HARVARDX: PH125.9X DATA SCIENCE CAPSTONE

MOVIELENS PROJECT

Alireza Jafari Arimi
arjafari@yahoo.com

JUNE 2024

Contents

1.Executive summary

1.1 Introduction:

Recommendation systems are well-known machine learning systems that use data to predict and provide suggestions for an item or items in such a way that users can choose it from a huge number of items offered to them. In industries and marketing several high rank companies such as Netflix and Amazon use this method to rate, promote and purchase items for recommend topics of interest to potential user or buyers. In the framework of the HarvardX: PH12 Data Science capstone course, this project is focused on MovieLens data set which is publicly available for download via the GroupLens research lab at the University of Minnesota.

We utilized MovieLens dataset and applied machine learning techniques to evaluate models for predicting movie ratings and recommend movies to users as well as calculating the Root Mean Square Error (RMSE). RMSE measures the average difference between the predicted values by a model and the observed values, which acts as an indicator of the accuracy of the model. It is also called prediction errors. During this data science journey, we explore data, visualize it, analyze it, and model it for the prediction of the RMSE value.

1.2 Objective:

The aim of this project is to build a model by using machine learning and predicting the RMSE value of less than 0.86490.

1.3 Data set:

The Movielens dataset contains 10,000,054 movie ratings applied across10,677 movies by 69,878 users classified into 797 unique genres.

1.4 Key Steps

The first step in this data science project is to inspect the structure and patterns of data by studying them thoroughly. The average effect of different factors, like movies as well as user's have been displayed through creation of several graphs.

The next step involved splitting the dataset into training and test sets, as well as a validation set. After splitting the data set with enhancement of machine learning, a number of different models were tested to find one that predicted the movie rating with the minimum RMSE. In the final step, the final model was used to calculate the RMSE on the validation data set, and to avoid overfitting of the model, regularization method used by adding a penalty to the model. By using this technique, the model's RMSE is lower than the project target was achieved.

2 Method

2.1 Data loading

The MovieLens dataset is downloaded from: ttps://grouplens.org/datasets/movielens/

```
# MovieLens 10M dataset:

# https://grouplens.org/datasets/movielens/10m/

# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- "ml-10M100K.zip"

if(!file.exists(dl))

download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"

if(!file.exists(ratings_file))

unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"

if(!file.exists(movies_file))

unzip(dl, movies_file)
```

2.2 Initial Data Wrangling

The initial wrangling code is provided by the Course:

```
ratings <- as.data.frame(str_split(readLines(ratings_file), fixed("::"), simplify = TRUE),

                  stringsAsFactors = FALSE)

colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")

ratings <- ratings %>%
mutate(userId = as.integer(userId),
       movieId = as.integer(movieId),
       rating = as.numeric(rating),
       timestamp = as.integer(timestamp))
movies <- as.data.frame(str_split(readLines(movies_file), fixed("::"), simplify = TRUE),
                 stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
mutate(movieId = as.integer(movieId))
movielens <- left_join(ratings, movies, by = "movieId")
```

2.3 Creating two Data Subsets

The MovieLens dataset will be divided into two subsets:

“edx,” serving for training, developing, then selecting the best algorithm;

“final_holdout_test” that will be used as a validation set for evaluating the RMSE of the selected algorithm.

```
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
semi_join(edx, by = "movieId") %>%
semi_join(edx, by = "userId")
# Add rows removed from final hold-out test set back into edx set
```

```
removed <- anti_join(temp, final_holdout_test)

edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

2.4 Data Exploration

#DataSet Structure

After data loading, we check the edx dataset structure, here its 7 first head rows as well as it structures info:

```
str(edx)

data.frame':    9000055 obs. of  6 variables:

$ userId   : int  1 1 1 1 1 1 1 1 1 1 ...

$ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...

$ rating   : num  5 5 5 5 5 5 5 5 5 5 ...

$ timestamp: int 838985046 838983525 838983421 838983392 838983392 838984474
838983653 838984885 838983707 838984596 ...

$ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...

$ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller"
"Action|Adventure|Sci-Fi" ..
```

We can see that the edX dataset is made of 6 features as classified on table 1,for a total of about 9,000,055 observations. The validation set which represents 10% of the 10M Movielens dataset contains the same features, but with a total of 999,999 occurrences. We made sure that userId and movieId in edx set are also in validation set.

Table 1. types of variables(features) in edx data set

quantitative features	**qualitative features**
userId : numerical ,discrete, Unique ID for the user	**title:** character, nominal, movie title (not unique)
movieId: numerical, discrete, Unique ID for the movie	**genres**: character, nominal, genres associated with the movie.
timestamp: numerical, discrete, Date and time the rating was given.	
rating: numerical, continuous, a rating between 0 and 5 for the movie	

Overview of data set:

```
# Load the DT package

> library(DT)

> datatable(head(edx, 7))
```

Table 2.An overview of edx data set

	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
8	1	356	5	838983653	Forrest Gump (1994)	Comedy|Drama|Romance|War

Summary Statistics

```
library(knitr)

library('kableExtra')

summary_edx <- summary(edx)

kable(summary_edx, caption = "Summary Statistics") %>%

kable_styling(position = "center", latex_options = c("scale_down", "striped"))
```

Table 3. summary Statistics

UserId	**movieId**	**rating**	**timestamp**	**Title**	**genres**
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
1st Qu.:18124	1st Qu: 648	1stQu.:3.000	1stQu.:9.468e+08	Class character	Class: character
Median : 35738	Median : 1834	Median: 4.000	Median: 1.035e+09	Mode :character	Mode :character
Mean: 35870	Mean : 4122	Mean : 3.512	Mean : 1.033e+09	NA	NA
3rdQu.: 53607	3rd Qu.: 3626	3rdQu.: 4.000	3rd Qu.: 1.127e+09	NA	NA
Max. : 71567	Max.: 65133	Max. : 5.000	Max. : 1.231e+09	NA	NA

2.5 Data Pre-processing

For using data set in any analysis, especially in machine learning techniques in the analysis steps, the data needs to be preprocessed (e.g. cleansed, filtered, transformed and etc.)

```
# searching for missing values

colSums(is.na(edx))

userId   movieId   rating timestamp

    0       0       0       0

  title   genres

  0       0
```

The dataset is clean with no missing/NA values.

