HarvardX: PH125.9x Data Science Capstone - Movielens Project

Golpira Elmi Assadzadeh, Ph.D.

03/04/2020

IMPORTANT COURSE INSTRUCTION:
The validation data should NOT be used for training your algorithm and should ONLY be used for evaluating the RMSE of your final algorithm. You should split the edx data into separate training and test sets to design and test your algorithm.

PROJECT OVERVIEW

The recommendation systems offered by companies such as Amazon utilize previously rated, browsed, or purchased items to recommend items of interest to potential buyers. In this project, our aim is to develop a machine learning algorithm to predict movie ratings, and use it to recommend movies to users. The Movielens dataset used in this project contains 10000054 movie ratings applied to 10677 movies by 69878 users and grouped into 797 unique genres. The dataset is collected by Harper and Konstan (2015) and is made available for public download through the GroupLens Research Group at the University of Minnesota.

For this project, the data were first inspected in order to understand the pattern and data structure. Several plots have been created in ordere to visualize the effect of movies, users, movie age, and genres on average ratings. The edx dataset were then splitted into training and test sets and several different algorithms were tested to find the movie predictions with lowest Root Mean Square Error (RMSE). The final model were then used to calculate the RMSE on validation datasets. The RMSE is a measure of the differences between values predicted by a model and the values observed (i.e., model accuracy), and is calculated as follows:

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

In calculating RMSE, one should be cognizant as larger errors have a disproportionately large effect on the result. In other words, RMSE is sensitive to outliers.

METHOD AND ANALYSIS

INSPECTING AND VISUALIZING THE EDX DATASET

A few rows from edx dataset is printerd in order to identify its variables. The datsset contains six columns (i.e., “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”), with each row representing a single rating of a user for a single movie.

```
head(edx)
```

```r
kable(head(edx), "pandoc", caption = "edx dataset")
```

Table 2: edx dataset

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

Number of edx distinct movieIds, userIds, and genres are provided in the table below:

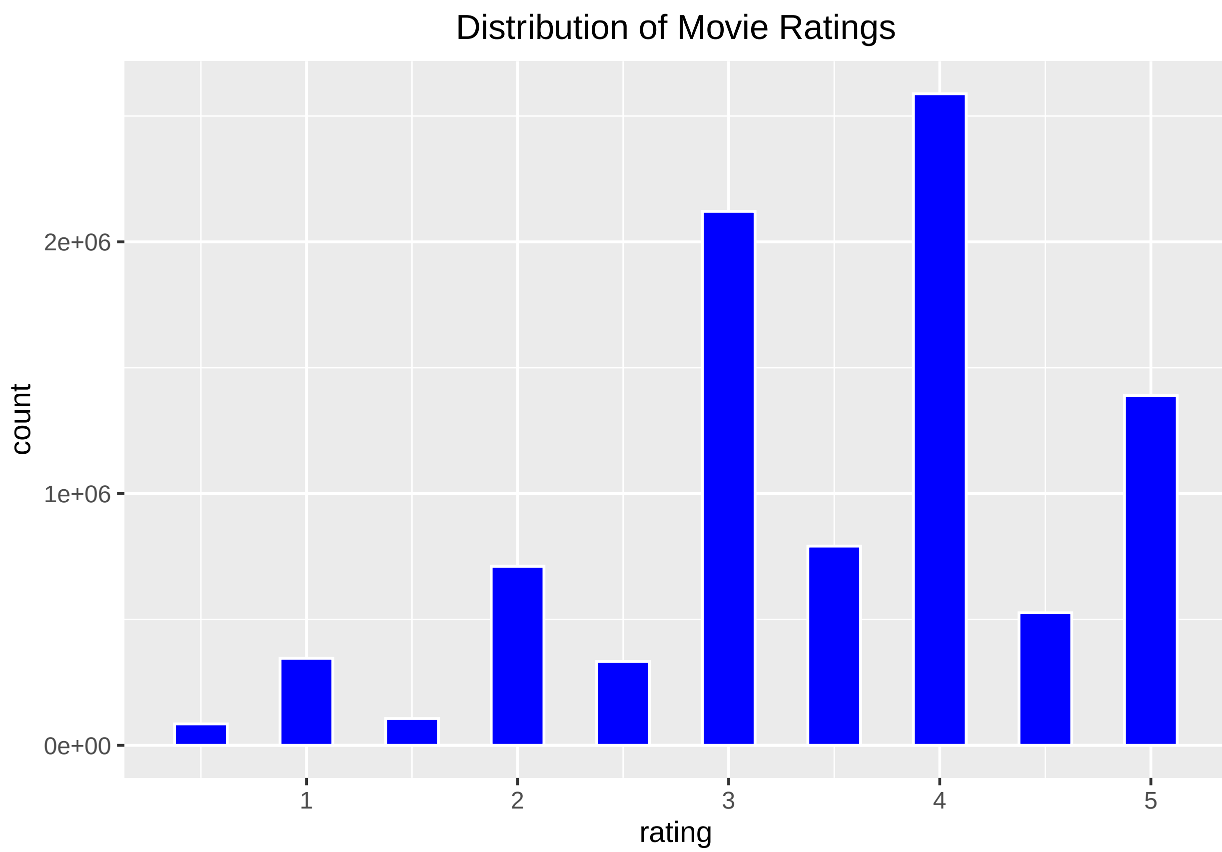
```r
distinct <- edx %>% summarize(n_users = n_distinct(userId),
    n_movies = n_distinct(movieId), n_genres = n_distinct(genres))
kable(distinct, "pandoc", caption = "number of unique users, movies, and genres")
```

Table 3: number of unique users, movies, and genres

n_users	n_movies	n_genres
69878	10677	797

The following plot shows the distribution of movie ratings. Four is the highest rating followed by 3 and 5, with half-ratings being less common than whole ratings.

```r
edx %>% ggplot(aes(rating)) + geom_histogram(binwidth = 0.25,
    fill = "blue", color = "white") + ggtitle("Distribution of Movie Ratings") +
    theme(plot.title = element_text(hjust = 0.5))  # centre the title
```



