Movielens Final Project

Rawin Chanpitak

26/11/2019

1.Introduction

This report is one of the compulsory assignment for **Capstones Projects** from **HarvardX Online Course**. The report is about building recommendation system that predicting the rating for specific user to watch more movie. The movie that is predicted to be high rating for the independent user will be suggested to that user.

The goal for this project is to build the movie recommendation system that have the minimum error by comparing lost function. In this case, it is Root Mean Square Error **(RMSE)**. The models in this assignment are developed from the evidences that appear in the data. In this particular assignment the goal is to yeild RMSE less than 0.86490. Therefore, there is no development after the model hit the barrier.

This report show that by considering all of the available variables, RMSE can be less than 0.86490 with normal regression model by adopting regularization technique. Furthermore, k-fold cross validation technique was used to analyse RMSE and choosing regularize factor.

2.Methodology

This section include content about evidences from the data set and analysis for evaluating the best method and model to use in the data set. The methods and analysing techniques that were adopted are explained here.

2.1.Pre-Training

In the Pre-training section, The data were explored before using any model for for selecting model. To avoid impossible models and too long calculation time, this section should be view seriously. Noted that there were no cleaning or generalising techniques here since the data is already tidy and they are all factor except timestamp variable

2.1.1.Pre-training analysis

The movielens data have 10 million data point. This data set is devine into two part which are *edx* (training set) and *validation* set which are 90% and 10% of total data respectively. The dimension of both set is showed below

```
dim(edx)
```

```
## [1] 9000055       7
```

```
dim(validation)
```

```
## [1] 999999      7
```

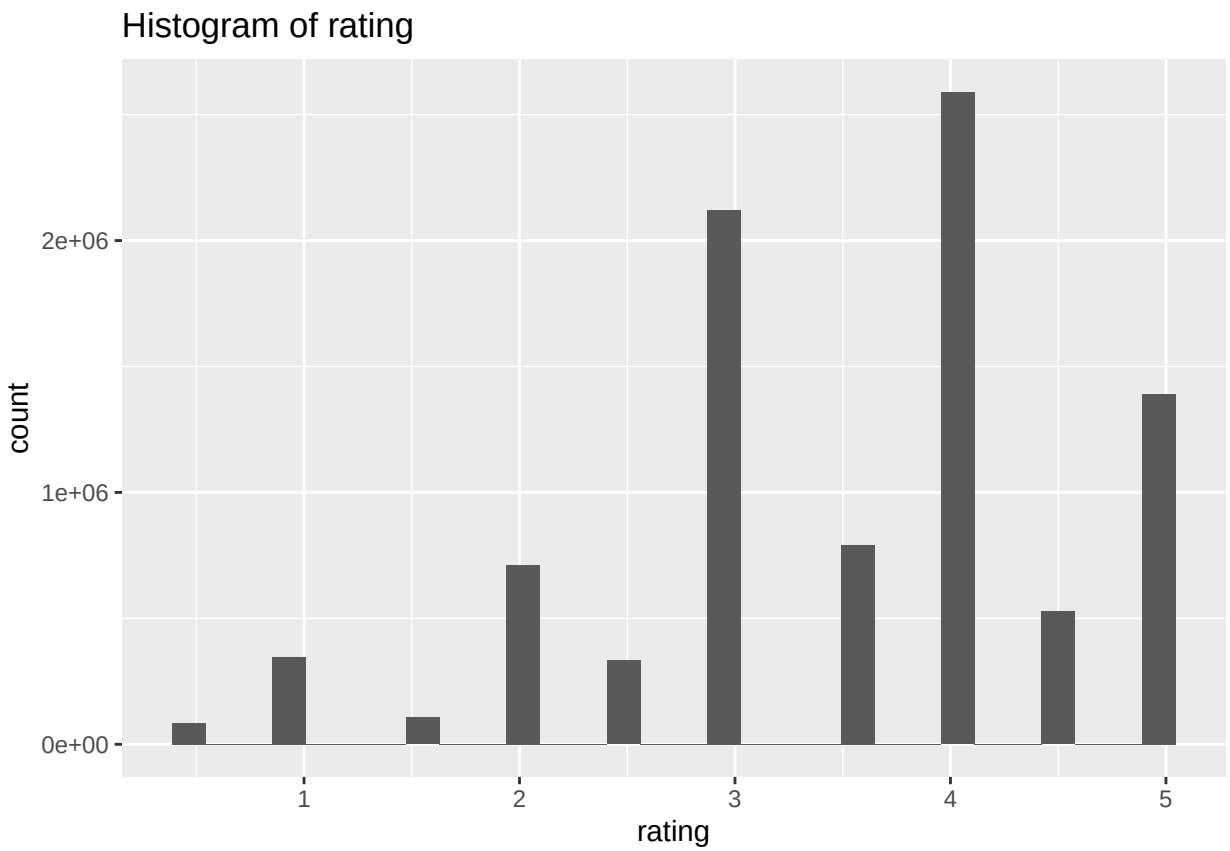
The variable that was predicted here is rating variable which range from 0 to 5. the distance between rating is 0.5. there are 4 variables that were analysed to predict rating namely; userId, movieId, genres, and timestamp. In this report, all of the independent variables were used except title. This is because it only indicate the name of the movie.

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                          genres     date_m
## 1                Comedy|Romance 1996-08-01
## 2          Action|Crime|Thriller 1996-08-01
## 3  Action|Drama|Sci-Fi|Thriller 1996-08-01
## 4       Action|Adventure|Sci-Fi 1996-08-01
## 5 Action|Adventure|Drama|Sci-Fi 1996-08-01
## 6       Children|Comedy|Fantasy 1996-08-01
```

```
edx %>% ggplot(aes(rating)) + geom_histogram() + ggtitle("Histogram of rating")
```

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



describetive statistic is provided below to describe the distribution of rating in edx training set

```
edx %>% summarise( avg = mean(rating), sd = sd(rating))
```

```
##        avg       sd
## 1 3.512465 1.060331
```

Furthermore, the others viraible should be looked to get the idea of how this data should be treated. The table below shows how many catagories in those variables. Noted that the timestamp was converted to months for grouping purpose

