Harvardx -PH125.9x Data Science

MovieLens Rating Project

Pritesh Contractor

24th June 2020

Table of Content

Introduction

In this project, we are asked to create movie recommedation system. Recommendation system plays a vital role this day within e-commerce and online streaming. Within this project we need to predict the rating a user will give to a movie in a validation set. The provided dataset is from MovieLens 10M set (i.e. 10 millions ratings). While the test data set called validation is rouhgly 1 million records long. The prediction is to be done using different models and it is also said that validation data set should only be use in final validation.

Objective

The main aim for this project is to train predict user ratings with the help of provided datasets and predict movie ratings in provided validation set which should be used only in the final validation.

We are using Root Mean Square Error or RMSE which is use to measure the difference between values predicted by a model and the values that are observed. RMSE is measure of accuracy, use in comparing forecasting errors of different models for a particular data set, a lower RMSE is treated as better than higher one. The effect of each error on RMSE is proportional to the size of the squared error; hence the larger errors have disproportionately large effect on RMSE. The expected RMSE provided is 0.8649000

The function that will compute the RMSE is

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

Process

The main steps in a data science project includes

1. Data Preparation
2. Data Exploration
3. Data Cleaning
4. Data Analysis
5. Communicate

As a part of this project, we will download the dataset from MovieLens website and split into two subset used for training and validation. The training validation will be called as edx and the validation subset is called validation. The edx is again split into two subset (as per the requirement) and use for training and testing purpose. Once the model reaches the

targeted RMSE in the testing set after that only we will finally validate the edx set with the model and use the validation set for final validation.

Secondly, we also create charts, tables and statistics summary to understand how different features have impact on the outcome. The information which we gather through exploration will be helpful in building the model.

In order to create a recommendation system, it is important to identify most important features that can help in predicting the rating any given user to any movie. In this project we will start building with simple model which will be just the mean of observed value. Then we will build user and movies effects within linear model, and check how much RMSE is improved. And finally we the user and movie effects will recieve the regularization parameters that penalizes samples with few ratings

Methods and Analysis

Part 1 - Data Preparation

In this section we will download and prepare dataset to be used for analysis. We will then split the dataset into two parts, the training set called edx and evaluation set called Validation with 90% and 10% of original dataset repectively.

After that we will split the edx set in to two parts - the train and the test set with 90% and 10% of edx set respectively. The model will be created and trained within train set and tested with test set unitl the RMSE target is achieved. Finally we will train the model again with entire edx set and validate that with validation set - which is called as cross validation

```
#### MovieLens Project - Generating movie rating and RMSE ####
#### Author: Pritesh Contractor ####

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

# Note: this process could take a couple of minutes


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

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------
---------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------
---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: tidyverse
## -- Attaching packages ------------------------------------------------
--------------------------- tidyverse 1.3.0 --
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ---------------------------------------------------------
---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
if(!require(data.table)) install.packages("data.table", repos = "http://cran.
us.r-project.org")

## Loading required package: data.table

##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## Loading required package: data.table
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
# 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/rati
ngs.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(movieI
d))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' s
ampler
## used

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' s
ampler
## used
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
```

```
ist = 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)

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es")

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es")
edx <- rbind(edx, removed)

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

As said earlier the edx set will be use for training and test and the validation set will be used only for final validation as per the project requirement. Here edx set will be split into training set and test set

The building of model will be carried out within Training Set and the Test Set will be use to test the model. When the model is completed we will use the validation set to calculate the final RMSE

We will be using the same procedure as we use in creating the edx and validation data sets

The training set will be 90% of edx data and the test set will be the remaining 10% of edx

```
# split the edx set into 2 parts the training set and the test set
#set.seed(1, sample.kind = "Rounding")
trainset_index <- createDataPartition(y=edx$rating, times=1, p=0.1, list=FALS
E)
train_set <- edx[-trainset_index,]
temp <- edx[trainset_index,]

# ensure that userid and movieid in test set are also in train set

test_set <- temp %>% semi_join(train_set, by = "movieId") %>% semi_join(train
_set, by="userId")

# adding rows removed from the test set back into the train set

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

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es")
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es")
train_set <- rbind(train_set,removed)

rm(trainset_index, temp, removed)
```

Part 2 - Data Exploration

Before we start building our model, it is necessary to understand the structure of the data, the distribution ratings and the relationship of the predictor. This will help in buidling better model.

```
##### Part 2: Data Exploration #####

# structure of dataset

str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  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 83898
4474 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" ...
```

From this exploration we come to know that edx has 6 columns

movieId integer userId integer rating numeric timestamp numeric title character genres character

Now we will find how many rows and columns are there within edx dataset?

```
# rows and columns within the edx dataset

dim(edx)
```

```
## [1] 9000055       6
```

The next table provides information about the structure and content of edx data set

From the table we can conclude that data is in tidy format i.e. each row has one observation and the column names are the features.

1. The desired outcome is stored within 'rating' column
2. User information is stored within 'úserId' column
3. Information with regards to movie is both in 'movieId'and 'title' columns
4. Rating date is available within 'timestamp' and it is measured in seconds since January 1st 1970
5. It is also identified that each movie is tagged with genre in the genres column

```
head(edx)
```

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

Now as we already know the structure of the dataset we will move forward and try to discover more details about each features and outcome

Feature 1 - Genres

As we conclude while understand the structure, each movie is tagged with genre in the 'genres' column. We will explore this information Overall dataset contain 797 different combination of genres and below is list of first six:

