HarvardX (PH125.9x) - Data Science: Capstone - MovieLens Project

Wayne Koong Wah Yan

10/13/2020

1 Abstract

1.1 Purpose of this report

This report is written as part of a submission requirement of Capstone course, the ninth course in HarvardX Data Science Professional Certificate Program.

1.2 Objective of this report

This objective of this report is to apply the knowledge acquired throughout the Harvardx Data Science courses, analyze the given MovieLens dataset, and suggests the best model to predict the preference of a user on a movie, at a rate from 0.5 to 5.0.

1.3 Project Goal

The goal of the project is to predict a user's movie rating (using MovieLens 10M Dataset) using models as such to obtain the lowest Rooted Mean Square Error (RMSE). The RMSE target is 0.86490 or lower.

2 Introduction

2.1 Recommendation Systems

Recommendation systems, a newer and wider definition of 80's Decision Support Systems, brings the logic or reasoning of a decision from the hand of a human to machine. This improvement from Decision Support Systems not only increase the accuracy and speed of recommendation to the business, but with a wider scale, with increased capability, scalability and affordability.

Today, recommendation system is part of our life. It is now widely adopted in commercial solutions to the general public. It forms the main engine of the video recommendation by tiktok, youtube, netflix, etc., traffic route recommendation by waze, google map, etc., products recommendation at ecommerce site like Amazon, Lazada, Shoppee, etc.. There's no surprise even the typical brick & mortal shops already using recommendation systems for the placement of products, store layout, promotion and etc.

The main objective of recommendation system is to predict either discrete or continuous probability of events based on given known historical data. It predicts rating or preference a typical user would decides by applying the best fit algorithms or models.

A recommendation systems is highly rely on given historical data that it's used to "train" the algorithm or model. It learns from the historical data, and models the outcome based on data points, which subsequently uses to derive or predict outcome of the given new dataset. The result is typically in a predefined scale, e.g. from 0 to 1, of which is subsequently interpreted into rating or preferences of choice, e.g. rating from 1 to 5, or choice of Yes or No, etc.

2.2 MovieLens (10M) Dataset

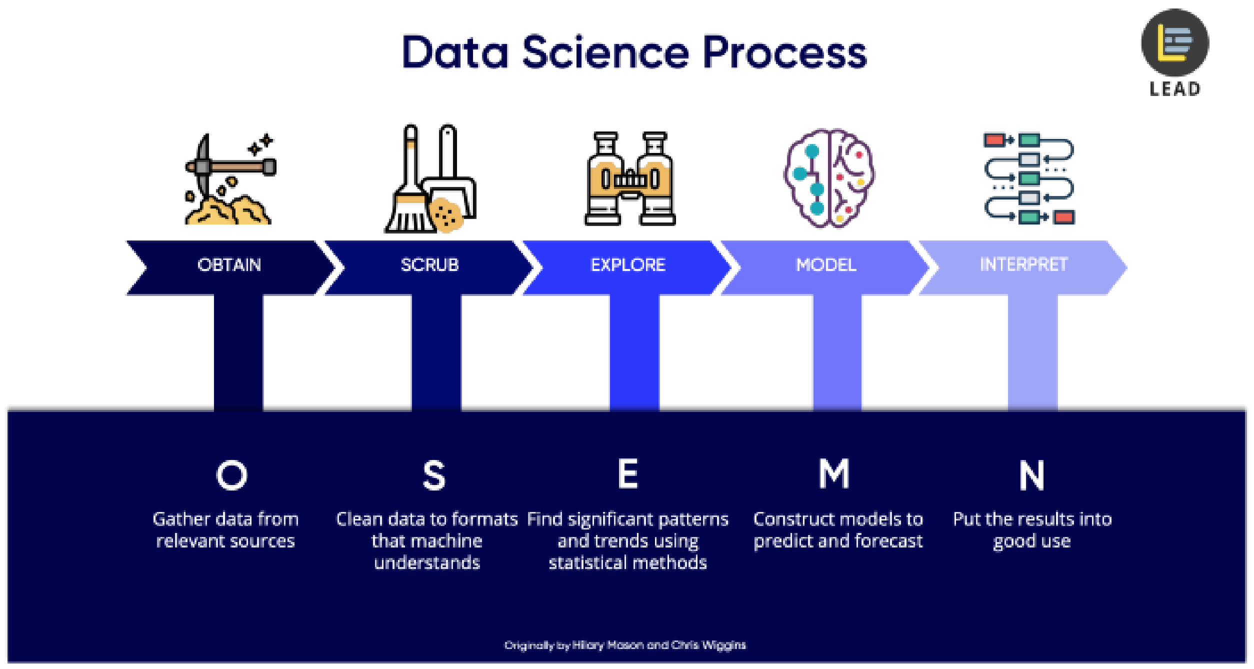
MovieLens is a dataset that contains users preferences of movies. The data is originated from eachmovie.org and subsequently becoming a research platform by GroupLens Research, a research lab in the University of Minnesota.

GroupLens published several MovieLens dataset. The full dataset (last updated 9/2018) contains 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users.

For the purpose of this report, the MovieLens 10M dataset is used. This dataset is released in Jan 2009, and contains 10,000,000 ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

2.3 Key steps

In order to achieve the project's objective and goal, proper defined steps are important ensuring the produced analysis and its results are correct. For this project, I am following the OSEMN Framework process:


Data Science Process
LEAD
OBTAIN
SCRUB
EXPLORE
MODEL
INTERPRET
O
Gather data from relevant sources
S
Clean data to formats that machine understands
E
Find significant patterns and trends using statistical methods
M
Construct models to predict and forecast
N
Put the results into good use
Originally by Hilary Mason and Chris Wiggins

OSEMN Framework process is a standardized process that is widely accepted and used among data science practitioners. This framework governs steps taken, ensuring probability of accurate analysis whilst allows steps to be backtrack-able and re-executable. OSEMN Framework process is consists of following 5 steps:

2.3.1 Step1 Obtain Data

Obtain data means identifying and acquiring the needed and correct dataset. This step is very important as a flawed dataset will miss-led and impact the probability of getting accurate prediction. After the complete dataset is obtained or downloaded, the data should be parsed and prepared in a form that is processable.

2.3.2 Step2 Scrub Data

Scrub data means to clean or filter unwanted "noise" from dataset. Depends on the quality of dataset, sometime a massive data cleansing may be required. This step needs to be executed properly and data should be explored from all angles, as "garbage in, garbage out" philosophy, not clean data with irrelevant or incorrectly parse data may rendered analyzed result null.

2.3.3 Step3 Explore Data

Explore data means examine data, or making sense of the dataset. This step involves careful data properties, e.g. data cardinality, relationship, factors, and data types like numerical, categorical, ordinal, nominal, etc. inspections.

Descriptive statistics are always compute to extract features and to test significant variables and their correlation. These extracted info are normally present in visualisation (e.g. chart) for patterns and trends identification.

2.3.4 Step4 Model Data

Model data is the step where models are selected, applied, tuned and executed to get the required outcome. This is the key step that resulted whether we able to produce a correct and high probability prediction, or a biased or wrong analysis.

Here, dimension of dataset is scrutinized and reduced if necessary, selecting the relevant features and value that contributes to the prediction of results. Various models are select and train:

- logistic regressions to perform classification to differentiate value,
- linear regression to forecast value,
- clustering by grouping data for better understanding of the logic,
- etc.

In short, regressions and predictions are typically use for forecasting future values, whilst classifications are to identify, and clustering to group values.

2.3.5 Step5 Interpreting Data

Interpreting Data means interpreting models and results, and presenting them in a human readable format. There is no standard format on how outcome should be presented. It can be simplified charts as those in newspaper, or series of highly technical charts for technical reader. A well-structured, clear and with actionable story report with relavent visual and data helps readers read and understands.

Regardless how good all other steps are performed, failure to present and communicate to the reader clearly & precisely means the efforts may not be appreciated (a.k.a getting continuous support, buy-in & fundings).

3 Methods & Analysis

3.1 Methods

The method that is used in this report is **loss function**. Loss function is a function that estimate parameter of an event or value of one or more variables of the difference between estimated and true values for an instance of data.

Commonly in use loss functions include squared loss , mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE).

In this project, Root Mean Squared Error (RMSE) is selected for reporting of prediction outcome.

3.1.1 Squared Loss Function

Squared loss function is the most commonly us loss function. if $\hat{y}_i$ is the predictor and y_i is the observed outcome, the squared loss function is:

$$(\hat{y}_i - y_i)^2$$

3.1.2 Mean Absolute Error (MAE)

Mean absolute error is the average of absolute differences between the predicted value and the true value, producing a linear value, which all errors are equally weighted. Thus, when predicting ratings in a scale of 1 to 5, the MAE assumes that an error of 2 is twice as bad as an error of 1.

If N is the number of observations, the MAE is given by this formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$$

3.1.3 Mean Squared Error (MSE)

Mean squared error is similar with squared loss function, averaging the squared error of the predictions, but squared. The formula give more weight to larger error. If N is the number of observations, the mean squared error formula is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

3.1.4 Root Mean Squared Error (RMSE)

Root mean squared error is similar to mean squared error and give more weight to larger error, minimizing the impact from smaller errors. RMSE is always use instead of MSE because RMSE preserve the original unit of measurement. The root mean squared error formula is:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

3.2 Processes & Techniques

OSEMN Framework process is followed in this project to ensure standardize procedures and good data organization that increase the probability of getting accurate outcome. In the following sections, I am executing the MovieLens modeling following the OSEMN Framework.

3.2.1 Step1 Obtain Data

The MovieLens 10M dataset for this project is downloaded from grouplens.org. The file ml-10m.zip is downloaded using following codes:

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

The downloaded ml-10m.zip contains following 2 files:

1. movies.dat (265,105,635 bytes)
2. ratings.dat (522,197 bytes)

File movies.dat contains 10,681 rows of reviewed movies and its genes, whilst the ratings.dat contains 10,000,054 rows of movies rating. These 2 files are read and merge into the main dataset for the project analysis. Following codes are used to prepare the dataset for analysis:

```
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(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

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

The output movielens (data.table class) parameter contains 10,000,054 rows of data and each row consists userId, movieId, rating, timestamp, title, genres data.

3.2.1.1 Prepare edx (training) and validation dataset

As setup in project guideline, the model exploration should only be performed on the training dataset, and the validation dataset should be only use for final model evaluation. As such, i incorporated following script to separate movielens dataset into edx (training) and validation dataset.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
validation <- movielens[test_index,]
```

3.2.2 Step2 Scrub Data

Prior performing data analysis and modeling, it's important to ensure the dataset is clean. The most important data scrubbing is to ensure that the partitioned data, especially the edx (training) dataset contains all movies and users of the validation dataset. This step is crucial preventing missing relationship during modeling that resulted in null (N.A.) value that is not handled by RMSE function.

```r
temp <- validation
validation <- validation %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation, by = c("userId", "movieId"))
edx <- rbind(edx, removed)
```

3.2.2.1 Ensure no null (NA) data

This step is important as null data will affect the overall analysis and may seriously jeopardize the outcome. The checking is perform by running follow syntax.

```r
anyNA(validation)
```

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

```r
anyNA(edx)
```

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

Based on above analysis, there is no null data thus, the dataset is good for further analysis.

