Movie recommendation system (HarvardX PH125 Capstone project)

Dawit Tsadik Berhe

12/26/2020

Contents

1. Overview

The purpose of this project is to build a model or an algorithm that would predict user's rating to a movie and recommends the movie to the user accordingly. MovieLens data, which consists of actual users' rating to movies over several years, was provided to be used for the project. The data is split into two part – 90% of the data (labeled as edx) will be used to develop/train the algorithm, while 10% of the data (labeled as validation) will be used to test the performance of the algorithm developed. The performance of the algorithm developed will be evaluated using the Residual Mean Square Error (RMSE) of the predictions made by the algorithm when compared to the actual rating users made in the validation data set. RMSE of 0.86999 or less is acceptable for this project.
As it is discussed in detail below, since the edx data set provided is very large, I was not able to use some of the standard algorithms such as lm, glm, knn, rt, etc. and run them in my laptop. Instead I have to come up with a very light weight model that can run in my laptop in a reasonable time amount and produce predictions with RMSE of 0.86999 or less.
I will start with a simple model – predicting all users' rating to all movies to be the average pf the overall ratings, and then progressively improve the prediction by introducing the effect of the other variables (such as Movie, user, genres etc), until we achieve the acceptable RMSE value. The edx data set was further split into two parts, one part was used to train/develop the model while the second part was used as a cross-validating data to tune the model.
This report has four sections. In section 2, the data is analyzed to get a better understanding and insight of the various variables and their relationships and effect on the ratings. And based on the analysis, the

approach to be used to develop the model and the variables to consider is decided. In section 3, the different models will be tested, and their results will be compared and the best performing model will be selected as a final model. The final model is then tested with the validation data set to get the final predictions and RMSE. In section 4, overall observations and lesson learned will be discussed.

2. Analysis and Model development approach

In this section we will try to explore and analysis the data to get more insight and understanding of the data and the effect of the different variables. And at the end we decide what approach to follow to build the model.

The MovieLens data was imported and split into edx (90%) and validation (10%) data sets with the script provided.

2.1. General data analysis and data validation.

```
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title                  genres
##     <int>   <dbl>  <dbl>     <int> <chr>                  <chr>
##  1      1     122      5 838985046 Boomerang (1992)       Comedy|Romance
##  2      1     185      5 838983525 Net, The (1995)        Action|Crime|Thriller
##  3      1     292      5 838983421 Outbreak (1995)        Action|Drama|Sci-Fi|T~
##  4      1     316      5 838983392 Stargate (1994)        Action|Adventure|Sci-~
##  5      1     329      5 838983392 Star Trek: Generation~ Action|Adventure|Dram~
##  6      1     355      5 838984474 Flintstones, The (199~ Children|Comedy|Fanta~
##  7      1     356      5 838983653 Forrest Gump (1994)    Comedy|Drama|Romance|~
##  8      1     362      5 838984885 Jungle Book, The (199~ Adventure|Children|Ro~
##  9      1     364      5 838983707 Lion King, The (1994)  Adventure|Animation|C~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The~ Action|Comedy
## # ... with 9,000,045 more rows
```

```
edx %>% summarize(users = n_distinct(userId), movies = n_distinct(movieId))
```

```
##   users movies
## 1 69878  10677
```

```
colSums(is.na(edx))
```