```
# genre extraction

edx %>% group_by(genres) %>% summarise(n=n()) %>% head()
```

```
## # A tibble: 6 x 2
##   genres                                                n
##   <chr>                                             <int>
## 1 (no genres listed)                                    7
## 2 Action                                            24482
## 3 Action|Adventure                                  68688
## 4 Action|Adventure|Animation|Children|Comedy         7467
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy  187
## 6 Action|Adventure|Animation|Children|Comedy|IMAX      66
```

From the table above we can see that serveral movies are classified in more than one genre. To identiy number of genre in each movie we will execute below and get the table for the same:

```
# extracting number of genres in each movie listed with the table

tibble(count=str_count(edx$genres, fixed("|")), genres=edx$genres) %>%
  group_by(count, genres) %>%
  summarise(n=n()) %>%
  arrange(-count) %>%
  head()
```

```
## # A tibble: 6 x 3
## # Groups:   count [3]
##   count genres                                                      n
##   <int> <chr>                                                   <int>
## 1     7 Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller   256
## 2     6 Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery    10975
## 3     6 Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery      355
## 4     6 Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance    515
## 5     5 Action|Adventure|Animation|Children|Comedy|Fantasy             187
## 6     5 Action|Adventure|Animation|Children|Comedy|IMAX                 66
```

Feature 2 - Date

The period of rating was collected over almost 14 years

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
```

```
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
tibble('Initial Date'=date(as_datetime(min(edx$timestamp), origin="1970-01-01
")),'Final Date'=date(as_datetime(max(edx$timestamp), origin="1970-01-01")))
%>%
  mutate(Period=duration(max(edx$timestamp)-min(edx$timestamp)))

## # A tibble: 1 x 3
##   `Initial Date` `Final Date` Period
##   <date>         <date>       <Duration>
## 1 1995-01-09     2009-01-05   441479727s (~13.99 years)
```

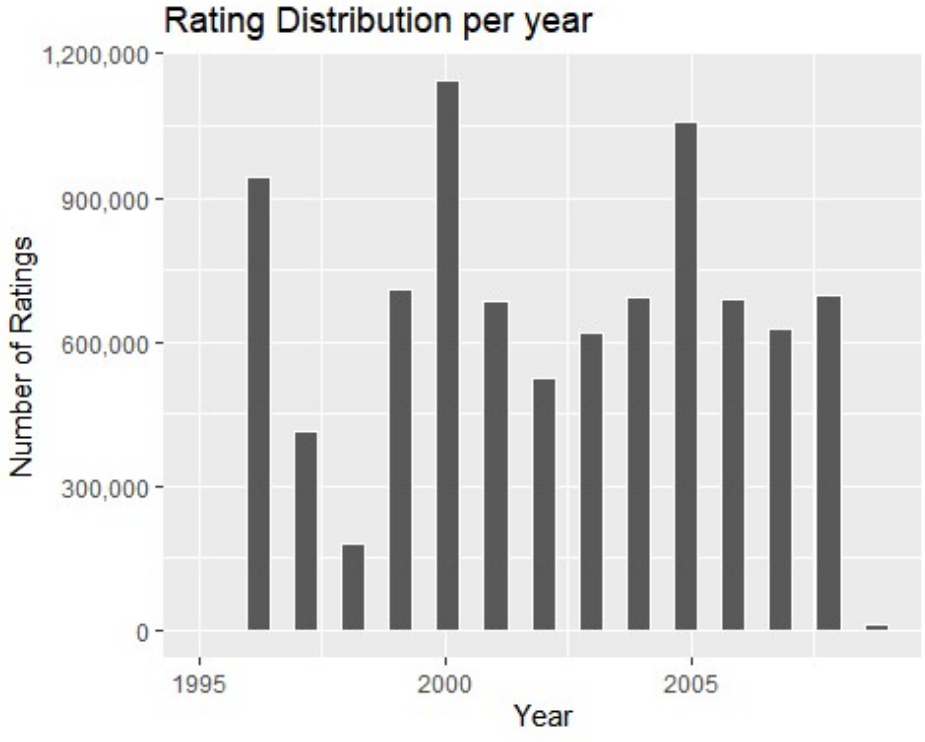
Plotting graph - number of ratings vs rating distribution over the year

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

## Loading required package: ggthemes

## Loading required package: ggthemes
if(!require(scales))
  install.packages("scales", repos = "http://cran.us.r-project.org")

## Loading required package: scales

##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## Loading required package: scales
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
edx %>% mutate(year=year(as_datetime(timestamp, origin="1970-01-01"))) %>%
  ggplot(aes(x=year)) +
  geom_histogram(color="white") +
  ggtitle("Rating Distribution per year") +
  xlab("Year") +
  ylab("Number of Ratings") +
  scale_y_continuous(labels=comma) +
  theme_grey()
```

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



plotting number of ratings vs rating distribution year

Following table provides list of the days with more ratings. We can see without any surprise the movies are well known blockbusters

```
# extracting table with more ratings

edx %>% mutate(date=date(as_datetime(timestamp, origin="1970-01-01"))) %>%
  group_by(date,title) %>%
  summarise(count=n()) %>%
  arrange(-count) %>%
  head(10)
```

