MovieLens Project Report

Raj Krishna, PMP, PMI-ACP, CSM

2019-11-05

Introduction

The objective of the MovieLens project is to build a recommendation system using the MovieLens dataset. The MovieLens dataset was created by the GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota.

The full dataset contains 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users[1]. For the purpose of this project we would be using a smaller stable benchmark 10M movie ratings dataset that contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens.

The MovieLens dataset is split into two datasets; the edx dataset, used to train the algorithms and predict the movie ratings in, the second, the validation dataset. Root Mean Squared Error (RMSE), also known as Residual Mean Squared Error, is used to gauge the accuracy of the said predictions. To avoid overtraining, the validation dataset is used only to report the **final** RMSE.

During the course of this project, various Machine Learning models would be assessed and their respective RMSEs would be compared to arrive at the final recommended model which would then be used to report the final RMSE using the validation dataset. The goal is to achieve **RMSE $\leq$ 0.8649**

Datasets

The MovieLens dataset is downloaded and split into **edx** and **validation** datasets. edx dataset contains 90% of the MovieLens dataset while the validation dataset contains 10%. Each of the datasets has 6 components namely *userId*, *movieId*, *rating*, *timestamp*, *title*, and *genres*. During this process we'll also ensure that the *userID* and *movieId* in the validation dataset are also present in the edx dataset.

Overview

Following code is used to download the MovieLens dataset and split it into two:

Load necessary libraries

```
##################################
# Create edx set, validation set #
##################################

# Note: this process could take a couple of minutes
# Installing/loading libraries necessary for the project and the report
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(kableExtra))
```

[1]Source: https://grouplens.org/datasets/movielens/latest/; data as of 2019-10-27

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

Download 10M MovieLens dataset and split to create edx and validation sets

```
# 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 <- 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
# if using R 3.5 or earlier, use `set.seed(1)` instead
set.seed(1, sample.kind="Rounding")
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)
```

Remove temporary objects

```
# Remove the objects that are not needed for further processing
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Both edx and validation datasets contain the following 6 variables:

1. **userId**: Anonymizes MovieLens user ids
2. **movieId**: MovieLens movie id
3. **rating**: Rating given by individual user and is made on a 5-star scale, with half-star increments
4. **timestamp**: Represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970
5. **title**: Title of the movie
6. **genre**: Pipe-separated list of genre(s) the movie belongs to

Each row represents the rating given to a single movie by a single user

edx

```
glimpse(edx, 85)
```

```
## Observations: 9,000,055
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, 46...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 8389844...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "Sta...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci-F...
```

As depicted by the snippet above, edx dataset contains ***9,000,055*** rows and 6 variables. The edx dataset would be further split into various train and test sets in order to train the various models and predict the ratings(Y).

validation

```
glimpse(validation, 85)
```

```
## Observations: 999,999
## Variables: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 85, 17...
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3.0, 3....
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 8682459...
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alone (1...
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|Comedy...
```

The validation dataset contains ***999,999*** rows and 6 variables. The validation dataset would only be used to obtain the final prediction and the RMSE therein.

Additional Datasets

As part of the assessment process, various additional datasets were necessitated. For ease of reference and in order to speed up code and reduce the processing time, they are being created and listed below. Where data partition is created, care has been taken to ensure that the *userID* and *movieId* in one partition is also present in the other.

```
set.seed(1, sample.kind="Rounding")
# creating additional train and test set from edx data
# obtaining test index with 20% of the edx data
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
temp <- edx[test_index, ]
train_set <- edx[-test_index, ]
```

```
# Ensuring movieId and userId in train set are also in test set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
```

1. train_set

```
dim(train_set)
```

```
## [1] 7200089       6
```

The **train_set** Contains 80% of the edx dataset. The train_set is used to train various models and predict ratings for the test set.

2. test_set

```
dim(test_set)
```

```
## [1] 1799966       6
```

The **test_set** contains 20% of the edx dataset. The test_set is used to validate ratings predicted using train_set and to calculate the RMSE.

3. genre_train_set

```
genre_train_set <- train_set %>%
  separate_rows(genres, sep = "\\|")

dim(genre_train_set)
```

```
## [1] 18696224        6
```

The **genre_train_set** is derived from the train_set where the pipe-separated genre variable has been split into separate rows.

4. genre_test_set

```
genre_test_set <- test_set %>%
  separate_rows(genres, sep = "\\|")

dim(genre_test_set)
```

```
## [1] 4675199       6
```

The **genre_test_set** is derived from the test_set where the pipe-separated genre variable has been split into separate rows.

5. genre_edx

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

```
dim(genre_edx)
```

```
## [1] 23371423        6
```

The **genre_edx_set** is derived from edx where the pipe-separated genre variable has been split into separate rows.

6. genre_validation

```
genre_validation <- validation %>%
  separate_rows(genres, sep = "\\|")