3.2.3 Step3 Explore Data

There are total of 10,000,054 rows of record in the movieLens dataset. Of each 9,000,055 rows partition into edx (training) dataset and 999,999 rows into the validation dataset.

Following are some sample records of the movielens:

Table 1: Some Movielens Data

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

** *As the project instruction, I have to assume that I only knows about the edx dataset, and performed all activities, including data exploration, data analysis, modeling and prediction only on edx dataset, thus, the following activities are only performing on edx dataset.* **

3.2.3.1 Columns Definition

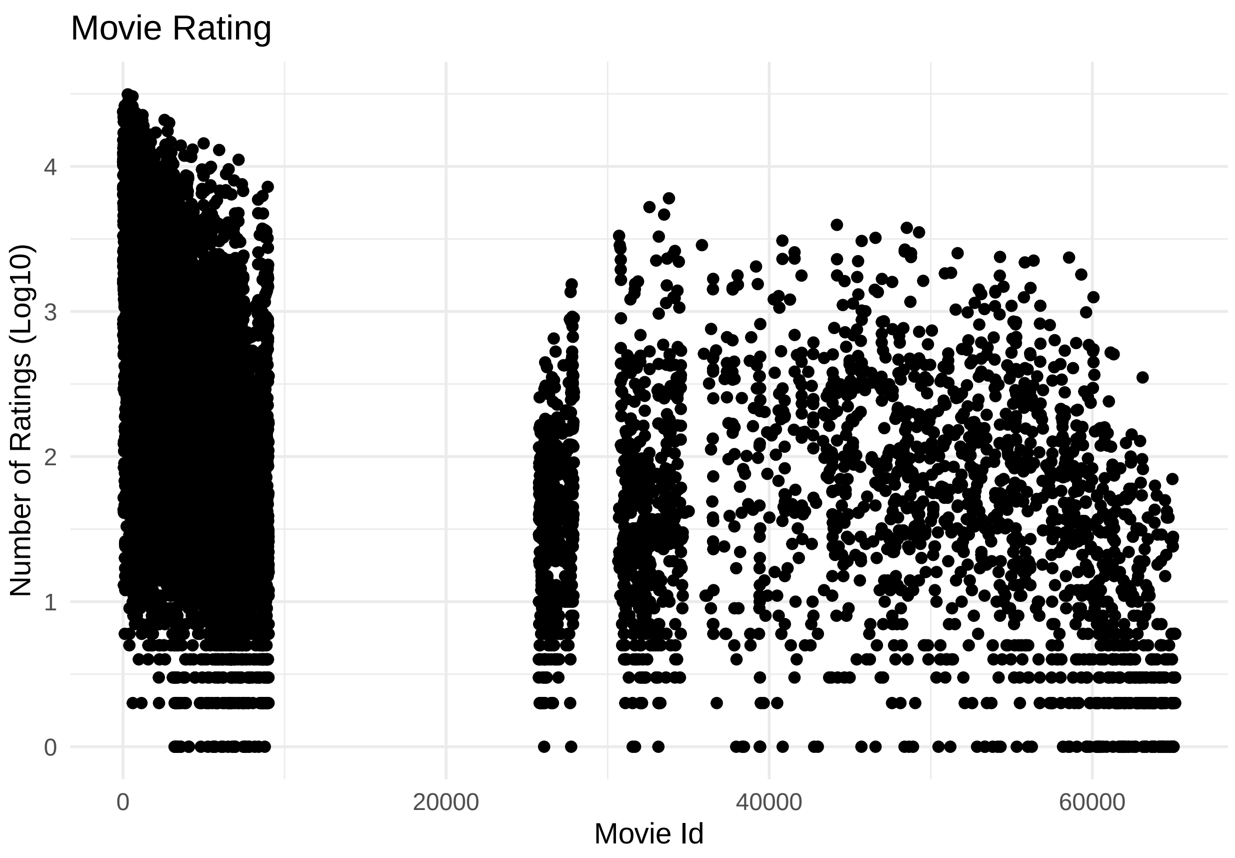
This is always a overlook step in data preparation. Sometime, due to input or parsing mistake, or sometimes due to the origin of dataset, some data are parse incorrectly, e.g. alphabet in timestamp, space in between number in id column, etc.

3.2.3.1.1 Movie ID (movieId)

Movie ID is the unique id for each movie. There are total of 10,677 movies in the dataset. Movie ID column (movieId) data type is numeric ranging from 1 to 65133.

Following are some sample movieId:
122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, 466, 520, 539, 588, 589, 594, 616, 110, 260, 376, 590, 648, 719, 733, 736, 780, 786, 802, 1049, 1073, 1210, 1356, 1391, 151, 213, 1148, 1246, 1252, 1276, 1288, 1408, 1552, 1564, 1597, 1674, 3408, 3684, 4535, 4677, 5299, 5505, 5527, 5952, 6287, 6377, 6539, 7153, 7155, 8529, 8533, 8783, 27821, 33750, 21, 39, 150, 153, 161, 165, 208, 231, 253, 266, 317, 344, 349, 367, 380, 410, 435, 440, 480, 500, 586, 587, 592, 595, 597, 1, 7, 25, 28, 30, 32, 47, 52, 57, 58

```r
edx %>% group_by(movieId) %>%
  summarise(count=n()) %>%
  ggplot(aes(movieId, log10(count))) + geom_point() +
  ggtitle("Movie Rating") +
  xlab("Movie Id") +
  ylab("Number of Ratings (Log10)") +
  theme_minimal()
```



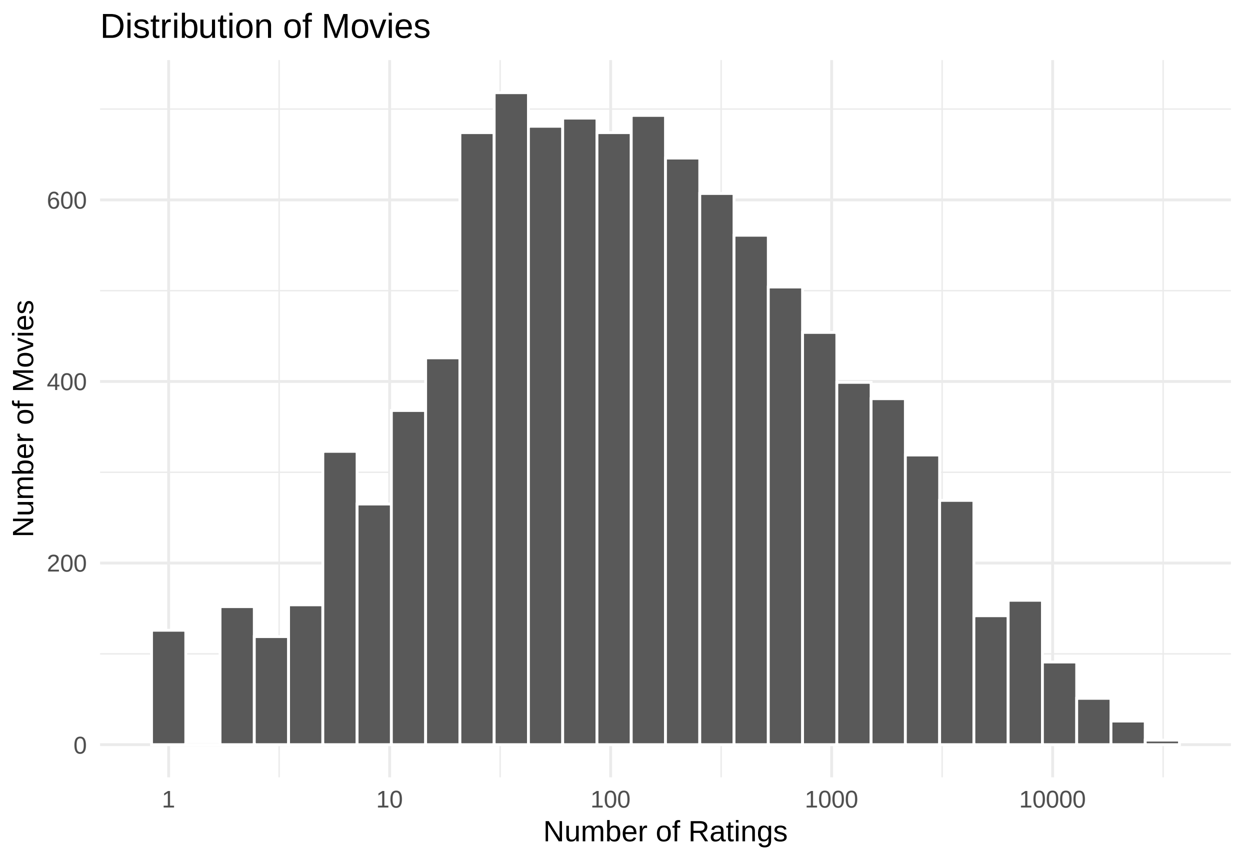
Based on above chart, I notice some movies are frequently rated whilst some only being rate a few times, movie id 294 rated 31,362 times whilst movie id 3107 only being rated 1 times:

- total of 1,933 movies rated less than 20 times,
- total of 1,131 movies rated more than 2000 times,
- sigma of count of movieId rating is 2238.48092,
- mean of count of movieId rating is 842.93856.

This mean, we have to regularise count of movieId rating if movieId is use for modeling.

By plotting movie id distribution, we notice the distribution is almost normal, which is support by above calcualted sigma and mean.

```r
edx %>% group_by(movieId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
    geom_histogram(color = "white") +
    scale_x_log10() +
    ggtitle("Distribution of Movies") +
    xlab("Number of Ratings") +
    ylab("Number of Movies") +
    theme_minimal()
```



3.2.3.1.2 User ID (userId)

User ID are the unique id for each user. There are total of 69,878 users in the dataset. User ID column (userId) data type is numeric ranging from 1 to 71567.