```
## # A tibble: 10 x 3
## # Groups:   date [4]
##    date       title
count
##    <date>     <chr>
<int>
##  1 1998-05-22 Chasing Amy (1997)
322
##  2 2000-11-20 American Beauty (1999)
277
##  3 1999-12-11 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
254
##  4 1999-12-11 Star Wars: Episode V - The Empire Strikes Back (1980)
251
##  5 1999-12-11 Star Wars: Episode VI - Return of the Jedi (1983)
241
##  6 2005-03-22 Lord of the Rings: The Two Towers, The (2002)
239
##  7 2005-03-22 Lord of the Rings: The Fellowship of the Ring, The (2001)
227
##  8 2000-11-20 Terminator 2: Judgment Day (1991)
221
##  9 1999-12-11 Matrix, The (1999)
210
## 10 2000-11-20 Jurassic Park (1993)
201
```

Feature - 3 - Rating

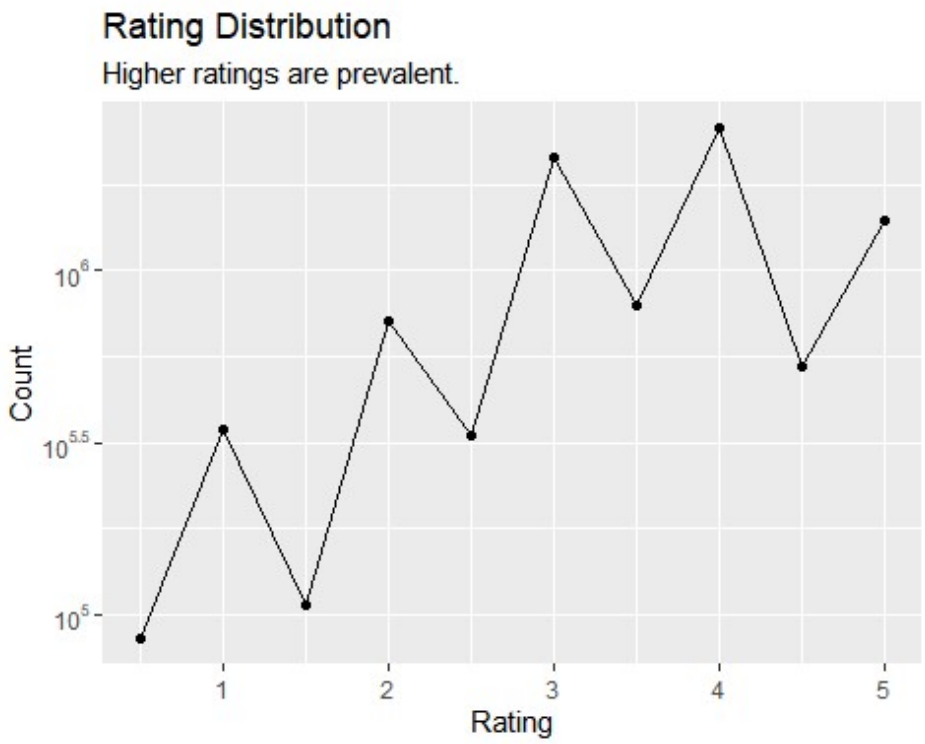
Users have the option to choose ratings value from 0.5 to 5.0 totalling 10 unusual possibilities. This is an unusual scale due to which most of the movies we can see below are getting rounded rating

```r
# extracting ratings - counting number of each ratings

edx %>% group_by(rating) %>% summarize(n=n())
```

```
## # A tibble: 10 x 2
##    rating       n
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```

Verifying number of ratings with edx subset

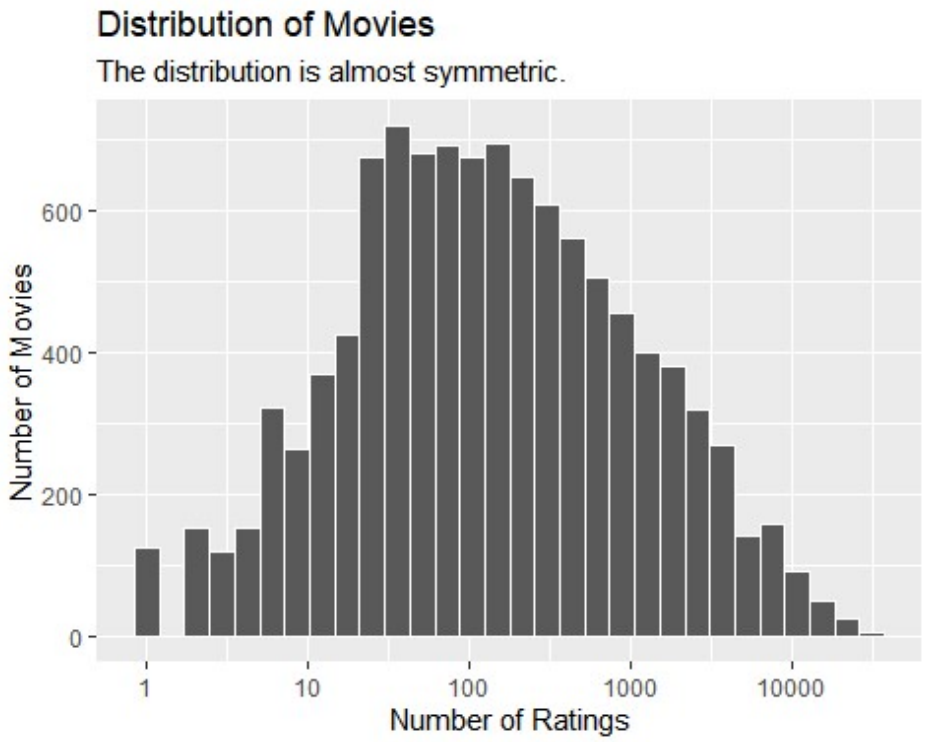
```r
# extracting ratings within edx data set

edx %>% group_by(rating) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=rating,y=count)) +
  geom_line() +
  geom_point() +
  scale_y_log10(breaks=trans_breaks("log10", function(x) 10^x),labels=trans_f
ormat("log10", math_format(10^.x))) +
  ggtitle("Rating Distribution", subtitle = "Higher ratings are prevalent.")
+
  xlab("Rating") +
  ylab("Count") +
  theme_grey()
```