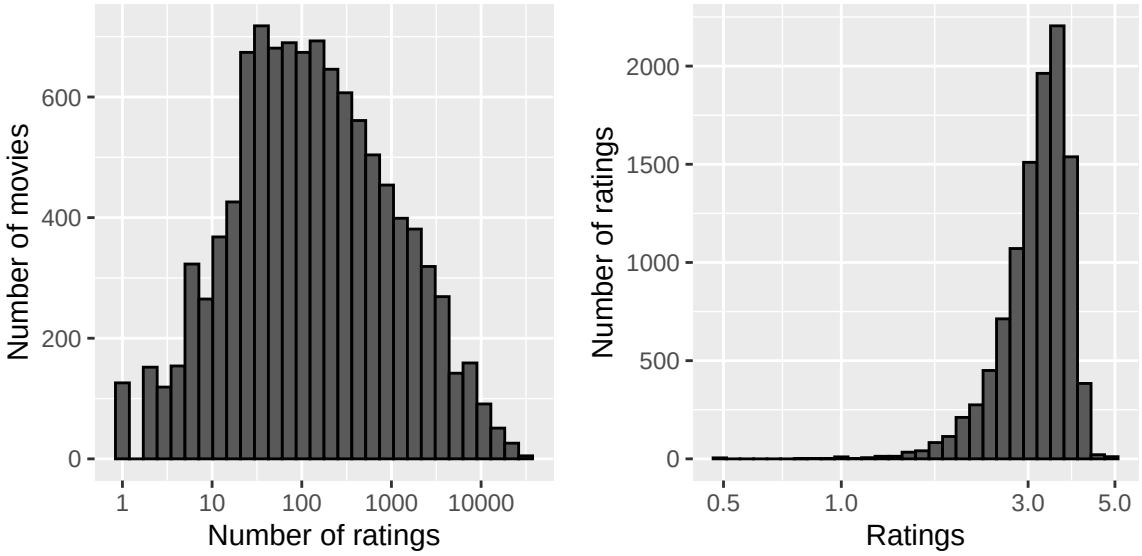
```
##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0
```

As it can be seen from the output of the above scripts, the edx data has 9,000,055 observations and 6 variables. 69,878 users rated one or more of the 10,677 movies. Each row represents one user's rating to a single movie. There are no n/a in any of the columns.

2.2. Analysis of the variables/predictors

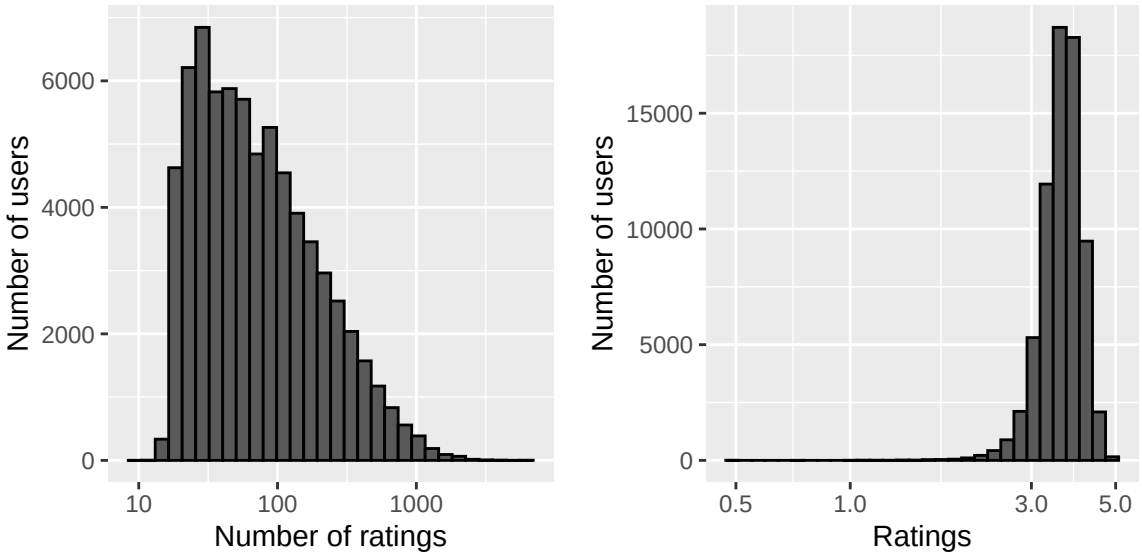
Here we will discuss the different variables/predictors that can be used as an input to the model.