```
anyNA(edx)
[1] FALSE
```

Note: The NAs indicated in summary table are not missing values, there are a character class variable which cannot be calculated. So, there are not any missing values on the edx data set and does not need to cleaning the data.

3. Data visualization and Analysis

After data pre-processing, we start analyzing each variable and exploring the features for determining of possible effects on our goal "rating".

3.1. Exploring unique features

#Number of unique values (**users, movies and genres**)

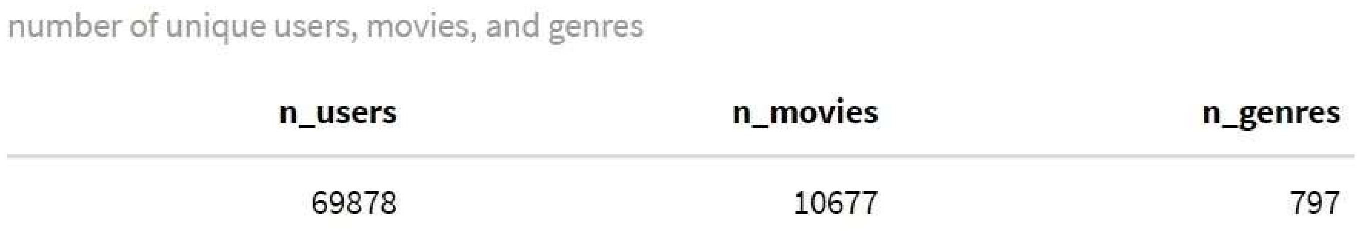
```
distinct <- edx %>% summarize(n_users = n_distinct(userId),

                    n_movies = n_distinct(movieId), n_genres = n_distinct(genres))

kable(distinct, caption = "number of unique users, movies, and genres") %>%

kable_styling(position = "center", latex_options = c("scale_down", "striped"))
```

table 4, number of unique values



number of unique users, movies, and genres

n_users	n_movies	n_genres
69878	10677	797

The above table reveals that there are 69,878 distinct users and 10,677 distinct movies as well as 707 unique genres. If we make a simple multiplication of the number of unique users (69,878) by the number of unique movies (10,677), it would yield over 746 million potential records. However, the edx dataset only contains approximately 9 million observations. This means that the ratings in the edx dataset represent just about 1.2%(9/747x100) of all conceivable ratings. This scientifically suggests that not every user rates every movie. Let's do more analysis and see the unique rating .

Unique rating

# unique rating

unique(edx$rating)

[1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5

As we can see, there are a total of 10 unique ratings ranging from 0.5 to 5 with intervals of 0.5. There is no rating with a value of 0. For further analysis we check the rating istributions of the above values and other features.

3.2 Visualization, rating distribution, and analysis of features

#Distribution Rating Values

rating_freq <- edx %>% group_by(rating) %>% summarize(count = n()) %>%

arrange(desc(count))

ggplot(rating_freq, aes(x = factor(rating), y = count))+

labs(title = "Distribution of Ratings", x = "Rating values", y = "Number of Ratings")+

geom_bar(stat = "identity", fill = "skyblue")+ scale_y_continuous(breaks = seq(0, 3000000,500000))

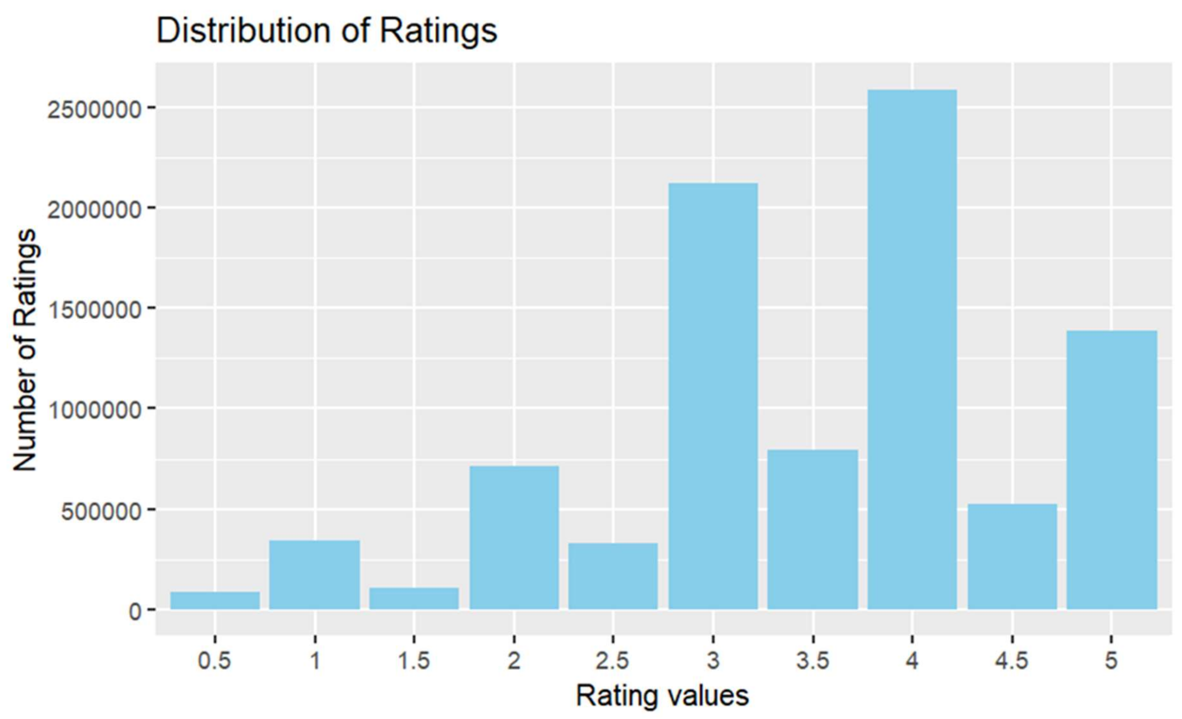


Fig 1. Histogram of distribution of rating

From above histogram. we notice the following facts:

i. The rating of “4” is the most popular rating followed by “3”, “5”, “3.5”, etc. while “0.5” is awarded the least frequently.
ii. The top 5 ratings from most to least are: 4, 3, 5, 3.5 and 2.
iii. The histogram shows that the half star ratings are less common than whole star ratings.

Now, we are going to explore the possibility of **bias on movies,user, genres and other effects on our edx data** set but before going to doing this analysis, lets do more exploration on data set.

Top 20 movies titles based on rating.

