method="Movie Year Bias",RMSE = movie_year_bias_rmse ),
                          data_frame(method = "User Genre Bias", RMSE = genre_bias_rmse),
                          data_frame(method="Cumulative Bias Model",RMSE = cumulative_bias_rmse ))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
rmse_results
```

```
## # A tibble: 7 x 2
##   method                        RMSE
##   <chr>                        <dbl>
## 1 Average Rating               1.06
## 2 Movie Bias Model             0.939
## 3 User Bias Model              0.966
## 4 Number of Ratings Bias Model 1.05
## 5 Movie Year Bias              1.04
## 6 User Genre Bias              0.935
## 7 Cumulative Bias Model        1.04
```

####Random Forest Model A random forest model was created to better fit the predictors to the rating. An initial random forest was run on a small percent of the data to do a quick check of processing time, evaluate variable importance, and to select the number of trees.

A new datafram was created containing only the movieId, userId, rating, year values and bias values

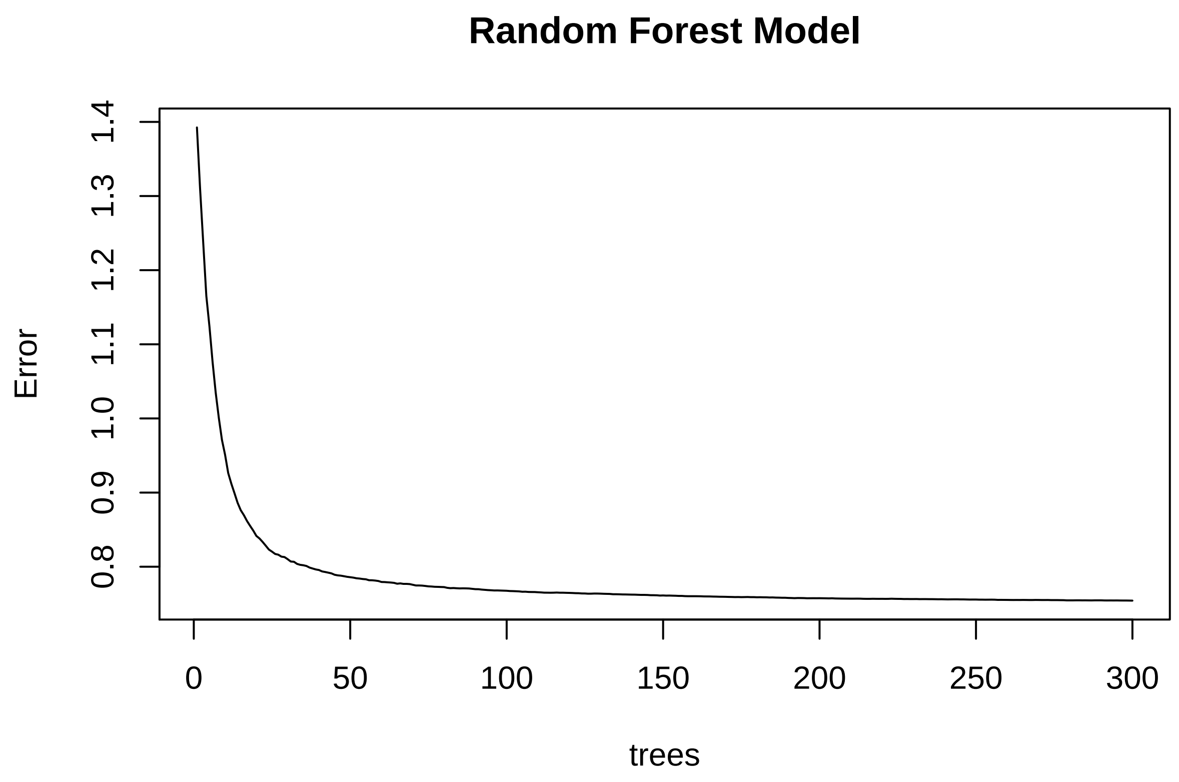
```
edx_bias<-predicted_ratings%>%
  select('rating','movie_bias','user_bias','n_ratings_bias', 'movie_year_bias','drama_bias','comedy_bia
         'thriller_bias','action_bias','genre_bias')

edx_bias<-edx_bias[complete.cases(edx_bias), ] #remove NaNs
```

A small random forest model was excuted on a random sample of 0.2% the training data to do a quick check of processing time, evaluate variable importance, and to select Ntrees