movieId (Movie) effect



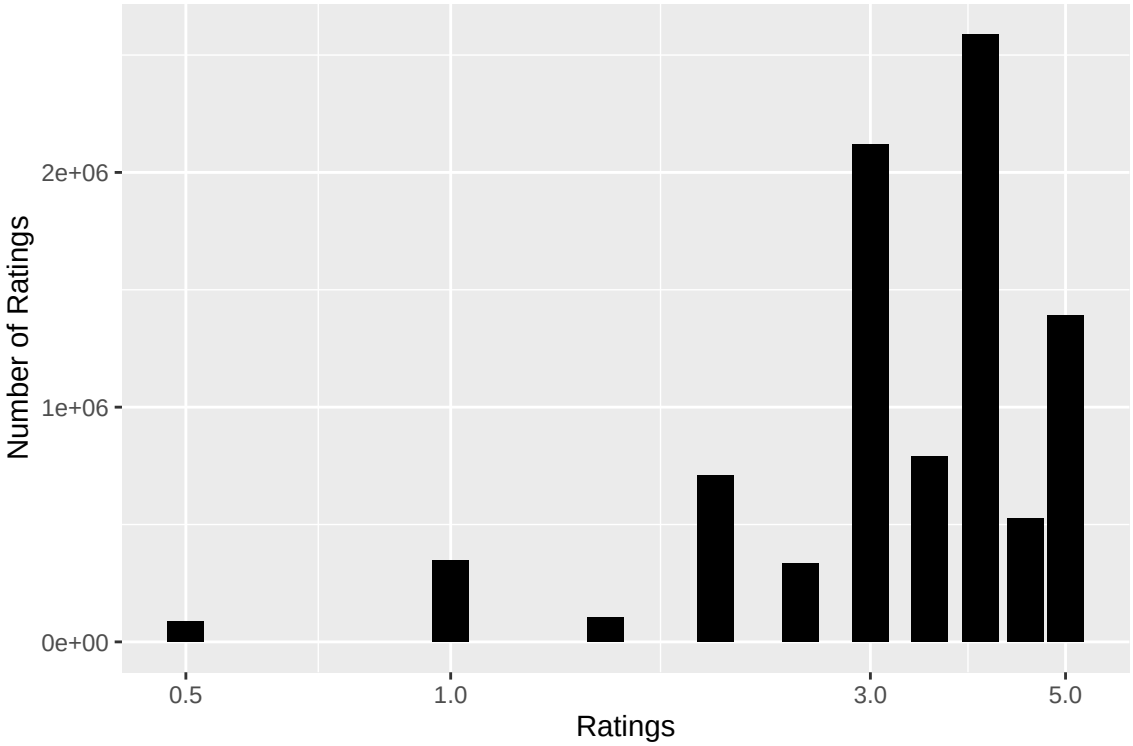
From the above two charts, we can observe that some movies have more rating than the others. And most of the movies with higher number of rating tend to have generally a higher rating (above the average (3.5124652) ratings).

userId (User) effect



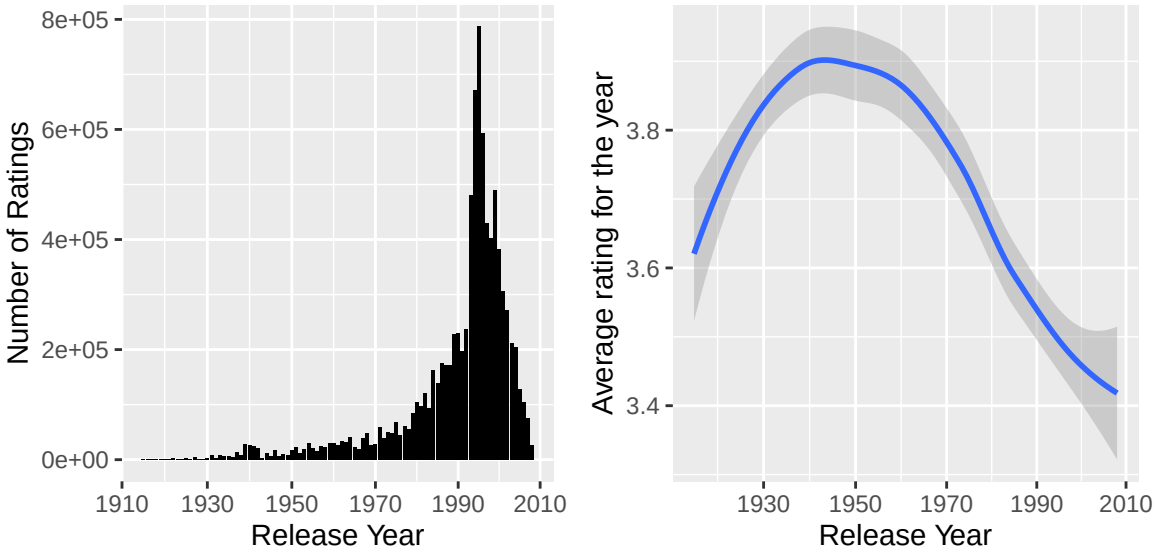
From the above, we can observe that there is a big difference on how frequent users rate a movie; also how critical they are when rating movies. Some tend to give generally a high rating to all/most of the movies, while others tend to give low ratings, and some are in between.