dim(genre_validation)
```

```
## [1] 2595771       6
```

The **genre_validation_set** is derived from edx where the pipe-separated genre variable has been split into separate rows.

Data Exploration

Before we dive deep into the project, let's explore the data that we have a bit. We'll be using the edx data set for this purpose.

Number of distinct users and movies

Let's first see how many distinct users and movies are there in the dataset.

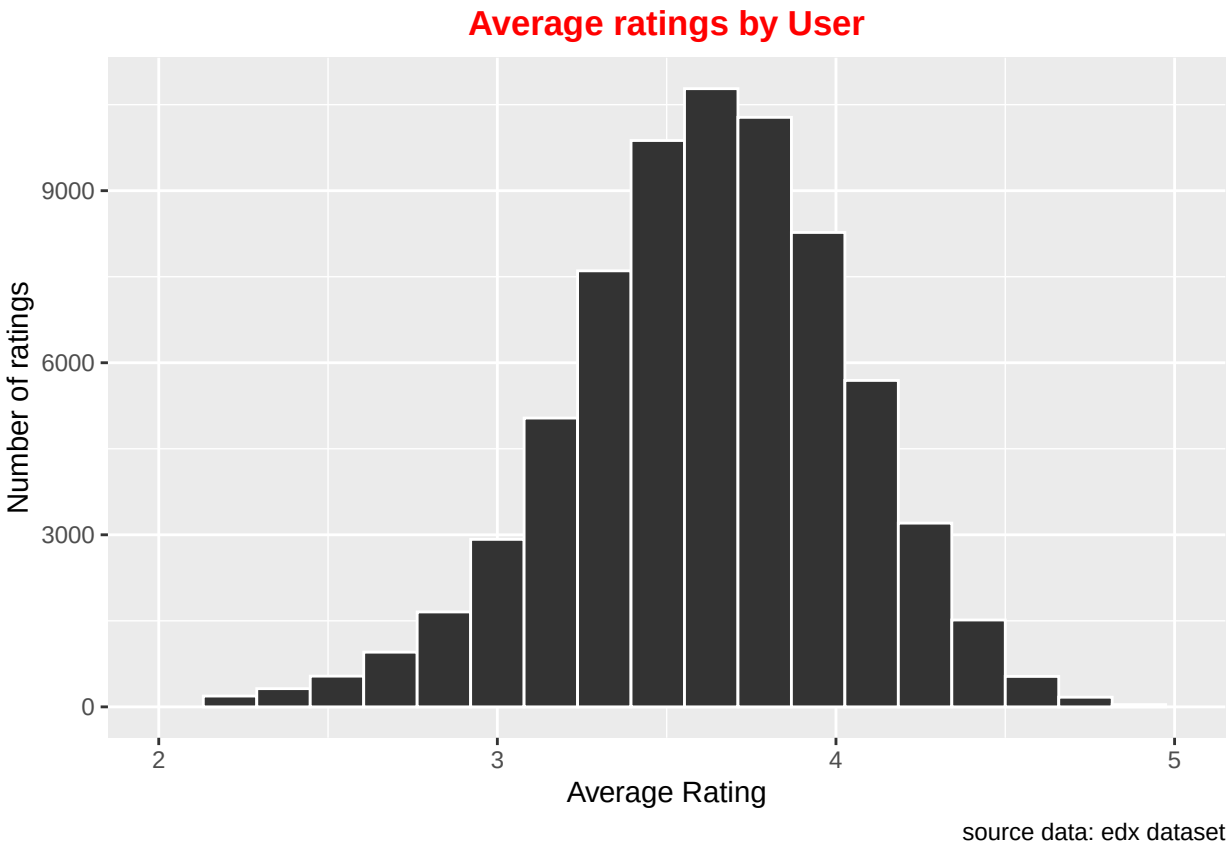
```
edx %>%
  summarise(Movies = n_distinct(movieId),
            Users = n_distinct(userId)) %>% knitr::kable()
```

Movies	Users
10677	69878

So, there are **10,677** movies and **69,878** users in the edx dataset. If each user had rated each of the movies, we should have a **746,087,406** records in out dataset where as we only have **9,000,055** records. We can infer from this that not all users have rated all the movies in the dataset.

Average ratings by User

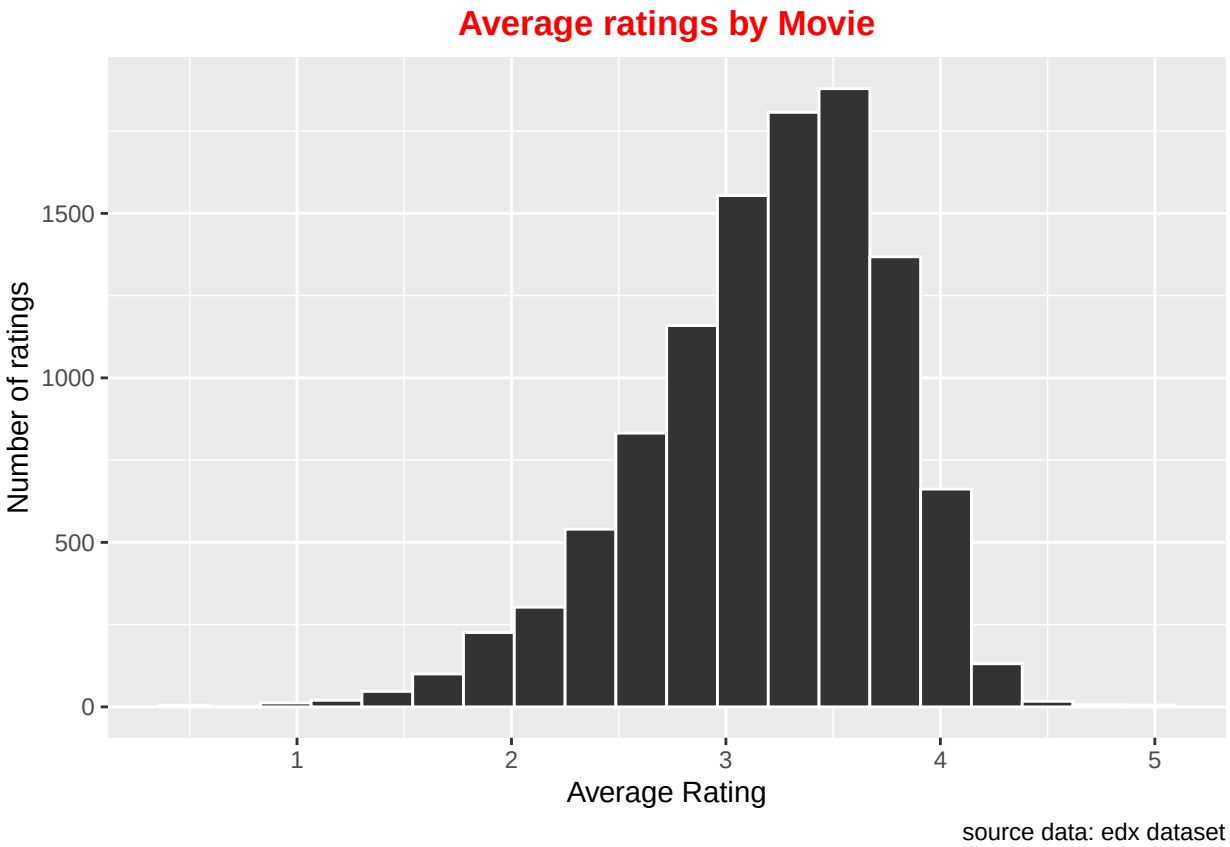
```
# Plot to show the average rating by user
edx %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(aes(b_u),bins = 20, fill = "#333333", color = "white") +
  scale_x_continuous(limits = c(2, 5)) +
  labs(x = "Average Rating", y = "Number of ratings",
       title = "Average ratings by User",
       caption = "source data: edx dataset") +
  theme(plot.title = element_text(color = "red", face = "bold", hjust = 0.5))
```



We can see variablity in the rating based on the users. Some users give high ratings to movies while others seem more cautious with their ratings. This would be another effect that we should be analyzing.

Average rating by Movie

