edx-project-report

yemi

10/29/2020

REPORT OF EDX PROJECT

##1. INTRODUCTIN

Machine learning is building of algorithms to make predictions using data. One or more features or predictors are processed to predict an outcome from a series of observations. The outcome can be divided into categorical and continuous. A categorical outcome is referred to as classification, while a continuous outcome is prediction. In this project, the outcome is continuous. The data set is series of observations of rating of various movies of one or multiple genres by users (viewers). The dataset is divided into edx with 9000055 rows of observations and validation set of 1000000 rows of observations.

The followings are definition of outcome and predictors in the dataset.

- Rating – the outcome. It range between 0.5 as the lowest to 5 as the highest.
- userId – a predictor. It's the unique identification for user who rated movies
- movieId –a predictor. It is the unique identification for a movie rated
- genres – a predictor. Genre is one or multiple classification of a movie
- timestamp – a predictor. The date and time the rating of a movie was given
- title – it is the name given to a movie. It is not unique, hence not considered as a predictor.

Notation

For this project, the outcome and predictors are denoted as follows

rating – Y

userId – u

movieId –i

genres – g

timestamp – d

The relation of the variables in data set can be represented with these notations as

Y(u,i) = i + u + g(u,i) + d(u,i)

Executive summary

The goal of this project is to make suggestions of movies for future users. The objective is to predict ratings of movies by users.

The method is to study how users rate movies, and make predictions. One or more predictors will be train to make rating predictions within the edx set. The best combinations (model) with the lowest root mean square error (RMSE) will be used to make rating predictions (suggestions) in validation set (future). The lower the RMSE on the validation shows how good the predictions are.

We have 9000055 ratings of 10677 movies rated by 69878 users in the edx data. This shows that some movies are more rated than others and there are some more active users who have rate more movies than others.

This project uses the RECOMMENDATION SYSTEM MODEL. This is due to the large data set and the small capacity of my computer system.

##2. METHODS ANALYSIS

The RECOMMENDATION SYSTEM MODEL is used. This model makes estimation of average of one or more predictors(effects) from the true mean of outcome and make predictions using the summation of the averages in the validation or test set. Five (5) models using the recommendation system were considered. The best with the lowest RMSE was then use at the final model to make predictions in validation set.

The metric – RMSE, is the root mean square error of the predicted rating and actual rating.

Pre-processing of the data

The following libraries are needed.

```
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 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     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-project.org")
```

```
## 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
```

```
library(tidyverse)
library(caret)
library(data.table)
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:base':
##
##     date, intersect, setdiff, union
```

The given data set can be downloaded here:

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")


# if using R 4.0 or later:
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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

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

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

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

```
edx <- rbind(edx, removed)

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

Having downloaded the given data set, both the edx and validation sets were examined.

I created a column, named date in both sets to convert timestamp from seconds to week. This is smoothing function of the timestamp later refer to in the project.

```
edx <- mutate(edx, date = as_datetime(timestamp)) %>% mutate(date = round_date(date, unit = "week"))
validation <- mutate(validation, date = as_datetime(timestamp)) %>% mutate(date = round_date(date, unit
```

edx dataset analysis

1. The dimension of the edx is 9000055 rows by 7 columns having added date column from timestamp

```
dim(edx)
```

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

2. A peek of the set shows first few rows and columns

```
glimpse(edx)
```

```
## Rows: 9,000,055
## Columns: 7
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
## $ date      <dttm> 1996-08-04, 1996-08-04, 1996-08-04, 1996-08-04, 1996-08-...
```

3. The class of the attributes identifies the different class of the predictors

```r
sapply(edx, class)
```

```
## $userId
## [1] "integer"
##
## $movieId
## [1] "numeric"
##
## $rating
## [1] "numeric"
##
## $timestamp
## [1] "integer"
##
## $title
## [1] "character"
##
## $genres
## [1] "character"
##
## $date
## [1] "POSIXct" "POSIXt"
```

4. The numbers of element in each variable shows the distinctive number of each variable involved in the dataset e.g movieId which is 10677, is the total number of movies rated.