rating effect



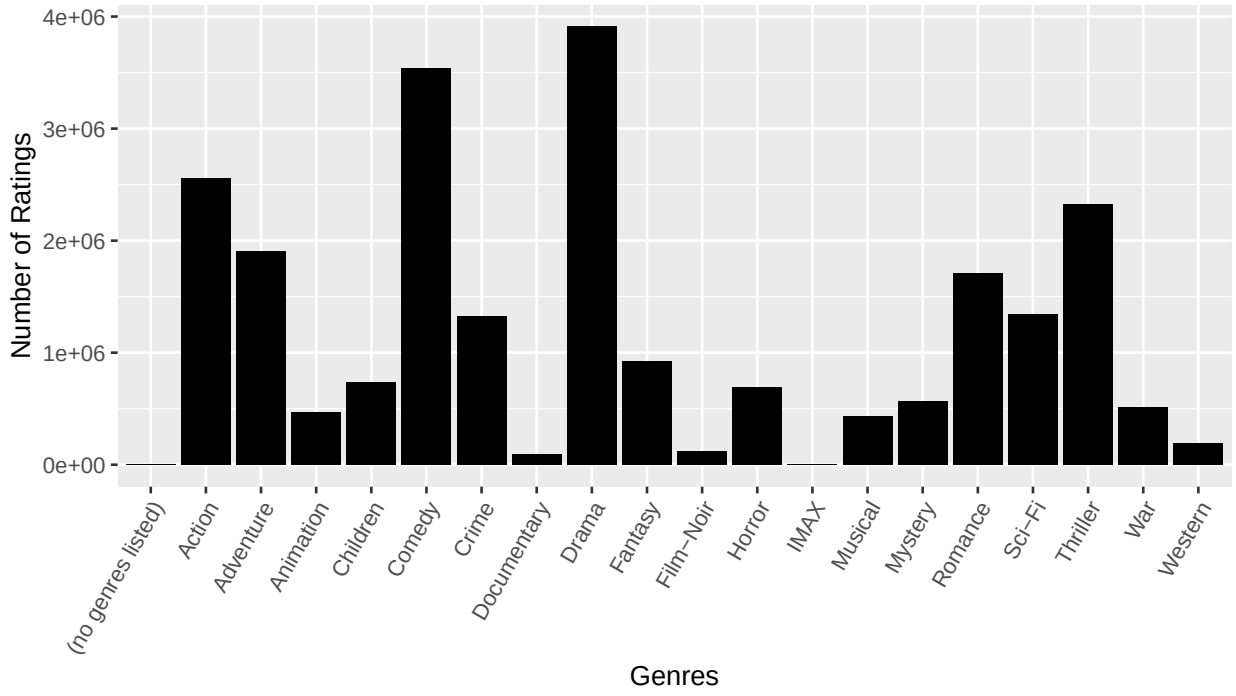
We can observe some ratings (4, 3, 5) are more commonly used than the other ratings. Also the half ratings (such as 2.5, 3.5 etc) are less used than the full ratings (2, 3 etc). Most of the movies with very high number of ratings tend to have a rating of greater than the overall average rating, which is 3.5124652.