```
data.frame( variable_name = c('movieId','title','userId','genres','timestamp_month'),
            n_catagory = c(nlevels(as.factor(edx$movieId)),
                           nlevels(as.factor(edx$title)),
                           nlevels(as.factor(edx$userId)),
                           nlevels(as.factor(edx$genres)),
                           nlevels(as.factor(edx$date_m))))
```

```
##     variable_name n_catagory
## 1         movieId      10677
## 2           title      10676
## 3          userId      69878
## 4          genres        797
## 5 timestamp_month        157
```

From these, it can be seen that there are several problem to consern here. Firstly, The data points are too large to calculate. This mean that a commercial computer might take days to calculate even only one train set for complicate model. Secondly, the matric that will store calculation will consume very large storage. The matrix that will store the calculation might take up to 100 Gb. Hence, the machine learning algorithm that consume calculation time should not be considered here. Even the simplest model such as linear model will crash R if it is used to estimate those parameters.

2.2.Training

This section provides the reason and the method for chosing the model from the avialable evidences that were obtained in previous section

The algorythm in this assignment seem not to have many option to choose. Due to the computational technology and storage, the best option is the simplest which is linear regression. However, even linear regression, in this case, cannot run on R because the massive size of the data. Therefore, the analytical solution was used for calculating the prediction.

The linear regression model can be seen as while Y = dependent variable, X = independent variables, and ϵ = error

$$Y = X_1 + X_2 + ... + X_n + \epsilon_i$$

2.2.1.Training techniques

The techniques that are described here include linear regression, k-fold cross-validation, and regularization.

linear regression .

This is one of the basic technique in data-science and also in economics to quantify causality. The method can be done by minimising the square error from the model. The proof of the formular will not included here since it is tedious to write for every model that is stated here.

k-fold cross-validation .

The cross-validation technique is the technique to evaluate the model after training by dividing the in-sample set into two set called training and validation set which is the general term. However, in this assignment the name *validation set* was been given to out-of-sample set. Hence, the validation set will be call *test set* to ease the confusion. The method is to partition data differently and then train various different datasets to obtain the mean of the accuracy or lost function from test set. Because the outputs from the model come from different partitioning, the mean of the various output will be more trustworthy than a single output from single variation of partition.

k-fold cross-validation is the technique that is developed from general cross-validation. The k-fold cross-validation partition data equally for each set. Let's say that there are 5-fold cross-validation and there are 10 data point. The first training set will contain c(1,2,3,4,5,6,7,8) and the test set will contain c(9,10). In the next fold, the training set is c(1,2,3,4,5,6,9,10) and the test set will contain c(7,8). This goes on until all of the data is used in test sets. Splitting data set this way seems to be maximise the benefit from all avialable data. However, there is a catch which is calculation time. Even though this seems not to be a concern for small data set. However, for the large data set this will consume too much calculation time.

Regularization .

Regularization is a practical technique for the model with the aim of generating better performance. This is because some data point should be penalise to reduce the effect from variables to the whole model. In here, the regularization was used for penalising the factor that have low datapoint. The example of this is show below. It can be seen that some movies were rated only once

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

```
## # A tibble: 6 x 2
##   title                                              n
##   <chr>                                          <int>
## 1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)      1
## 2 100 Feet (2008)                                    1
## 3 4 (2005)                                           1
## 4 Accused (Anklaget) (2005)                          1
## 5 Ace of Hearts (2008)                               1
## 6 Ace of Hearts, The (1921)                          1
```

2.2.2.Training methodology

The models were built from the criterai. The RMSE has to be less than 0.86490. Therefore, the simplest model was the first model which is the calculation of mean of rating.

$$\widehat{Y} = \mu$$

After this, the RMSE that contain from the model was compared with the goal RMSE. If the model is not good enough, another variable will be add to increase the performance of the model. The independent

variables can be added util no variables left. For example, the movieId variable can be added to find the average effect on rating from each movie. Thus, the model will be

$$\widehat{Y}_i = \mu \ + \ b_i$$

More detail about each model will be state in model Specification section

following this, if the RMSE still not below the RMSE tarket after using all variables, the regularization will be another technique to help performance as it can be seen that some of the variable average effect is calculate from one or two data point which is reasonable to use average to be predicter. Therefore, the RMSE should be reduce if those effect indeed decreasing the performance.

All of the models adapted k-fold cross-validation to generate better RMSE on the grounds that they were compared to each other. The k-fold cross-validation was also used to select regularization factor before the model was applyed to validation set.

2.3.Post-Training

After training, the RMSE from each model was compared with each other and then find the best performer to apply with the validation dataset. The RMSE from each models were report in a table. There are several technique that can be used here such as bootstrap resampling for constructing confident interval of RMSE. However, it was leaved out from this analysis.

3.Model Specification and Analysis

In this section, each model is explained and given general idea of the formular. Moreover, the model was developed based on the evidenced that inherite in the data set

3.1.Model Explanation and Analysis

each model was developed with the aim of improving the performance of RMSE. Noted that every models in here was adopted k-fold cross validation technique

3.1.1.The single constant model

The easiest way to construct prediction is calculating the mean. This evidence is show in the derivation of linear regression when using only constant as a variable to predict the outcome. Therefore, the first model was single constant model which can be describe as

$$Y_n = c + \epsilon_n$$

where c is constant and ϵ_i is each individual datapoint error.

From this the outcome of the prediction is

$$\widehat{Y}_n = \mu$$

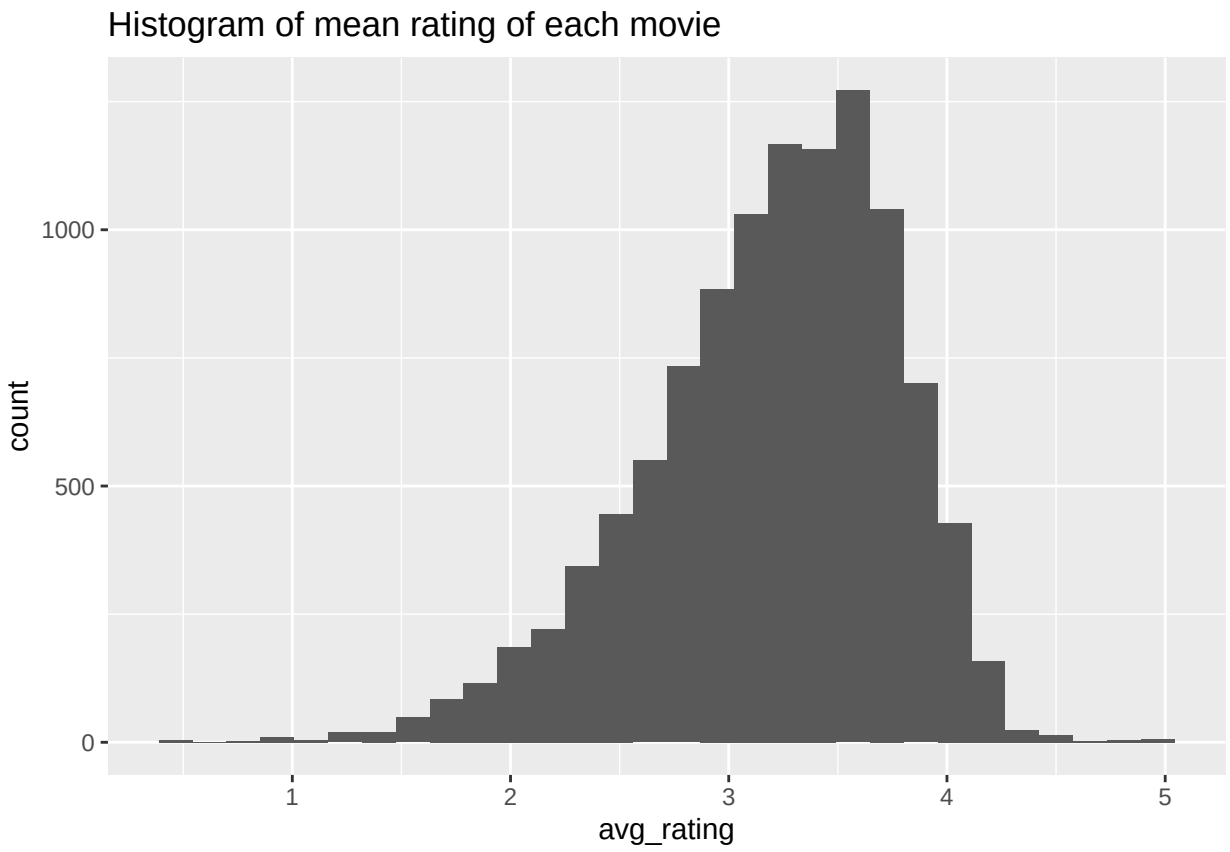
while μ is denoted as the mean of the vector $\mathbf{Y}$

3.1.2.The movie model

Each movie should have different rating since there are good movies and bad movies. Thinking about the different between *Shawshank Redemtion* and *Disastor Movie* might give an initial idea. The distribution of average rating for each movie can be seen below