To evaluate the movie age effect, the movie released date is first extracted from movie title as follows:

```
# extracting the movie released date
released_date <- stringi::stri_extract(edx$title, regex = "(\\d{4})",
    comments = TRUE) %>% as.numeric()
```

In addition, the year each movie has been rated is calculated based on timestamp column. The timestamp column is no longer needed, so will be removed from edx dataset.

```
# Add the released date
edx <- edx %>% mutate(year = released_date, year_rated = year(as_datetime(timestamp))) %>%
    select(-timestamp)
head(edx)
```

```
kable(head(edx), "pandoc", caption = "edx dataset with extracted year_released and calculat
```

Table 4: edx dataset with extracted year_released and calculated year_rated

userId	movieId	rating	title	genres	year	year_rated
1	122	5	Boomerang (1992)	Comedy|Romance	1992	1996
1	185	5	Net, The (1995)	Action|Crime|Thriller	1995	1996
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	1996
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1994	1996
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994	1996
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1994	1996

The following code is to check if the released_date is correctly extracted from title. All released years in future and those with released dates before 1900 are wrong and need to be corrected.

```
any_errors <- edx %>% group_by(movieId, title, year) %>%
    filter(year > 2020 | year < 1900) %>% distinct(year)
kable(any_errors, "pandoc", caption = "Movie entries with incorrect release date extraxted")
```

Table 5: Movie entries with incorrect release date extraxted

year	movieId	title
1000	6290	House of 1000 Corpses (2003)
1600	1422	Murder at 1600 (1997)
3000	671	Mystery Science Theater 3000: The Movie (1996)
1000	8198	1000 Eyes of Dr. Mabuse, The (Tausend Augen des Dr. Mabuse, Die) (1960)
1138	6645	THX 1138 (1971)
5000	5310	Transylvania 6-5000 (1985)
3000	4159	3000 Miles to Graceland (2001)
1492	8905	1492: Conquest of Paradise (1992)
2046	27266	2046 (2004)
3000	8864	Mr. 3000 (2004)
1408	53953	1408 (2007)
1776	5472	1776 (1972)
9000	2308	Detroit 9000 (1973)
1732	4311	Bloody Angels (1732 Høtten: Marerittet Har et Postnummer) (1998)

This piece of code fixes all years that were wrongly extracted:

```
# Fix the incorrect dates
edx[edx$movieId == "6290", "year"] <- 2003
edx[edx$movieId == "1422", "year"] <- 1997
edx[edx$movieId == "671", "year"] <- 1996
edx[edx$movieId == "8198", "year"] <- 1960
edx[edx$movieId == "6645", "year"] <- 1971
edx[edx$movieId == "5310", "year"] <- 1985
edx[edx$movieId == "4159", "year"] <- 2001
edx[edx$movieId == "8905", "year"] <- 1992
edx[edx$movieId == "27266", "year"] <- 2004
edx[edx$movieId == "8864", "year"] <- 2004
edx[edx$movieId == "53953", "year"] <- 2007
edx[edx$movieId == "5472", "year"] <- 1972
edx[edx$movieId == "2308", "year"] <- 1973
edx[edx$movieId == "4311", "year"] <- 1998
```

Age if each movie is calculated and added to the edx dataset as follows:

```
edx <- edx %>% mutate(movie_age = 2020 - year)
head(edx)
```

```r
kable(head(edx), "pandoc", caption = "edx dataset with calculated movie_age column")
```

Table 6: edx dataset with calculated movie_age column

userId	movieId	rating	title	genres	year	year_rated	movie_age
1	122	5	Boomerang (1992)	Comedy|Romance	1992	1996	28
1	185	5	Net, The (1995)	Action|Crime|Thriller	1995	1996	25
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	1996	25
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1994	1996	26
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994	1996	26
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1994	1996	26

MOVIE EFFECT BY TITLE

Average rating for each movie and rating frequency are calculated.

```r
movie_avgs <- edx %>% group_by(title) %>% summarize(number_of_movie_ratings = n(),
    avg_movie_rating = mean(rating)) %>% arrange(desc(avg_movie_rating))
kable(head(movie_avgs), "pandoc", caption = "Calculated movie average rating & rating frequ
```

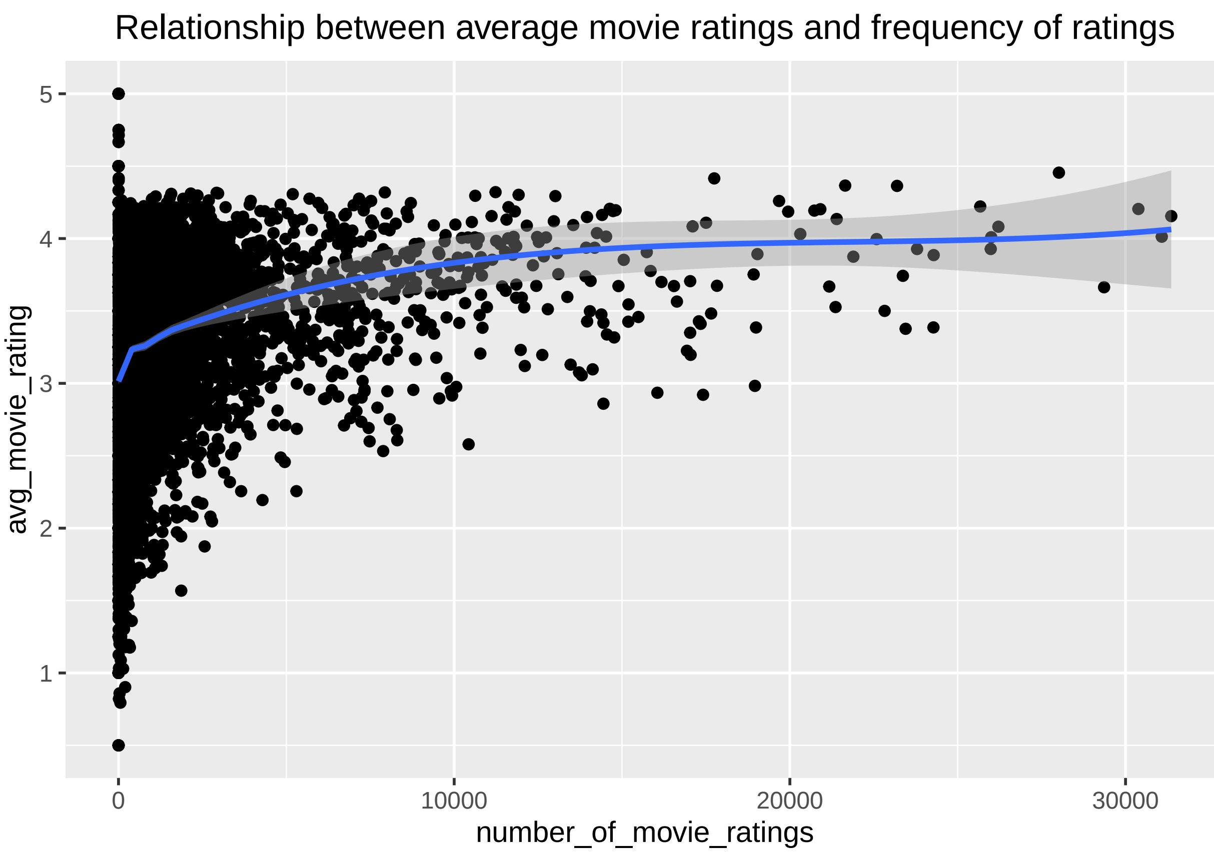
Table 7: Calculated movie average rating & rating frequency