```r
sapply(edx, n_distinct)
```

```
##    userId   movieId    rating timestamp     title    genres      date
##     69878     10677        10   6519590     10676       797       671
```

5. The distribution of rating in percentage shows te frequency of each rating.

```r
percentage <- prop.table(table(edx$rating))*100
```

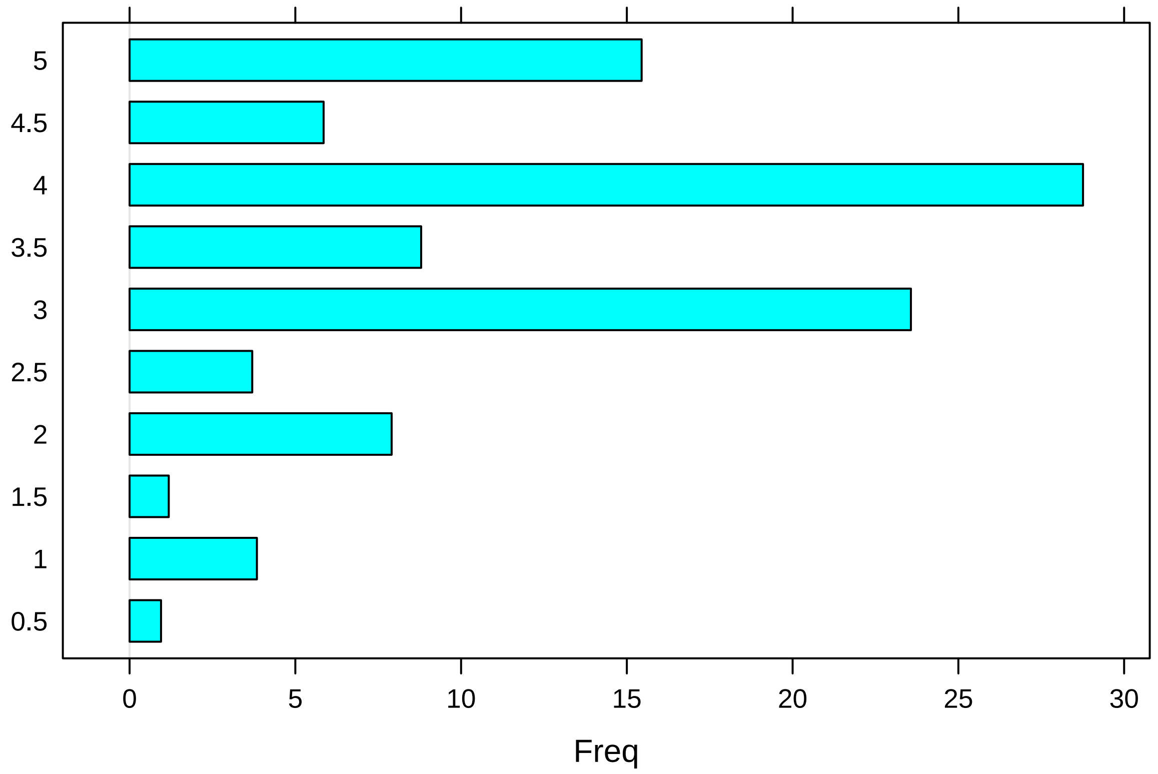
6. table of the frequency

```r
cbind(freq = table(edx$rating),  percentage = percentage)
```

```
##        freq percentage
## 0.5   85374  0.9485942
## 1    345679  3.8408543
## 1.5  106426  1.1825039
## 2    711422  7.9046406
## 2.5  333010  3.7000885
## 3   2121240 23.5691893
## 3.5  791624  8.7957685
## 4   2588430 28.7601576
## 4.5  526736  5.8525865
## 5   1390114 15.4456167
```

7. The barchart visualizes the distribution of rating in the edx set.