```
edx %>% group_by(movieId) %>%
  summarise( avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) + geom_histogram() + ggtitle("Histogram of mean rating of each movie")
```

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



Because the movie number or title name cannot be quantified, each movie was had there own predictor. The easiest way to calculate this is to find the average effect on rating of each movie. The analytical solution can be found when we consider regression of using only dummies variables to regress $(Y - \mu)$

The linear model has a form as

$$Y_n = c + X_1 + \epsilon_n$$

where $X1$ denote the vector of movieId variable.

From this the outcome of the prediction is

$$\widehat{Y}_n = \mu + b_i$$

where i is the index of each movie and b_i is the average effect on rating from each movie

The analytical formular for b_i is

$$b_i = \sum_{j=1}^{m} (Y - \mu)/n_j$$

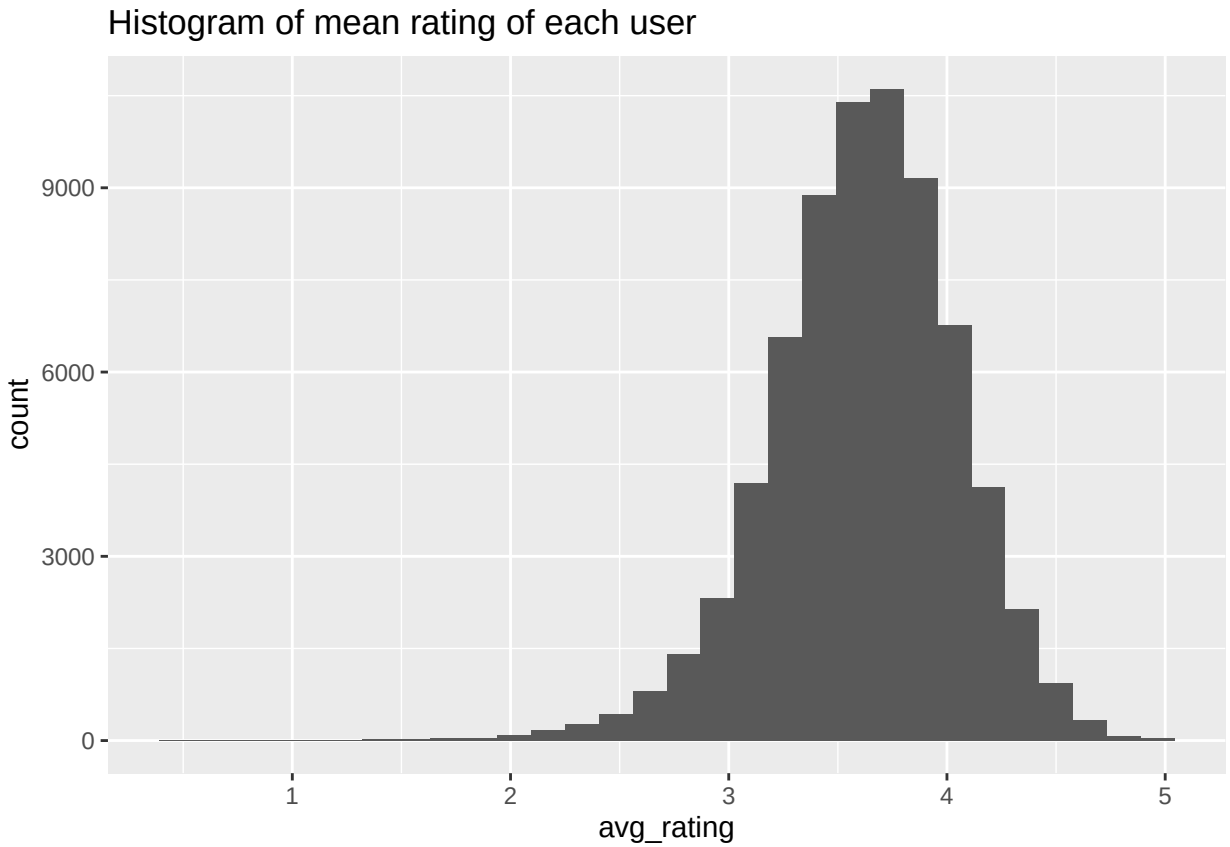
where m is the total number of data point in each movie.

3.1.3.The movie and user model

Each individual user have different preference and standard. As a result, the means of rating from individual user are different