```
# Plot to show the average rating by movie
edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating)) %>%
  ggplot(aes(b_i)) +
  geom_histogram(aes(b_i),bins = 20, fill = "#333333", color = "white") +
  labs(x = "Average Rating", y = "Number of ratings",
       title = "Average ratings by Movie",
       caption = "source data: edx dataset") +
  theme(plot.title = element_text(color = "red", face = "bold", hjust = 0.5))
```

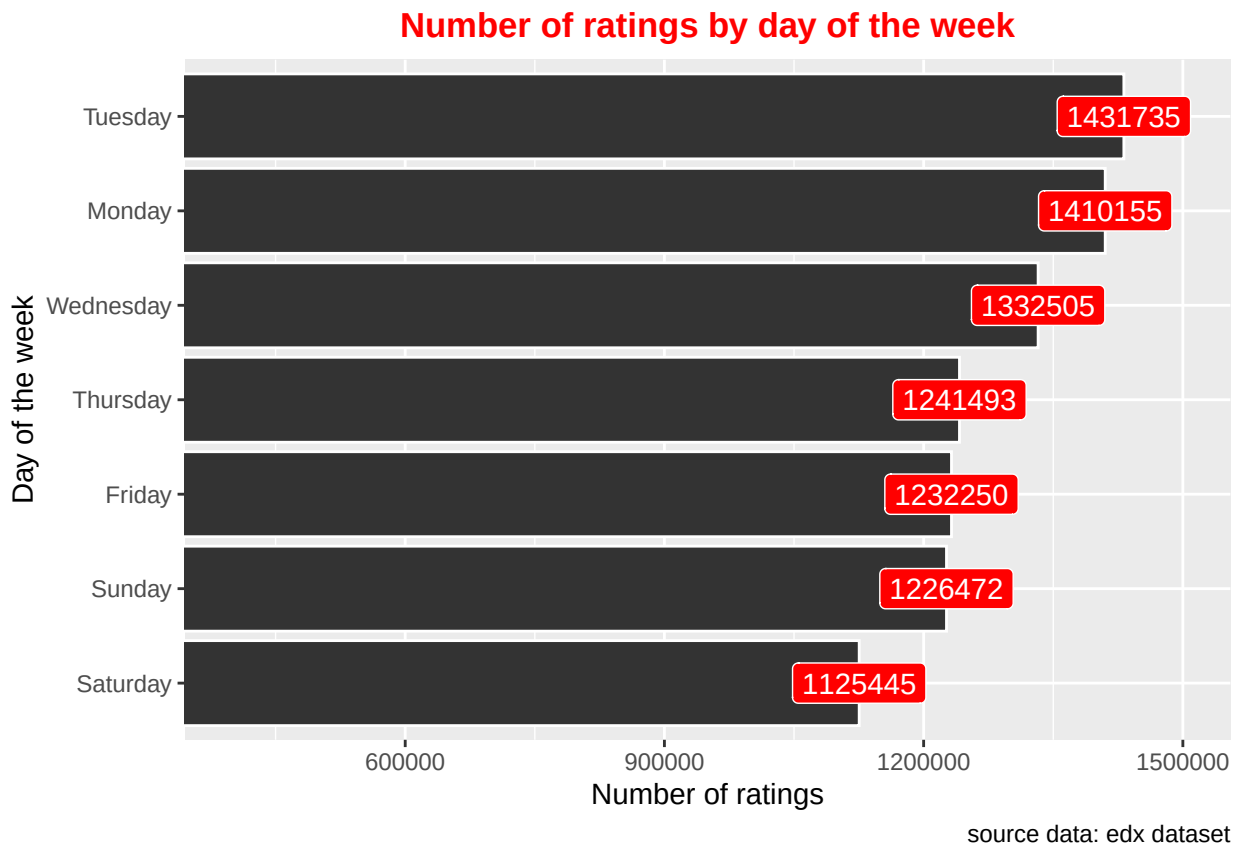


As we can see there is variability in the average rating based on the movie. Some movies are rated higher than others. We would explore this effect later in the project.

Day of the week

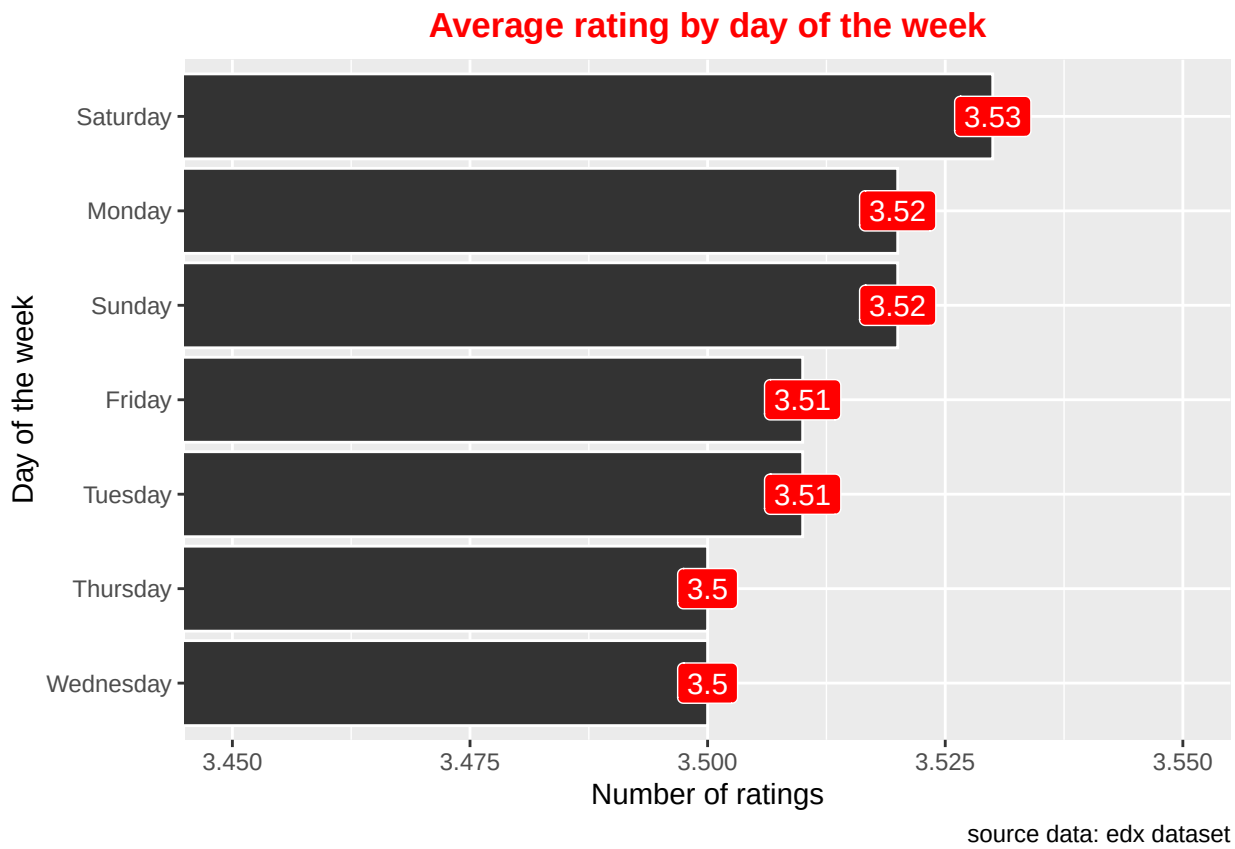
The ratings for a movie could be dependent on when it was rated, let's explore this possibility.