Following are some sample userId:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 22, 23, 24, 26, 27, 28, 29, 30, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108

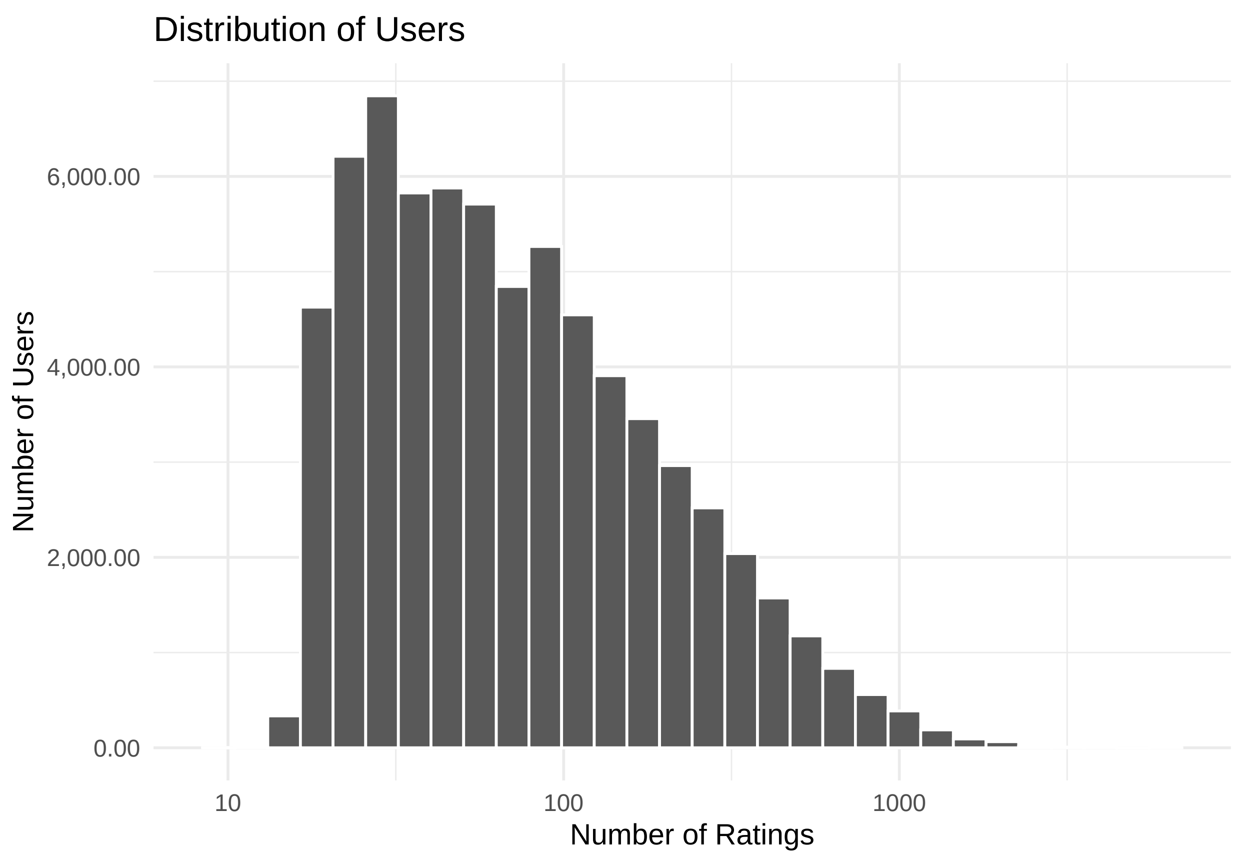
I notice number of rating per user is not consistent, user id 57960 rated 6,616 movies whilst user id 61147 only rated 10 movies:

- total of 4,966 users rated less than 20 movies,
- total of 611 users rated more than 1000 movies,
- sigma of count of userId rating is 195.06019,
- mean of count of userId rating is 128.79669.

Count of userid need to be regularised if it is used for subsequent modeling.

By plotting rating against users, I notice the distribution is skewed toward right.

```r
edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
    geom_histogram(color = "white") +
    scale_x_log10() +
    ggtitle("Distribution of Users") +
    xlab("Number of Ratings") +
    ylab("Number of Users") +
    scale_y_continuous(labels = comma) +
    theme_minimal()
```



3.2.3.1.3 Timestamp (timestamp)
Timestamp is the Unix date time the rating is made by a user for a movie. In order to make sense of the timestamp data, I convert the data into year format using following codes:

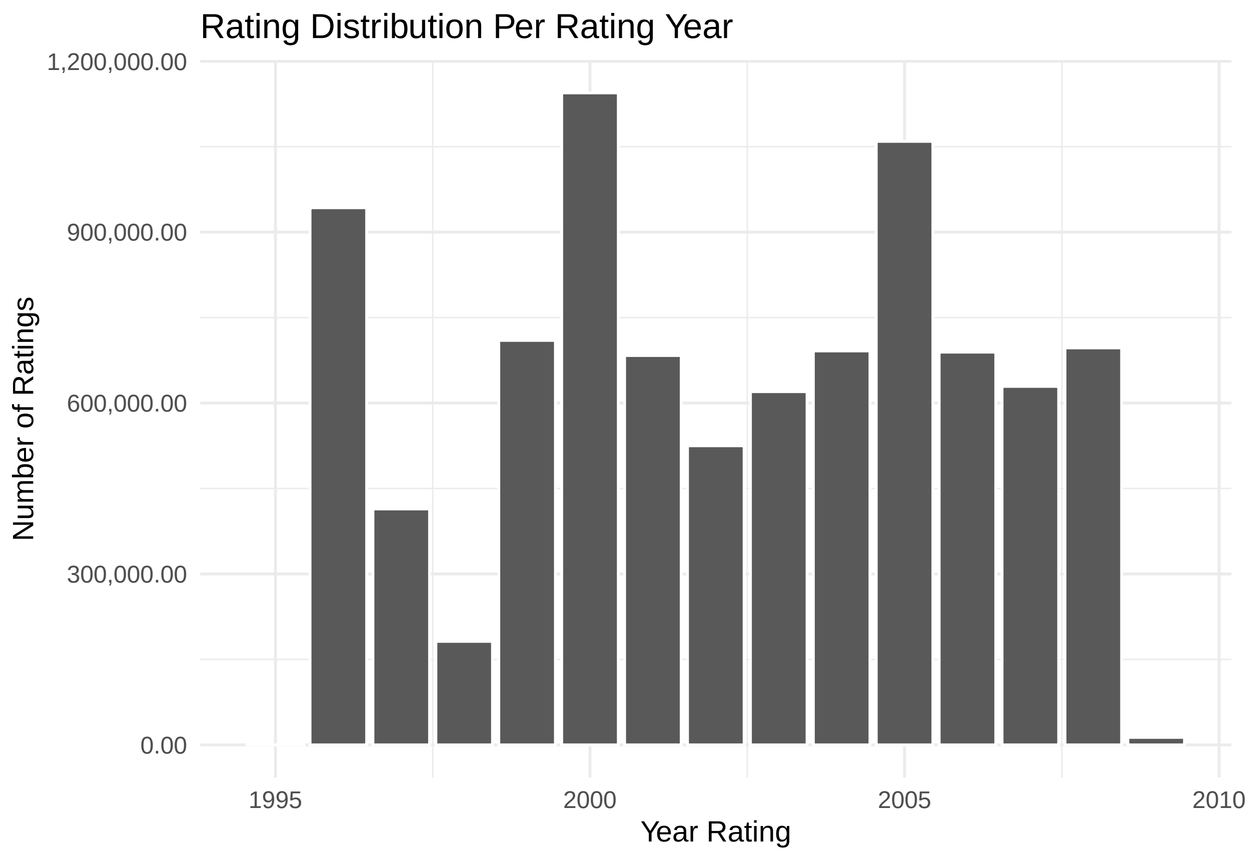
```r
edx <- edx %>% mutate(yr_rating=year(as_datetime(timestamp,origin="1970-01-01")))
```

Based on the data, the earliest rating is rated in 1995 and the latest rating is rated in 2009. This means, rating of movies are done for a span of 15 years, where:

- In year 2000, 1,144,349 rating done,
- In year 2009, only 13,123 rating done,
- sigma of year rating performed is 3.7102,
- mean of year rating perfoemd is 2002.20007.

Although, the figure above shows that the different between year are not drastic, however, regularise of yr_rating may help smoothed data and reduced bias.

```r
edx %>% ggplot(aes(x=yr_rating)) +
    geom_bar(color = "white") +
    ggtitle("Rating Distribution Per Rating Year") +
    xlab("Year Rating") +
    ylab("Number of Ratings") +
    scale_y_continuous(labels = comma) +
    theme_minimal()
```



3.2.3.1.4 Movie Title (title) Column title contains the title of the movie suffix with release year. As movie title itself is not unique, suffixing release year is a common practice in the industry. Following are some movie title that are reuse throughout the years:

```r
title_unq <- unique(edx$title)
title_unq_wo_year <- substr(title_unq,1,str_length(title_unq)-7)
title_dup <- title_unq_wo_year[which(duplicated(title_unq_wo_year))]
title_dup[1:10]
```

```
##  [1] "Psycho"                        "Of Mice and Men"
##  [3] "Father of the Bride"           "Cheaper by the Dozen"
##  [5] "Punisher, The"                 "Texas Chainsaw Massacre, The"
##  [7] "Bandits"                       "Peter Pan"
##  [9] "Sabrina"                       "Cape Fear"
```