```
set.seed(1)
rf_index <- createDataPartition(y = edx_bias$rating, times = 1, p = 0.002, list = FALSE)
x<-as.matrix(edx_bias[rf_index,2:10])
y<-as.matrix(edx_bias[rf_index,1])
fit_rf<-randomForest(x,as.vector(y), ntree = 300)
varImp(fit_rf)
```

```
##                   Overall
## movie_bias       4299.884
## user_bias        2603.967
## n_ratings_bias      0.000
## movie_year_bias  1234.905
## drama_bias      1245.624
## comedy_bias     1103.538
## thriller_bias    710.047
## action_bias      816.240
## genre_bias      3607.501
```

```
plot(fit_rf, main = "Random Forest Model")
```



Reviewing the variables in the random forest showed that the number of ratings bias was not very important so it was removed from the dataframe in order to improve processing time.

```
edx_bias<-edx_bias[,-4]
```

The data is now ready for fitting the final random forest model. The number of trees is 50. The model is being fit on a random sample of 10% of teh training data because of RAM/memory limitations.

```
set.seed(1)
rf_index <- createDataPartition(y = edx_bias$rating, times = 1, p = 0.1, list = FALSE)
x<-as.matrix(edx_bias[rf_index,2:9])
y<-as.matrix(edx_bias[rf_index,1])
fit_rf<-randomForest(x,as.vector(y), ntree= 50)
varImp(fit_rf, scale = TRUE)
```

```
##                   Overall
## movie_bias      191018.5
## user_bias       117137.0
## movie_year_bias  47109.7
## drama_bias       58886.1
## comedy_bias      51488.8
## thriller_bias    32541.9
## action_bias      40339.6
## genre_bias      161696.3
```