```r
barchart(percentage)
```



8. The summary shows the statistical details of all the variables.

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title               genres                  date
##  Length:9000055     Length:9000055     Min.   :1995-01-08 00:00:00
##  Class :character   Class :character   1st Qu.:2000-01-02 00:00:00
##  Mode  :character   Mode  :character   Median :2002-10-27 00:00:00
##                                        Mean   :2002-09-21 12:42:48
##                                        3rd Qu.:2005-09-18 00:00:00
##                                        Max.   :2009-01-04 00:00:00
```

9. The memory limit was set to 56000 to create more memory to accommodate the large data on my system.

```r
memory.limit(56000)
```

```
## [1] 56000
```

Prepare the edx set

Creation of Test and Train sets from the edx Using the createDataPartition function of caret package, the test and train sets were created from the edx data in ratio 2:8 respectively.

```r
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train <- edx[-test_index,]
test_set <- edx[test_index,]
```

To ensure that both the movieId and userId in the test set are in train set, semi_join function was used and the removed rows were row bind to train set. This is because a rating of a movie must be by a user.

```r
test <- test_set %>%
  semi_join(train, by = "movieId") %>%
  semi_join(train, by = "userId")
```

```r
removed <- anti_join(test_set, test)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres", "date")
```

```r
train <- rbind(train, removed)
```

Evaluate models

1. The baseline model – The model assume same rating for all movies by all users.

So, the recommendation system to predict rating - y(u,i) by finding the true mean - μ(mu) of all ratings

Y(u,i) = μ + (u,i)

Where:

Y(u,i) – represent the predicted rating

μ - mu, is the true mean of all the ratings

 (u,i) – represents independent errors sampled from same distribution centered at zero.

1. The baseline model

```r
mu <- mean(train$rating)
```

The RMSE for the baseline

```r
naive_rmse <- RMSE(test$rating, mu)
```

2. Model_1 – This model looks at the effect (bias) of movie (i) could add in making predictions. The mean of movie(i) denoted as b_i is estimated from the true mean.

Y(u,i) = µ + b(i)+ (u,i)

Where:

Y(u,i) – predicted_ratings, represent the predicted ratings

µ - mu, is the mean of all the ratings

b_i - is the average of Y(u,i) minus the overall mean for each movie(i).

 (u,i) – represents independent errors sampled from same distribution centered at zero.

The model_1

```r
mu <- mean(train$rating)

movie_avgs <- train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

The Prediction of model_1

```r
predicted_ratings <- test %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
```

The RMSE for model 1

```r
model_1_rmse <- RMSE(predicted_ratings, test$rating)
```

3. Model_2 – This model looks at the effect (bias) of user - b(u) in addition to model 1 on determining the rating. The model estimate the average rating of users (u) .

Y(u,i) = µ + b(i) + b(u) + (u,i)

Where:

Y(u,i) – predicted_ratings, represent the predicted ratings

µ - mu, is the mean of all the ratings

b_i - is the average of Y(u,i) minus the overall mean for each movie(i).

b_u – is the average of Y(u,i) from overall mean for each user(u).

 (u,i) – represents independent errors sampled from same distribution centered at zero.

The model_2

```
mu <- mean(train$rating)

user_avgs <- train %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

The prediction of model_2

```
predicted_ratings <- test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

The RMSE of model_2