```
#bar chart of top 20 movies based on rating

#  calculate the number of ratings for each movie

movie_ratings <- edx %>%

group_by(title) %>%

summarise(count = n(), .groups = "drop")

#  top 20 movies title based on the number of ratings

 top_movies <- movie_ratings %>%   top_n(20, count)

# Create the bar chart (invers cone) of top 20 movies

ggplot(top_movies, aes(x=reorder(title, -count), y=count)) +   geom_bar(stat='identity',
fill="blue") +   coord_flip() +    labs(x="", y="Number of ratings") +
geom_text(aes(label=count), hjust=-0.1, size=3) +   labs(title="Top 20 movies title based on
number of ratings", caption = "source data: edx set")
```

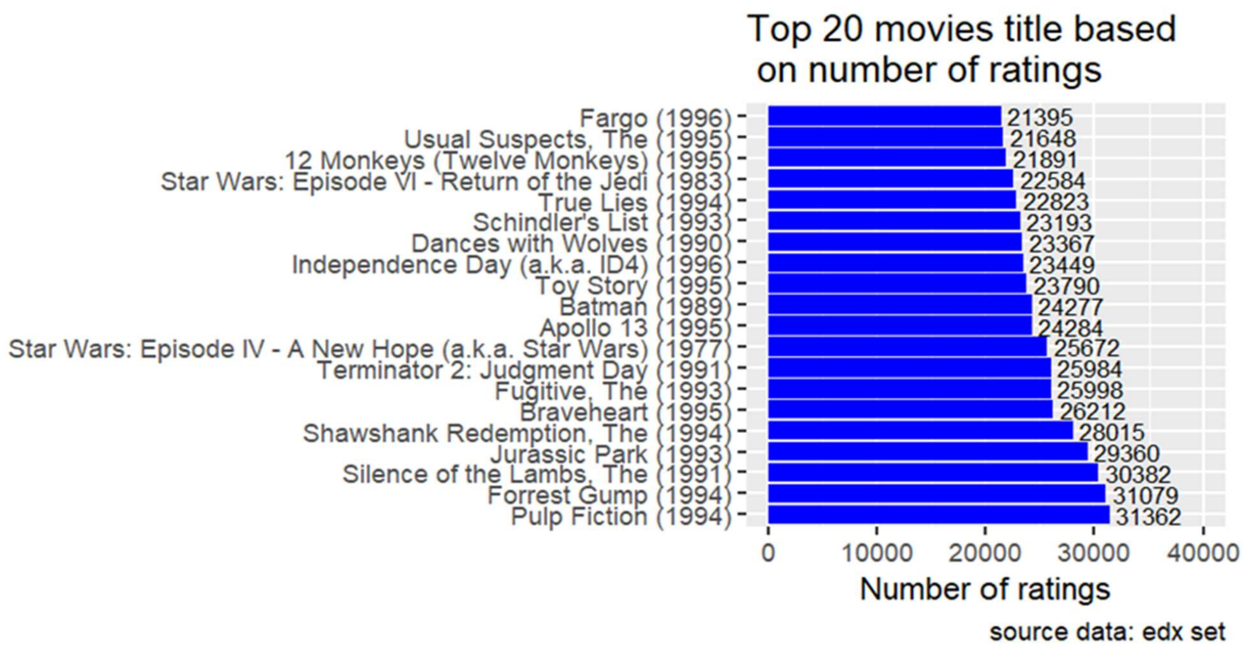


Fig 2. Top 20 movie titles based on number of ratings.

As can be seen from the above cone of the top 20 movies "title" characteristic aligns with the prior analysis. The films that have gained the most ratings fall into the top genre categories. For instance, films such as Pulp Fiction (1994), Forrest Gump (1994), Silence of the Lambs, Jurassic Park (1993) and Shawshank Redemption, which are among the top 5 in terms of the number of movie ratings, belong to the Drama, Comedy, or Action genres.

Top 20 movies based on genres.

```
top_genres <- edx %>%

group_by(genres) %>%

summarize(n = n(), .groups = "drop") %>%

arrange(desc(n))

# Select the top 20 genres

top_20_genres <- top_genres %>%   top_n(20)

# Create the plot

ggplot(top_20_genres, aes(x = reorder(genres, n), y = n)) +

geom_bar(stat = "identity") +   coord_flip() +   labs(x = "Genres", y = "Movie Count", title =
"Top 20 Genres by Movie Count") +
```

theme_minimal()

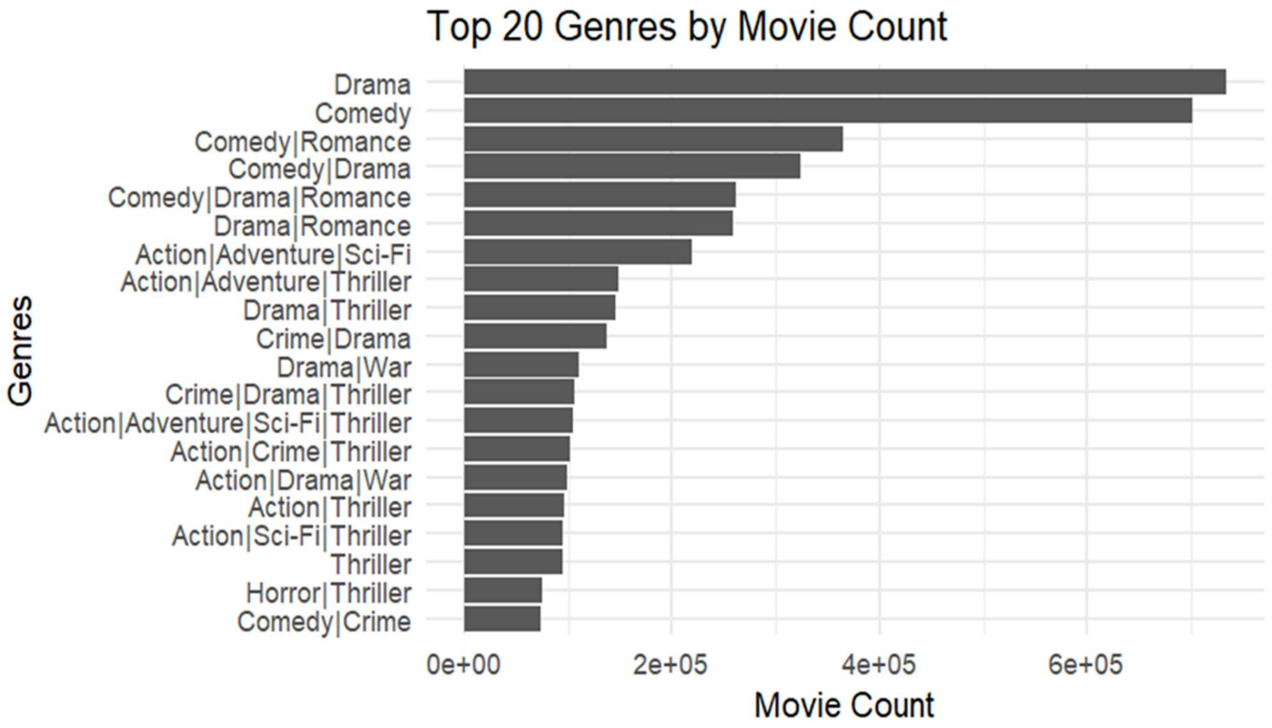


Fig 3. Top 20 popular genres

The above plot shows obviously that most of movies were with genre of “Drama” and “Comedy” respectively and only few of them were with genre of “Thriller. Then we plot the percentage of top 20 genres.

Calculate the number of movies percentage per genre

top_genres <- edx %>%

group_by(genres) %>%

summarize(n = n(), .groups = "drop") %>%

mutate(total = sum(n), percentage = n / total * 100) %>%

arrange(desc(percentage))

# Select the top 20 genres

top_20_genres <- top_genres %>%   top_n(20, percentage)

# Create the plot

ggplot(top_20_genres, aes(x = reorder(genres, percentage), y = percentage)) +