title	number_of_movie_ratings	avg_movie_rating
Blue Light, The (Das Blaue Licht) (1932)	1	5
Fighting Elegy (Kenka erejii) (1966)	1	5
Hellhounds on My Trail (1999)	1	5
Satan's Tango (Sátántangó) (1994)	2	5
Shadows of Forgotten Ancestors (1964)	1	5
Sun Alley (Sonnenallee) (1999)	1	5

The figure below shows the relationship between average movie ratings and frequency of ratings. The variation in movie ratings are much higher for movies that have been rated less often.

```r
movie_avgs %>% ggplot(aes(number_of_movie_ratings,
    avg_movie_rating)) + geom_point() + geom_smooth(method = "loess") +
    ggtitle("Relationship between average movie ratings and frequency of ratings") +
    theme(plot.title = element_text(hjust = 0.5))  # centre the title
```

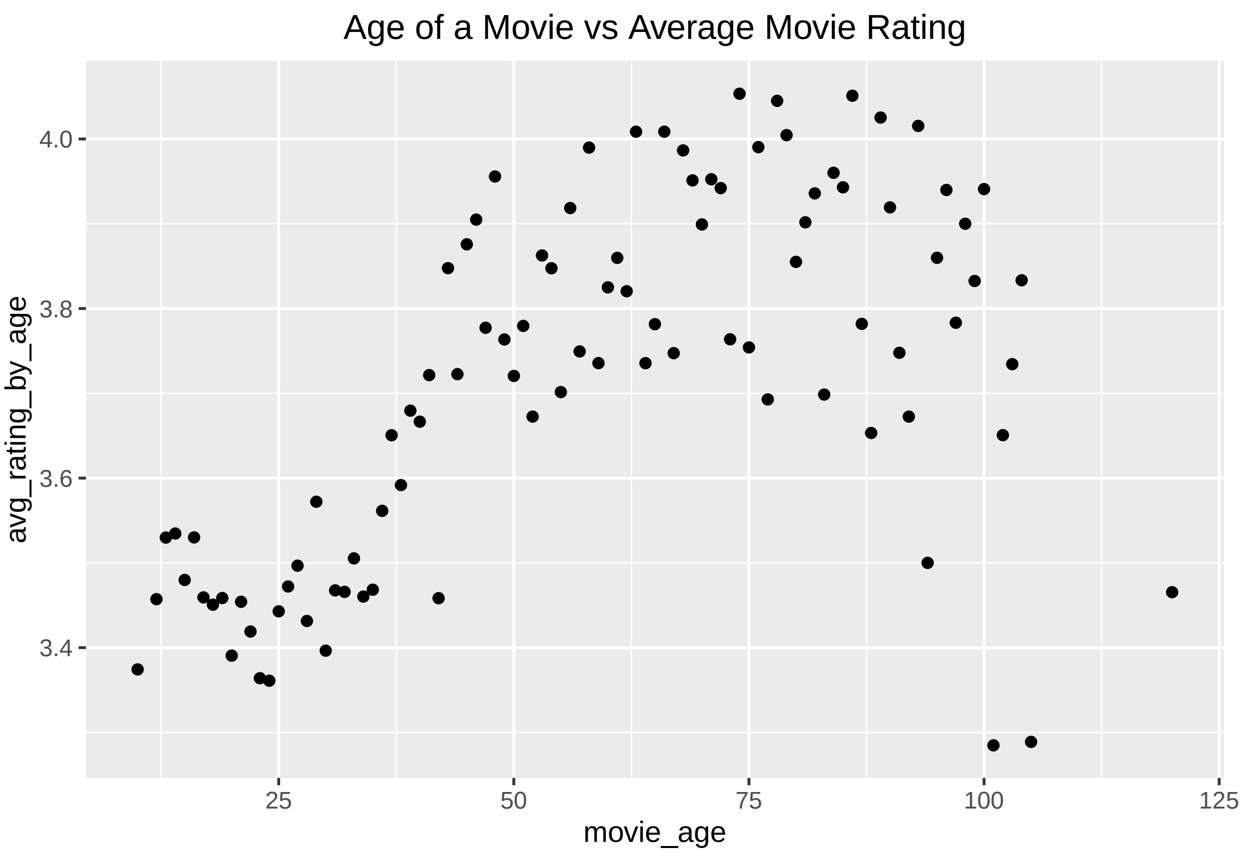
```
## `geom_smooth()` using formula 'y ~ x'
```