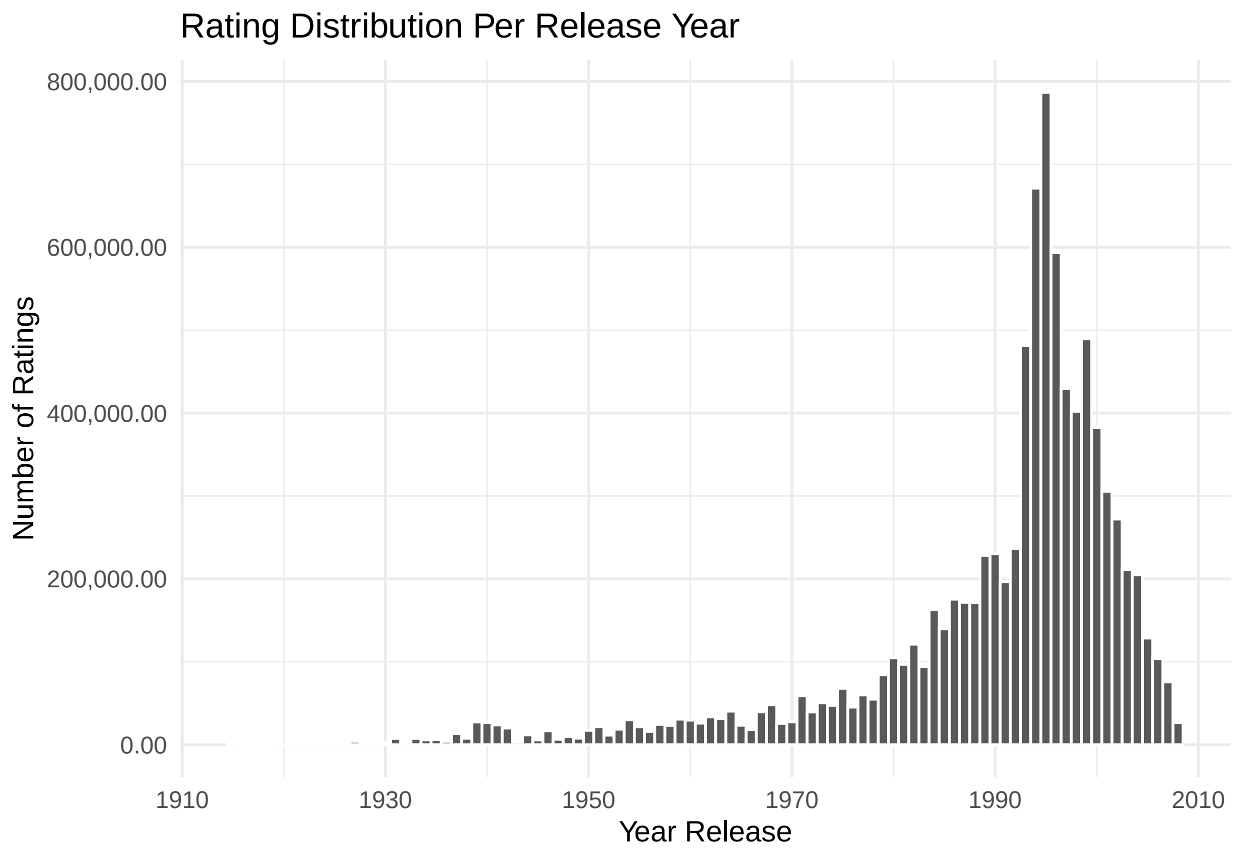
- Movie with title "Psycho": Psycho (1960), Psycho (1998)
- Movie with title "Of Mice and Men": Of Mice and Men (1992), Of Mice and Men (1939)

The release year that suffix in this column is a data variant with interest. I separate this information into column named "yr_release"

```r
edx <- edx %>% mutate(title=str_trim(title)) %>%
  mutate(yr_release=as.numeric(substr(title,str_length(title)-4,
                                      str_length(title)-1)))
```

After yr_release extracted from title column, I notice that the most movies release in year 2002, more movies release after 1993 and less movies release before 1929.

```r
edx %>% ggplot(aes(x=yr_release)) +
    geom_histogram(color = "white", binwidth = 1) +
    ggtitle("Rating Distribution Per Release Year") +
    xlab("Year Release") +
    ylab("Number of Ratings") +
    scale_y_continuous(labels = comma) +
    theme_minimal()
```



Breakdown of movie release before or on 1930:

Table 2: Number of movie release before or in 1929

	1915	1916	1917	1918	1919	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929
count	1	2	2	2	4	5	3	7	6	6	10	10	19	10	7

Breakdown of movie since 2002:

Table 3: Number of movie release after or in 2002

	2002	2003	2004	2005	2006	2007	2008
count	441	366	342	331	345	363	251

3.2.3.1.5 Rating (rating)

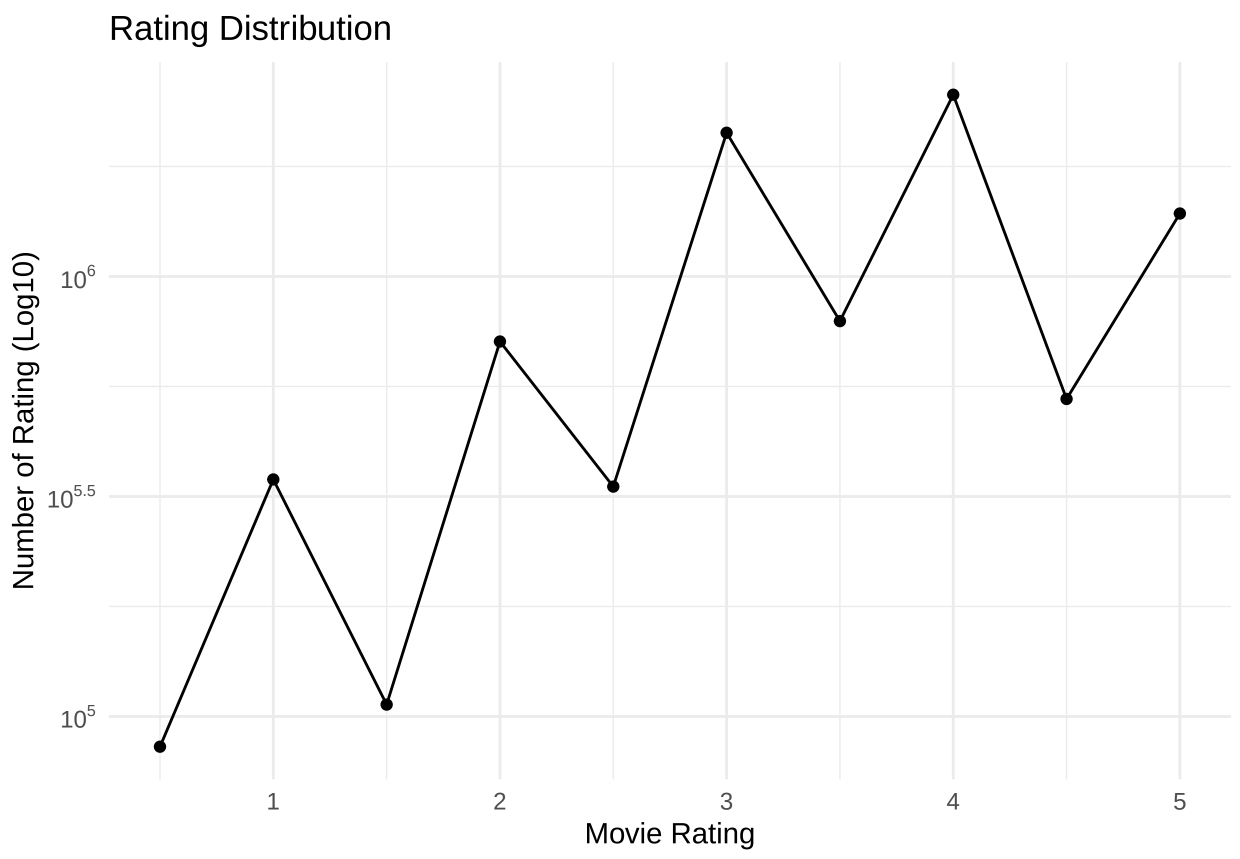
Rating is the movie rating by each user for a movie. It is a numeric data with discrete value from 0.5 to 5.0 as shown in following table with the records count for each rating.

Table 4: Number of Rating

0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
85374	345679	106426	711422	333010	2121240	791624	2588430	526736	1390114

Rating is the main variant that I am modeling on, predicting the rating value for each user-movie rating. Following is the rating distribution chart:

```r
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_format("log10", math_format(10^.x))) +
    ggtitle("Rating Distribution") + xlab("Movie Rating") + ylab("Number of Rating (Log10)") +
  theme_minimal()
```



3.2.3.1.6 Genes (genes)

Genes are refer to the categorization of movies characterists. Each value of genes may contains concatenation of multiple gene seperate by pipe ("|"). This information is helpful to made better predictions, especially with correlating users. However, I don't use this value in my modeling.

The data set contains 797 different combinations of genres. Here is the list of the first 10.

Table 5: Sample Genres

genres	count
(no genres listed)	7
Action	24482
Action|Adventure	68688
Action|Adventure|Animation|Children|Comedy	7467

genres	count
Action|Adventure|Animation|Children|Comedy|Fantasy	187
Action|Adventure|Animation|Children|Comedy|IMAX	66
Action|Adventure|Animation|Children|Comedy|Sci-Fi	600
Action|Adventure|Animation|Children|Fantasy	737
Action|Adventure|Animation|Children|Sci-Fi	50
Action|Adventure|Animation|Comedy|Drama	1902

Interestingly, there are 7 movies without any genre defined.

3.2.3.2 Data Relationship Exploration

In addition of above columns levels data exploration, following data combination also examined to evaluate their relationship.

3.2.3.2.1 Year Lapse Between Release Year and Review Year

I hypothesed that user will not rate a movie using the same perspective, if a movies is viewed and rated immediately after the movie released, or years after its released. This could be due to influence from the press and external information, or the release of better or worst movies of the similar topic or gerne, or remade.

For the purpose, i introduced a new column yr_lapse, which is the year lapse between movie release year and rating year.

```
edx <- edx %>%
  mutate(yr_lapse=yr_rating-yr_release)
```

Strangely, I notice 175 records which the rating is done before the movie release, i assume this is due to either the reviewed is done prior movie release, or error in data input:

Table 6: Movies With Negative Year Lapse

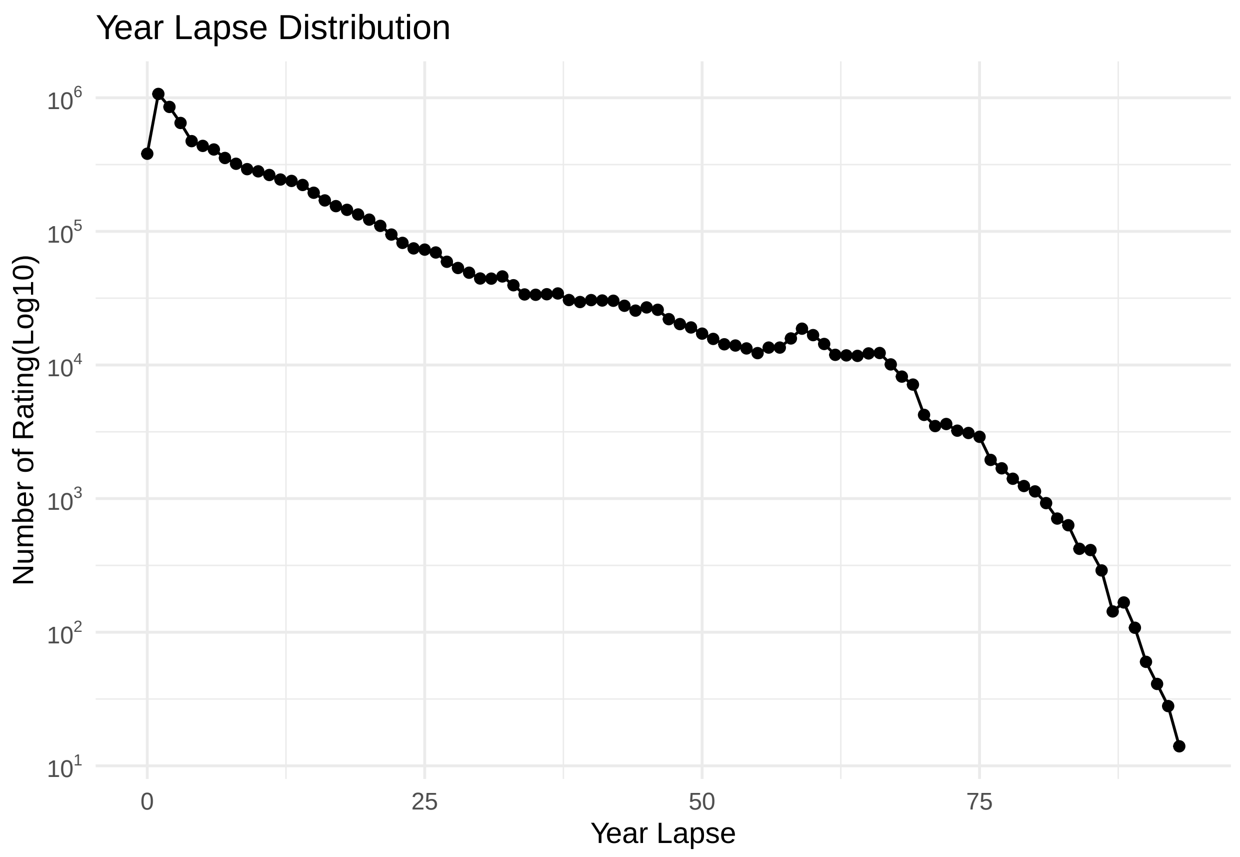
userId	movieId	rating	title	genres	yr_rating	yr_release	yr_lapse
785	981	3	Dangerous Ground (1997)	Drama	1996	1997	-1
1468	879	2	Relic, The (1997)	Horror|Thriller	1996	1997	-1
1583	981	1	Dangerous Ground (1997)	Drama	1996	1997	-1
1766	870	5	Gone Fishin' (1997)	Comedy	1996	1997	-1
1766	879	5	Relic, The (1997)	Horror|Thriller	1996	1997	-1
1766	981	3	Dangerous Ground (1997)	Drama	1996	1997	-1
2080	2235	2	One Man's Hero (1999)	Drama|War	1998	1999	-1
2219	2769	4	Yards, The (2000)	Crime|Drama	1999	2000	-1
3052	779	3	'Til There Was You (1997)	Drama|Romance	1996	1997	-1
3370	1055	5	Shadow Conspiracy (1997)	Thriller	1996	1997	-1

To mitigate the impact of this data, I reset these yr_lapse value to zero (0) using following codes:

```
  edx <- edx %>% mutate(yr_lapse=ifelse(yr_lapse<=0,0,yr_lapse))
```

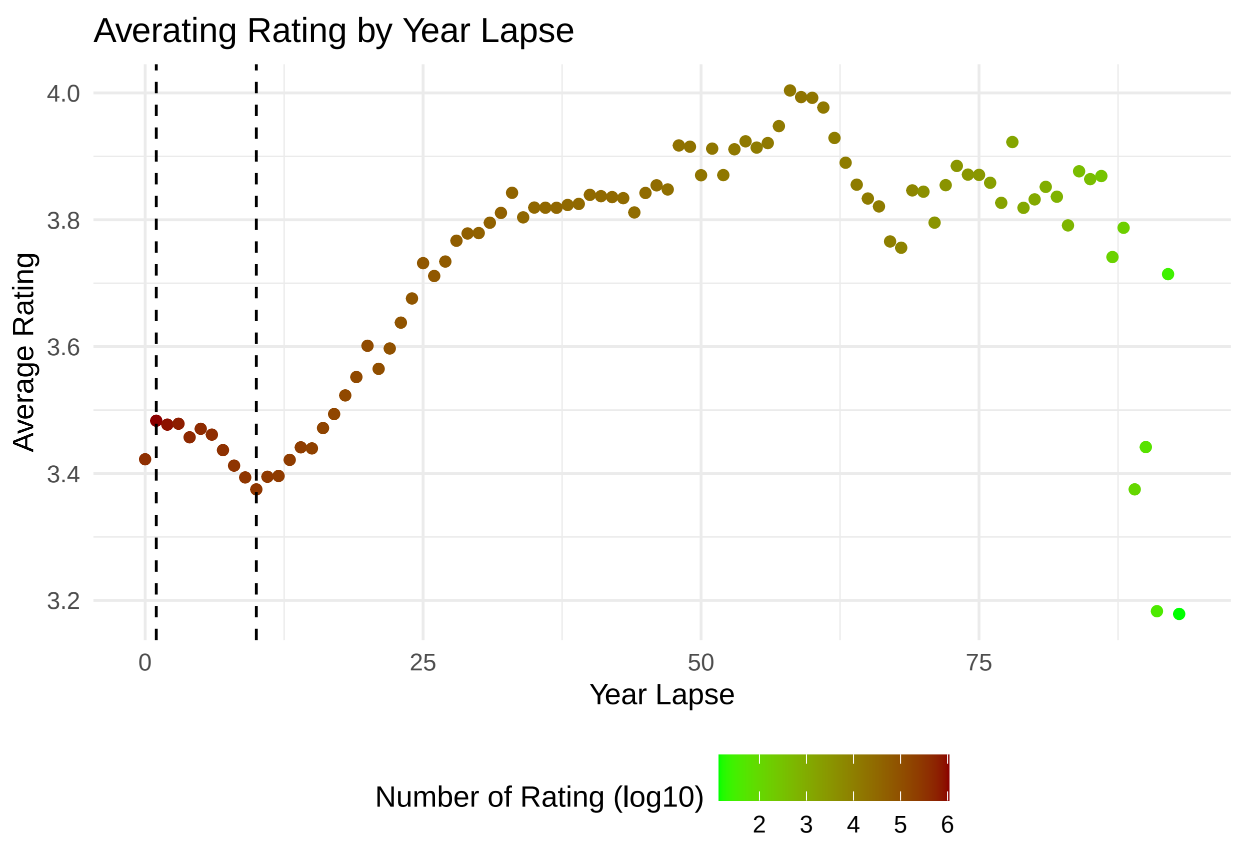
Following is the yr_lapse distribution:

```r
edx %>% group_by(yr_lapse) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=yr_lapse, y=count)) + geom_line() + geom_point() +
  scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x))) +
    ggtitle("Year Lapse Distribution") + xlab("Year Lapse") +
  ylab("Number of Rating(Log10)") + theme_minimal()
```



Plotting year lapse against average rating give an interesting fact. There's a trend reversal when yr_lapse = 10, users tend to rate movies lower yearly from year 1 toward year 10, and increase rating from year 11 onward. This could be due to the fact that, users perceived movie released 10 years or above as classical movie and only best movie are watched and reviewed.

```r
edx %>% group_by(yr_lapse) %>% summarise(m=mean(rating), n=n()) %>%
  ggplot(aes(yr_lapse, m)) + geom_point(aes(color=log10(n))) + theme_minimal() +
  scale_color_gradient(low="green", high="darkred") + xlab("Year Lapse") +
  ylab("Average Rating") + ggtitle("Averating Rating by Year Lapse") +
  labs(color = "Number of Rating (log10)") +
  theme(legend.position = "bottom") + geom_vline(xintercept = c(1,10),linetype=2)
```



3.2.3.2.2 Users & Movies

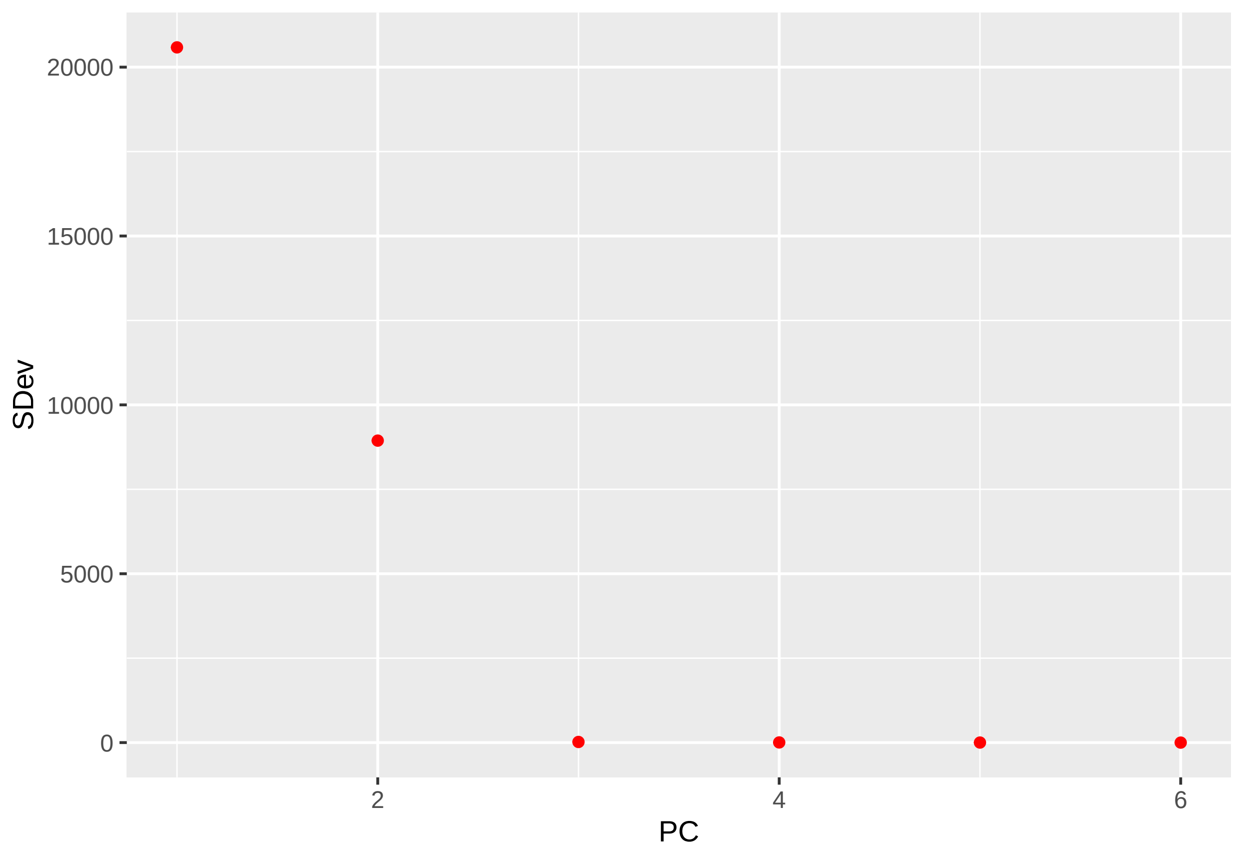
According to Principal Component Analysis, I notice that users and movies are 2 most important variants:

```
edx_minimal <- edx %>% select(userId,movieId,rating,yr_release,
                              yr_rating, yr_lapse)
edx_pca <- prcomp(edx_minimal)
edx_pca$rotation %>% knitr::kable(caption="Principal Component Analysis")
```