```
  geom_bar(stat = "identity", fill = "steelblue") +

  coord_flip() +   labs(x = "Genres", y = "Percentage", title = "Top 20 Genres by Movie
Percentage") +

theme_minimal()
```

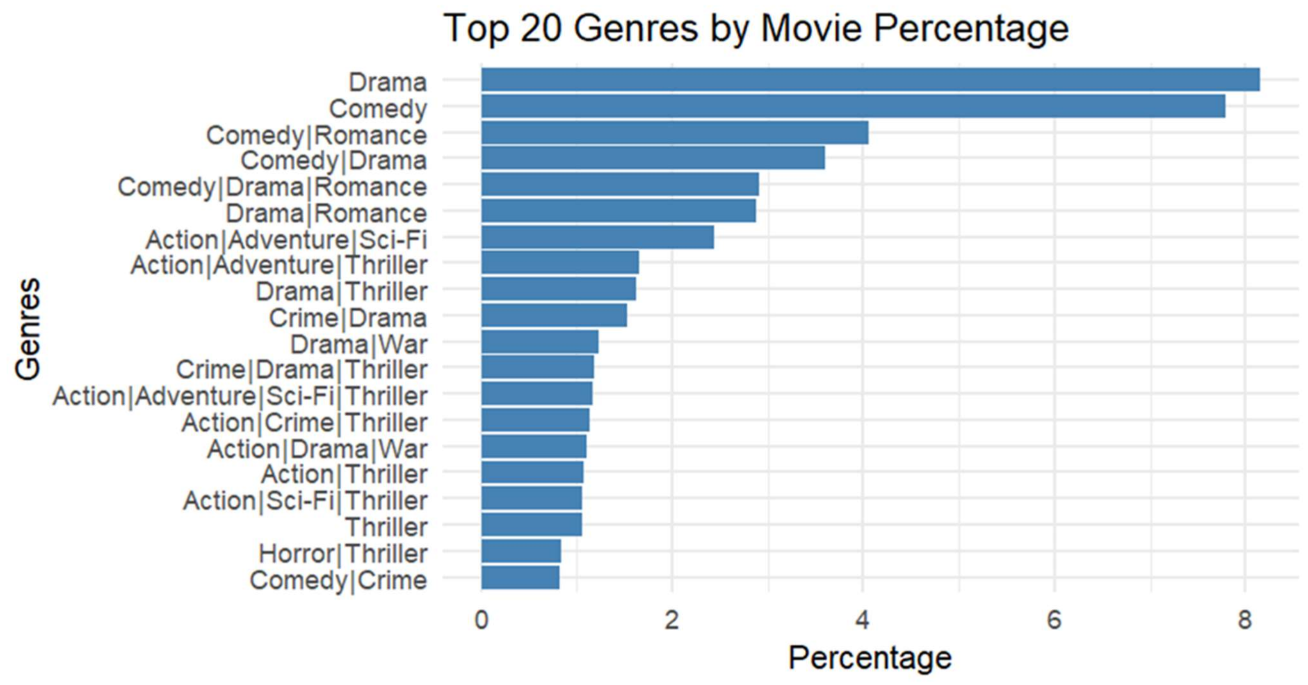


Fig 4. Percentage of top 20 movie genres

As figure 4 shows, we calculate the total number of movies and the percentage of movies for each genre. The graph provides insights into the popularity distribution of different genres, indicating which are most rated within the dataset. This can help to compare the relative frequency of genre ratings, with Drama-related genres appearing notably. The graph can be a useful tool for analyzing trends in genre popularity and make decisions in the film industry.

Top 20 movie genres by average rating

```
avg_rating_genres <- edx %>%

group_by(genres) %>%   summarize(avg_rating = mean(rating), .groups = "drop") %>%

arrange(desc(avg_rating))