movie release year effect



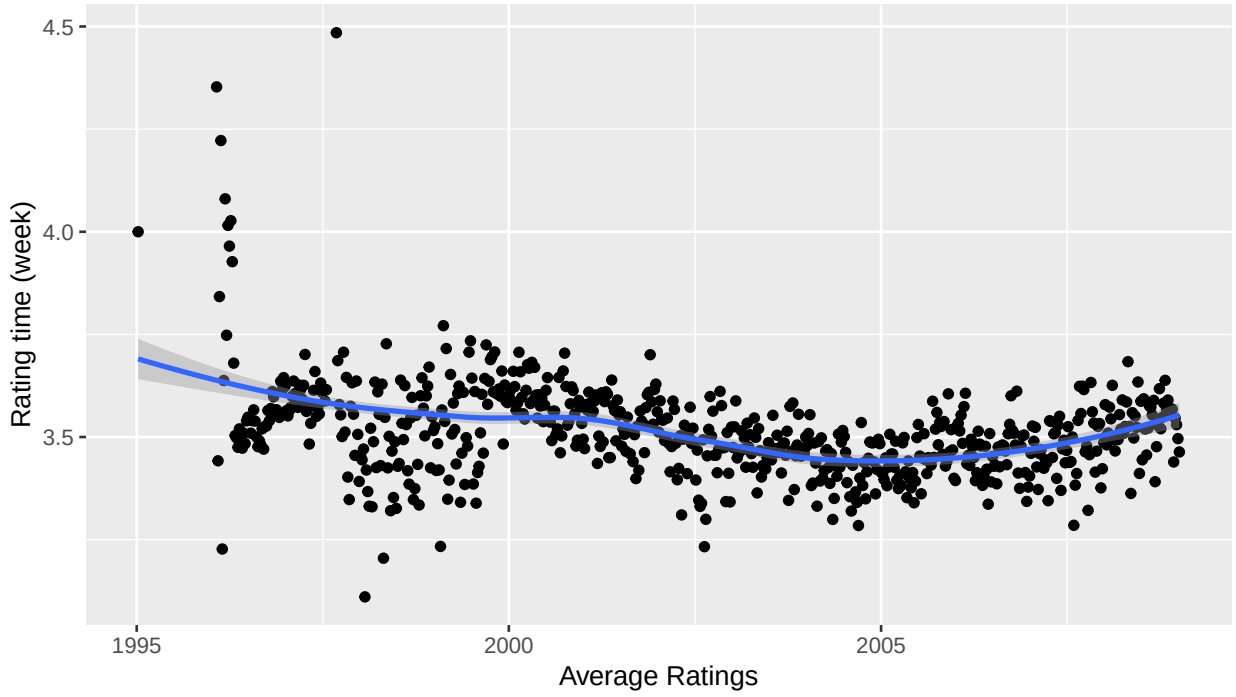
We can observe that the number ratings were high in the mid 90's, then decreasing every year since then. The recent movies have less number of ratings. We can also observe that the average rating tends also to decrease over time, the old movies have higher average ratings than the recent movies.

genres effect



From the above chart, we can observe there is a big difference in the number of rating to the different genres.

timestamp (user rating time) effect



From the above timestamp (grouped by week) chart, we can observe that the timestamp (rating time) has small effect of the ratings.

2.3. Model developing approach

As already mentioned above, the edx data set is a very large data set and we cannot use the standard algorithms functions such as lm, glm, knn, rf etc. So I will start with a very simply model that predicts all users' ratings to any movie to be the same as the overall average rating, and then keep improving the model by incorporating the effect of the different variables.
As it can be seen from the analysis in section 2, the different variables have different effect on the ratings. So we can include or exclude the different variables as needed - depending on the improvement and complexity they will introduce to the model.
As it can be seen, the effect of the timestamp (rating time) is minimum, so we can exclude it from the model. The genres has significant effect on the ratings, but to have a meaningful analysis of genres' effect, it requires splitting the composite genres into individual genres, and calculating the effect of each genres using relatively more complex calculations. So we will introduce the genres effect into the model only if we cannot achieve the required RMSE with the other variables.
So we will start with simple average model, and then include the effect of the other variables in the following order until we get the desired RMSE: movie, user, year, genres, rating time. As we saw in the analysis section, some movies have a very high number of ratings, while other have very few numbers of rating. We also saw some users rate very rarely while other rats more frequently. The very high and very low ratings that are coming from small samples (few numbers of ratings) might affect the predication negatively. So we will use a method known as Regularization to penalize (decrease the effect) of the very high or low ratings that are coming from small samples.
We will further split the edx data set into two parts – edx_train (80%) and edx_test (20%), then use edx_train for training the model and edx_test for cross-validating and tuning the model to get the best value of lambda (the regularization constant) that results in minimum RMSE.