Table 7: Principal Component Analysis

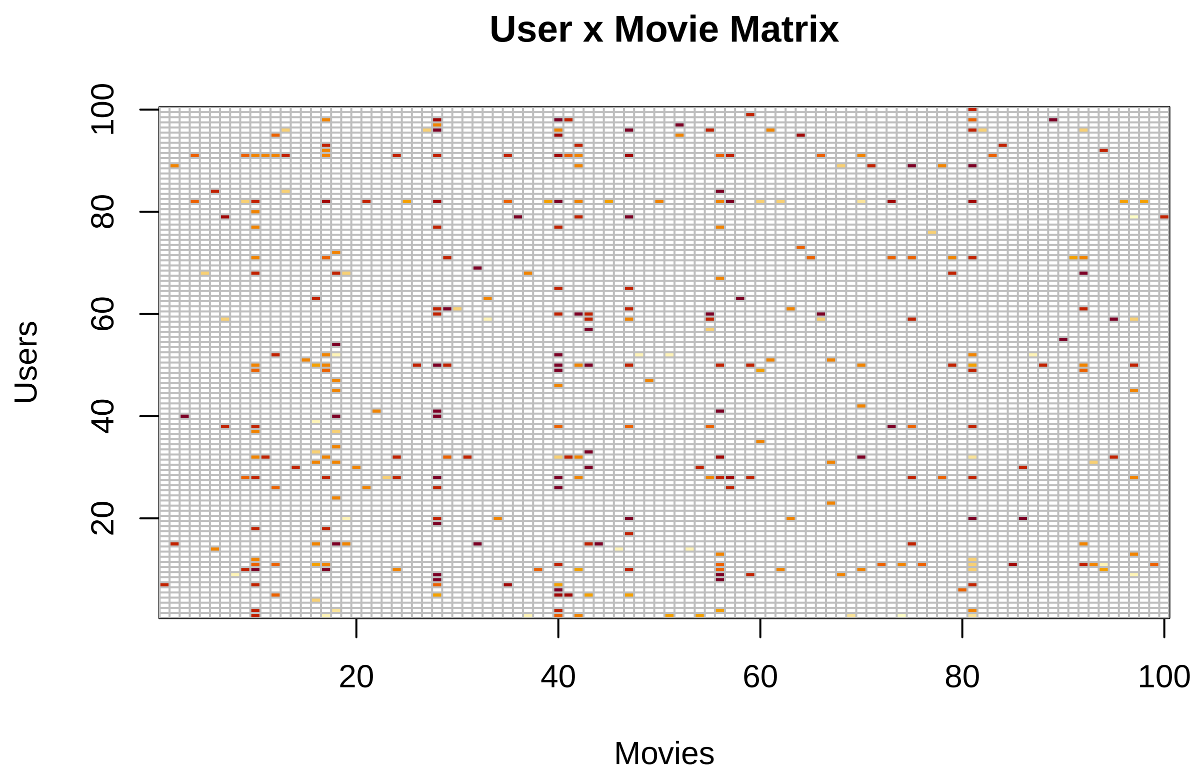
	PC1	PC2	PC3	PC4	PC5	PC6
userId	1.00000	0.00264	0.00000	0.00000	0.00000	0.00000
movieId	0.00264	-1.00000	0.00040	-0.00020	0.00000	0.00000
rating	0.00000	0.00000	0.00660	-0.01406	0.99988	0.00000
yr_release	0.00000	-0.00036	-0.69427	0.42959	0.01062	0.57735
yr_rating	0.00000	-0.00015	0.02499	0.81604	0.01131	-0.57735
yr_lapse	0.00000	0.00021	0.71925	0.38644	0.00068	0.57735

```
qplot(1:length(edx_pca$sdev), edx_pca$sdev, xlab = "PC",
      ylab="SDev", colour = I("red"))
```



Via below (sampled) user-movie matrix, i notice the matrix is sparse, with the vast majority of empty cells.

```
set.seed(1,sample.kind = "Rounding")
users <- sample(unique(edx$userId), 100)
edx %>% filter(userId %in% users) %>%
  select(userId, movieId, rating) %>%
#  mutate(rating = 1) %>%
  spread(movieId, rating) %>%
  select(sample(ncol(.), 100)) %>%
  as.matrix() %>% t(.) %>%
  image(1:100, 1:100,. , xlab="Movies", ylab="Users")
abline(h=0:100+0.5, v=0:100+0.5, col = "grey")
title("User x Movie Matrix")
```



3.2.4 Step4 Model Data

3.2.4.1 Data Preparation

As discussed in Processes & Tehniques section above, 2 dataset edx and validation are prepared. As instructed, the validation dataset can only be use for final model evaluation, thus the modeling analysis & training is done on edx dataset.

The edx dataset is partitioned into training dataset (edx_train) and test dataset (edx_test). The dataset is partition in such a way the training dataset (edx_train) contains 90% of edx data, and the test dataset (edx_test) contains 10% of edx data, without data overlapping, using following codes:

```
set.seed(1,sample.kind = "Rounding")
test_index <- createDataPartition(y = edx$rating,
                                  times = 1, p = 0.1, list = FALSE)
edx_test <- edx[test_index]
edx_train <- edx[-test_index]

temp <- edx_test
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

removed <- anti_join(temp, edx_test, by = c("userId", "movieId"))
edx_train <- rbind(edx_train, removed)

rm(temp,removed)
```

Similar with the codes use in partitioned movielens main dataset to edx (training) and validation dataset, dataset are scrub to ensure that the newly partitioned data, the edx_test dataset contains all movies and users of the edx_train dataset. This step is crucial preventing missing relationship during modeling that resulted in null (N.A.) value that is not handled by RMSE function

3.2.4.2 Modeling Technique

I am modeling movielens dataset first by calculating the mean from the entire dataset, and then applying Regularized Linear Regression.

The objective of modeling is to achieve RMSE below 0.8649

```r
result <- tibble(Method = "Project Target", RMSE = 0.86490)
result %>% knitr::kable(caption="RMSE Result")
```

Table 8: RMSE Result

Method	RMSE
Project Target	0.8649

3.2.4.2.1 Using the mean

I predict that the same rating will be given by all movies regardless of user, which demonstrated by following formula:

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

where u is user and m is movie.
The $\epsilon_{u,i}$ is the independent errors sampled from the same distribution centered at 0 and μ is the true rating for all movies.

Based on edx_traing dataset, the mean and RMSE of this estimation calculated using following syntax,and are respectively:

```r
mu <- mean(edx_train$rating)
mu_rmse <- RMSE(edx_test$rating,mu)
result <- rbind(result, c(Method = "Mean", RMSE = round(mu_rmse,5)))
result %>% knitr::kable(caption="RMSE Result")
```

Table 9: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005

3.2.4.2.2 Using the Regularized Movie Effect

As analyzed in Data Exploration section, I notice some movies are rated less than 20 times. Regularised of Movie effect is necessary to reduce prediction error causing by this data. The prediction formula with regularised movie effect is as follow:

$$Y_{u,m} = \mu + \hat{b}_m(\lambda) + \epsilon_{u,m}$$

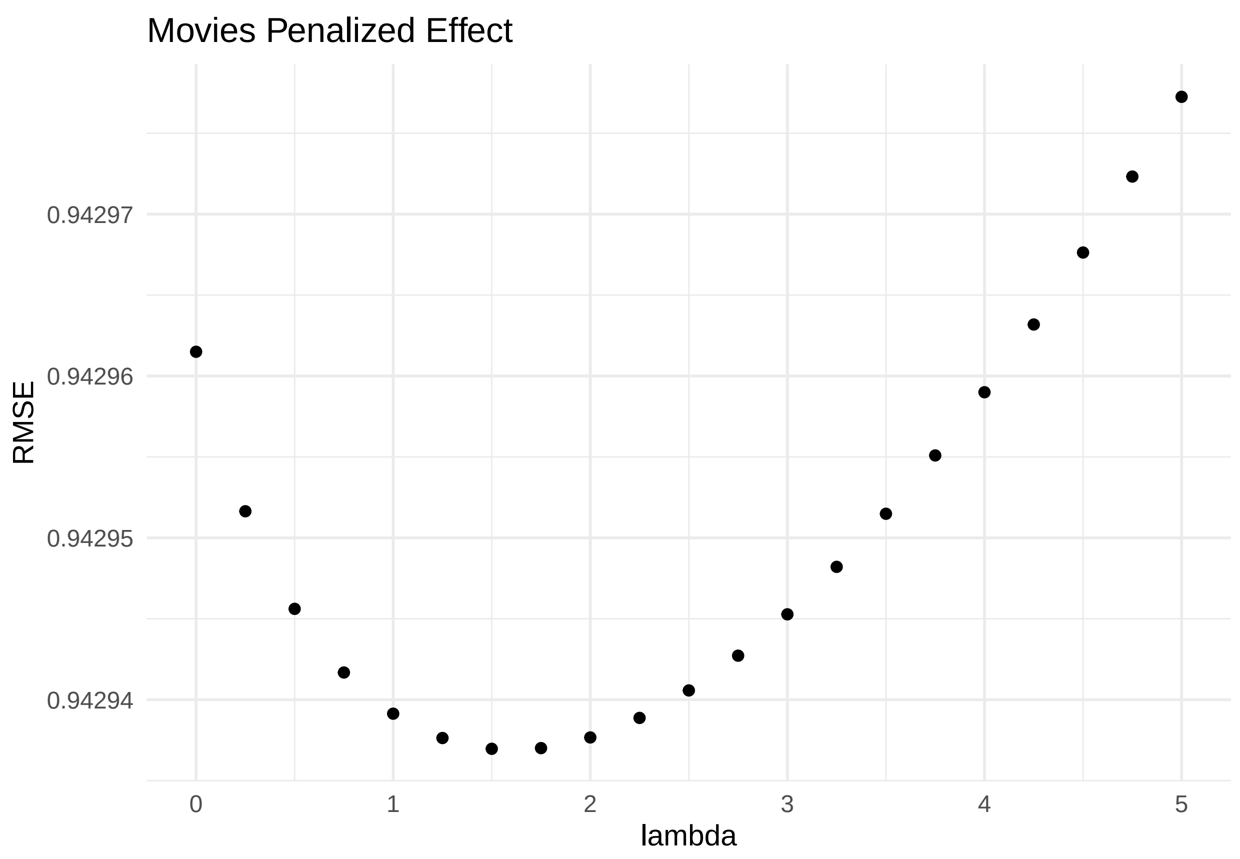
where $\hat{b}_m(\lambda)$ is

$$\hat{b}_m(\lambda) = \frac{1}{N_m + \lambda} \sum_{m=1}^{N_m} (Y_{u,m} - \mu)$$

and N_m is the number of ratings made for movie m.