# Select the top 20 genres

top_20_genres <- avg_rating_genres %>%   top_n(20, avg_rating)
```

# Create the plot

ggplot(top_20_genres, aes(x = reorder(genres, avg_rating), y = avg_rating, fill = avg_rating)) +

geom_bar(stat = "identity") +   coord_flip() +   labs(x = "Genres", y = "Average Rating", title = "Top 20 Genres by Average Rating") +   theme_minimal() +   scale_fill_gradient(low = "blue", high = "red")

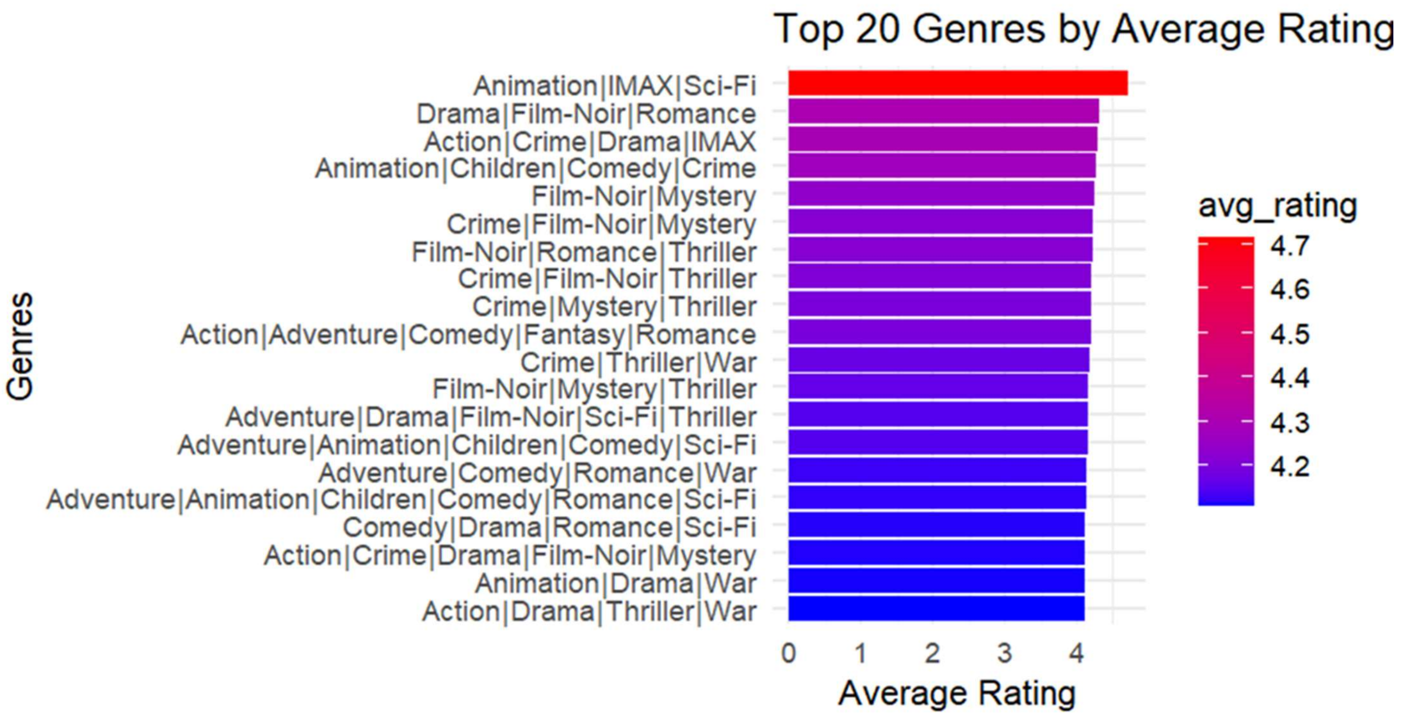


Fig 5. Top 20 movie genres based on average rating

The graph shows that some genres such as “Animation IMAX” and “Dama Film-Noir Drama” have higher ratings than the general average ratings while others such as ““Animation IDrama”” and “Action” have lower ratings than the general average ratings.

After above exploration, now, we come back to do explore the possibility of bias on movies and user and other effects on our edx data set

#Distribution of ratings by movieID

# Number of rating by movieID

edx %>%

group_by(movieId) %>%

summarize(n = n(), .groups = "drop") %>%

ggplot(aes(n)) +   geom_histogram(bins = 30, color = "blue") +   scale_x_log10() +

xlab("Number of ratings") +   ylab("Number of movies") +   ggtitle("Number of ratings given by movieID")

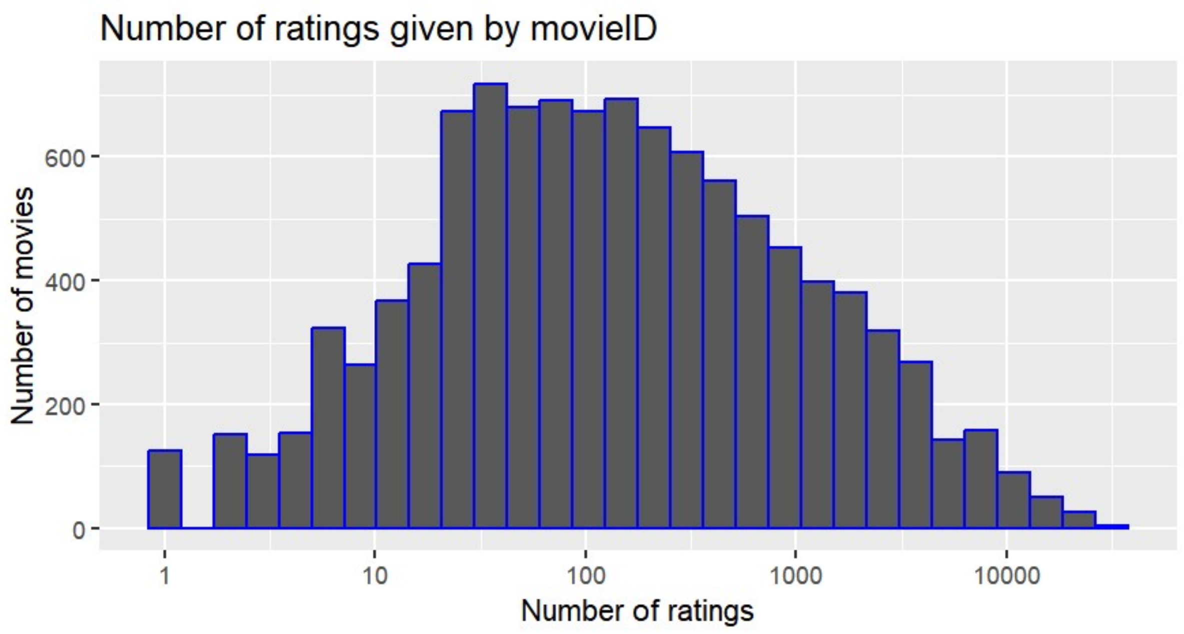


Fig 6. number of ratings received by different movies(movie ID)

This histogram also reveals that there is a variation in the frequency of ratings received by different movies as well as certain movies with more ratings than others. Similarly, some users gave more rating movies compared to others (following histogram). These variations can potentially be related to the effects of movies and users.

Distribution of rating by user ID

# userid rating

edx %>%

group_by(userId) %>%   summarize(n = n(), .groups = "drop") %>%

ggplot(aes(n)) +   geom_histogram(bins = 30, color = "blue") +   scale_x_log10() +

xlab("Number of ratings") +   ylab("Number of users") +   ggtitle("Number of ratings given by users")

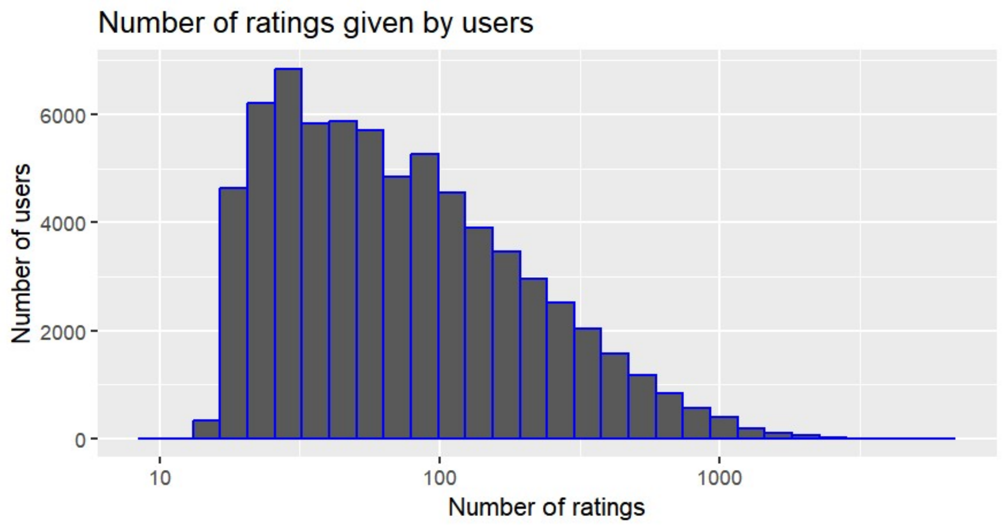


Fig 7. Rating distribution by different users (user ID**)**

Plot rating by user ID also shows that some movies get rated more than others, and some users are more active than others at rating movies. This also verifies the presence of movies and users effects.

Up to now, we've investigated the possible influences on ratings based on factors such as the movie and user. With the insights gained from this analysis, we're now ready to progress to the next phase, which involves building models and calculation of RSME.

4. Modeling and calculation of RSME

Let's assume a model that predicts ratings, we have two vectors: actual ratings and predicted ratings. actual ratings from the edx dataset and predicted ratings, predicted by our model. The first step of this analysis is data splitting .

4.1 Data Splitting for modeling

In order to modeling and analysis, we split the data in two parts: edx and validation dataset, with 90% allocated to edx, 10% to the validation. The edx part will be used for training/test set of various models and machine learning with portion of 80/20. The validation data is the final holdout dataset and will be used on the chosen algorithm for performance evaluation using RMSE.