Feature - 4 - Movies

In total there are 10677 different movies in the edx data set. Through our analysis we can see that some of them are rated more than others, because many movies are watched by few users compare to blockbusters that tend to have higher ratings

```
# distribution of movies

edx %>% group_by(movieId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color="white") +
  scale_x_log10() +
  ggtitle("Distribution of Movies", subtitle = "The distribution is almost sy
mmetric.") +
  xlab("Number of Ratings") +
  ylab("Number of Movies") +
  theme_grey()

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



Feature 5 - Users

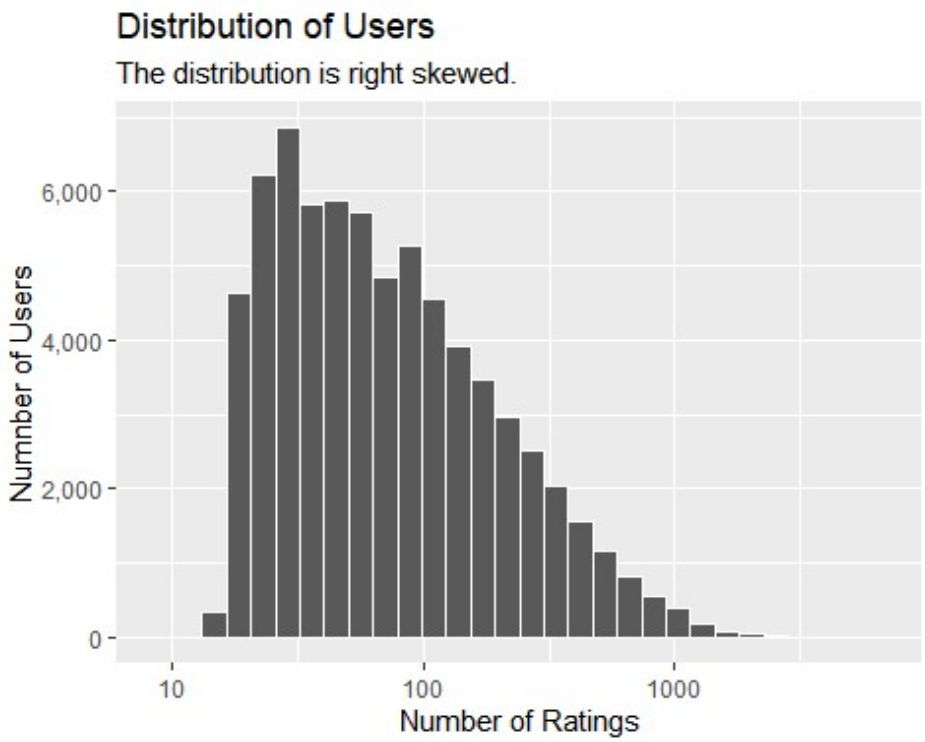
In total there are 69878 different users within the edx dataset Majority of users rate few movies while few users rate more than a thousand movies

5% users rated less than 20 movies

```
# Extraction of users

edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  arrange(n) %>%
  head()
```

```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
## 1  62516    10
## 2  22170    12
## 3  15719    13
## 4  50608    13
## 5    901    14
## 6   1833    14
```

```r
# user distribution

edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color="white") +
  scale_x_log10() +
  ggtitle("Distribution of Users", subtitle = "The distribution is right skew
ed.") +
  xlab("Number of Ratings") +
  ylab("Numnber of Users") +
  scale_y_continuous(labels = comma) +
  theme_grey()

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



Part 3 - Data Cleaning

As we discussed earlier, that serveral features can be used to predict the rating for a given user. However, as predictors increases the complexity of model requires more computer resources, so in this project the estimated rating uses only movie and user information

```
##### Part 3: Data Cleaning #####

# extracting estimated rating for movie and user only - many predictors incre
ases the model complexity and requires more computer resources

train_set <- train_set %>% select(userId, movieId, rating, title)
test_set <- test_set %>% select(userId, movieId, rating, title)
```

Part 4 - Modelling

We will be modelling through various models like

1. **Random Prediction**

Simple model which is use to just randomly predict the rating using the probability distribution observed during the data exploration. Such prediction may give worst error.

2. **Linear Model**

Simplest model that predicts all users who give the same ratings to all movies and then assume movie to movie variation as randomly distributed error. Although predicted value could be any value the statistic theory says that average minimizes the RMSE, so the initial prediction is just the average of all observed ratings, as described in this formula

$$\hat{Y}_{u,i} = \mu + \epsilon_{i,u}$$

Where $\hat{Y}$ is the predicted rating, μ is the mean of observed data and $\epsilon_{i,u}$ is the error distributions. Any other value other than mean increases the RMSE, so this could be good initial estimation.