```
# Plot to show the number of ratings by Weekday
edx %>%
  mutate(Weekday = lubridate::wday(lubridate::as_datetime(timestamp),
                                   label = TRUE, abbr = FALSE)) %>%
  group_by(Weekday) %>%
  summarise(n = n()) %>%
  ggplot(aes(reorder(Weekday, n), n)) +
  coord_flip(y = c(400000, 1500000)) +
  geom_bar(stat = "identity", fill ="#333333", color = "white") +
  labs(x = "Day of the week", y = "Number of ratings",
       title = "Number of ratings by day of the week",
       caption = "source data: edx dataset") +
  geom_label(aes(label = n),fill = "red", color = "white") +
  theme(plot.title = element_text(color = "red", face = "bold", hjust = 0.5))
```



Interestingly, most movie ratings seem to have been given on a Tuesday. Now, let's see if there is any impact of the day of the week when the movie was rated on the rating itself.

```
# Plot to show the average ratings by weekday
edx %>%
  mutate(Weekday = lubridate::wday(lubridate::as_datetime(timestamp),
                                   label = TRUE, abbr = FALSE)) %>%
  group_by(Weekday) %>%
  summarise(avg_rating = round(mean(rating),2)) %>%
  ggplot(aes(reorder(Weekday, avg_rating), avg_rating)) +
  coord_flip(y = c(3.45, 3.55)) +
  geom_bar(stat = "identity", fill ="#333333", color = "white") +
  labs(x = "Day of the week", y = "Number of ratings",
       title = "Average rating by day of the week",
       caption = "source data: edx dataset") +
  geom_label(aes(label = avg_rating),fill = "red", color = "white") +
  theme(plot.title = element_text(color = "red", face = "bold", hjust = 0.5))
```



As we can see there does seem to be some impact due to the day of the week on the ratings, but it may not be significant. We'll explore related impact further in the course of our project.

Genre of the movie

Let's now explore the data based on the genre and see if we can identify some variability which we can use in our model. We'd be using the *genre_edx*, dataset that we created by splitting pipe-separated genre data, for this purpose.

First let's see how many movies do we have by each genre.

```
# Code to list out the number of movies by Genre
genre_edx %>%
  group_by(genres) %>%
  summarise(Movies = n()) %>%
  rename(Genre = genres) %>%
  arrange(desc(Movies)) %>%
  knitr::kable("latex", caption = "Number of Movies by Genre",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  footnote(general = "genre edx dataset",
           general_title = "Source Data:", footnote_as_chunk = TRUE)
```

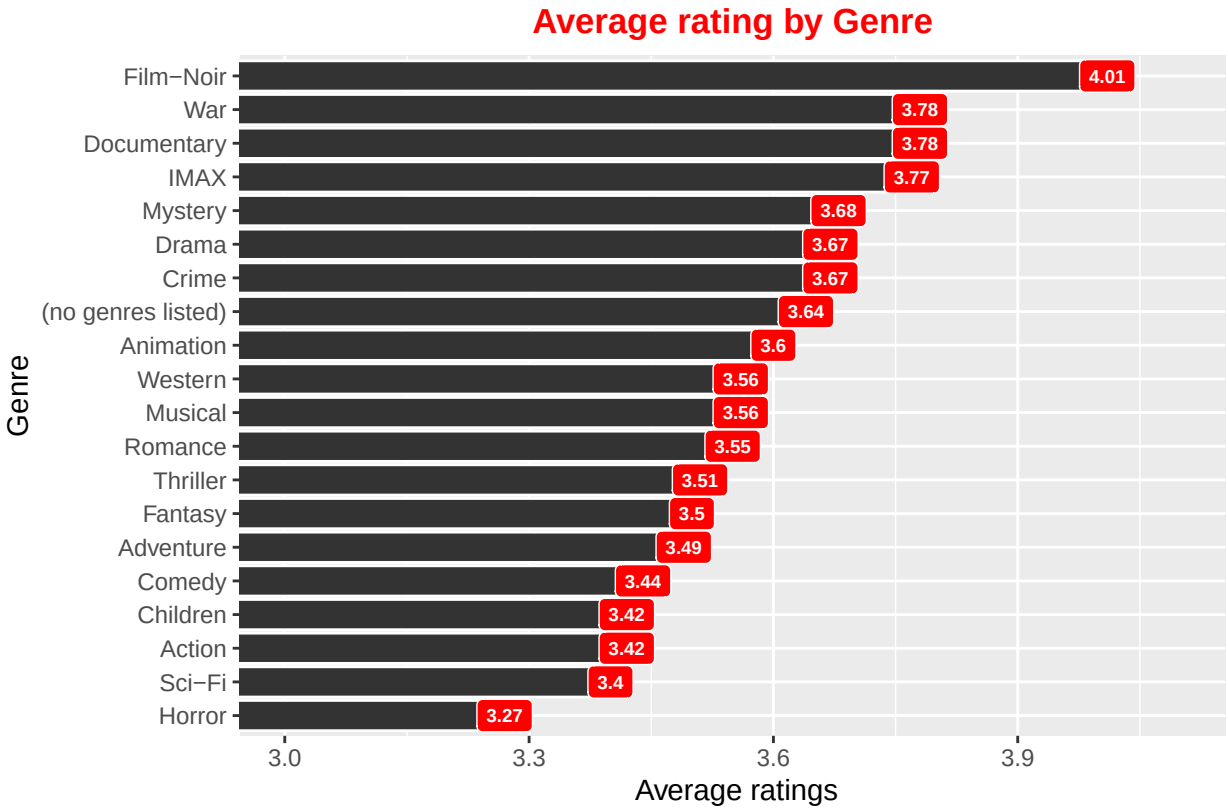
Let's now identify the average rating by genre

Table 1: Number of Movies by Genre

Genre	Movies
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332
War	511147
Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7

Source Data: genre edx dataset