3. Results

In this section we will test and compare the different models and select the best performing model as the Final model. We will then apply the edx and validation data set as training and testing data sets respectively to the Final model to get the final predictions and RMSE result.

```
# let's start building different models and check the RMSE results
# Partition edx into two parts: edx_train (80%) and edx_test (20%).
# edx_train will be used to build/train the model, while edx_test
# is used for cross-validating the model.

set.seed(2)
edx_test_index <- createDataPartition(y = edx$rating, times = 1,
                                      p = 0.2, list = FALSE)
edx_train <- edx[-edx_test_index,]
edx_test_tmp <- edx[edx_test_index,]

# get all records in edx_test_tmp with corresponding records in
# edx_train and set it to edx_test data set
edx_test <- edx_test_tmp %>%
semi_join(edx_train, by = "movieId") %>%
semi_join(edx_train, by = "userId")

# remove those records in edx_test_tmp without corresponding records
```

```r
#in edx_train, and put them back to edx_train.
removed_tmp <- anti_join(edx_test_tmp, edx_test)
edx_train <- rbind(edx_train, removed_tmp)

# the following function will be used to calculate the RMSE, for a
# given true_ratings and predicted_ratings, throughout the script.
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# the following would be used to store the RMSE values of the different model
rmse_results <- tibble(Method=character(), RMSE=double())

# "Overall average" model - use the overall average rating as prediction to
# every user/movie combinations.

# get the overall average value, mu.
mu<-mean(edx_train$rating)

# Calculate the RMSE for the above "Overall average" model and store the
# result in the rmse_results table.
rmse<-RMSE(edx_test$rating, mu)
rmse_results <- bind_rows(rmse_results,tibble(Method="Overall average", RMSE=rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.060273

```r
# "Overall average + Movie" Model - add the movie effect to the "Overall average" model.
mu <- mean(edx_train$rating)
movie_avgs <- edx_train %>%
group_by(movieId) %>%
summarize(b_i = mean(rating - mu))

predicted_ratings <- edx_test %>%
left_join(movie_avgs, by = "movieId") %>%
mutate(pred = mu + b_i) %>% .$pred

# Calculate the RMSE for the above "Overall average + Movie" model and store the result
# in the rmse_results table.
rmse<-RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Overall average + Movie", RMSE=rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.0602732
Overall average + Movie	0.9440821

```r
# regularize the prediction to get better result - penalize large estimates that are
# based on small sample sizes. Test the model with different values of lambda to get
# the lambda value that minimize the RMSE.
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_train$rating)
  movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+l))

  predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(pred = mu + b_m) %>% .$pred
  return(RMSE(predicted_ratings, edx_test$rating))
})
# Take the lowest RMSE and store the result in the results table.
rmse_results <- bind_rows(rmse_results,
          tibble(Method="Overall average + Movie + Regularize", RMSE=min(rmses)))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.0602732
Overall average + Movie	0.9440821
Overall average + Movie + Regularize	0.9440254

```r
# add the user's effect to the model.
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_train$rating)
  movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))
  user_avgs <- edx_train %>%
    left_join(movie_avgs, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating -mu -b_i)/(n()+l))

  predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>% .$pred
  return(RMSE(predicted_ratings, edx_test$rating))
})
rmse_results <- bind_rows(rmse_results,
      tibble(Method="Overall average + Movie + User + Regularize", RMSE=min(rmses)))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.0602732
Overall average + Movie	0.9440821

Method	RMSE
Overall average + Movie + Regularize	0.9440254
Overall average + Movie + User + Regularize	0.8662526

```
# add movie release year effect
lambdas <-seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_train$rating)
  movie_avgs <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  user_avgs <- edx_train %>%
    left_join(movie_avgs, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating -mu -b_i)/(n()+l))
  year_avgs <- edx_train %>%
    left_join(movie_avgs, by="movieId") %>%
    left_join(user_avgs, by="userId") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating -mu -b_i -b_u)/(n()+l))

  predicted_ratings <- edx_test %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    left_join(year_avgs, by = "year") %>%
    mutate(pred = mu + b_i + b_u + b_y) %>% .$pred
  return(RMSE(predicted_ratings, edx_test$rating))
})
rmse_results <- bind_rows(rmse_results,
tibble(Method="Overall average + Movie + User + year + Regularize", RMSE=min(rmses)))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.0602732
Overall average + Movie	0.9440821
Overall average + Movie + Regularize	0.9440254
Overall average + Movie + User + Regularize	0.8662526
Overall average + Movie + User + year + Regularize	0.8659700