```
# Final hold-out test set will be 10% of edx data set as mentioned in the course and will be served as  validation data set.

validation_set <- final_holdout_test

# creating training/test(80/20)  set
```

```
set.seed(123)

#loading libraries for data partitioning process

library(caret)

library(lattice)

#creating an index to pick a sample for training and test set

edx_test_index <- createDataPartition(y = edx$rating,   times = 1, p = 0.2, list = FALSE)

training_set <- edx[-edx_test_index, ] #creating the training set

test_set <- edx[edx_test_index, ] #creating the test set
```

4.2 Models:

The Root Mean Square Error (RMSE) measures the discrepancy between the predicted values from a model and the actual observed values. In other words, it's a measure of the model's accuracy. The lower RMSE is better for predicting actual value. The calculation of RMSE is as follows:

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

Σ is the sum of the series.

$\hat{y}_{u,i}$ is the predicted value for the ith observation in the dataset.

$y_{u,i}$ is the observed value for the ith observation in the dataset.

N is the sample size

4.2.1 Model1: Modeling with average rating :

For the first simple model, we assume that the predicted value is always the mean value of all ratings from the dataset. There are no differences between users or movies in this setting.the equation of first model will be like follows.

Yu,i=μ+εu,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.

```
# Creating RMSE Functions

RMSE <- function(true_ratings, predicted_ratings) {
```

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

The simple estimate μ of μ is the mean of all ratings in training set, or mu:

```
mu <- mean(training_set$rating)
>   mu
[1] 3.512371
```

This model gives the following RMSE values when applied to edx_test:

```
RMSE1 <- RMSE (test_set$rating, mu)
RMSE1
```

[1] 1.060012   which is too high. So, we consider the movie effect for second model.

4.2.2 Model 2 : Movie effects modeling

For the second model, we consider movie bias (effects). It will be like the following equation.

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

Yu,i = predict the rating , μ= average rating , b_m=movie bias and $\epsilon_{u,i}$=independent error.

```
# Calculate the bias for each movie
bm <- training_set %>% group_by(movieId) %>%
summarize(bm = mean(rating - mu))
# Plot the distribution of movie bias
ggplot(bm, aes(x = bm)) +
geom_histogram(bins = 20, fill = "green", color = "black") +
labs(title = "Distribution of movie bias", x = "Movie Bias(bm)", y = "Number of movies") +
theme_minimal()
```

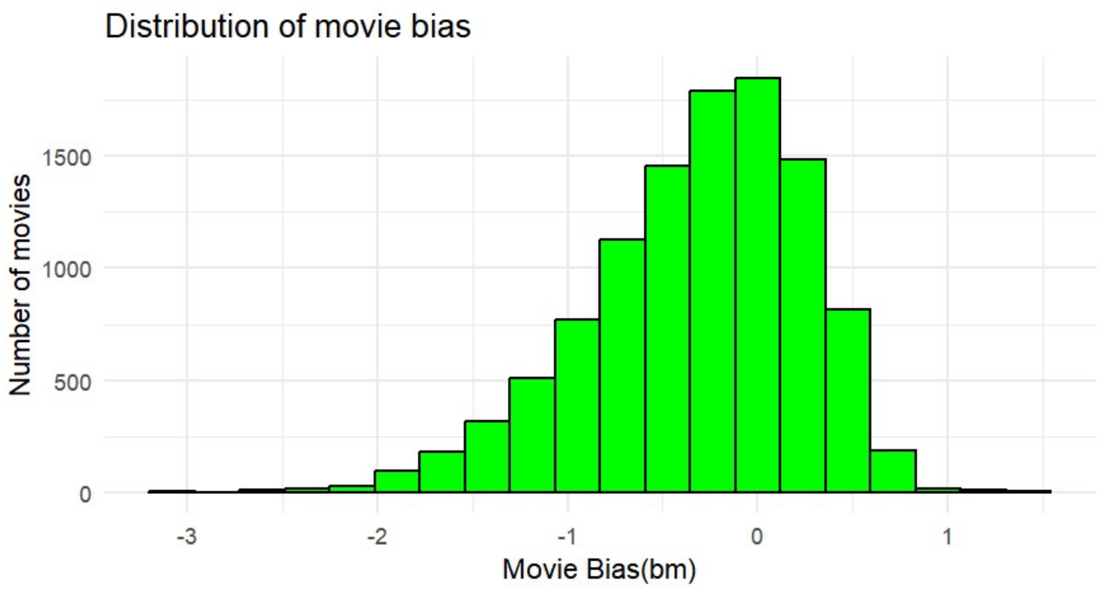


Fig 8. Distribution of movie bias(effect)

```
# Join bm to the test set

test_set <- test_set %>%

left_join(bm, by = "movieId")

# Predict ratings as mu + bm

bm_prediction <- mu + test_set$bm

# Calculate RMSE on the ratings.

 RMSE_movies <- RMSE(test_set$rating, bm_prediction)

RMSE_movies

[1] 0.9436204

RMSE_table <- data.frame(     RMSE = c(RMSE1, RMSE_movies),

Stage = c("Rating model ", " Movie effects model ")    )

library(knitr)

library(kableExtra)