```
fit_rf
```

```
##
## Call:
##  randomForest(x = x, y = as.vector(y), ntree = 50)
##                Type of random forest: regression
##                      Number of trees: 50
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.749096
##                     % Var explained: 32.55
```

The test data needed to be updated to match the same format as the training data. This included removing NaN, adding the genre columns, and calculating the bias values.

Once the test data was formatted, then the predicted ratings were preticted with the fitted random forest model.

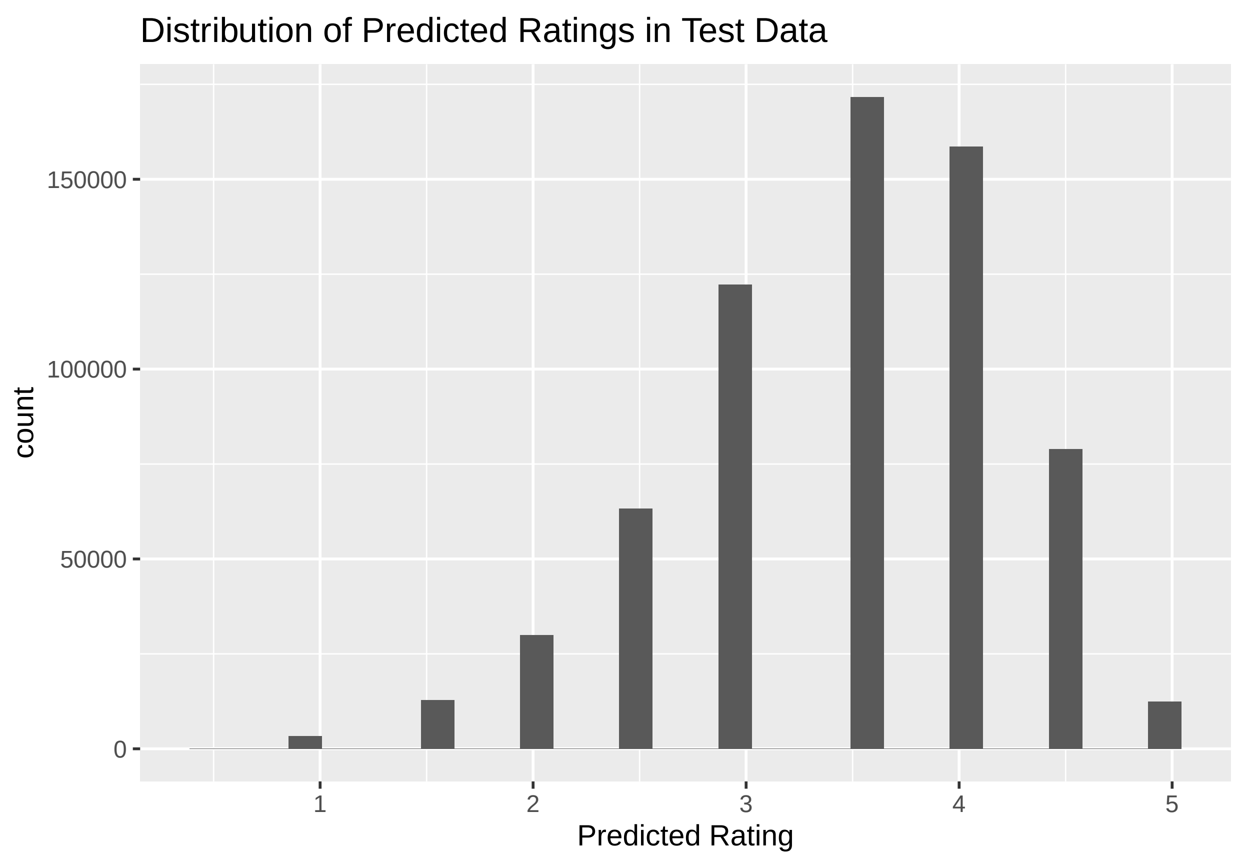
```
x<-as.matrix(test_bias[,2:9])
y<-as.matrix(test_bias[,1])
y_hat <- predict(fit_rf, x, type = "class")

rf_test_rmse <- RMSE(as.numeric(y_hat),as.numeric(y))
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "RF model Test Data", RMSE = rf_test_rmse))
rf_test_rmse
```

```
## [1] 0.790813
```

A histogram of the predicted ratings shows a similar distribution to the actual ratings.

```
y_hat_round <- round_any(y_hat,0.5)

as.data.frame(y_hat_round)%>%
  ggplot(aes(y_hat_round))+
  geom_histogram()+
  ggtitle("Distribution of Predicted Ratings in Test Data")+
  xlab("Predicted Rating")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