```
# Plot to show the average rating by Genre
genre_edx %>%
  group_by(genres) %>%
  summarise(avg_rating = round(mean(rating), 2)) %>%
  ggplot(aes(reorder(genres, avg_rating), avg_rating)) +
  coord_flip(y = c(3.0, 4.10)) +
  geom_bar(stat = "identity", fill ="#333333", color = "white") +
  labs(x = "Genre", y = " Average ratings",
       title = "Average rating by Genre",
       caption = "source data: genre edx dataset") +
  geom_label(aes(label = avg_rating), size = 2.5,
             fontface = "bold", fill = "red", color = "white") +
  theme(plot.title = element_text(color = "red", face = "bold", hjust = 0.5))
```



As we can see there seems to be a relationship between the rating and the genre of the movie. We would use this insight to further improve our model.

We now have a good insight into the data that we are dealing with, let's put what we have learned into developing models for our project.

Methods and Analysis

Now that the necessary datasets have been setup, we can start exploring the various models that would be built. For the MovieLens projects various ***Linear Regression Models*** would be used.

Linear Regression Models

Linear regression is the simplest of the statistical models. In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables)[2].

In Machine Learning Linear Regression can be used to predict a response(Y) based on one or more predictors(X). In the simplest of forms it can be mathematically respresented as

$$Y = \beta_0 + \beta_1 X$$

where:

- β_0 The slope of the regression line
- β_1 The intercept of the regression line

[2]https://en.wikipedia.org/wiki/Linear_regression

Naive model

The first model to be explored assumes the same rating for all movies by all users with any deviation explained by random variation. The model can be represented as

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

where:

- $\epsilon_{u,i}$ independent errors sampled from the same distribution centred at 0
- μ the actual rating for all the movies

Modeling movie effect

Modeling the movie effect takes into consideration that some movies are rated higher than others, we can thus supplement our model with b_i to represent this bias or effect.

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

where:

- $\epsilon_{u,i}$ independent errors sampled from the same distribution centred at 0
- μ the actual rating for all the movies
- b_i movie effects or movie bias

For this model we know that the least square estimate b_i is the average of $Y_{u,i} - \mu$ for each movie i as using lm() function would be very slow due to the number b_i that needs to be accounted for. Thus:

$$b_i = Y_{u,i} - \mu$$

Modeling movie + user effect

The above model doesn't account for the bias of a user or the user effect. Thus we can further improve on our model by accounting for b_u. So our enhanced model can be represented as

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

where:

- $\epsilon_{u,i}$ independent errors sampled from the same distribution centred at 0
- μ the actual rating for all the movies
- b_i movie effects or movie bias
- b_u user effects or user bias

Follwoing the reasoning in the earlier model we would derive the user effect b_u as the average of $Y_{u,i} - \mu - b_i$. Thus:

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

Modeling movie + user + time effect

An additional improvement to the model could be the effect of time on the rating. So if $d_{u,i}$ is defined as the day that user u rates the movie i then the time effect can be defined as per the model below

$$Y_{u,i} = \mu + b_i + b_u + f(d_{u,i}) + \epsilon_{u,i}$$

with f a smooth function of $d_{u,i}$

where:

- $\epsilon_{u,i}$ independent errors sampled from the same distribution centred at 0
- μ the actual rating for all the movies
- b_i movie effects or movie bias
- b_u user effects or user bias
- $d_{u,i}$ the day for user u's rating of movie i

And the time effect represented by b_t can be arrived at by obtaining μ, b_i, and b_u and calculating the average of $Y_{u,i} - \mu - b_i - b_u$. Thus:

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

Modeling movie + user + genre effect

A similar effect on the rating could be that of genre, which could be derived as per the model below

$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} x_{u,i\beta_k} + \epsilon_{u,i}$$

with $x_{u,i}^k = 1$ if $g_{u,i}$ is genre k

where:

- $\epsilon_{u,i}$ independent errors sampled from the same distribution centred at 0
- μ the actual rating for all the movies
- b_i movie effects or movie bias
- b_u user effects or user bias
- $g_{u,i}$ the genre for user u's rating of movie i

And the genre effect represented by b_g can be arrived at by obtaining μ, b_i, and b_u and calculating the average of $Y_{u,i} - \mu - b_i - b_u$. Thus:

$$b_g = Y_{u,i} - \mu - b_i - b_u$$

Regularization[3]

Regularization is a regression technique that helps us correct any overfitting in our model. It permits us to penalize large estimates that are induced by small sample sizes. The general idea of the technique is to minimize the variance of effect estimates.

Regularized movie + user + genre effect

As we would see in subsequent steps below, the movie + user + genre effect model would provide the least RMSE. We would thus regularize this model in order to improve it.

This regularized model can be defined as below

$$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u - b_g)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2)$$

where $\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u - b_g)^2$ is the least squares while $\lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2)$ is the penalty. Larger the effect estimates larger the penalty.

[3]Introduction to Data Science, Rafael A. Irizarry (2019)

We would use cross validation to arrive at the best λ that corresponds to the least RMSE. Using calculus it can be derived that the values of b_i, b_u, and b_g that minimizes the above equation will be

$$b_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu)$$

$$b_u(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i)$$

$$b_g(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u)$$

where n_i is the number of ratings made for movie i.

RMSE Function

We will evaluate the accuracy of our model by determining by how much our predicted ratings from the respective models, deviate from the actual ratings.

If $y_{u,i}$ is the rating for movie i by user u, and $\hat{y}_{u,i}$ denotes the predicted rating then the RMSE would be defined as:

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

with N being the number of user and movie combinations

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

Results

We would now assess our models and their performance to arrive at the best model to achieve our objective of an **RMSE $\leq$ 0.8649**

Assessment of the models

Naive model

```
# Naive Model: RMSE using average movie rating

# Calculate the average
mu <- mean(train_set$rating)

# Evaluate the performance
naive_rmse <- RMSE(test_set$rating, mu)

# Store the result
rmse_results <- data_frame(Method = "Naive model",
                           RMSE = round(naive_rmse, 7), Improvement = "NA")
```

```r
# Display the results in a tabular format
rmse_results %>%
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Naive Model", escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red")
```

Table 2: Naive Model

Method	RMSE	Improvement
Naive model	1.0599043	NA

So, we now have our very first model, however the RMSE of 1.0599043 can definitely be improved upon. Let's continue with the assessment of our next model.

Modeling movie effect

```r
# Movie Effect Model: Modeling movie effect (b_i)