Movie to movie variability can be explained by the fact that different movies have different rating distribution. This is fairly easy to understand because some movies are more popular than others and the public preference views. This is called Movie Effect or Movie Bias and is expressed through b_i in below formula

$$\hat{Y}_{u,i} = \mu + b_i + \epsilon_{i,u}$$

Also, the movie effect can be calculated as the mean of the difference between the observed rating y and the mean μ.

$$\hat{b}_i = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\mu})$$

Similar to movie effect, different users also have different rating pattern or distribution. To calculate this, we will be using below formula and it is called user bias:

$$\hat{b}_u = \frac{1}{N}\sum_{i=1}^{N}(y_{u,i} - \hat{b}_i - \hat{\mu})$$

Hence the prediction model that includes the user effect will then becomes

$$\hat{Y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

3. Regularization

The linear model will provide a good estimation for the ratings, but it may not be considered that many movies have few numbers of ratings and some users rate very few movies. Which means that sample size is very small for these movies and users. Statistically it leads to large estimated error

The estimated value can be improved by adding a factor that penalizes small sample sizes and have little or no impact. Hence the estimated movie and user effects can be calculated with below formula:

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

$$\hat{b}_u = \frac{1}{n_u + \lambda} \sum_{i=1}^{n_u} (y_{u,i} - \hat{b}_i - \hat{\mu})$$

For values N smaller than or similar to λ, $\hat{b}_i$, and $\hat{b}_u$ is smaller than the original values, whereas for values of N much larger than λ, $\hat{b}_i$, and $\hat{b}_u$ change very little

An effective method to choose λ that minimizes the RMSE is running simulations with several values of λ

Part 5 - Result

```
# Evaluating Model Functions


# defining Root Mean Squared Error (RMSE)
RMSE <- function(true_ratings, pred_ratings){
  sqrt(mean((true_ratings-pred_ratings)^2))
}
```

Random Prediction Model

Our first model is the simplest one which will randomly predicts the ratings with the help of observed probablities in the training set. For this we will calculate the probability of each rating within training set and then we will predict the rating for the test set and compare and actual rating.

We will be using Monte Carlo Simulation with replacements that provides a good approximation of the rating distribution

```
# random prediction model - rating distribution

set.seed(4321, sample.kind = "Rounding")
```

```
## Warning in set.seed(4321, sample.kind = "Rounding"): non-uniform 'Rounding
'
## sampler used
```

```
## Warning in set.seed(4321, sample.kind = "Rounding"): non-uniform 'Rounding
'
## sampler used
# produce probability of each rating

p <- function(x,y) mean(y==x)
rating <- seq(0.5,5,0.5)

# estimating probability of each rating with Monte Carlo Simulation

B <- 10^3
M <- replicate(B,{
  s <- sample(train_set$rating,100,replace = TRUE)
  sapply(rating,p,y= s)
})

prob <- sapply(1:nrow(M), function(x) mean(M[x,]))

# predicting random ratings distribution

pred_random <- sample(rating,size=nrow(test_set),
                      replace=TRUE, prob=prob)

# Table with the error results for random prediction model

result <- tibble(Method="Goal of Project", RMSE=0.8649)
result <- bind_rows(result,tibble(Method="Random Prediction Model",
                                  RMSE=RMSE(test_set$rating, pred_random)))
```

RMSE of random prediction is very high

```
print.data.frame(result)
```

```
##                    Method     RMSE
## 1         Goal of Project 0.864900
## 2 Random Prediction Model 1.500892
```

Linear Model Prediction

Linear model is build based on below formula

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

Initial Prediction

The initial prediction is mean of the ratings μ

$$\hat{y} = \mu + \epsilon_{u,i}$$

```
# Linear Model Prediction for rating distribution

# Mean of observed values

mu <- mean(train_set$rating)

# Updating the error table

result <- bind_rows(result,
                    tibble(Method="Mean",
                           RMSE=RMSE(test_set$rating, mu)))

# intermediate RMSE results

print.data.frame(result)
```

```
##                     Method     RMSE
## 1         Goal of Project 0.864900
## 2 Random Prediction Model 1.500892
## 3                    Mean 1.061135
```