RMSE_table <- data.frame(   RMSE = c(RMSE1, RMSE_movies),

Stage = c("Average rating model", "Movie effects model") )
```

RMSE_table %>%

kable("html") %>% kable_styling("responsive")

Table 5. RMSE of simple and movie effects model

RMSE	Stage
1.0600125	Average rating model
0.9436204	Movie effects model

The RSME of Movie effects model is **0.9436204,** approximately **0.943** which is quite lower than first model.

4.2.3 Model3: Modeling with combination of movie and user effects

The third model combines movie and user bias., By using the same approach with the user effects, it leads to the following equation,

$Y_{u,i} = \mu + b_m + b_u + \epsilon_{u,i}$ (3)

b_m=movie effect and b_u = user effect

As we show on plat (User ID), some movies get rated more than others, and some users are more active than others at rating movies. **This should apparently explain the presence of movies effects and users effects.** We compute the average rating for user (bu).

# Calculate user(bias) average rating as bu

bu <- training_set %>% left_join(bm, by = "movieId") %>% group_by(userId) %>%

summarize(bu = mean(rating - mu - bm))

# Create plot(histogram) of user bias

ggplot(user_avgs, aes(x = bu)) +

geom_histogram(bins = 20, fill = "green", color = "black") + labs(title = " Distribution of User Bias ", x = "bu", y = "Number of movies") +

theme_minimal()

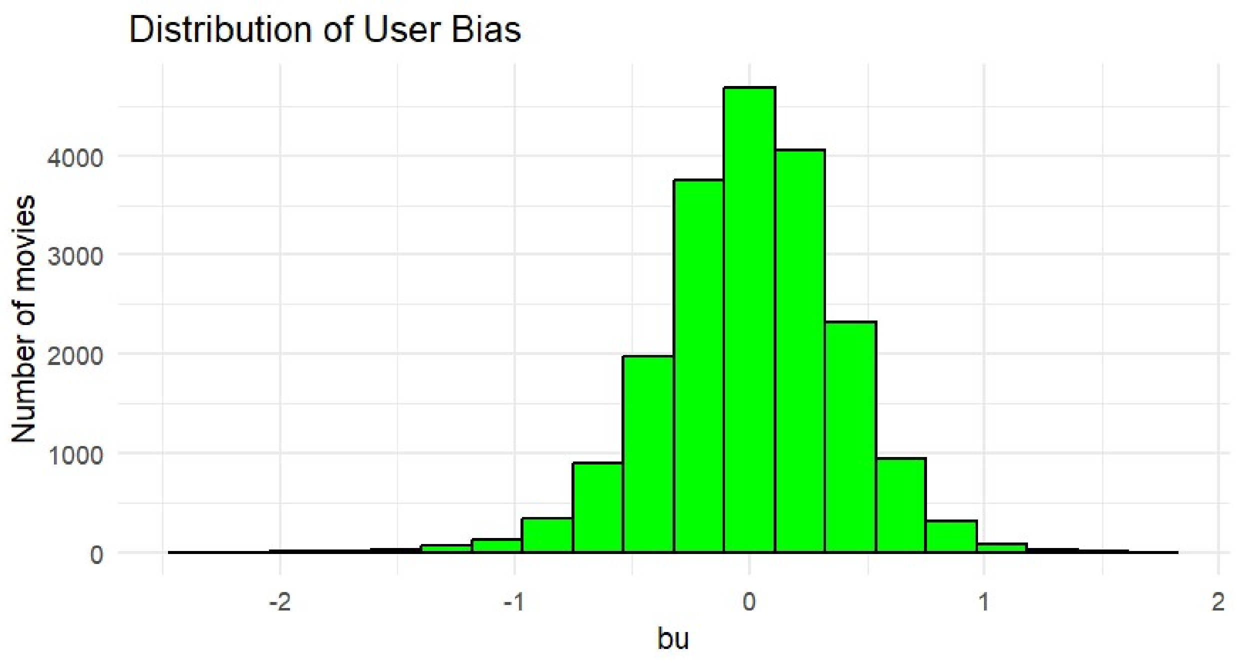


Fig 9. Distribution of user bias(effect)

```
  # Construct predictors and improves RMSE
bu_prediction <- validation_set %>%
 left_join(bm, by='movieId') %>%
 left_join(bu, by='userId') %>%
 mutate(pred = mu + bm + bu) %>%  .$pred
# Calculation of RMSEs
RSME_user_movies <- RMSE(bu_prediction, validation_set $rating)
RSME_user_movies
 [1] 0.8665207
```

The third model, which contains both movie and user effects, yields a Root Mean Square Error (RMSE) of 0.8665, demonstrating a more accurate prediction compared to the model with solely movie effects, which has an RMSE of 0.943. The result present in the table 6.

```
 RMSE_table <- data.frame(   RMSE = c(RMSE1, RMSE_movies,RSME_user_movies)
                          , Stage = c("Average rating model", "Movie effects model","Movie_User effects
model" )  )
 RMSE_table %>%
   kable("html") %>% kable_styling("responsive")
```

Table 6. RMSE of different stages modeling

RMSE	Stage
1.0600125	Average rating model
0.9436204	Movie effects model
0.8665207	Movie_User effects model

As can be seen from the above table, the new RMSE by considering movieId and userId effects together drops from 0.9436 to 0.8665. By using these two effects we improve predication accuracy compared to only movie effects and simple average rating model, but it needs further consideration. In order to reach the goal value 0.8649 or lower, we should take into account the overfitting may happen when we use the models, so for avoiding overfitting, we use regularization concept which permits to penalize large estimates that come from small sample sizes. Regularization is a method used to reduce the effect of overfitting.

4.2.4 Model4: Modeling user + movie effects using regularization.

Here a penalty term(lambda) can be applied for tuning the model and effects. For our model the terms are squared. Regularization applied to movie and user effects (bm and bu). To determine the regularization parameter lambda, we also take into consideration the different number of ratings per movie and per user.

The regularization for movie and user effects is given by:

$$Y_{u,i} = \mu + \frac{1}{\lambda + n_m} b_m + \frac{1}{\lambda + n_u} b_u + \epsilon_{u,i}$$

In this case, we minimize the following objective function:

$$\sum_{u,i} ⬚ (y_{u,i} - \mu - b_m - b_u)^2 + \lambda(\sum_{i} ⬚ b_m^2 + \sum_{u} ⬚ b_u^2)$$

In these formulas:

- (Y{u,i}) is the predicted rating of user (u) for item (i).
- (μ) is the overall average rating.
- (bmi) and (bu) are the bias terms for movie (m) and user (u), respectively.
- (lambda) is the regularization parameter.
- (nm) and (nu) are the number of ratings for movie (m) and user (u), respectively.
- (epsilon{u,i}) is the residual error for user (u) and movie (i).

Lambda has two benefits:

- ➢ can be used to prevent over-learning in any analysis method.
- ➢ As a regularization parameter adjusts the influence of the regularization term.

# Create a sequence of values for lambda ranging from 0 to 10 with 0.25 increments

# regualization with lambda

# Create a sequence of values for lambda ranging from 0 to 10 with 0.25 increments

lambda <- seq(0, 10, 0.25)

# After building the regularization model, the ratings are predicted and the RMSE at each lambda value.

# Apply and validate the regularized model against the validation data set

RMSES <- sapply(lambda, function(l){

bm <- edx %>%

group_by(movieId) %>%

summarise(bm = sum(rating - mu)/(n()+l))

bu <- edx %>%

left_join(bm, by="movieId") %>%

group_by(userId) %>%

summarise(bu = sum(rating - bm - mu)/(n()+l))

# Ratings prediction for validation_set

predicted_ratings <- validation_set %>%