# Calculate the movie effect (b_i)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

# Obtain the prediction of movie ratings
me_rating <- mu + test_set %>%
  left_join(movie_avgs, by = "movieId") %>% # Combining movie averages with the test set
  pull(b_i)

# Evaluate the performance
model_1_rmse <- RMSE(test_set$rating, me_rating)

# Calculate the improvement in percentage
improvement <- round((naive_rmse - model_1_rmse) * 100 / naive_rmse, 4)

# Append the result for visual comparison
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie effect",
                                     RMSE = round(model_1_rmse, 7),
                                     Improvement = as.character(improvement)))

# Display the results in a tabular format
rmse_results %>%
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Comparison of 2 models",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
```

```
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  footnote(general = "Improvement in percentage",
           general_title = "Note:", footnote_as_chunk = TRUE)
```

Table 3: Comparison of 2 models

Method	RMSE	Improvement
Naive model	1.0599043	NA
Movie effect	0.9437429	10.9596

Note: Improvement in percentage

When we account for the movie effect we immediately see an improvement of 10.9596% which is significant, though there is still scope for improvement. Let's move on.

Modeling movie + user effect

```
# Movie and user effect model: Modeling movie effect and user effect (b_i + b_u)

# Calculate the user effect (b_u)
user_avgs <- train_set %>%
  left_join(movie_avgs, by = "movieId") %>%  # Combining movie averages with the train set
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

# Obtain the prediction of movie ratings
ue_rating <- test_set %>%
  left_join(movie_avgs, by = "movieId") %>% # Combining movie averages with the test set
  left_join(user_avgs, by = "userId") %>% # Combining user averages with the test set
  mutate(uer = mu + b_i + b_u)%>%
  pull(uer)

# Evaluate the performance
model_2_rmse <- RMSE(test_set$rating, ue_rating)

# Calculate the improvement in percentage
improvement <- round((model_1_rmse - model_2_rmse) * 100 / model_1_rmse, 4)

# Append the result for visual comparison
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie + User effect",
                                     RMSE = round(model_2_rmse, 7),
                                     Improvement = as.character(improvement)))

#Display the results in a tabular format
rmse_results %>%
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Comparison of 3 models",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
```

```
  footnote(general = "Improvement in percentage",
           general_title = "Note:", footnote_as_chunk = TRUE)
```

Table 4: Comparison of 3 models

Method	RMSE	Improvement
Naive model	1.0599043	NA
Movie effect	0.9437429	10.9596
Movie + User effect	0.8659319	8.2449

Note: Improvement in percentage

As we can see, including the user effect has further improved our model. There is an improvement of 8.2449% from our previous model. Now let's see how much does time bias impact our model.

Modeling movie + user + time effect

```
# Movie, user and time effect model: Modeling movie effect, user effect,
# and time effect (b_i + b_u + b_t)

# Mutate test set to create a date column which would be used for the join
test_set <- test_set %>%
  mutate(date = lubridate::round_date(lubridate::as_datetime(timestamp), unit = "week"))

# Calculate the time effect (b_t)
time_avgs <- train_set %>%
  left_join(movie_avgs, by = "movieId") %>%  # Combining movie averages with the train set
  left_join(user_avgs, by = "userId") %>% # Combining user averages with the train set
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarise(b_t = mean(rating - mu - b_i - b_u))

# Obtain the prediction of movie ratings
te_rating <- test_set %>%
  left_join(movie_avgs, by = "movieId") %>%  # Combining movie averages with the test set
  left_join(user_avgs, by = "userId") %>% # Combining user averages with the test set
  left_join(time_avgs, by = "date") %>%
  mutate(ter = mu + b_i + b_u + b_t)%>%
  pull(ter)

# Evaluate the performance
model_3_rmse <- RMSE(test_set$rating, te_rating)

# Calculate the improvement in percentage
improvement = round((model_2_rmse - model_3_rmse) * 100 / model_2_rmse, 4)

# Append the result for visual comparison
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie + User + Time effect",
                                     RMSE = round(model_3_rmse, 7),
                                     Improvement = as.character(improvement)))

#Display the results in a tabular format
rmse_results %>%
```

```
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Comparison of 4 models",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  footnote(general = "Improvement in percentage",
           general_title = "Note:", footnote_as_chunk = TRUE)
```

Table 5: Comparison of 4 models

Method	RMSE	Improvement
Naive model	1.0599043	NA
Movie effect	0.9437429	10.9596
Movie + User effect	0.8659319	8.2449
Movie + User + Time effect	0.8658367	0.011

Note: Improvement in percentage

The impact of time bias seems to be insignificant. We see an improvement of only 0.011% which doesn't help us achieve our target RMSE of ***0.8649***. Consequently we would be ignoring this model and would verify the impact of genre effect on Movie + User model instead.

Modeling movie + user + genre effect

```
# Modeling movie effect, user effect and genre effect (b_i + b_u + b_g)

# Calculate the genre effect (b_g)
genre_avgs <- genre_train_set %>%
  # Combining movie averages with the genre train set
   left_join(movie_avgs, by = "movieId") %>%
  # Combining user averages with the genre train set
   left_join(user_avgs, by = "userId") %>%
   group_by(genres) %>%
   summarise(b_g = mean(rating - mu - b_i - b_u))

# Obtain the prediction of movie ratings
ge_rating <- genre_test_set %>%
  # Combining movie averages with the genre test set
  left_join(movie_avgs, by = "movieId") %>%
  # Combining user averages with the genre test set
  left_join(user_avgs, by = "userId") %>%
  # Combining genre averages with the genre test set
  left_join(genre_avgs, by ="genres") %>%
  mutate(ger = mu + b_i + b_u + b_g)%>%
  pull(ger)

# Evaluate the performance
model_4_rmse <- RMSE(genre_test_set$rating, ge_rating)

# Calculate the improvement in percentage
improvement <- round((model_2_rmse - model_4_rmse) * 100 / model_2_rmse, 4)
```

```r
# Append the result for visual comparison
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Movie + User + Genre effect",
                                     RMSE = round(model_4_rmse, 7),
                                     Improvement = as.character(improvement)))