```
edx %>% group_by(userId) %>%
  summarise( avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) + geom_histogram() + ggtitle("Histogram of mean rating of each user")
```

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



As the same reason as movie variables, the predicted value that will be used here is means of effect on rating from individual user. Therefore, the analytical of this variable is similar with the previous one except that instead of considering $(Y - \mu)$ now it is $(Y - \mu - b_i)$

The linear model has a form as

$$Y_n = c + X_1 + X_2 + \epsilon_n$$

where $X2$ denote the vector of userId variable.

From this the outcome of the prediction is

$$\widehat{Y}_n = \mu + b_i + b_u$$

where u is the index of each user and b_u is the average effect on rating from each user

The analytical formular for b_u is

$$b_u = \sum_{j=1}^{m} (Y - \mu - b_i)/n_j$$

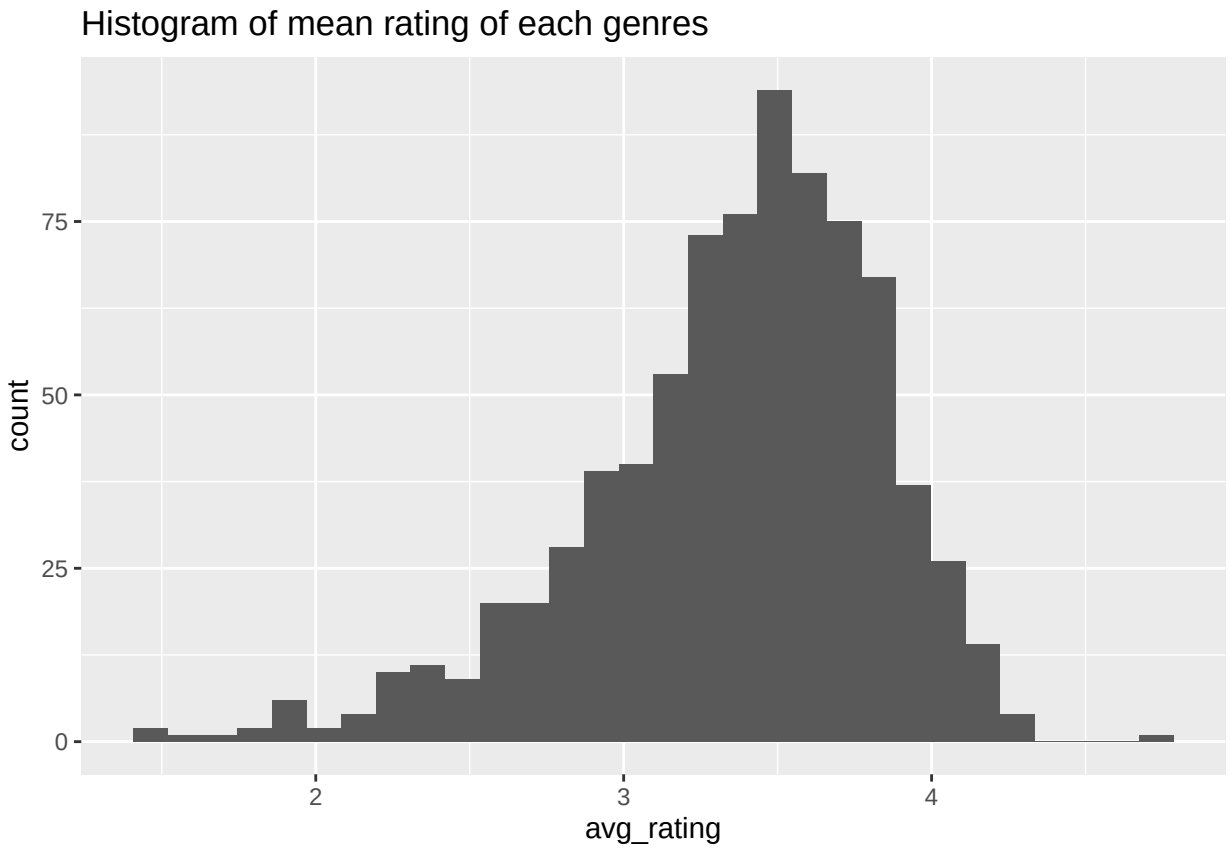
where m is the total number of data point in each user.

3.1.4.The movie, user and genres model

Again, each genres seem to have different rating scores. Hence, the means of rating from individual genres were ploted to explore any evidence of the genres effect.

