FinalReport

Khai Hung Luong

05/06/2021

Introduction

In this project, I try to build a simple machine learning model that can predict the rating of the movies based on their various factors from the data set. This project and data set are taken from the final project of Edx data science courses.

The data set (MovieLens) has 10 millions ratings with the following columns:

```
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

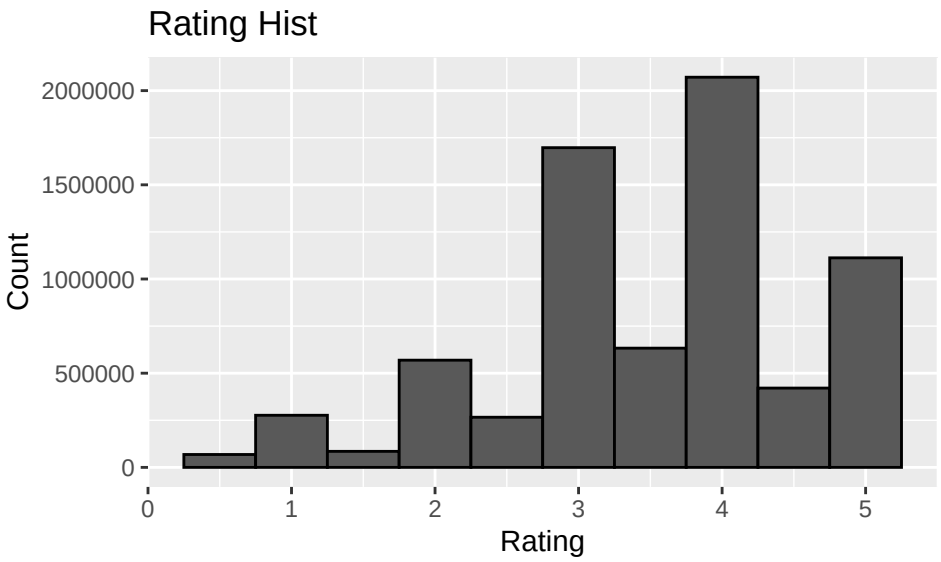
One important thing to notice is that the rating values range from 0 to 5, with the interval of 0.5.

I divide it into two sets, one for creating the model (90% of the data set) and one for validation (10% of the data set). Furthermore, in the first data set, I use the same method to create train set and test set.

The algorithm used in this project will use linear regression model and include "bias" such as users, genre to predict the ratings. Regularization will be implemented as a tool to remove occasional cases and prevent overfitting. Please keep in mind that this project is just a simple so there will be errors or inefficient methods or algorithm that can be improved.

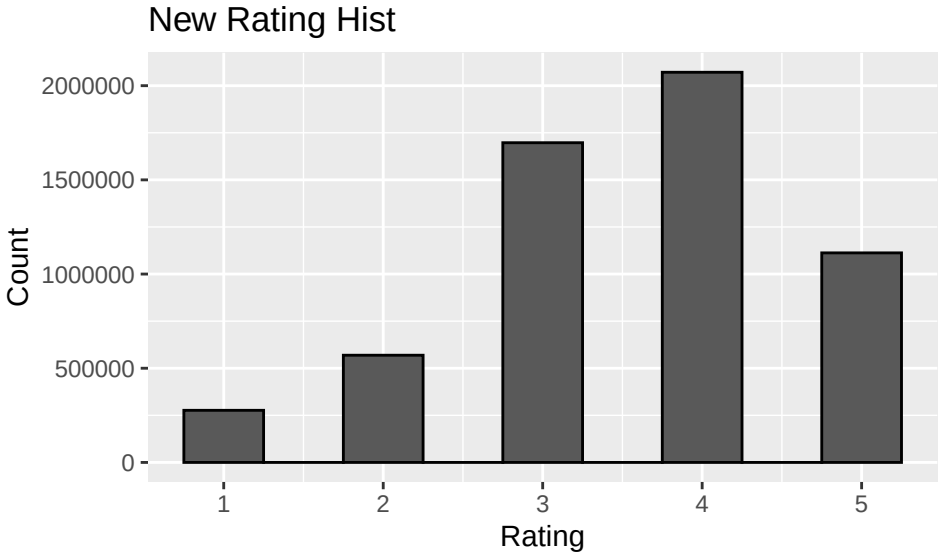
Data Exploration and Processing

First we try to analyze some basic characteristics of the data set:



As we can see, the rating distribution follows normal distribution with rating 3 and 4 have the highest use. In addition, we see that the number of user who put ratings as integer is siginificantly higher than the ones who put ratings as decimal. For training purpose, we will only use the data that has rating is an integer in both trainSet and testSet.

Figure for new train set:



Model Development

The algorithm that we use here includes the mean rating μ, the bias terms: b_m for movie bias, b_u for user bias, b_g for genre bias and residual error ε:

$$Y_{u,i} = \mu + b_m + b_u + b_g + \varepsilon_{i,u,g}$$