```
model_2_rmse <- RMSE(predicted_ratings, test$rating)
```

4. Model_3 – This model looks at the effect (bias) of genre - b(g) in addition to model 2 on determining the rating. The model finds the average rating of each genre (g). It take mean of summation of each genre (k) of movies by a user.

Y(u,i) = µ + b(i) + b(u) + k,(k=1) x(u,i),k (k) + (u,i) with x(u,i),k = 1 if g(u,i) is genre k

this can be summarise as

Y(u,i) = µ + b(i) + b(u) + b_g(u,i) + (u,i)

Where:

Y(u,i) – predicted_ratings, represent the predicted ratings

µ - mu, is the mean of all the ratings

b_i - is the average of Y(u,i) minus the overall mean for each movie(i).

b_u – is the average of Y(u,i) from overall mean for each user(u).

b_g – is the mean from the summation of different genre averages of movies by user

 (u,i) – represents independent errors sampled from same distribution centered at zero.

The model_3

```
mu <- mean(train$rating)

genres_avgs <- train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by= 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

The predictions of model_3

```
predicted_ratings <- test %>%
  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) %>%
  pull(pred)
```

The RMSE of model_3

```
model_3_rmse <- RMSE(predicted_ratings, test$rating)
```

5. Model_4 – This model looks at the effect (bias) of date - b(d) in addition to model 3 on determining the rating. The model finds the average rating of date (d) after using round_date as the smoothing function (f) on timestamp as the reason for creating the date column.

Y(u,i) = µ + b(i) + b(u) + b_g(u,i) + f(d(u,i)) + (u,i) with f a smooth function of d(u,i)

Where:

Y(u,i) – predicted_ratings, represent the predicted ratings

µ - mu, is the mean of all the ratings

b_i - movie average from train set used in making predictions in test set.

b_u – is the user average from train set used in making predictions in test set.

b_g – is the mean from the summation of different genre averages.

b_d - is the average from the smoothing of the timestamp to weeks.

 (u,i) – represents independent errors sampled from same distribution centered at zero.

The model_4

```
mu <- mean(train$rating)

date_avgs <- train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by= 'userId') %>%
  left_join(genres_avgs, by= 'genres') %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u - b_g))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

The prediction of model_4

```
predicted_ratings <- test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by= 'genres') %>%
  left_join(date_avgs, by= 'date') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
  pull(pred)
```

The RMSE of model_4

```
model_4_rmse <- RMSE(predicted_ratings, test$rating)
```

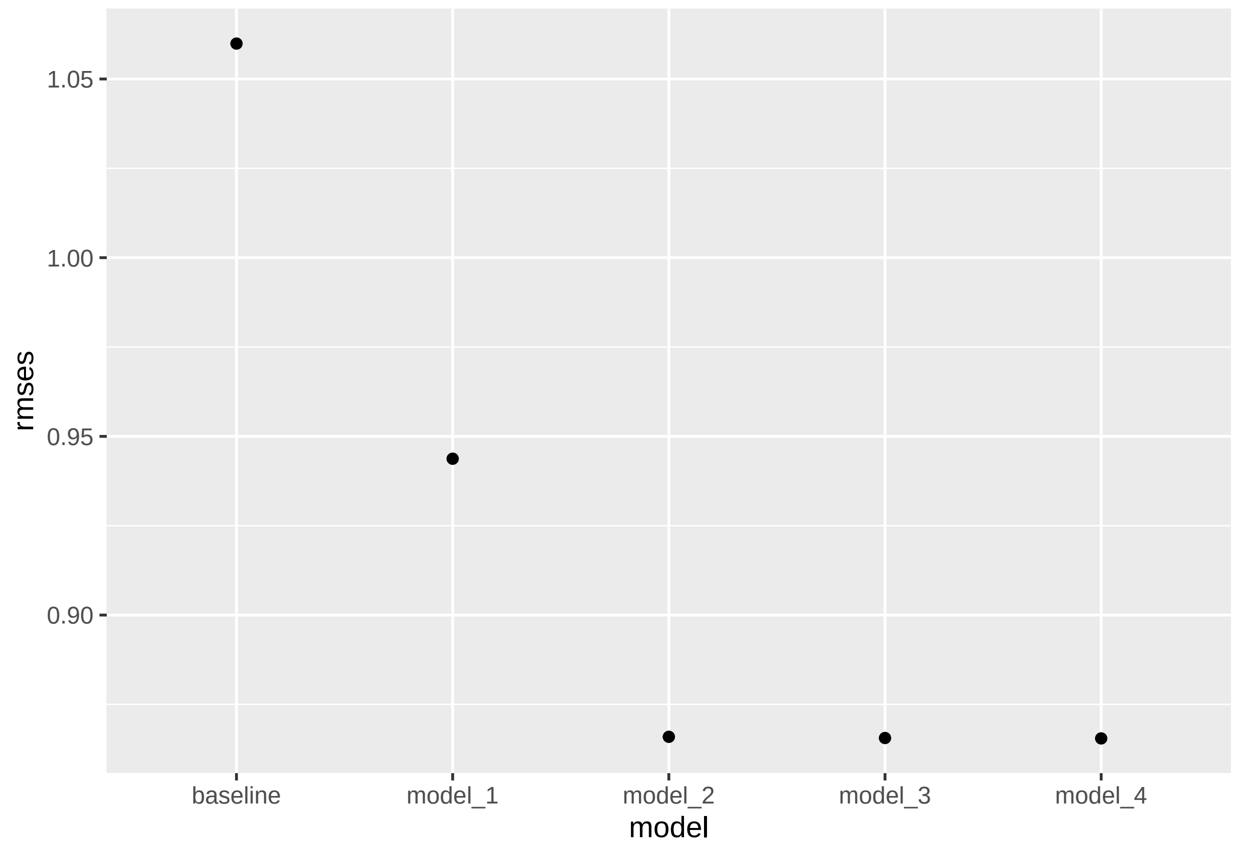
##3. RESULTS