```
edx %>% group_by(genres) %>%
  summarise( avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) + geom_histogram()+ ggtitle("Histogram of mean rating of each genres")
```

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



From sceptical analysis, it can be seen that there are differences between each genres. The analytical solution of mean of effect on rating from each genres can be find similary with other previous two.

The linear model has a form as

$$Y_n = c + X_1 + X_2 + X_3 + \epsilon_n$$

where $X3$ denote as the vector of genres variable.

From this the outcome of the prediction is

$$\widehat{Y}_n = \mu + b_i + b_u + b_g$$

where g is the index of each genre and b_g is the average effect on rating from each user

The analytical formular for b_g is

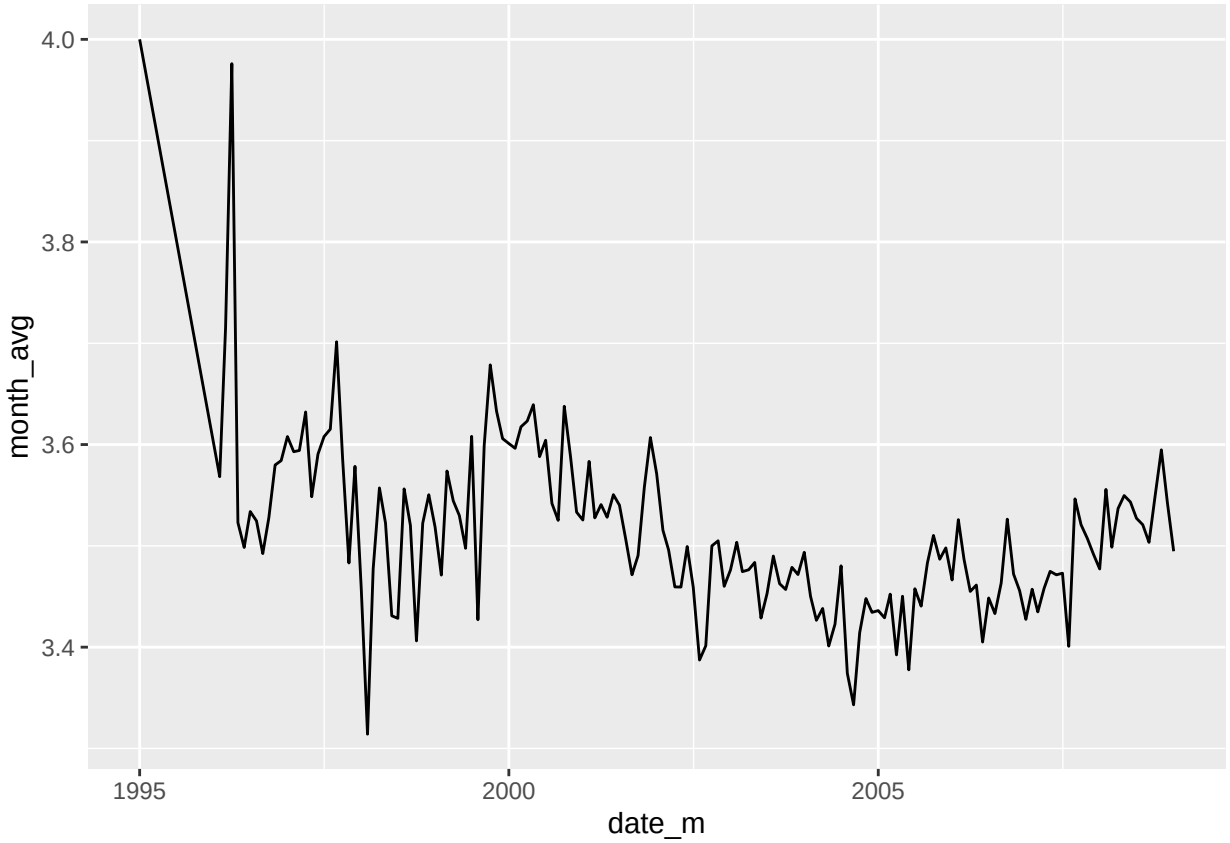
$$b_g = \sum_{j=1}^{m} (Y - \mu - b_i - b_u)/n_j$$

where m is the total number of data point in each genres.

3.1.5.All four variables model

From some point of views, there might be sometime that the time have a specific trend on the rating. Therefore, considering the time to be one of the predicting variable appear to make sence. The time series plot of average rating is shown below. There is a test for this which is unit root analysis. However, this technique was not used here.

```
edx %>% mutate( date_m = round_date(as_datetime(timestamp),"month")) %>%
  group_by(date_m) %>% summarise( month_avg = mean(rating))%>%
  ggplot(aes(date_m,month_avg)) + geom_line()
```



The data shown that there is a variation when the rating is given in different time period. Thus, it was included in this regression

The time that was given in the timestamp variable has a very small time interval. To group as a levels in prediction, the timestamp was rounded to month.

The linear model has a form as

$$Y_n = c + X_1 + X_2 + X_3 + X_4 + \epsilon_n$$

where $X4$ denote as the vector of timestamp variable.

From this, the outcome of the prediction is

$$\widehat{Y}_n = \mu + b_i + b_u + b_g + b_d$$

where d is the index of each month and b_d is the average effect on rating from each month

The analytical formular for b_d is

$$b_d = \sum_{j=1}^{m} (Y - \mu - b_i - b_u - b_g)/n_j$$

where m is the total number of data point in each genres.

3.1.6 Regularized model

The regularize technique was adopted to penalise meaningless predictors such that it is reduce the effect from that predictor and rely on other variables that are more meaningful. In this case, there are some movie that were rated only once or twice. The same goes with rating but with higher number. Therefore, all the evidence will be provided here

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

from userId

```
## # 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
```

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

from movie title

```
## # A tibble: 6 x 2
##   title                                                 n
##   <chr>                                             <int>
## 1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)         1
## 2 100 Feet (2008)                                       1
## 3 4 (2005)                                              1
## 4 Accused (Anklaget) (2005)                             1
## 5 Ace of Hearts (2008)                                  1
## 6 Ace of Hearts, The (1921)                             1
```

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

from genres

```
## # A tibble: 6 x 2
##   genres                                      n
```

```
##   <chr>                                        <int>
## 1 Action|Animation|Comedy|Horror                   2
## 2 Action|War|Western                               2
## 3 Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi       2
## 4 Adventure|Mystery                                2
## 5 Crime|Drama|Horror|Sci-Fi                        2
## 6 Documentary|Romance                              2
```

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

from date month