AGE OF MOVIE

The following plot shows the average movie rating versus movie age. It's evident from the plot that the movies of 50-100 years in age generally rated higher than newer movies.

```
# age of movie vs average movie rating
age_avgs <- edx %>% group_by(movie_age) %>% summarize(avg_rating_by_age = mean(rating))

age_avgs %>% ggplot(aes(movie_age, avg_rating_by_age)) +
    geom_point() + ggtitle("Age of a Movie vs Average Movie Rating") +
    theme(plot.title = element_text(hjust = 0.5))   # centre the title
```



USER EFFECT

The average movie rating grouped by userId for user that rated over 100 movies is calculated as follows:

```
user_avgs <- edx %>% group_by(userId) %>% summarize(number_of_user_ratings = n(),
    avg_user_rating = mean(rating)) %>% filter(number_of_user_ratings >
    100) %>% arrange(desc(avg_user_rating))

head(user_avgs)
```

```
kable(head(user_avgs), "pandoc", caption = "Calculated average user rating and number of ra
```

Table 8: Calculated average user rating and number of ratings

userId	number_of_user_ratings	avg_user_rating
5763	214	4.934579
59987	202	4.896040
36896	149	4.892617
19010	140	4.850000
16033	102	4.843137
48518	130	4.838462

The results show that userId #5763 has given higher ratings to movies with average of 4.93.

```r
max(user_avgs$avg_user_rating)
```

```
## [1] 4.934579
```

```r
user_avgs$userId[which.max(user_avgs$avg_user_rating)]
```

```
## [1] 5763
```

and userId #24176 has given lowest average rating to movies with average of 1.

```r
min(user_avgs$avg_user_rating)
```

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

```r
user_avgs$userId[which.min(user_avgs$avg_user_rating)]
```

```
## [1] 24176
```

GENRE EFFECT

In order to investigate the effect of individual genres on movie ratings, the genre column is separated into single genres as follows:

```r
single_genres <- edx %>% separate_rows(genres, sep = "\\|")
head(single_genres)
```

```r
kable(head(single_genres), "pandoc", caption = "edx dataset with seperated genres")
```

Table 9: edx dataset with seperated genres

userId	movieId	rating	title	genres	year	year_rated	movie_age
1	122	5	Boomerang (1992)	Comedy	1992	1996	28
1	122	5	Boomerang (1992)	Romance	1992	1996	28
1	185	5	Net, The (1995)	Action	1995	1996	25
1	185	5	Net, The (1995)	Crime	1995	1996	25
1	185	5	Net, The (1995)	Thriller	1995	1996	25
1	292	5	Outbreak (1995)	Action	1995	1996	25

The total number of movies in each genre is calculated as follows:

```
number_of_movies_genres <- single_genres %>% group_by(genres) %>%
    summarize(number_movies_genre = n())
kable(head(number_of_movies_genres), "pandoc", caption = "Total number of movies in each ge
```

Table 10: Total number of movies in each genre

genres	number_movies_genre
(no genres listed)	7
Action	2560545
Adventure	1908892
Animation	467168
Children	737994
Comedy	3540930

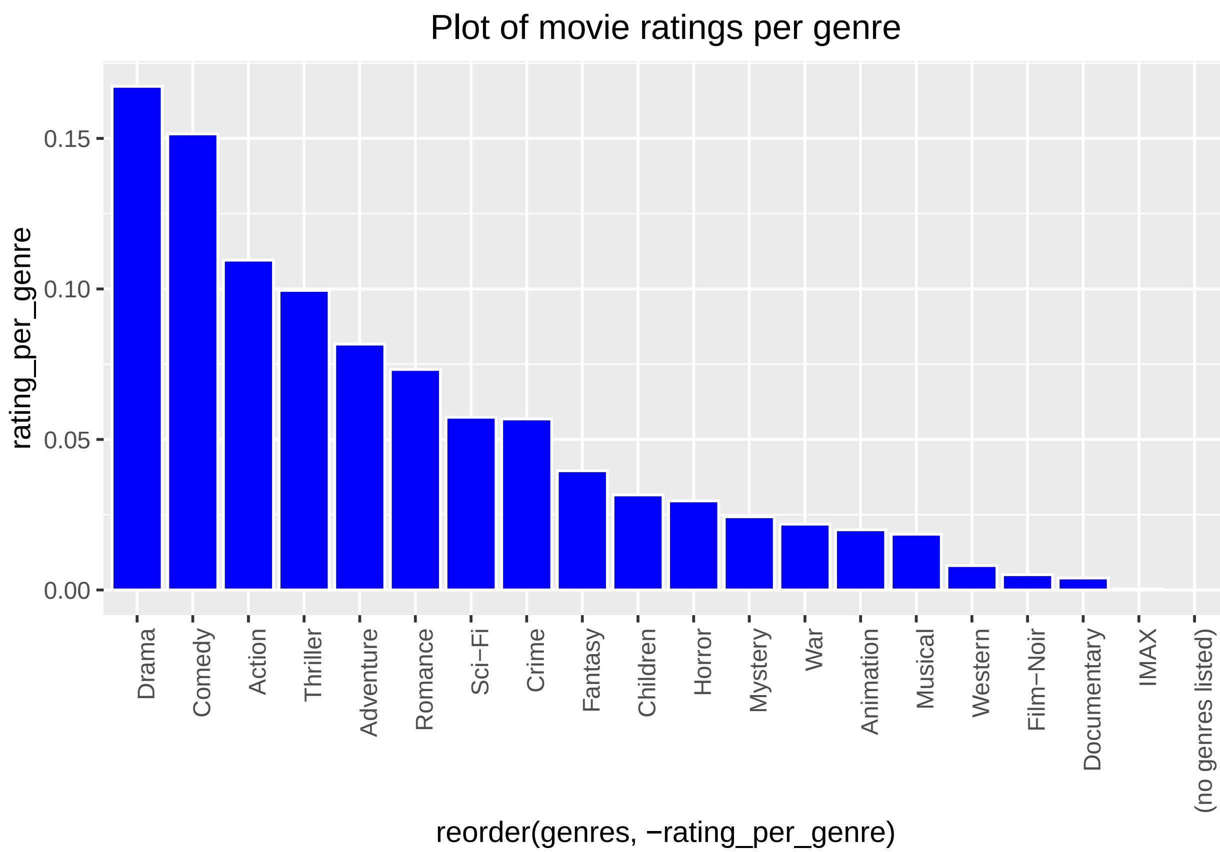
Distribution of ratings per genre

```
genre_distribution <- single_genres %>% group_by(genres) %>%
    summarize(n = n()) %>% ungroup() %>% mutate(rating_per_genre = n/sum(n)) %>%
    arrange(desc(rating_per_genre)) %>% select(-n)
```