# Display the results in a tabular format
rmse_results %>%
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Comparison of 5 models",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  row_spec(4, strikeout = TRUE) %>%
  footnote(general = "Improvement in percentage",
           general_title = "Note:", footnote_as_chunk = TRUE)
```

Table 6: Comparison of 5 models

Method	RMSE	Improvement
Naive model	1.0599043	NA
Movie effect	0.9437429	10.9596
Movie + User effect	0.8659319	8.2449
~~Movie + User + Time effect~~	~~0.8658367~~	~~0.011~~
Movie + User + Genre effect	0.8641894	0.2012

Note: Improvement in percentage

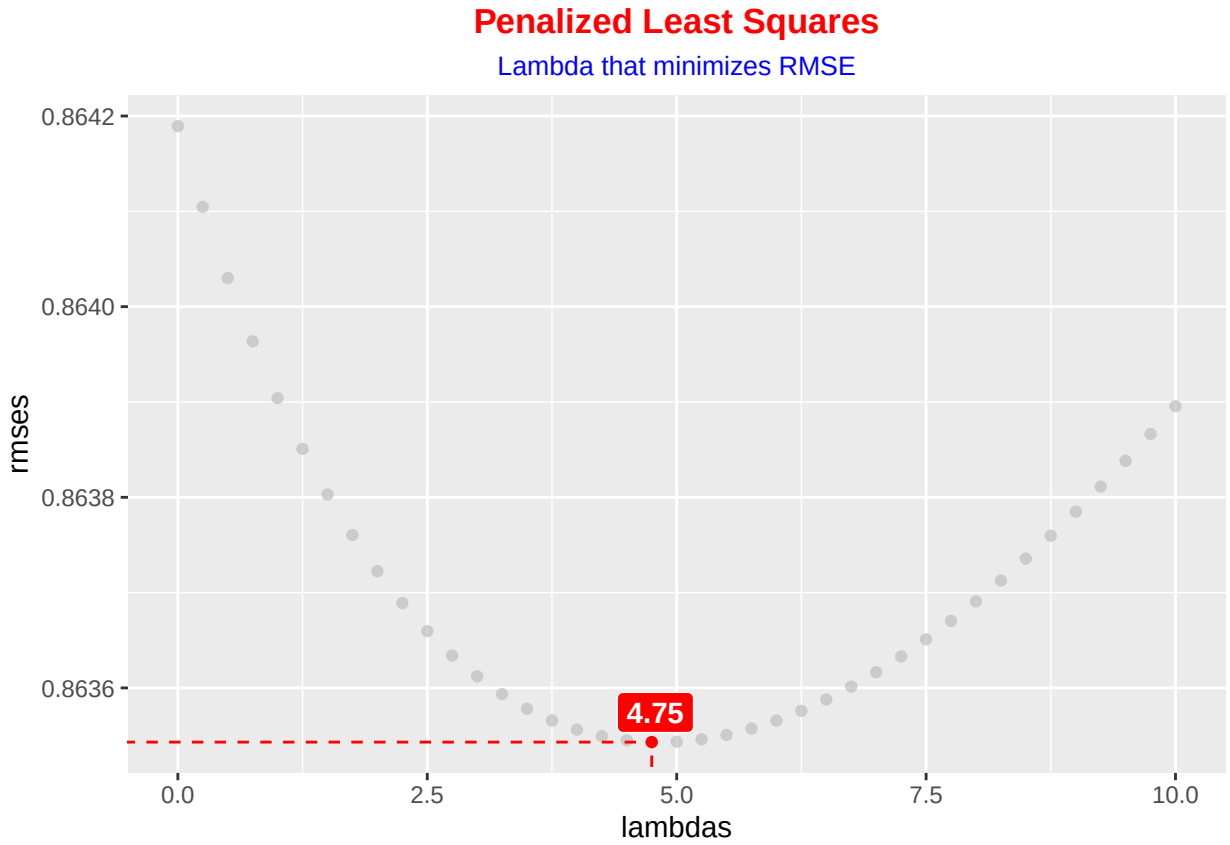
This model gives us a much better improvement as compared to that of the previous model. We see an improvement of 0.2012%. This is by far the best of our models as it provides an RMSE that actually exceeds our target of ***0.8649***. Now let's apply regularization to this model and see how much of an improvement that brings.

Regularized movie + user + genre effect

```r
# Penalized Least Square

# Create a set of possible lambdas
lambdas <- seq(0, 10, 0.25)

mu <- mean(genre_train_set$rating)

# Using cross validation to obtain the lambda that minimizes the RMSE
# Calculate the RMSE for each of the effects or each lambda

rmses <- sapply(lambdas, function(lambda){
  b_i <- train_set %>%
   group_by(movieId) %>%
   summarize(b_i = sum(rating - mu) / (n() + lambda))
```

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

  b_g <- genre_train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + lambda))

# Predict the ratings based on the penalized effects
  predicted_ratings <- genre_test_set %>%
    left_join(b_i, by = "movieId") %>% # Combining movie averages with genre test set
    left_join(b_u, by = "userId") %>% # Combining user averages with the genre test set
    left_join(b_g, by ="genres") %>%
    mutate(ger = mu + b_i + b_u + b_g)%>%
    pull(ger)

  return(RMSE(predicted_ratings, genre_test_set$rating))

})

# Determine which value of lambda provides the minimum RMSE
lambda <- lambdas[which.min(rmses)]
```

As we can see $\lambda = 4.75$ provides the minimum RMSE 0.8635432. The plot below depicts λ and the RMSE it generates. The λ for minimum RMSE is highlighted.



Let's calculate the improvement based on the RMSE of 0.8635432

```
# Calculate the improvement in percentage
improvement <- round((model_4_rmse - min(rmses)) * 100 / model_4_rmse, 4)

# Append the result for visual comparison
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Regularized Movie + User + Genre Effect",
                                     RMSE = round(min(rmses), 7),
                                     Improvement = as.character(improvement)))

#Display the results in a tabular format
rmse_results %>%
   mutate(
    RMSE = cell_spec(RMSE, "latex", color = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", caption = "Comparison of 6 models",
               escape = FALSE, linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  row_spec(4, strikeout = TRUE) %>%
  row_spec(6, bold = TRUE, color = "blue") %>%
  footnote(general = "Improvement in percentage",
           general_title = "Note:", footnote_as_chunk = TRUE)
```

The improvement of the regularized model is 0.0748%. Note though, that we have so far been applying our

Table 7: Comparison of 6 models

Method	RMSE	Improvement
Naive model	1.0599043	NA
Movie effect	0.9437429	10.9596
Movie + User effect	0.8659319	8.2449
~~Movie + User + Time effect~~	~~0.8658367~~	~~0.011~~
Movie + User + Genre effect	0.8641894	0.2012
Regularized Movie + User + Genre Effect	**0.8635432**	**0.0748**

Note: Improvement in percentage

models in the train and test sets we created for the purpose by splitting the edx dataset. We are now ready to apply the model on the full edx set.