```
## # A tibble: 6 x 2
##   date_m                  n
##   <dttm>              <int>
## 1 1995-01-01 00:00:00     2
## 2 1996-02-01 00:00:00   183
## 3 1997-09-01 00:00:00   700
## 4 1996-03-01 00:00:00  1019
## 5 1998-05-01 00:00:00  4776
## 6 1999-07-01 00:00:00  5172
```

from this, it can be seen that the variable movieId and genres need to be reguralise due to the pour average from not enough sampled datapoint. The variables that will be used in regularization are userId, movieId, and genres variables.

After adopted the regularization technique, The estimators of each formular change their form into

$$b_i = \sum_{j=1}^{m}(Y - \mu)/(n_j + \lambda)$$

$$b_u = \sum_{j=1}^{m}(Y - \mu - b_i)/(n_j + \lambda)$$

$$b_g = \sum_{j=1}^{m}(Y - \mu - b_i - b_u)/(n_j + \lambda)$$

where λ is regularization factor

The choice of selecting regularization factor (lambda) can be found by trial and error. After trying different lambda. The lambda that yeild the lowest RMSE was chosed and used to estimate the validation set.

3.2.Training Formular As Coding Function

In this section, the coding functions are stated here to show the formular of each predictors which were used and analysed in the result section. Because these equations were used many time in models, It is more convenian and easier for reader to understand the code.

- 1.index for regularization partition function

```
ind_cv <- function(k){
  ind <- round(seq(1,nrow(edx),nrow(edx)/k),digits = 0)
  return(ind)
}
```

- 2.average of rating function

```
mu_cal <- function(rating){
  mean(rating)
}
```

- 3.average effect to rating of each movie

```
movie_reg <- function(ts,mu_l,lamb){
  ts %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_l)/(n()+lamb))
}
```

- 4.average effect to rating of each user

```
user_reg <- function(ts,mu_l,mov_avgs,lamb){
  ts %>%
    left_join(mov_avgs, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu_l - b_i)/(n()+lamb))
}
```

- 5.average effect to rating of each genre

```
genres_reg <- function(ts,mu_l,mov_avgs,use_avgs,lamb){
  ts %>%
    left_join(mov_avgs, by='movieId') %>%
    left_join(use_avgs, by= 'userId') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu_l - b_i - b_u)/(n()+lamb))
}
```

- 6.average effect to rating of each month

```
month_reg <- function(ts,mu_l,mov_avgs,use_avgs,gen_avgs,lamb){
  ts %>%
    left_join(mov_avgs, by='movieId') %>%
    left_join(use_avgs, by='userId') %>%
    left_join(gen_avgs, by='genres') %>%
    group_by(date_m) %>%
    summarize(b_d  = sum(rating - mu_l - b_i - b_u - b_g)/(n()+lamb))
}
```

- 7.RMSE calculation

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

The function for finding predictors contain lambda in there function. If the regularization is not needed it can be set to 0

4.Result

The result of each model will be shown in this section. However, it is not the result from validation set. **The validation set (or out-of-sample set) will be used with the best performer after compared with each others**. Therefore, every decision that was made was from in-sample set.

4.1 Model Performance

In here, every model was compaired in train set and test set which were splitted from edx set. Every model were adopted the k-fold cross validation technique, even the last model that contain traial and error for lambda. 5 fold cross validation was used to generate RMSE. The best performer was the model that have the lowest RMSE.

The code below is for preparing the dataset for splitting and setting the number of fold. The input is the number of fold.

```
ind_cv <- function(k){
  ind <- round(seq(1,nrow(edx),nrow(edx)/k),digits = 0)
  return(ind)
}
set.seed(2019, sample.kind="Rounding")
```

```
## Warning in set.seed(2019, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
random_index <- sample(seq(1:nrow(edx)),nrow(edx), replace = FALSE)
k <- 5
ind <- ind_cv(k)
```

The RMSE function is

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(Y - \widehat{Y})^2}{n}}$$

```
RMSE_model_1 <- sapply(ind,function(n){
  train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
  test_set <- temp_set %>%
    semi_join(train_set, by = "movieId") %>%
```

```
    semi_join(train_set, by = "userId") %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  mu <- mu_cal(train_set$rating)
  predicted_ratings <- test_set %>%
    mutate(pred = mu) %>%
    .$pred
  RMSE(predicted_ratings, test_set$rating)
})
mean(RMSE_model_1)
```

The single constant model (model 1)

```
## [1] 1.06033
```

```
RMSE_result <- data_frame(method = "Plain average", RMSE = mean(RMSE_model_1))
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
RMSE_result
```

```
## # A tibble: 1 x 2
##   method         RMSE
##   <chr>         <dbl>
## 1 Plain average  1.06
```

```
rm(RMSE_model_1)
```

The number show that it is not a good predictor as it is still far from the target which is 0.86490

```
RMSE_model_2 <- sapply(ind,function(n){
  train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
  test_set <- temp_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId") %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  mu <- mu_cal(train_set$rating)
  movie_avgs <- movie_reg(train_set,mu, lamb = 0)
  predicted_ratings <- test_set %>%
    left_join(movie_avgs, by='movieId') %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  RMSE(predicted_ratings, test_set$rating)
})
mean(RMSE_model_2)
```

The movie model (model 2)

```
## [1] 0.9437067
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Plus Movie Avg", RMSE = mean(RMSE_model_2)))
RMSE_result
```

```
## # A tibble: 2 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Plain average  1.06
## 2 Plus Movie Avg 0.944
```

```
rm(RMSE_model_2)
```

```
RMSE_model_3 <- sapply(ind,function(n){
  train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
  test_set <- temp_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId") %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  mu <- mu_cal(train_set$rating)
  movie_avgs <- movie_reg(train_set,mu, lamb = 0)
  user_avgs <- user_reg(train_set,mu,movie_avgs, lamb = 0)
  predicted_ratings <- test_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  RMSE(predicted_ratings, test_set$rating)
})
RMSE_model_3
```

The movie and user model (model 3)

```
## [1] 0.8661721 0.8663967 0.8660558 0.8662241 0.8660648
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Plus User Avg", RMSE = mean(RMSE_model_3)))
RMSE_result
```

```
## # A tibble: 3 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Plain average  1.06
## 2 Plus Movie Avg 0.944
## 3 Plus User Avg  0.866
```