Plot of movie ratings per genre

The following graph shows that movie ratings are also a function of genres, with Drama and Comedy having being the most frequently rated genres.

```
genre_distribution %>% ggplot(aes(reorder(genres, -rating_per_genre),
    rating_per_genre)) + geom_bar(stat = "identity",
    color = "white", fill = "blue") + ggtitle("Plot of movie ratings per genre") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    theme(plot.title = element_text(hjust = 0.5))  # centre the title
```



The average movie rating per genre is calculated as follows:

```
mean_rating_per_genre <- single_genres %>% group_by(genres) %>%
    summarize(mean_rating_by_genre = mean(rating)) %>%
    arrange(-mean_rating_by_genre)
kable(head(mean_rating_per_genre), "pandoc", caption = "Average ratings for each genre")
```

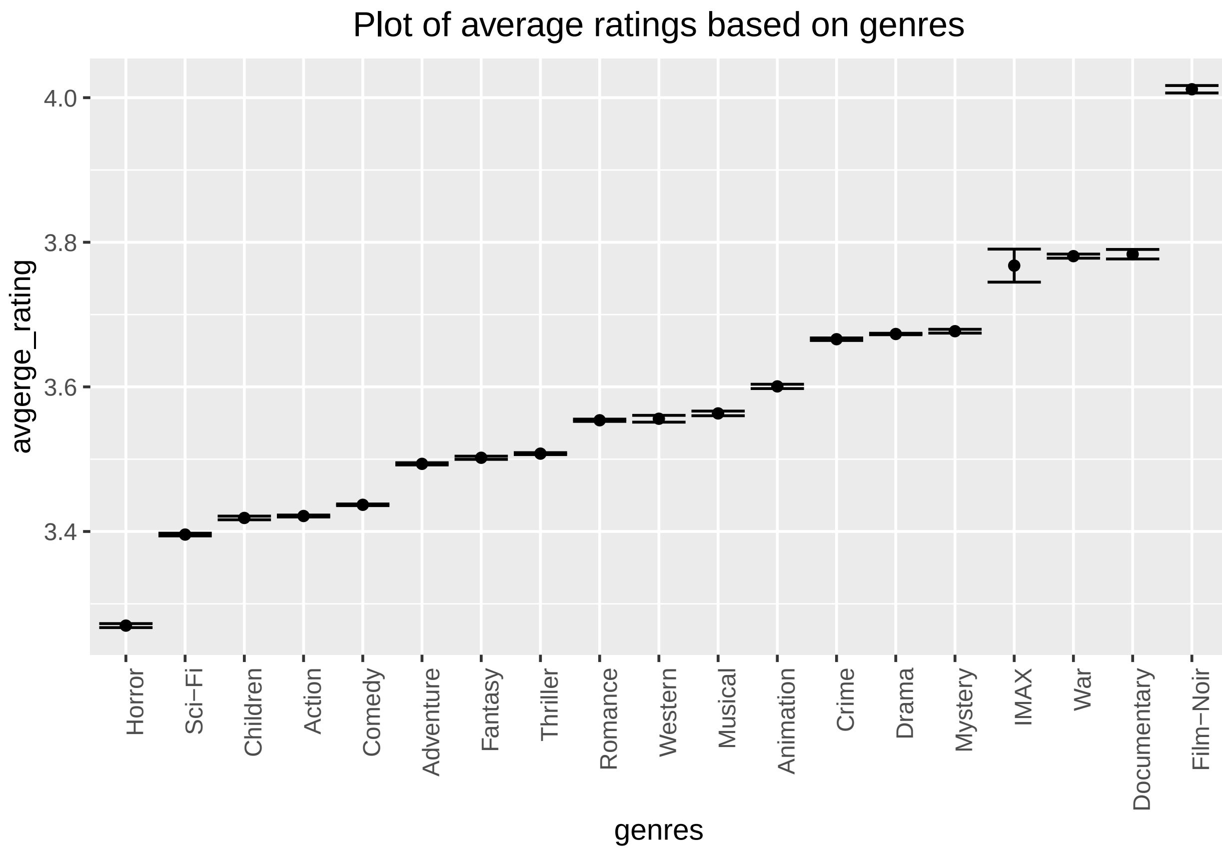
Table 11: Average ratings for each genre

genres	mean_rating_by_genre
Film-Noir	4.011625
Documentary	3.783487
War	3.780813
IMAX	3.767693
Mystery	3.677001
Drama	3.673131

Below is the plot showing the average rating based on genres.

```
single_genres %>% group_by(genres) %>% summarize(n = n(),
    avgerge_rating = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 1000) %>% mutate(genres = reorder(genres,
    avgerge_rating)) %>% ggplot(aes(x = genres, y = avgerge_rating,
    ymin = avgerge_rating - 2 * se, ymax = avgerge_rating +
        2 * se)) + geom_point() + geom_errorbar() +
```

```r
    ggtitle("Plot of average ratings based on genres") +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    theme(plot.title = element_text(hjust = 0.5))  # centre the title
```



Based on this graph, Film-Noir has the highest average rating while Horror movies have the lowest average rating.