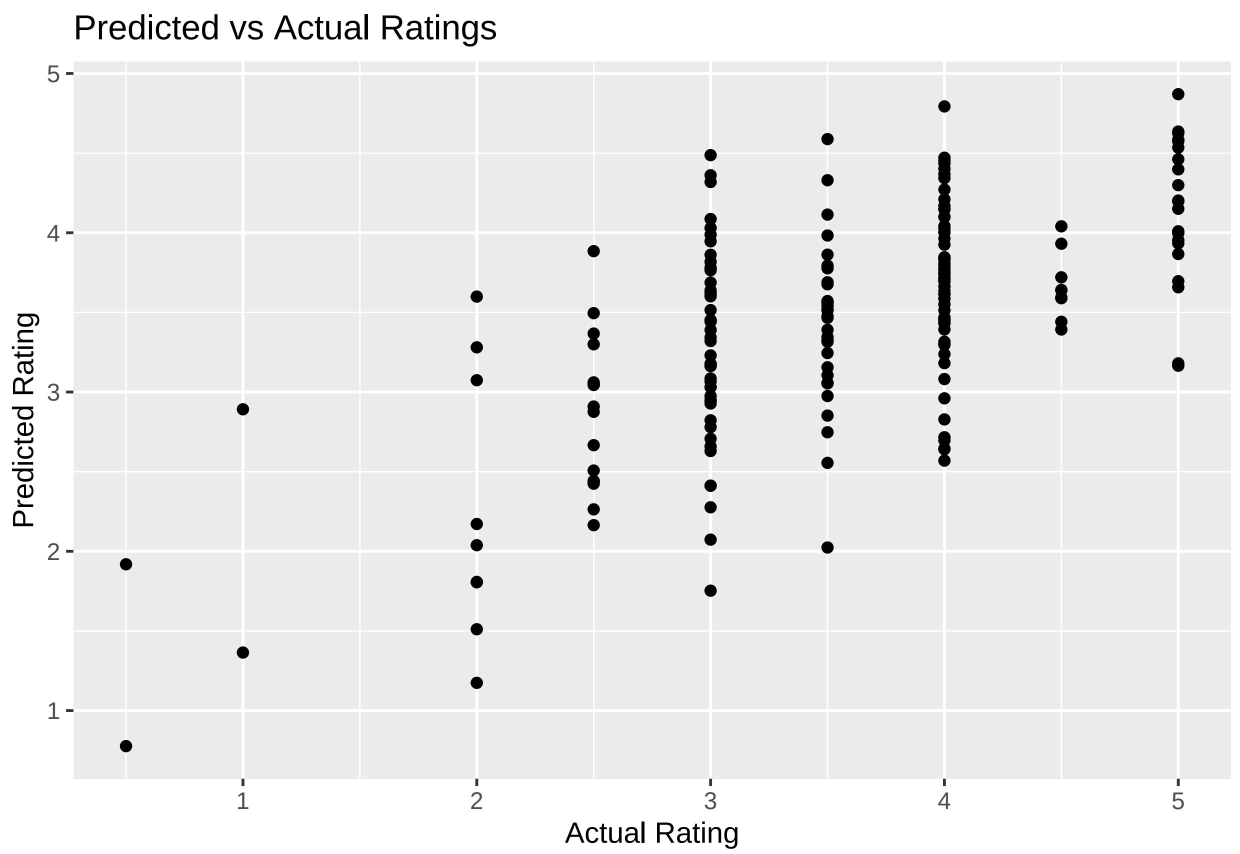
##RESULTS: The validation data set was converted into the same format and then the final results were calculated.

The predicted ratings for the validation dataset give a RMSE that is well below the required value of 0.86490.

```r
#Create predictions and evaluate on the accuracy and RMSE
x<-as.matrix(val_bias[,2:9])
y<-as.matrix(val_bias[,1])
y_hat <- predict(fit_rf, x, type = "class")
rf_val_rmse <- RMSE(as.numeric(y),as.numeric(y_hat))
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "RF model Validation Data", RMSE = rf_val_rmse))
rmse_results
```

```
## # A tibble: 9 x 2
##   method                          RMSE
##   <chr>                          <dbl>
## 1 Average Rating                 1.06
## 2 Movie Bias Model               0.939
## 3 User Bias Model                0.966
## 4 Number of Ratings Bias Model 1.05
## 5 Movie Year Bias                1.04
## 6 User Genre Bias                0.935
## 7 Cumulative Bias Model          1.04
## 8 RF model Test Data             0.791
## 9 RF model Validation Data      0.801
```

Addtionally, a scatter plot reveals a strong correlation between the actual ratings and the predicted ratings.



##CONCLUSION: The random forest model returned an RMSE well below the required value of 0.86490. The solution was created from the exploratory data analysis. The bias predictors for movie ratings, user ratings, and user-genre preferences are major contributors to the low RMSE score. The main limitation to these results was the processing capability of the computer. This prevented the randow forest from being fit to a large percentage of the training data set. Future work could be done to predict movie ratings for a specific user based on how users with similar movie preferences have rated movies.