```
rm(RMSE_model_3)
```

```
RMSE_model_4 <- sapply(ind,function(n){
  train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
  test_set <- temp_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId") %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  mu <- mu_cal(train_set$rating)
  movie_avgs <- movie_reg(train_set,mu, lamb = 0)
  user_avgs <- user_reg(train_set,mu,movie_avgs, lamb = 0)
  genres_avgs <- genres_reg(train_set,mu,movie_avgs,user_avgs, lamb = 0)
  predicted_ratings <- test_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genres_avgs, by='genres') %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred
  RMSE(predicted_ratings, test_set$rating)
})
RMSE_model_4
```

The movie, user and genres model (model 4)

```
## [1] 0.8658132 0.8660949 0.8657088 0.8658566 0.8657052
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Plus Genres Avg", RMSE = mean(RMSE_model_4)))
RMSE_result
```

```
## # A tibble: 4 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Plain average   1.06
## 2 Plus Movie Avg  0.944
## 3 Plus User Avg   0.866
## 4 Plus Genres Avg 0.866
```

```
rm(RMSE_model_4)
```

```
RMSE_model_5 <- sapply(ind,function(n){
  train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
```

```
  test_set <- temp_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId") %>%
    mutate( date_m = round_date(as_datetime(timestamp),"month"))
  mu <- mu_cal(train_set$rating)
  movie_avgs <- movie_reg(train_set,mu, lamb = 0)
  user_avgs <- user_reg(train_set,mu,movie_avgs, lamb = 0)
  genres_avgs <- genres_reg(train_set,mu,movie_avgs,user_avgs, lamb = 0)
  month_avgs <- month_reg(train_set,mu,movie_avgs,user_avgs,genres_avgs, lamb = 0)
  predicted_ratings <- test_set %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genres_avgs, by='genres') %>%
    left_join( month_avgs, by = 'date_m') %>%
    mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
    .$pred
  RMSE(predicted_ratings, test_set$rating)
})
RMSE_model_5
```

The All four variables model (model 5)

```
## [1] 0.8657717 0.8660400 0.8656660 0.8658341 0.8656722
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Plus Month Avg", RMSE = mean(RMSE_model_5)))
RMSE_result
```

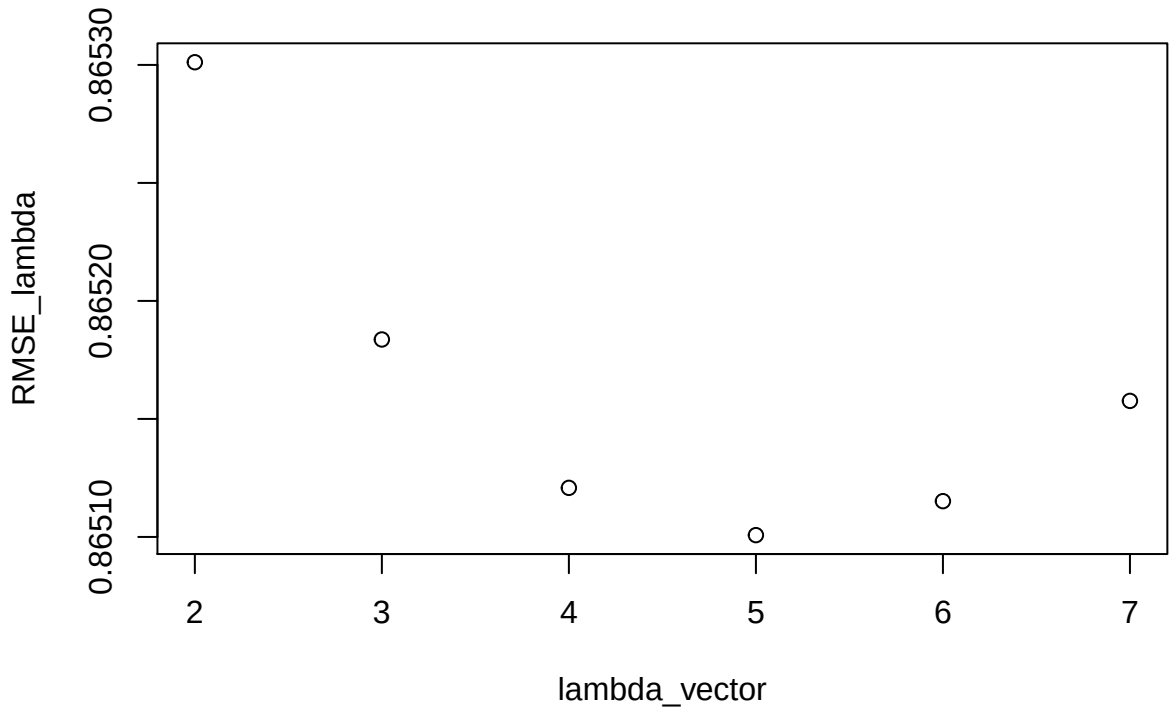
```
## # A tibble: 5 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Plain average   1.06
## 2 Plus Movie Avg  0.944
## 3 Plus User Avg   0.866
## 4 Plus Genres Avg 0.866
## 5 Plus Month Avg  0.866
```

```
rm(RMSE_model_5)
```

Regularized model The resularized model was taking more time than the others. This was because the model have to run with various regularise factor to find the best factor that yeilded lowest RMSE.

```
lambda_vector <- seq(2,7)
RMSE_lambda <- c()
for (lambda in lambda_vector) {
  RMSE_cv <- c()
  for (n in ind) {
    train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
      mutate( date_m = round_date(as_datetime(timestamp),"month"))
    temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
    test_set <- temp_set %>%
```

```
      semi_join(train_set, by = "movieId") %>%
      semi_join(train_set, by = "userId") %>%
      mutate( date_m = round_date(as_datetime(timestamp),"month"))
    mu <- mu_cal(train_set$rating)
    movie_avgs <- movie_reg(train_set,mu,lambda)
    user_avgs <- user_reg(train_set,mu,movie_avgs,lambda)
    genres_avgs <- genres_reg(train_set,mu,movie_avgs,user_avgs,lambda)
    month_avgs <- month_reg(train_set,mu,movie_avgs,user_avgs,genres_avgs, lamb = 0)
    predicted_ratings <- test_set %>%
      left_join(movie_avgs, by='movieId') %>%
      left_join(user_avgs, by='userId') %>%
      left_join(genres_avgs, by='genres') %>%
      left_join(month_avgs, by='date_m') %>%
      mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
      .$pred
    RMSE_cv <- append(RMSE_cv,RMSE(predicted_ratings, test_set$rating))
  }
  RMSE_lambda <- append(RMSE_lambda,mean(RMSE_cv))
  rm(mu,user_avgs,movie_avgs,genres_avgs,month_avgs,
     train_set,test_set,temp_set,predicted_ratings, RMSE_cv)
}
plot(lambda_vector,RMSE_lambda)
```