RESULTS: PREDICTIONS (TESTING DIFFERENT MODELS)

According to course instructions, the edx dataset is splitted into training and test sets.

```r
edx_test_index <- createDataPartition(y = edx$rating,
    times = 1, p = 0.2, list = FALSE)
training_set <- edx[-edx_test_index, ]
test_set <- edx[edx_test_index, ]
```

DEFINING RMSE FUNCTION

As mentioned earlier, the goal of the project is to determine an algorithm that minimizes RMSE. The RMSE is calculated as:

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

and is a measure of model accuracy. The lower RMSE, the higher the accuracy.

```
RMSE <- function(true_ratings, predicted_ratings) {
    sqrt(mean((true_ratings - predicted_ratings)^2,
        na.rm = TRUE))
}
```

BASE MODEL

This basic model predicts the same rating for all movies by all users (i.e., calculating mean rating for entire dataset). That is our base model and is represented by the following formula:

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

The $\epsilon_{u,i}$ ia an independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies.

In this model all differences in movie ratings are explained by random variation alone, and is calculated by averaging all movie ratings in the entire dataset:

```
mu_hat <- mean(training_set$rating)
mu_hat
```

[1] 3.512574

```
naive_rmse <- RMSE(test_set$rating, mu_hat)

rmse_results <- tibble(method = "Base model_Averaging",
    RMSE_on_training_set = naive_rmse, RMSE_on_validation_set = "NA")
kable((rmse_results[1:1, ]), "pandoc", align = "c",
    caption = "RMSE Results")
```

Table 12: RMSE Results

method	RMSE_on_training_set	RMSE_on_validation_set
Base model_Averaging	1.060708	NA

RMSE result for this model is 1.0607 which is too high.

MOVIE EFFECTS

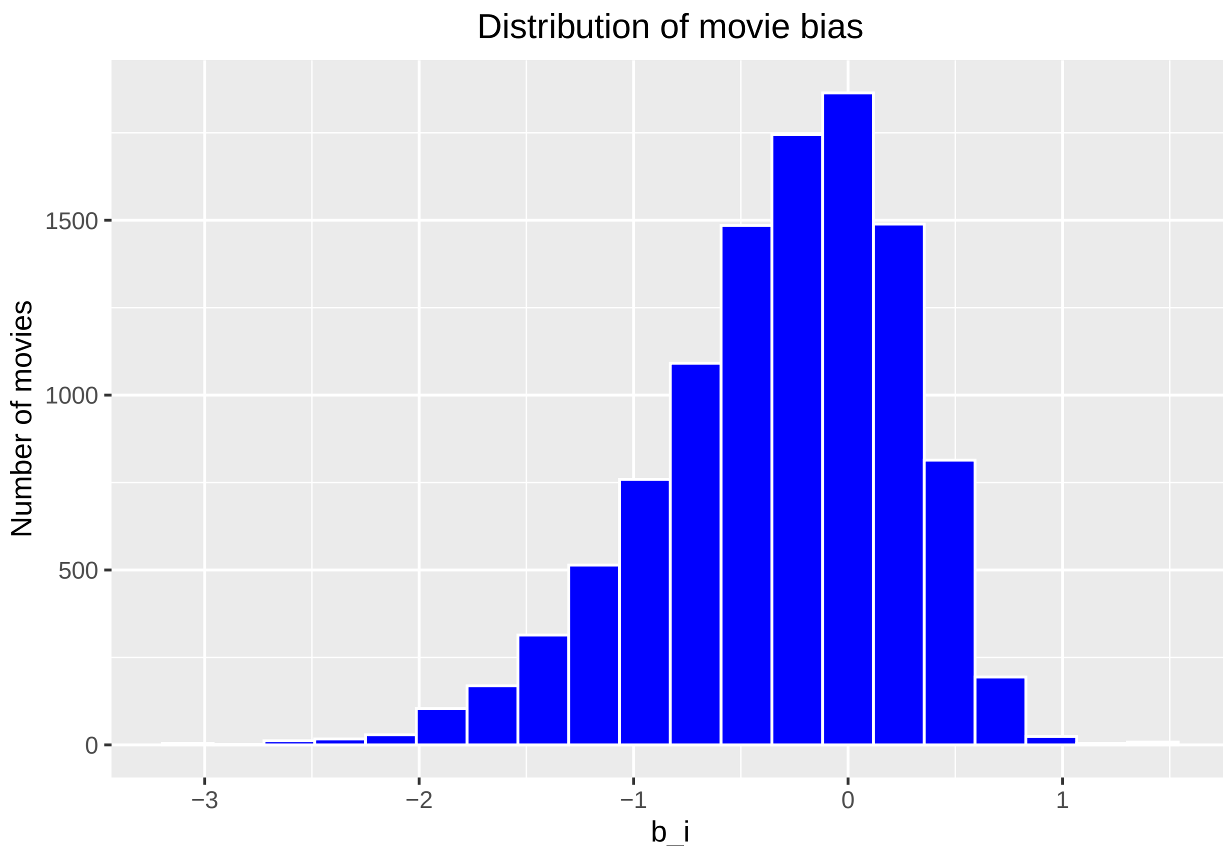
Because popular movies usually rate higher than non-popular movies, it's not correct to average ratings for all movies altogether, rather, movie rating bias should be taken into account. This bias is calculated as follows:

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

The distribution of movie bias is plotted below:

```
mu <- mean(training_set$rating)
movie_avgs <- training_set %>% group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))
```

```r
qplot(b_i, data = movie_avgs, bins = 20, color = I("white"),
    fill = I("blue"), ylab = "Number of movies", main = "Distribution of movie bias") +
    theme(plot.title = element_text(hjust = 0.5))
```



Here is our prediction and our prediction accuracy based on movie effect alone. By adding the computed b_i to μ, we have included movie rating bias to our predictive model. That is the difference between individual movie average from the total movie average is taken into account. We will predict higher rating for a movie which has generally rated higher than average of all movies, and lower rating for a movie that rated lower than overall average.

```r
predicted_ratings <- mu + test_set %>% left_join(movie_avgs,
    by = "movieId") %>% pull(b_i)

RMSE_movies <- RMSE(test_set$rating, predicted_ratings)

# adding the results to the rmse tibble for
# comparison
rmse_results <- add_row(rmse_results, method = "Movie_Effect",
    RMSE_on_training_set = RMSE_movies, RMSE_on_validation_set = "NA")
kable((rmse_results[1:2, ]), "pandoc", align = "c",
    caption = "RMSE Results")
```