Include Movie Effect

bi is the movie effect or bias for the movie i

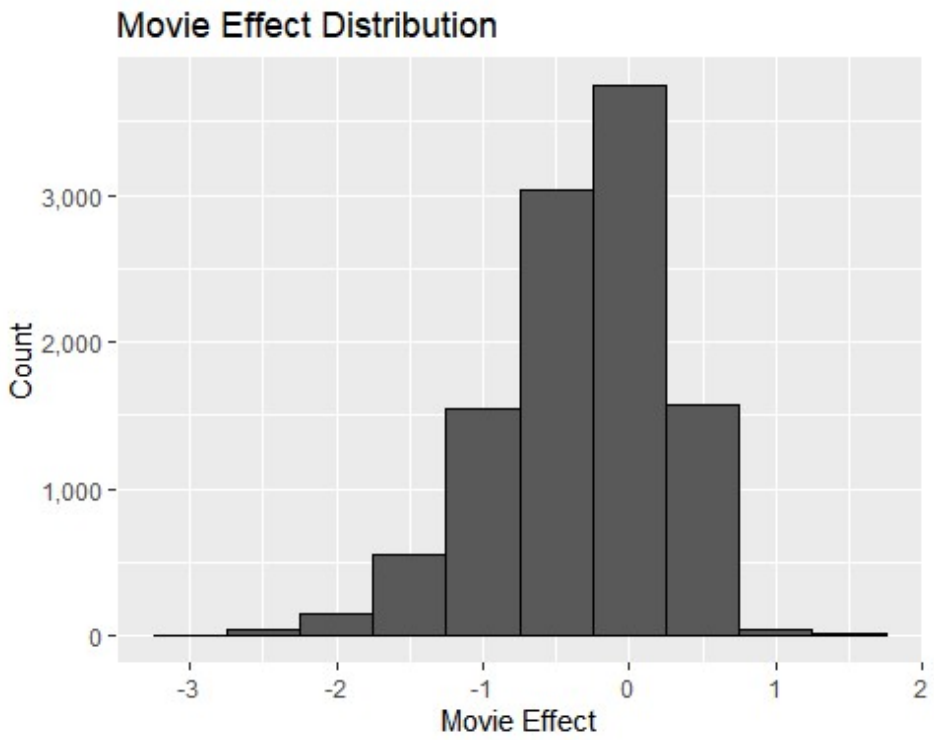
$$\hat{y} = \mu + b_i + \epsilon_{u,i}$$

```
# including movie effect
```

```r
bi <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i=mean(rating-mu))
head(bi)
```

```
## # A tibble: 6 x 2
##   movieId    b_i
##     <dbl>  <dbl>
## 1       1  0.415
## 2       2 -0.303
## 3       3 -0.359
## 4       4 -0.631
## 5       5 -0.443
## 6       6  0.303
```

```r
# plotting movie effect distribution

bi %>% ggplot(aes(x=b_i)) +
  geom_histogram(bins=10, col=I("black")) +
  ggtitle("Movie Effect Distribution") +
  xlab("Movie Effect") +
  ylab("Count") +
  scale_y_continuous(labels=comma)+
  theme_grey()
```



```r
# prediction of rating distribution with mean and bi

pred_bi <- mu + test_set %>%
  left_join(bi, by="movieId") %>%
  .$b_i

# calculating RMSE

result <- bind_rows(result,
                    tibble(Method="Mean + bi",
                           RMSE=RMSE(test_set$rating, pred_bi)))

# intermediate RMSE Improvement result check

print.data.frame(result)
```

```
##                     Method      RMSE
## 1         Goal of Project 0.8649000
## 2 Random Prediction Model 1.5008922
## 3                    Mean 1.0611350
## 4               Mean + bi 0.9441568
```

Include User Effect

bu is the user effect (or bias) for the user u

$$\hat{y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

predicting the rating with mean+bi+bu

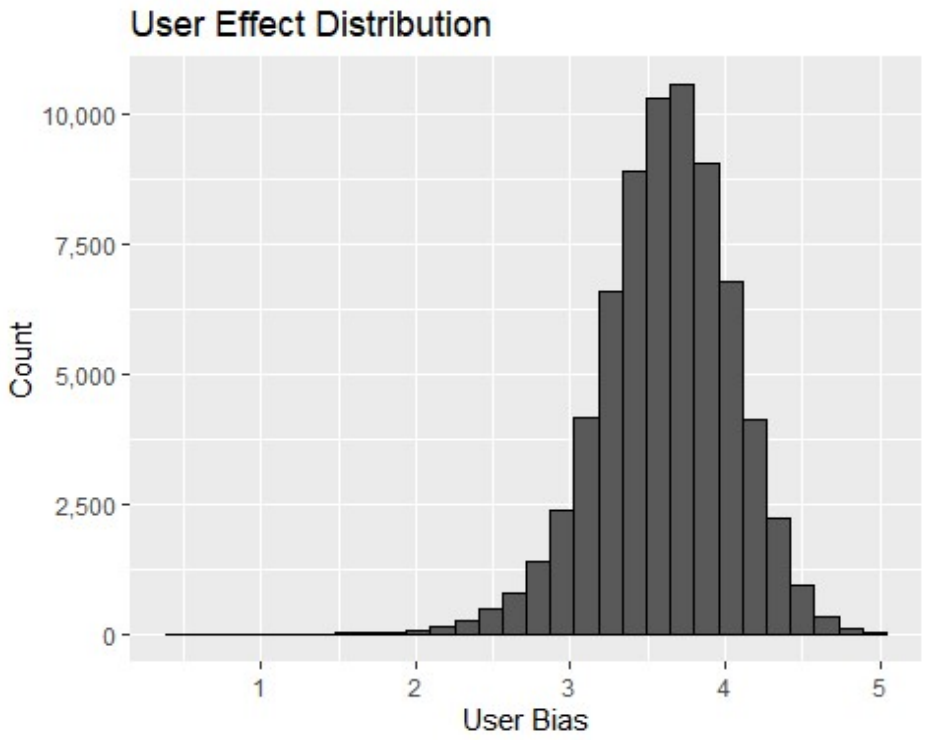
```r
# including user effect for rating distribution generation

bu <- train_set %>%
  left_join(bi, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating-mu-b_i))

# prediction including user effect
```

```r
pred_busr <- test_set %>%
  left_join(bi, by="movieId") %>%
  left_join(bu, by="userId") %>%
  mutate(pred=mu + b_i + b_u) %>%
  .$pred

# updating the result in the table

result <- bind_rows(result,
                    tibble(Method="Mean + bi + bu",
                           RMSE=RMSE(test_set$rating,pred_busr)))

# intermediate results RMSE Improvement

print.data.frame(result)
```

```
##                     Method      RMSE
## 1         Goal of Project 0.8649000
## 2 Random Prediction Model 1.5008922
## 3                    Mean 1.0611350
## 4               Mean + bi 0.9441568
## 5          Mean + bi + bu 0.8659736
```

```r
# plot user effect distribution normally distributed

train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n() >= 100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(color="black") +
  ggtitle("User Effect Distribution") +
  xlab("User Bias") +
  ylab("Count") +
  scale_y_continuous(labels=comma) +
  theme_grey()
```