```
best_lambda <- lambda_vector[which.min(RMSE_lambda)]
min(RMSE_lambda)
```

```
## [1] 0.8651008
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Regularization", RMSE = min(RMSE_lambda)))
message("Best lambda: ", best_lambda)
```

```
## Best lambda: 5
```

```
RMSE_result
```

```
## # A tibble: 6 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Plain average   1.06
## 2 Plus Movie Avg  0.944
## 3 Plus User Avg   0.866
## 4 Plus Genres Avg 0.866
## 5 Plus Month Avg  0.866
## 6 Regularization  0.865
```

After the regularisation, the RMSE was equal to 0.86510. This can be lower by increasing the fold. Up until now, the trained data was 80% of the in-sample set due to the fact that the 5-fold cross validation was used. Therefore, increasing the percentage for training set to be the same with the data that was splitted. The original data was splitted into 90%. As a consequence, the 10-fold cross-validation was used. The edx set was splitted for 90%.

```
k <- 10
ind <- ind_cv(k)
lambda_vector <- best_lambda
RMSE_lambda <- c()
for (lambda in lambda_vector) {
  RMSE_cv <- c()
  for (n in ind) {
    train_set <- edx[-(random_index[n:(n+nrow(edx)/k-2)]),] %>%
      mutate( date_m = round_date(as_datetime(timestamp),"month"))
    temp_set <- edx[random_index[(n:(n+nrow(edx)/k-2))],]
    test_set <- temp_set %>%
      semi_join(train_set, by = "movieId") %>%
      semi_join(train_set, by = "userId") %>%
      mutate( date_m = round_date(as_datetime(timestamp),"month"))
    mu <- mu_cal(train_set$rating)
    movie_avgs <- movie_reg(train_set,mu,lambda)
    user_avgs <- user_reg(train_set,mu,movie_avgs,lambda)
    genres_avgs <- genres_reg(train_set,mu,movie_avgs,user_avgs,lambda)
    month_avgs <- month_reg(train_set,mu,movie_avgs,user_avgs,genres_avgs, lamb = 0)
    predicted_ratings <- test_set %>%
      left_join(movie_avgs, by='movieId') %>%
      left_join(user_avgs, by='userId') %>%
      left_join(genres_avgs, by='genres') %>%
```

```
      left_join(month_avgs, by='date_m') %>%
      mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
      .$pred
    RMSE_cv <- append(RMSE_cv,RMSE(predicted_ratings, test_set$rating))
  }
  RMSE_lambda <- append(RMSE_lambda,mean(RMSE_cv))
  rm(mu,user_avgs,movie_avgs,genres_avgs,month_avgs,
     train_set,test_set,temp_set,predicted_ratings, RMSE_cv)
}
RMSE_lambda
```

Regularized model with 10-fold cross-validation

```
## [1] 0.8646113
```

```
RMSE_result <- bind_rows(RMSE_result,
                         data_frame(method = "Regularization with best lambda", RMSE = max(RMSE_lambda)))
RMSE_result
```

```
## # A tibble: 7 x 2
##   method                           RMSE
##   <chr>                           <dbl>
## 1 Plain average                   1.06
## 2 Plus Movie Avg                  0.944
## 3 Plus User Avg                   0.866
## 4 Plus Genres Avg                 0.866
## 5 Plus Month Avg                  0.866
## 6 Regularization                  0.865
## 7 Regularization with best lambda 0.865
```

From the last table, We can see that all of the past model is not good enough to generate RMSE less than 0.86490 except the regularise model. Hence, this model will be used in the next section.

4.2 Model Performance

The best model that was used is regularised model with lambda = 5. The model trained edx set and will predict validation set. **This is the only step that validation set was used**

```
train_set <- edx %>%
  mutate( date_m = round_date(as_datetime(timestamp),"month"))
test_set <- validation %>%
  mutate( date_m = round_date(as_datetime(timestamp),"month"))
mu <- mu_cal(train_set$rating)
movie_avgs <- movie_reg(train_set,mu,best_lambda)
user_avgs <- user_reg(train_set,mu,movie_avgs,best_lambda)
genres_avgs <- genres_reg(train_set,mu,movie_avgs,user_avgs,best_lambda)
month_avgs <- month_reg(train_set,mu,movie_avgs,user_avgs,genres_avgs,lamb = 0)
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
```

```
  left_join(month_avgs, by='date_m') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_d) %>%
  .$pred
final_RMSE <- RMSE(predicted_ratings, test_set$rating)
final_RMSE
```

```
## [1] 0.8643759
```

Now, the RMSE is indeed less than 0.86490

5.Conclusion

The objective of this report is to minimise the RMSE to be less than 0.86490. The model that was used here is develop from the evidences from the data itself by visualisation. There are several technique that were used apart from linear regression which are k-fold cross-validation and regularization. The best model is the regularized model. the model have RMSE = 0.86438 which is less than 0.86490. Thus, the goal is reached.

There are several limitation. Firstly, the datapoint is too large. As a consequence, other method of prediction seems to be impossible for commercial computer to perform prediction. Secondly, the statistical inferance is not conveyed here. Lastly, there might be some variable that could be added such as the time that the movie is release. This is due to the fact that the different between released time and timestamp can effect the rating. The example of this is that the movie that adopted CGI technique in 30 years ago might not seem enjoyable for some user.

There can be some improvements for this report. First and formost, the data size should be smaller. If the smaller data size is used, the other technique could be able to perform. Secondly, the statistical inferance can be adopted here as well. This is for making sure that it is indeed less than 0.86490 which is not just by luck. Finally, the other technique might help the model perform better such as matrix factorization technique