Table 13: RMSE Results

method	RMSE_on_training_set	RMSE_on_validation_set
Base model_Averaging	1.0607079	NA
Movie_Effect	0.9437144	NA

MOVIE_USER EFFECTS

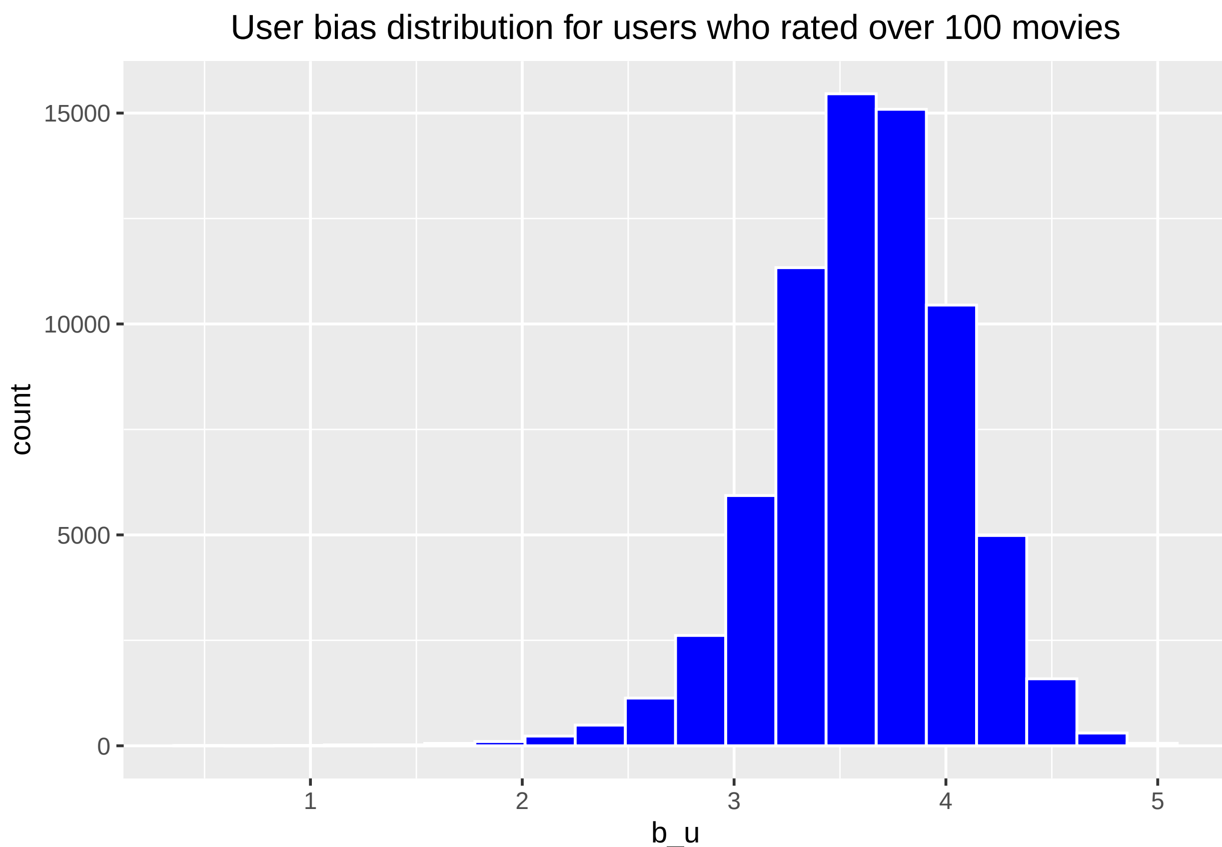
Some users give higher rating to movies in generall than others, which will also creates a bias. This bias needs to be taken into account. This shows that further improvement to our model my be:

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

where b_u is a user-specific effect.

Here we compute and plot the average rating for user u for those that have rated over 100 movies.

```r
training_set %>% group_by(userId) %>% summarize(b_u = mean(rating)) %>%
    filter(n() >= 100) %>% ggplot(aes(b_u)) + geom_histogram(bins = 20,
    color = "white", fill = "blue") + ggtitle("User bias distribution for users who rated o
    theme(plot.title = element_text(hjust = 0.5))  # centre the title
```



Based on our knowledge, we can calculate our predictions and RMSE for combined movie and user effect as follows:

```
user_avgs <- training_set %>% left_join(movie_avgs,
    by = "movieId") %>% group_by(userId) %>% summarize(b_u = mean(rating -
    mu - b_i))

predicted_ratings <- test_set %>% left_join(movie_avgs,
    by = "movieId") %>% left_join(user_avgs, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>% pull(pred)

RMSE_user_movie <- RMSE(test_set$rating, predicted_ratings)
# addind the results to the rmse tibble for
# comparison
rmse_results <- add_row(rmse_results, method = "User_Movie_Effect",
    RMSE_on_training_set = RMSE_user_movie, RMSE_on_validation_set = "NA")
```

```
kable((rmse_results[1:3, ]), "pandoc", caption = "RMSE Results",
    align = "c")
```

Table 14: RMSE Results

method	RMSE_on_training_set	RMSE_on_validation_set
Base model_Averaging	1.0607079	NA
Movie_Effect	0.9437144	NA
User_Movie_Effect	0.8661625	NA

REGULARIZED MOVIE_USER

In order to improve our predictions, we also need to consider that some movies were rated more often than others, and some users rated more movies than others. The general idea behind regularization is to constrain the total variability of the effect sizes. In regularized movie_user model, we penalize low rated movies and user who rated less frequently.