Based on edx_train dataset, first I derive the tuning parameter λ that minimises the RMSE. I assume the λ is a value between 0 and 5.

```r
movie_lambdas <- seq(0,5,0.25)

movie_sums <- edx_train %>%
  group_by(movieId) %>%
  summarise(movie_rating_sum=sum(rating-mu), movie_rating_count=n())

movie_rmses <- sapply(movie_lambdas, function(lambda){
  predicted_movie_rating <- edx_test %>%
    left_join(movie_sums, by='movieId') %>%
    mutate(movie_effect = movie_rating_sum/(movie_rating_count+lambda)) %>%
    mutate(predicted = movie_effect+mu) %>%
    pull(predicted)
  return(RMSE(predicted_movie_rating,edx_test$rating))
})

tibble(lambda = movie_lambdas, RMSE = movie_rmses) %>%
  ggplot(aes(x = lambda, y = RMSE)) +
    geom_point() +
    ggtitle("Movies Penalized Effect") +
    theme_minimal()
```



Based on above chart, the best λ value for predicting regularised movie effect is 1.5 which give the RMSE value 0.94294.

```
rmse_movie <- min(movie_rmses)
lambda_movie <- movie_lambdas[which.min(movie_rmses)]
result <- rbind(result, c(Method = "Regularised Movie",
                          RMSE = round(rmse_movie,5)))
result %>% knitr::kable(caption="RMSE Result")
```

Table 10: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005
Regularised Movie	0.94294

3.2.4.2.3 Using the Regularized User Movie Effect

As analyzed in Data Exploration section, there are some users rated less than 20 movies. Regularised User effect is also necessary . The new formula with regularised user effect, with b_u is the average ranking for user u:

$$Y_{u,m} = \mu + \hat{b}_u(\lambda) + \hat{b}_m(\lambda) + \epsilon_{u,m}$$

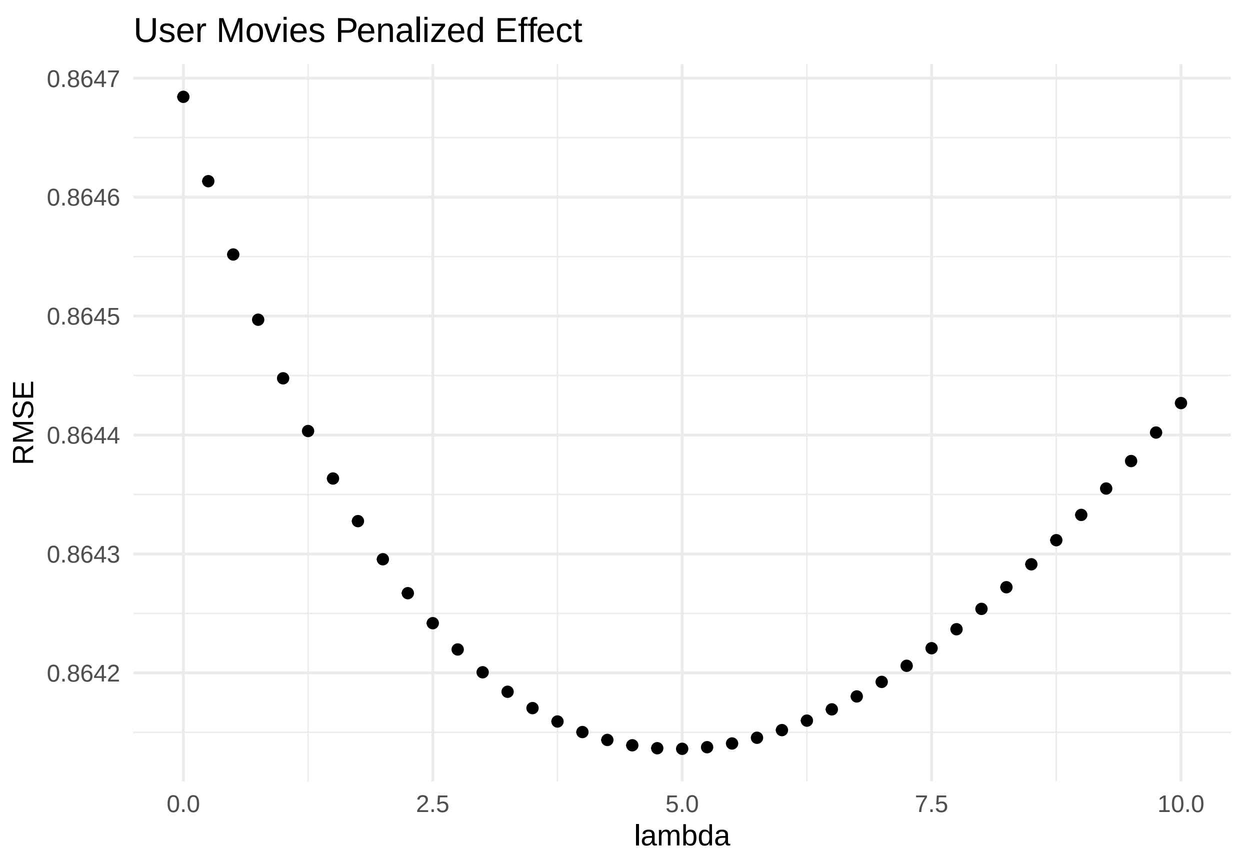
where $\hat{b}_u(\lambda)$ is

$$\hat{b}_u(\lambda) = \frac{1}{N_u + \lambda} \sum_{u=1}^{N_u} (Y_{u,m} - \mu)$$

and n_u is the number of ratings made for user u.

Based on edx_train dataset, the tuning parameter λ is simulated using following codes:

```r
user_movie_lambdas <- seq(0,10,0.25)

user_movie_rmses <- sapply(user_movie_lambdas, function(lambda){
  movie_sums <- edx_train %>%
    group_by(movieId) %>%
    summarise(movie_effect = sum(rating-mu) / (n()+lambda)) %>%
    select(movieId,movie_effect)

  user_sums <- edx_train %>%
    left_join(movie_sums, by="movieId") %>%
    group_by(userId) %>%
    summarise(user_effect = sum(rating-mu-movie_effect) / (n()+lambda))

  predicted_rating <- edx_test %>%
    left_join(user_sums, by='userId') %>%
    left_join(movie_sums, by="movieId") %>%
    mutate(predicted = mu + user_effect + movie_effect) %>%
    pull(predicted)
  return(RMSE(predicted_rating,edx_test$rating))
})

tibble(lambda = user_movie_lambdas, RMSE = user_movie_rmses) %>%
  ggplot(aes(x = lambda, y = RMSE)) +
    geom_point() +
    ggtitle("User Movies Penalized Effect") +
    theme_minimal()
```



Based on above chart, 5 is the best λ which give RMSE value 0.86414.

```r
rmse_usermovie <- min(user_movie_rmses)
lambda_usermovie <- user_movie_lambdas[which.min(user_movie_rmses)]

result <- rbind(result, c(Method = "Regularised User Movie",
                          RMSE = round(rmse_usermovie,5)))
result %>% knitr::kable(caption="RMSE Result")
```

Table 11: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005
Regularised Movie	0.94294
Regularised User Movie	0.86414

3.2.4.2.4 Regularized using Other available variant

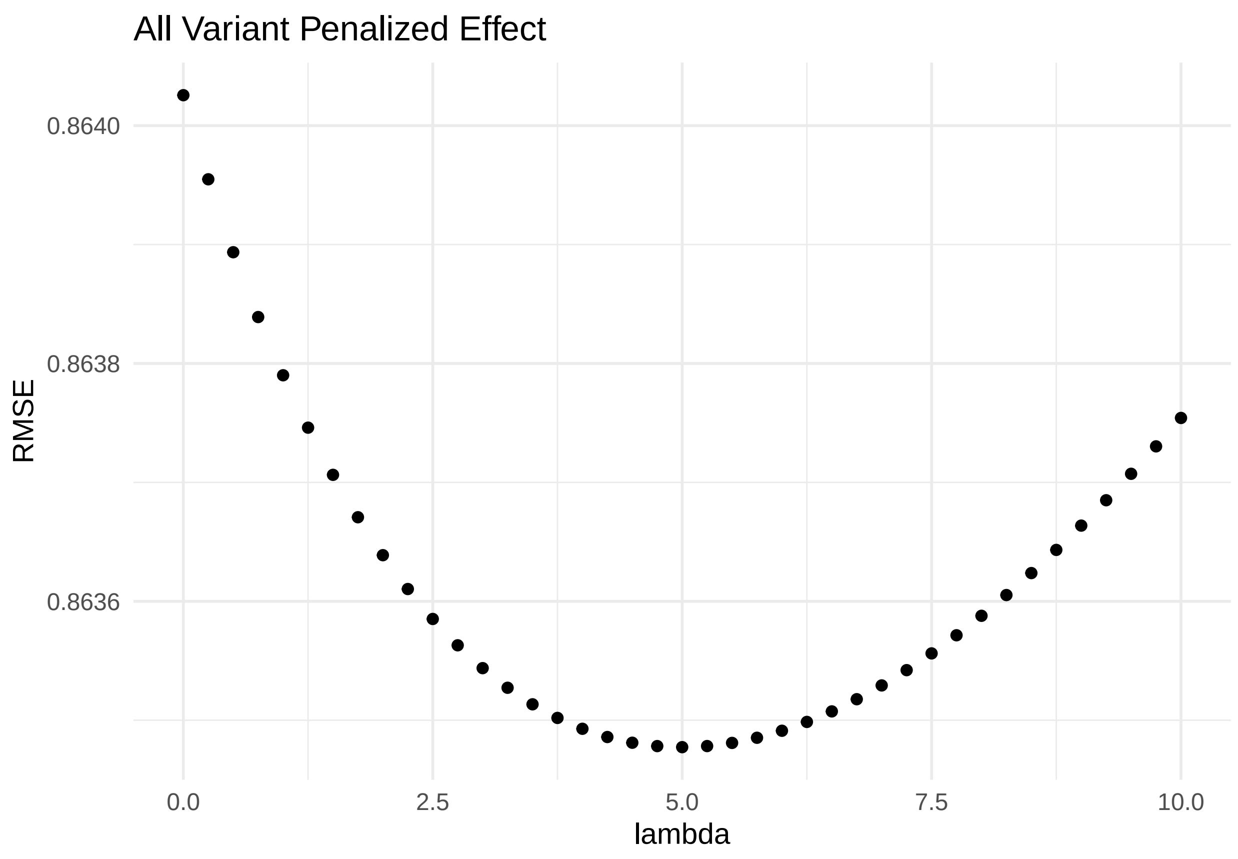
Based on step 3 results as above, and also Principal Components Analysis, both movies & users are significant variants that affect rating prediction. In order to further minimise RMSE, increase prediction accuracy, I regularised following variants into the main formula:

- Release Year,
- Rating Year, and

- Year Lapse between release and rating.