```
left_join(bm, by="movieId") %>%
left_join(bu, by="userId") %>%
mutate(pred = mu + bm + bu) %>%  pull(pred)
# Calculate RMSE and evaluate accuracy
return( RMSE(validation_set $rating, predicted_ratings))
})
RMSE_Rg_Movie_User <- min(RMSES)
RMSE_Rg_Movie_User
[1] 0.864817
#Draw a plot of RMSES verses lambda to see this value.
# Plot RMSES vs lambdas to select the optimal lambda
qplot(lambda, RMSES,  color = I("#51A8FF"),       main = "RMSEs vs. Lambdas") +
theme_minimal()
```

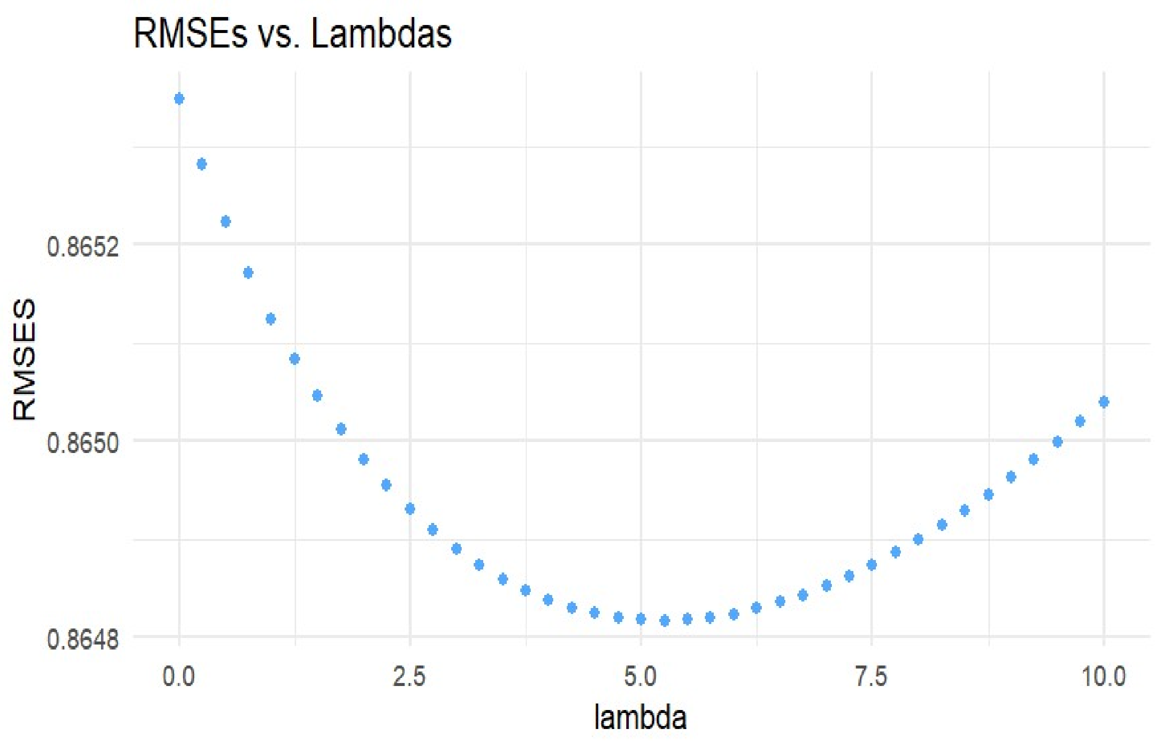


Fig 10. RMSEs plot verses Lambda

```
#lambda that minimizes RMSEs for Movie + User

lambda <- lambda[which.min(RMSES)]

lambda

 [1] 5.25
```

The optimal regularization parameter(lambda) is 5.25.

5.Results:

The result of RMSE values from applied models are shown in Table 7

```
RMSE_table <- data.frame(   RMSE = c(RMSE1, RMSE_movies,RSME_user_movies,
RMSE_Rg.Movie_User) , Stage = c("Average rating model", "Movie effects
model","Movie_User effects model","Rg.Movie_User effects model" )  )

 RMSE_table %>%

   kable("html") %>% kable_styling("responsive")
```

Table 7. RSME values with different method of modeling

RMSE	Stage
1.0600125	Average rating model
0.9436204	Movie effects model
0.8665207	Movie_User effects model
0.8648170	Rg.Movie_User effects model

6. Conclusion

In conclusion, the machine learning process of this MovieLens project has been a success. The objective of the project was to build a model that could predict movie ratings with a Root Mean Square Error (RMSE) of less than **0. 8649**.Through the regularization techniques on both movie and user effects, we were able to achieve an RMSE of **0.8648**, which is lower than our project goal. This indicates that our model's(model4) predictions are quite close to the actual values, demonstrating its effectiveness.

7. Future work

The success of this project opens up opportunities for further improvements and exploration. For instance, we could consider incorporating additional features such as movie genres, user demographics, years, and ages to enhance the model's performance but in our case the goal has been achieved. As a future work, other methods that could be used for this task are advanced machine learning techniques, deep learning models, principal component analysis (PCA) and matrix factorization approach.

8. References

GroupLens. 2009. "MovieLens 10M Dataset."
https://grouplens.org/datasets/movielens/10m/

Most of the content material was learned through Professor Rafael Irizarry and his team from HarvardX's Data Science course, book, and GitHub.

Irizarry, Rafael A., "Introduction to Data Science: Data Analysis and Prediction Algorithms in R" https://rafalab.github.io/dsbook/

https://rafalab.dfci.harvard.edu/dsbook/large-datasets.html#regularization