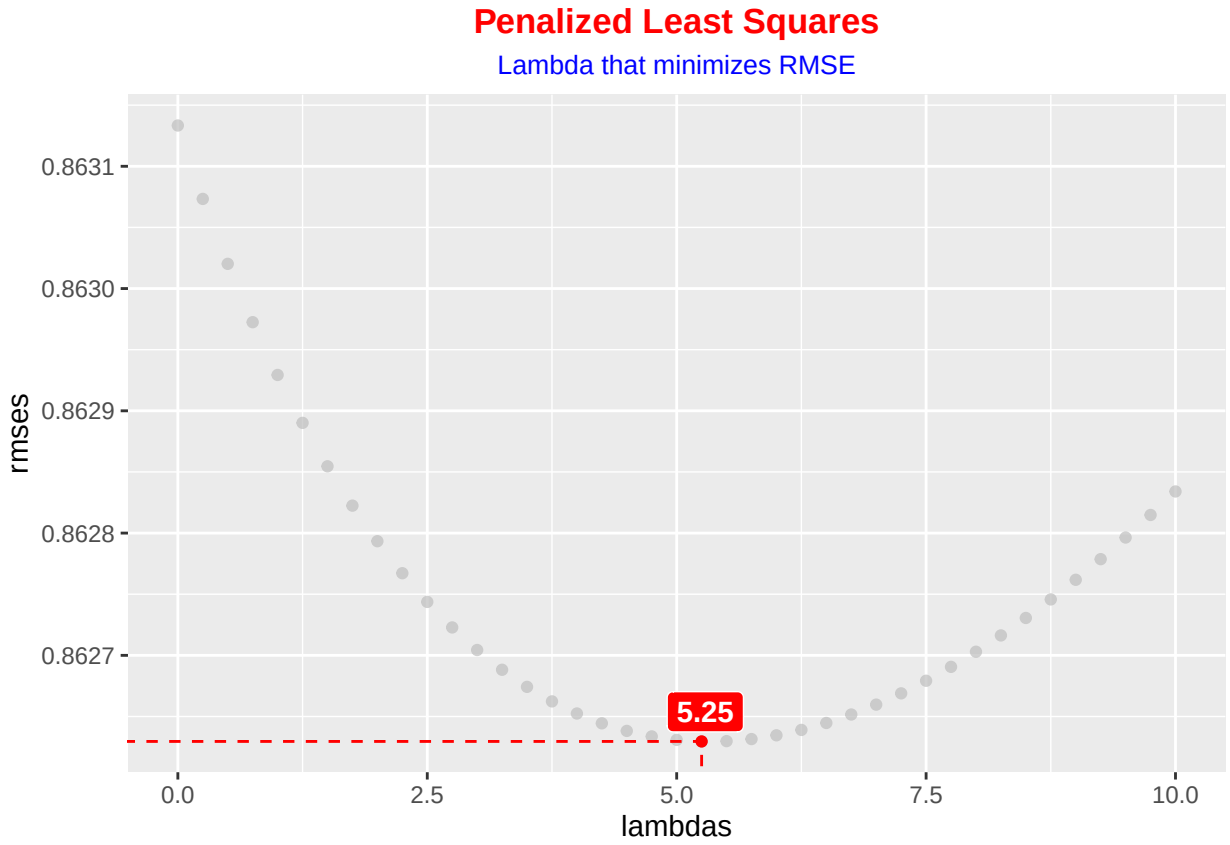
Final Result

For the final result of our project we would apply the **Regularized Movie + User + Genre Effect Model** to the full edx dataset and use the validation dataset to arrive at the final RMSE.

```
# Create a set of possible lambdas
lambdas <- seq(0, 10, 0.25)

# Calculating the various penalized effects based on the lambda that minimizes the RMSE
rmses <- sapply(lambdas, function(lambda){
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + lambda))

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

b_g <- genre_edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + lambda))

# Obtain the prediction of movie ratings
predicted_ratings <- genre_validation %>%
  left_join(b_i, by = "movieId") %>% # Combining movie averages with the validation set
  left_join(b_u, by = "userId") %>% # Combining user averages with the validation set
  left_join(b_g, by ="genres") %>% # Combining genre averages with the validation set
  mutate(ger = mu + b_i + b_u + b_g)%>%
  pull(ger)

return(RMSE(predicted_ratings, genre_validation$rating))
})

# Determine which value of lambda provides the minimum RMSE
lambda <- lambdas[which.min(rmses)]
```

So $\lambda = 5.25$ provides the minimum RMSE 0.8626296. The plot below depicts λ and the RMSE it generates.

The λ for minimum RMSE is highlighted



So the final RMSE that we have arrived at is 0.8626296 which meets our target of achieving $\boldsymbol{RMSE \leq 0.8649}$

```
final_result <- data_frame(
  Method = "Regularized Movie + User + Genre Effect on full dataset",
  RMSE = round(min(rmses), 7))

#Display the final results in a tabular format
final_result %>%
  mutate(
    RMSE = cell_spec(RMSE, "latex", color = "white",
                     background = ifelse(RMSE <=0.8649, "#006600","red"))
  ) %>%
  knitr::kable("latex", escape = FALSE, caption = "Final Result", linesep = "") %>%
  kable_styling(latex_options = c("striped", "hold_position"),
                full_width = F , position = "center") %>%
  row_spec(0, bold = TRUE, color = "white" , background ="red") %>%
  row_spec(1, bold = TRUE)
```

Table 8: Final Result

Method	RMSE
Regularized Movie + User + Genre Effect on full dataset	**0.8626296**

Conclusions

As part of the project we assesed 6 different models on the stable benchmark 10M movie ratings MovieLens dataset. We trained our models to predict the movie ratings on a subset (train_set) which we had obtained by partitioning the edx dataset. 5 of these were Linear Regression models, we then applied regularization on the best of these 5 to arrive at our 6^{th} model, and proceeded to predict the ratings in the other partition (test_set). Next we determined the RMSE to ascertain the performance of each of these models and compared the RMSEs to arrive at our best model which provided the least RMSE of the lot. Two of the models actually exceeded our target RMSE.

Through this exercise we deduced that the **Regularized Movie + User + Genre Effect Model** was our best model. As a final check we then applied this model on the full **edx** data set and obtained the RMSE on the **validation** data set to obtain the Final RMSE of 0.8626296.

References

1. 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.
2. Introduction to Data Science, Rafael A. Irizarry (2019)
3. Create Awesome LaTeX Table with knitr::kable and kableExtra, Hao Zhu (2019)
4. An Example R Markdown (2017)
5. Linear Regression Models
6. Introduction to gghighlight, Hiroaki Yutani (2018)