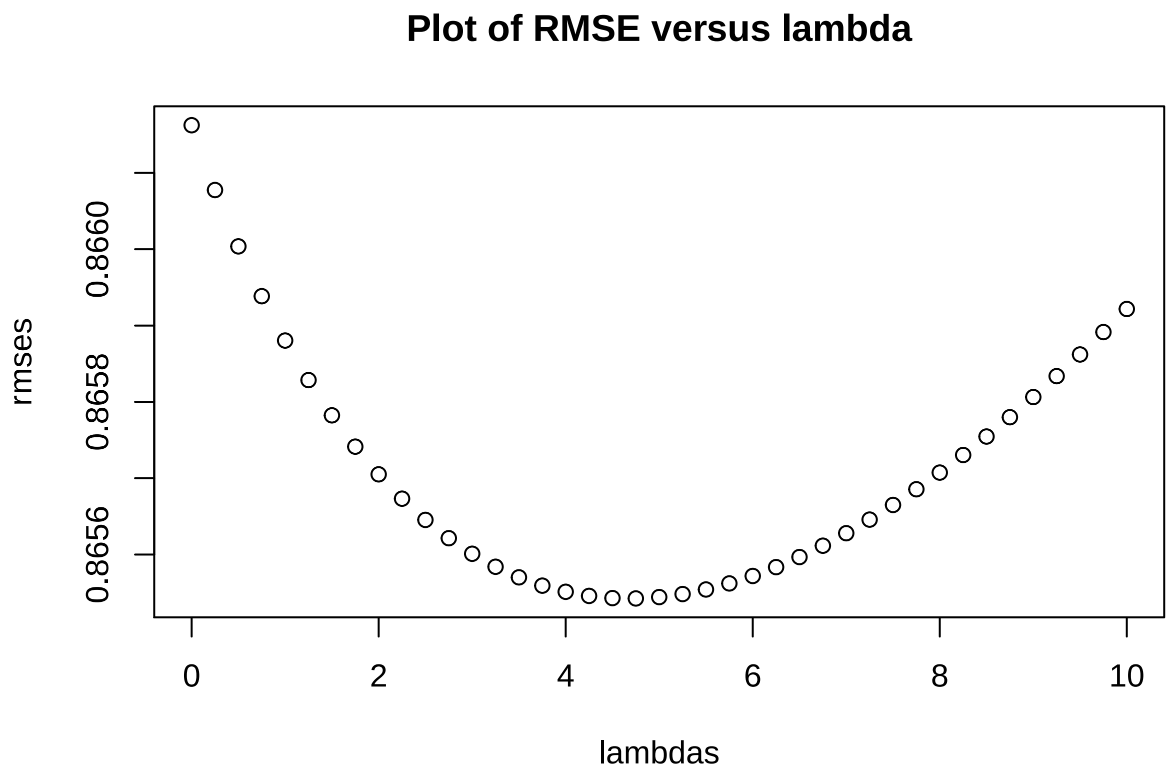
We need to add a tuning parameter to our calculation of bias as follows:

$$b_{i,u}(\lambda) = \frac{1}{\lambda + n_{i,u}} \sum_{i,u=1}^{n} (Y_{i,u} - \mu)$$

This shrinks the b_i and b_u in case of small number of ratings.

In order to determine the tuning parameter, the plot of RMSE vs lamnda is constructed as follows:

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(lam) {
    mu <- mean(training_set$rating)

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

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

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

    return(RMSE(predicted_ratings, test_set$rating))
    # Note that the test_set here is part of edx
    # dataset and is different from final validation
    # set
})

plot(lambdas, rmses, main = "Plot of RMSE versus lambda")
```



The optimal lambda is:

```r
RMSE_REG_MOVIE_USER <- min(rmses)
lam <- lambdas[which.min(rmses)]  #lambda that minimizes RMSEs for MOVIE + USER
lam
```

[1] 4.75

Now we calculate the RMSE based on regularized user and movie effects as follows:

```r
rmse_results <- add_row(rmse_results, method = "regularized_User_Movie",
    RMSE_on_training_set = RMSE_REG_MOVIE_USER, RMSE_on_validation_set = "NA")
rmse_results
```

```r
kable((rmse_results[1:4, ]), "pandoc", caption = "RMSE Results",
    align = "c")
```

Table 15: RMSE Results

method	RMSE_on_training_set	RMSE_on_validation_set
Base model_Averaging	1.0607079	NA
Movie_Effect	0.9437144	NA
User_Movie_Effect	0.8661625	NA
regularized_User_Movie	0.8655425	NA

USE THE REGULARIZED_MOVIE_USER TO PREDICT VALIDATION SET

We calculae the RMSE on validation set based on optimized lambda value:

```r
# lam OBTAINED FROM TUNING RMSE_REG_MOVIE_USER
mu <- mean(validation$rating)

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

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

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

RMSE_validation <- RMSE(predicted_ratings, validation$rating)
RMSE_validation <- round(RMSE_validation, 7)

rmse_results <- add_row(rmse_results, method = "regularized_User_Movie",
    RMSE_on_training_set = RMSE_REG_MOVIE_USER, RMSE_on_validation_set = RMSE_validation)

rmse_results
```

```r
kable((rmse_results[1:5, ]), "pandoc", digits = 7,
    caption = "RMSE Results", align = "c")
```

Table 16: RMSE Results

method	RMSE_on_training_set	RMSE_on_validation_set
Base model_Averaging	1.0607079	NA
Movie_Effect	0.9437144	NA
User_Movie_Effect	0.8661625	NA
regularized_User_Movie	0.8655425	NA
regularized_User_Movie	0.8655425	0.8395881

CONCLUSION AND FUTURE WORK

In this assignment a machine learning algorithm was successfully build in order to predict movie ratings with a subset of MovieLens dataset. It is determined that the regularized model for combined user and movie effect is sufficient to reduce RSME to 0.82585. Therefore the final model for this project is:

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

In data visualization section, it is also shown that both genre and movie age introduce bias to our predictions and the effect of this bias on rating predictions can be investigated in future work.

REFERENCE

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872