With the above 3 additional variants, I repeat the step above, looking for the best λ that minimise the RMSE.

```r
all_in_lambdas <- seq(0,10,0.25)

all_in_rmse <- sapply(all_in_lambdas, function(lambda){
  movie_sums <- edx_train %>%
    group_by(movieId) %>%
    summarise(movie_effect = sum(rating-mu) / (n()+lambda)) %>%
    select(movieId,movie_effect)

  user_sums <- edx_train %>%
    inner_join(movie_sums, by="movieId") %>%
    group_by(userId) %>%
    summarise(user_effect = sum(rating-mu-movie_effect) / (n()+lambda))

  yrrelease_sums <- edx_train %>%
    inner_join(user_sums, by="userId") %>%
    inner_join(movie_sums, by="movieId") %>%
    group_by(yr_release) %>%
    summarise(yrrelease_effect = sum(rating-mu-movie_effect-user_effect) / (n()+lambda))

  rellapse_sums <- edx_train %>%
    inner_join(user_sums, by="userId") %>%
    inner_join(movie_sums, by="movieId") %>%
    inner_join(yrrelease_sums, by="yr_release") %>%
    group_by(yr_lapse) %>%
    summarise(rellapse_effect = sum(rating-mu-movie_effect-user_effect-
                                  yrrelease_effect) / (n()+lambda))

  yrrate_sums <- edx_train %>%
    inner_join(user_sums, by="userId") %>%
    inner_join(movie_sums, by="movieId") %>%
    inner_join(yrrelease_sums, by="yr_release") %>%
    inner_join(rellapse_sums, by="yr_lapse") %>%
    group_by(yr_rating) %>%
    summarise(yrrate_effect = sum(rating-mu-movie_effect-user_effect-
                                yrrelease_effect-rellapse_effect) / (n()+lambda))

  predicted_rating <- edx_test %>%
    inner_join(user_sums, by='userId') %>%
    inner_join(movie_sums, by="movieId") %>%
    inner_join(yrrelease_sums, by="yr_release") %>%
    inner_join(rellapse_sums, by="yr_lapse") %>%
    inner_join(yrrate_sums, by="yr_rating") %>%
    mutate(predicted = mu + user_effect + movie_effect + yrrelease_effect +
             rellapse_effect + yrrate_effect) %>%
    pull(predicted)
  return(RMSE(predicted_rating,edx_test$rating))
})
tibble(lambda = all_in_lambdas, RMSE = all_in_rmse) %>%
  ggplot(aes(x = lambda, y = RMSE)) +
    geom_point() +
    ggtitle("All Variant Penalized Effect") +
    theme_minimal()
```



According to above chart, the best λ value for predicting by regularised all variant effect is 5 which give the RMSE value 0.86348.

```
rmse_all_in <- min(all_in_rmse)
lambda_all_in <- all_in_lambdas[which.min(all_in_rmse)]

result <- rbind(result, c(Method = "Regularised All Variants", RMSE = round(rmse_all_in,5)))
result %>% knitr::kable(caption="RMSE Result")
```

Table 12: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005
Regularised Movie	0.94294
Regularised User Movie	0.86414
Regularised All Variants	0.86348

3.2.4 Step5 Interpreting Data

After taking into consideration of regularised user, movie, release year, rating year and rating lapse effects, the final RMSE 0.86414 is below the project goal of 0.8649.

The final prediction script for the above modeling with final lambda value 5 is as below:

```r
final_lambda <- 5.0
movie_sums <- edx_train %>%
  group_by(movieId) %>%
  summarise(movie_effect = sum(rating-mu) / (n()+final_lambda)) %>%
  select(movieId,movie_effect)

user_sums <- edx_train %>%
  inner_join(movie_sums, by="movieId") %>%
  group_by(userId) %>%
  summarise(user_effect = sum(rating-mu-movie_effect) / (n()+final_lambda))

yrrelease_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  group_by(yr_release) %>%
  summarise(yrrelease_effect = sum(rating-mu-movie_effect-user_effect) / (n()+final_lambda))

rellapse_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  group_by(yr_lapse) %>%
  summarise(rellapse_effect = sum(rating-mu-movie_effect-user_effect-yrrelease_effect) /
              (n()+final_lambda))

yrrate_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  inner_join(rellapse_sums, by="yr_lapse") %>%
  group_by(yr_rating) %>%
  summarise(yrrate_effect = sum(rating-mu-movie_effect-user_effect-
                                  yrrelease_effect-rellapse_effect) / (n()+final_lambda))

predicted_rating <- edx_test %>%
  inner_join(user_sums, by='userId') %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  inner_join(rellapse_sums, by="yr_lapse") %>%
  inner_join(yrrate_sums, by="yr_rating") %>%
  mutate(predicted = mu + user_effect + movie_effect + yrrelease_effect +
           rellapse_effect + yrrate_effect) %>%
  pull(predicted)
```

4 Modeling Result

Table 13: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005
Regularised Movie	0.94294
Regularised User Movie	0.86414
Regularised All Variants	0.86348

As we can see from the result table, regularization and matrix factorization achieved the target RMSE. In this section, we will use the validation dataset to validate the prediction script. The target is to achieve reqired RMSE goal less than 0.865.

4.1 Prepare Dataset

Validation dateset need to be parsed to ensure it's consists the required columns, data types and data cardinality. 3 data mutation exercses are executed:

1. Prepare yr_release column for Movie Release year,
2. Prepare yr_rating column for Rating year, and
3. Prepare yr_lapse column for the year lapse between Rating year and Release year.

```
validation <- validation %>%
  mutate(title=str_trim(title)) %>%
  mutate(yr_release=as.numeric(substr(title,str_length(title)-4,str_length(title)-1)))

validation <- validation %>%
  mutate(yr_rating=year(as_datetime(timestamp,origin="1970-01-01")))

validation <- validation %>%
  mutate(yr_lapse=yr_rating-yr_release) %>%
  mutate(yr_lapse=ifelse(yr_lapse<=0,0,yr_lapse))
```

4.2 Execute Model

After dataset is prepared, I run the following modeling script that developed using training data set (edx_training and edx_test). The λ value 5.0 is selected, which is as simulated and confirmed in previous section.

```
final_lambda <- 5.0
mu <- mean(edx_train$rating)
movie_sums <- edx_train %>%
  group_by(movieId) %>%
  summarise(movie_effect = sum(rating-mu) / (n()+final_lambda)) %>%
  select(movieId,movie_effect)

user_sums <- edx_train %>%
  inner_join(movie_sums, by="movieId") %>%
  group_by(userId) %>%
  summarise(user_effect = sum(rating-mu-movie_effect) / (n()+final_lambda))
```

```
yrrelease_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  group_by(yr_release) %>%
  summarise(yrrelease_effect = sum(rating-mu-movie_effect-user_effect) /
              (n()+final_lambda))

rellapse_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  group_by(yr_lapse) %>%
  summarise(rellapse_effect = sum(rating-mu-movie_effect-user_effect-
                                    yrrelease_effect) / (n()+final_lambda))

yrrate_sums <- edx_train %>%
  inner_join(user_sums, by="userId") %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  inner_join(rellapse_sums, by="yr_lapse") %>%
  group_by(yr_rating) %>%
  summarise(yrrate_effect = sum(rating-mu-movie_effect-user_effect-
                                  yrrelease_effect-rellapse_effect) / (n()+final_lambda))

predicted_rating <- validation %>%
  inner_join(user_sums, by='userId') %>%
  inner_join(movie_sums, by="movieId") %>%
  inner_join(yrrelease_sums, by="yr_release") %>%
  inner_join(rellapse_sums, by="yr_lapse") %>%
  inner_join(yrrate_sums, by="yr_rating") %>%
  mutate(predicted = mu + user_effect + movie_effect + yrrelease_effect +
           rellapse_effect + yrrate_effect) %>%
  pull(predicted)

validation_rmse <- RMSE(predicted_rating,validation$rating)
```

4.3 Validation Result

The script takes approximate 18 minutes to execute. With λ value at 5.0, the validation dataset RMSE is calculated as 0.86454, which satisfied the project goal.

```
result <- rbind(result, c(Method = "Validation", RMSE = round(validation_rmse,5)))
result %>% knitr::kable(caption="RMSE Result")
```

Table 14: RMSE Result

Method	RMSE
Project Target	0.8649
Mean	1.06005
Regularised Movie	0.94294
Regularised User Movie	0.86414

Method	RMSE
Regularised All Variants	0.86348
Validation	0.86454

5 Conclusion

5.1 Brief Summary

The objective of this report is to apply the knowledge acquired throughout the Harvardx Data Science courses, to analyze the given 10 million MovieLens dataset (downloaded from GroupLens Research) using Recommendation System approach, and to suggest the best model to predict the rating of a user on a movie, at a rate from 0.5 to 5.0. To ensure the probability of accurate analysis, I selected and followed the OSEMN Framework standardized process. The process consists of 5 steps, (1) Obtain Data, (2) Scrub Data, (3) Explore Data, (4) Model Data and (5) Intepreting Data.

Linear Modeling with rata Regularization technique is used for the analysis and prediction of data, with final resulted reported in Root Mean Squared Error (RMSE). The MovieLens columns userid, movieid, release year and rating year are regularised and used in my analysis in predicting rating.

After extensive modeling, the validation result using the regularised regressive model is 0.86454.

5.2 Limitation

There are few limitation encounters in research, analyze, training and choosing models for the project modeling:

5.2.1 Hardware limitation

The training and modeling are perform on a laptop that has limited memory and processing powers. I have attempted to evaluate RandomForest, k-NN, and others models, but all of them failed due to hardware limitation. This limitation can be overcome with a better hardware or machine.

5.2.2 Algorithm limitation

According to MovieLens website, the are fews modeling that manage to achieve RMSE less than 0.7. Most of these modeling are performed on specialize program. Apart of these algorithm, there's also other R package that may resulted better accuracy, e.g. Matrix Factorization using recosystem package.

This limitation can be overcome with more details exploration and research.

5.2.3 Exploration limitation

The modeling is performed on most of the available data columns, except genres. I hypotheses user-genres have some collation that affects how a user perceived and rated a movie. The modeling of genres is not performed due to limitations above.

In addition, I encountered some strange data e.g. year review is before year release. Understanding of how this data exist will further helps in tuning the modeling.

5.3 Future Works

More research, analysis, and training of models are needed to get the best result in predicting user-movie rating from MovieLens dataset. Following modeling techniques warrants further studies to improve the overall accuracy,

1. Collaborative filter with recommenderlab package,
2. k-NN collaborative filtering algorithm,
3. Collaborative filtering & matrix factorization with rrecsys package.