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



Evaluating Model Result

As we model it and calculate the prediction we found that RMSE is improved from the initial estimation based on the mean. However, we will still need to check if the model makes good ratings prediction or not

```
# evaluating the model result

# identigy the 10 largest residual differences

train_set %>%
  left_join(bi,by="movieId") %>%
  mutate(residual=rating-(mu+b_i)) %>%
  arrange(desc(abs(residual))) %>%
  slice(1:10)
```

```
##    userId movieId rating                           title        b_i  resi
dual
## 1   26423    6483    5.0      From Justin to Kelly (2003) -2.6169046  4.10
4396
## 2   29924    6483    5.0      From Justin to Kelly (2003) -2.6169046  4.10
4396
## 3    5279    6371    5.0           PokÃ©mon Heroes (2003) -2.4688582  3.95
```

```
6349
## 4   57863    6371    5.0           PokÃ©mon Heroes (2003) -2.4688582  3.95
6349
## 5    2507     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
## 6    7708     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
## 7    9214     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
## 8    9568     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
## 9    9975     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
## 10  10680     318    0.5 Shawshank Redemption, The (1994)  0.9431528 -3.95
5662
```

```
titles <- train_set %>%
  select(movieId, title) %>%
  distinct()

# identifying top 10 best movies

bi %>%
  inner_join(titles, by="movieId") %>%
  arrange(-b_i) %>%
  select(title) %>%
  head()
```

```
## # A tibble: 6 x 1
##   title
##   <chr>
## 1 Hellhounds on My Trail (1999)
## 2 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)
## 3 Shadows of Forgotten Ancestors (1964)
## 4 Fighting Elegy (Kenka erejii) (1966)
## 5 Sun Alley (Sonnenallee) (1999)
## 6 Bullfighter and the Lady (1951)
```

```
# identifying top 10 worst movies

bi %>%
  inner_join(titles,by="movieId") %>%
  arrange(b_i) %>%
  select(title) %>%
  head()
```

```
## # A tibble: 6 x 1
##   title
##   <chr>
## 1 Besotted (2001)
## 2 Hi-Line, The (1999)
## 3 Accused (Anklaget) (2005)
## 4 Confessions of a Superhero (2007)
## 5 War of the Worlds 2: The Next Wave (2008)
## 6 SuperBabies: Baby Geniuses 2 (2004)
```

```
# identifying number of ratings for 10 best movies

train_set %>%
  left_join(bi, by="movieId") %>%
  arrange(desc(b_i)) %>%
  group_by(title) %>%
  summarise(n=n()) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    title                                                          n
##    <chr>                                                      <int>
##  1 "'burbs, The (1989)"                                        1199
##  2 "'night Mother (1986)"                                       177
##  3 "'Round Midnight (1986)"                                      39
##  4 "'Til There Was You (1997)"                                  232
##  5 "\"Great Performances\" Cats (1998)"                           3
##  6 "*batteries not included (1987)"                             395
##  7 "...All the Marbles (a.k.a. The California Dolls) (1981)"     20
##  8 "...And God Created Woman (Et Dieu... crÃ©a la femme) (1956)"  61
##  9 "...And God Spoke (1993)"                                     20
## 10 "...And Justice for All (1979)"                              495
```

```
train_set %>% count(movieId) %>%
  left_join(bi,by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull(n)
```