```
model <- c("baseline", "model_1", "model_2", "model_3", "model_4")
rmses <- c(naive_rmse, model_1_rmse, model_2_rmse, model_3_rmse, model_4_rmse)
data.frame(model, rmses)
```

```
##      model     rmses
## 1 baseline 1.0599043
## 2  model_1 0.9437429
## 3  model_2 0.8659319
## 4  model_3 0.8655941
## 5  model_4 0.8654875
```

qplot

```
qplot(model,rmses)
```



## The best model

```r
model[which.min(rmses)]
```

```
## [1] "model_4"
```

The results of the models as presented above by a data frame and a plot. The model 4 has the best performance judging by the metric – RMSE. It has the lowest RMSE of 0.86548.

FINAL MODEL (EDX AND VALIDATION)

The model 4 being the best of the models was train with the entire edx set and makes the final predictions of rating from the validation set. The final RMSE of the final prediction of rating from the validation set is 0.8648392.

USING MODEL_4 ON EDX SET

```r
mu <- mean(edx$rating)

movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
genres_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by= 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
date_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by= 'userId') %>%
  left_join(genres_avgs, by= 'genres') %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u - b_g))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

THE PREDICTION ON VALIDATION SET

```r
final_predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by= 'genres') %>%
  left_join(date_avgs, by= 'date') %>%
  mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
  pull(pred)
```

THE FINAL RMSE ON VALIDATION SET

```r
RMSE(final_predicted_ratings, validation$rating)
```

```
## [1] 0.8648392
```

4. CONCLUSION

The project is about making predictions for movies rating by users. The predictions will enable making movies suggestion to users perfectly. This project through the recommendation system was able to achieve the set RMSE.

Limitations

1. The recommendation system used an approximation by computing true mean of rating (mu) and estimating every other effect(bias) - b mean from mu.

Other algorithms can do better in this respect.

2.The project did not constrain the sizes effect of large estimate that might come from small sample sizes. These effect known as outliner effects means that, in this project, large number of inactive (rate less than 100 movies) users might rate a certain movie more.

Regularization can be used to constrain each effect(bias) to its size.

3.This model could not provide the most important feature in determining the rating predictions.

Further use of matrix factorization for the numeric biases can evaluate the importance of the predictors.