```
#Get the lambda that results in minimum RMSE, and use it for the final model.
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 4.75
```

From the above results, we select our Final model to be "Overall average + Movie + User + year + Regularize" with lambda=4.75.
We will apply the whole edx data to train the Final model and test it with the validation data set. In order to have a valid prediction, we need to make sure the validation data set doesn't contain n/a in user, movie, year columns, and all user and movies in the validation data set are also included in the edx data set.

```
# Final model - "Overall average + Movie + User + year + Regularize" with lambda=4.75
# we will use the whole edx data set to train the final model and use the validation
# data set it and get the final RMSE value.

# since we extract the movie release year from the edx data set and used it for
# predication, we will need to extract the movie release years from the validation data too.
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation %>% as_tibble()
```

```
## # A tibble: 999,999 x 7
##    userId movieId rating  timestamp title                 genres            year
##     <int>   <dbl>  <dbl>      <int> <chr>                 <chr>            <dbl>
##  1      1     231    5    838983392 Dumb & Dumber (1994)  Comedy            1994
##  2      1     480    5    838983653 Jurassic Park (1993)  Action|Adventur~  1993
##  3      1     586    5    838984068 Home Alone (1990)     Children|Comedy   1990
##  4      2     151    3    868246450 Rob Roy (1995)        Action|Drama|Ro~  1995
##  5      2     858    2    868245645 Godfather, The (1972) Crime|Drama        1972
##  6      2    1544    3    868245920 Lost World: Jurassic~ Action|Adventur~  1997
##  7      3     590    3.5 1136075494 Dances with Wolves (~ Adventure|Drama~  1990
##  8      3    4995    4.5 1133571200 Beautiful Mind, A (2~ Drama|Mystery|R~  2001
##  9      4      34    5    844416936 Babe (1995)           Children|Comedy~  1995
## 10      4     432    3    844417070 City Slickers II: Th~ Adventure|Comed~  1994
## # ... with 999,989 more rows
```

```
# Make sure userId and movieId in validation set are also in edx set
#(i.e. to make sure we have a valid prediction when using validation data set)
validation_exist_in_edx <- validation %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
identical(validation_exist_in_edx,validation)
```

```
## [1] TRUE
```

```
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
  user_avgs <- edx %>%
    left_join(movie_avgs, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating -mu -b_i)/(n()+lambda))
  year_avgs <- edx %>%
    left_join(movie_avgs, by="movieId") %>%
    left_join(user_avgs, by="userId") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating -mu -b_i -b_u)/(n()+lambda))

  predicted_ratings <-
    validation %>%
    left_join(movie_avgs, by = "movieId") %>%
    left_join(user_avgs, by = "userId") %>%
    left_join(year_avgs, by = "year") %>%
```

```
    mutate(pred = mu + b_i + b_u + b_y) %>% .$pred

Final_RMSE<-(RMSE(predicted_ratings, validation$rating))

rmse_results <- bind_rows(rmse_results, tibble(Method="Final model", RMSE=Final_RMSE))
rmse_results %>% knitr::kable()
```

Method	RMSE
Overall average	1.0602732
Overall average + Movie	0.9440821
Overall average + Movie + Regularize	0.9440254
Overall average + Movie + User + Regularize	0.8662526
Overall average + Movie + User + year + Regularize	0.8659700
Final model	0.8645223

4. Conclusion

As it can be seen form the project, we see we were able to successfully build a very simple and light weight model and still achieve a reasonable low (0.8645223) RMSE. If further improvement is need to the model, the gender and timestamp effect could be included. We saw the prediction results get better with large size samples, but at the same time we saw the challenges associated with dealing with a very large data set. So whenever possible we should consider reducing the size of the data set by removing data, which doesn't have much importance to the predictions. We also saw how the ratings coming from small size of sample could affect the prediction result and how Regularization method was used to effectively reduce the effect of these small sized sample data sets.