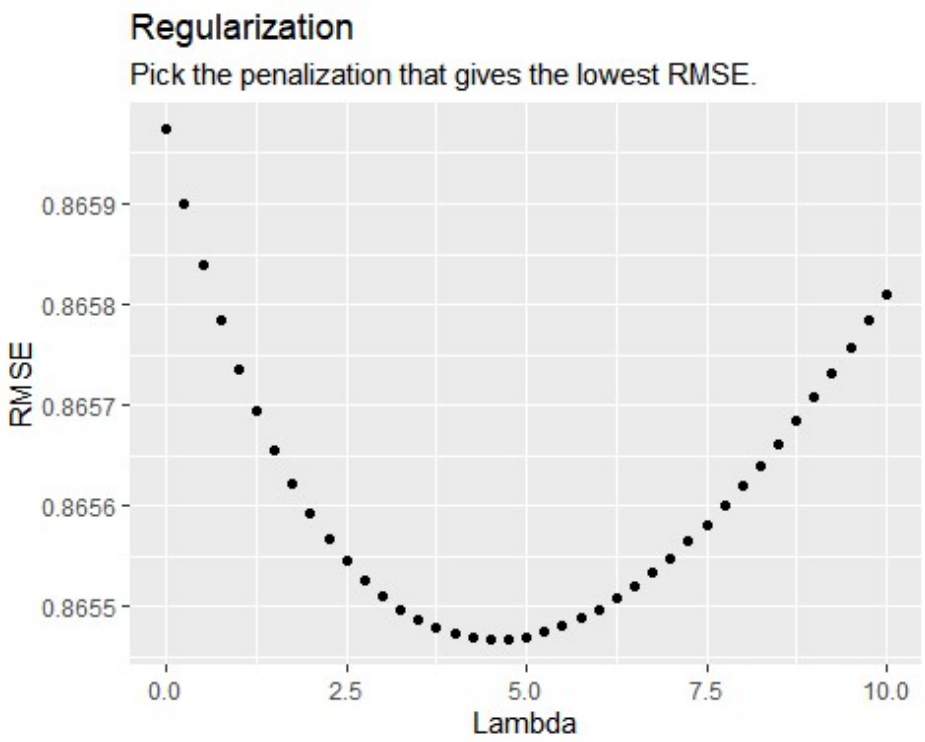
```
##  [1] 1 1 1 1 1 1 1 4 2 4
```

Regularization Model Prediction

In this section we will regularize the user and movie effects adding a penalty factor λ which is tuning parameter and based on the output we will pick the best value that minimizes the RMSE

```
# Regularization the user and movies effects calculation

regularization <- function(lambda, trainset, testset){
  # Mean calculation
  mu <- mean(trainset$rating)

  # predicting Movie Effect (bi)
  b_i <- trainset %>%
    group_by(movieId) %>%
    summarise(b_i=sum(rating-mu)/(n()+lambda))

  # predicting user effect calculation (bu)
  b_u <- trainset %>%
    left_join(b_i, by="movieId") %>%
    filter(!is.na(b_i)) %>%
    group_by(userId) %>%
    summarize(b_u=sum(rating-b_i-mu)/(n()+lambda))

  # prediction calculation mu + bi + bu
  prediction_ratings <- testset %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    filter(!is.na(b_i), !is.na(b_u)) %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(prediction_ratings, testset$rating))

}

# defining the set of lambdas to tune

lambdas <- seq(0,10,0.25)


# tuning the lambda

rmses <- sapply(lambdas,
                regularization,
                trainset=train_set,
                testset=test_set)
```

```
# plot the lambda vs RMSE

tibble(Lambda=lambdas, RMSE=rmses) %>%
  ggplot(aes(x=Lambda, y=RMSE)) +
  geom_point() +
  ggtitle("Regularization", subtitle = "Pick the penalization that gives the
lowest RMSE.") +
  theme_grey()
```



In next step we applied the best penalty factor of regularization to the linear model

```
# picking up the Lambda which will returns Lowest RMSE

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

# predicting rating using parameters achieved through regularization model

mu <- mean(train_set$rating)

# predicting Movie Effect - bi

b_i <- train_set %>%
```

```
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

# predicting user effect - bu

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

# prediction regularization model output

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

# predicting regularization bi and bu

result <- bind_rows(result,
                    tibble(Method = "Regularized bi and bu",
                           RMSE = RMSE(test_set$rating, reg_model)))

# intermediate results showing  improvement for RMSE

print.data.frame(result)
```

```
##                     Method      RMSE
## 1         Goal of Project 0.8649000
## 2 Random Prediction Model 1.5008922
## 3                    Mean 1.0611350
## 4               Mean + bi 0.9441568
## 5          Mean + bi + bu 0.8659736
## 6   Regularized bi and bu 0.8654673
```

Part 6 - Final Validation - using the validation set with edx

From the output we got we can conclude that regularization achievevd the target RMSE. Hence, in final stage we will train the complete edx set and calculate RMSE in the validation set. The project goal is to achieve if the RMSE stays below the target

```
#### final validation #####
# linear model with regularization output

mu_edx <- mean(edx$rating)

# predicting Movie effect (bi)
b_i_edx <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_edx)/(n()+lambda))

# predicting User effect (bu)
b_u_edx <- edx %>%
  left_join(b_i_edx, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu_edx)/(n()+lambda))

# final edx vs validation prediction
final_edx <- validation %>%
  left_join(b_i_edx, by = "movieId") %>%
  left_join(b_u_edx, by = "userId") %>%
  mutate(pred = mu_edx + b_i + b_u) %>%
  pull(pred)

# Updating the results table

result <- bind_rows(result,
                    tibble(Method = "Final Regularization (edx vs validation)
",
                           RMSE = RMSE(validation$rating, final_edx)))

# Show final result with RMSE achieved or not

print.data.frame(result)
```

```
##                                     Method      RMSE
## 1                          Goal of Project 0.8649000
## 2                  Random Prediction Model 1.5008922
## 3                                     Mean 1.0611350
## 4                                Mean + bi 0.9441568
## 5                           Mean + bi + bu 0.8659736
```

```
## 6                     Regularized bi and bu 0.8654673
## 7 Final Regularization (edx vs validation) 0.8648242
```

As expected, the RMSE calculated on the validation set ***(0.8648242)*** is lower than the target ***0.8649***.

Conclusion

We started by collecting and preparing the dataset for analysis, after that we explored the information seeking for insights that may help in model buidling exercise.

Once that was done we created random model to predicts the rating bassed on the probability distribution of each rating. The model does not gave proper result.

After that we started with linear model just with calculating mean of the observed ratings and from there, we added movie, user effects and with the help of regularization and linear model achieved the **RMSE of 0.8648242** successfully passing the target of **0.8649000**

Limitation

Some of the algorithms are computationally expensive to run on a normal laptop and therefore were unable to test. Only two predictors used, the movie and user information, and not considered other features. Modern recommendation system use many predictors.

The current model works only on existing users, movies and rating values. So in case a new user or movie is introduce then algorithm should run at that time.

Future Work

This report provides information with regards to simple models that predict ratings. There could be other approaches too.

References

1. Rafael A. Irizarry (2019), Introduction to Data Science: Data Analysis and Prediction Algorithms with R
2. https://cran.r-project.org/web/packages/available_packages_by_name.html↩