```
#Calculate mean of the new train set
mu <- mean(newTrainSet$rating)
mean_RMSE <- RMSE(testSet$rating, mu)
mean_accuracy <-  sum(newTestSet$rating == round(mu)) / length(newTestSet$rating)
#Calculate median of the new train set
med <- median(newTrainSet$rating)
median_RMSE <- RMSE(testSet$rating, med)
median_accuracy <- sum(newTestSet$rating == med) / length(newTestSet$rating)
```

Type	Accuracy	RMSE
Mean	0.361651	1.060728
Median	0.361651	1.166678

From previous data, we notice that if we only use mean as a tool for prediction, the accuracy is around 36%. The same goes for median. Next, we try to apply more bias terms to improve our prediction, which is b_m (movie bias), b_u (user bias) and b_g (genre bias):

```
#Calculate movie bias b_m
b_mData <- newTrainSet %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu))
#Calculate user bias b_u
b_uData <- newTrainSet %>%
  left_join(b_mData, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m))
#Calculate genre bias b_g
b_gData <- newTrainSet %>%
  left_join(b_mData, by='movieId') %>%
  left_join(b_uData, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_m - b_u))
#Combine all bias into new the prediction
predictRating <- newTestSet %>%
  left_join(b_mData, by='movieId') %>%
  left_join(b_uData, by='userId') %>%
  left_join(b_gData, by='genres') %>%
  mutate(prediction = mu + b_m + b_u + b_g) %>%
  mutate(roundedPrediction = round(prediction))
#Remove columns that have NA data due to loss of some certain movie when do data processing
naList <-which(is.na(newTestSet$rating == predictRating$roundedPrediction))
predictRating <- predictRating[-naList,]
newTestSet <- newTestSet[-naList,]
# Calculate new accuracy and RMSE
predict_mean_accuracy <- sum( predictRating$roundedPrediction == newTestSet$rating) / length(newTestSet$
predict_RMSE <- RMSE(newTestSet$rating, predictRating$prediction)
```

Type	Accuracy	RMSE
Mean	0.3616510	1.0607280
Median	0.3616510	1.1666776

Type	Accuracy	RMSE
Include biases	0.4583417	0.8702087

Regularization

As we can see, even though the accuracy is good for a simple model, the RMSE is still high. Therefore, we try to apply regularization to remove records that have few ratings and affect the model. We will use the following function for each bias:

For b_m:

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

For b_u:

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

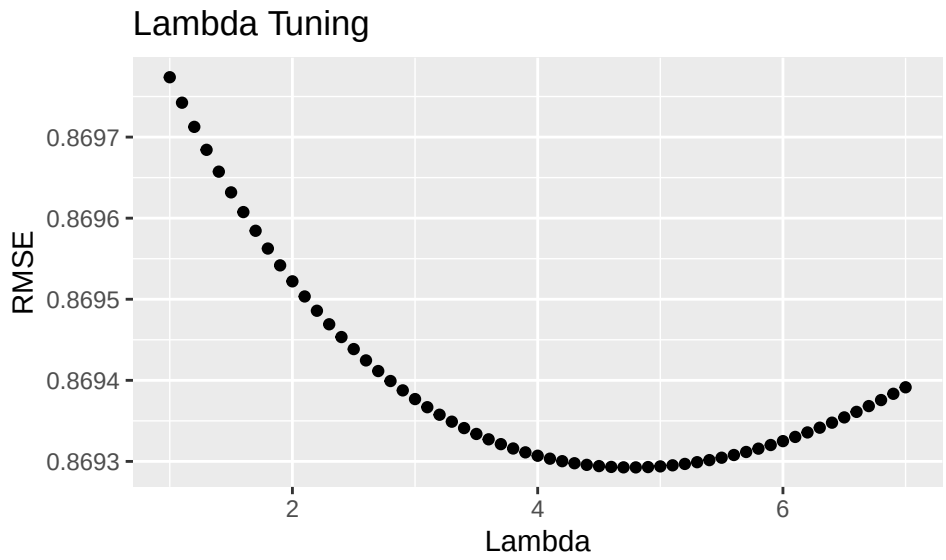
For b_g:

$$\hat{b}_g(\lambda) = \frac{1}{\lambda + n_g} \sum_{g=1}^{n_g} \left(Y_{i,u,g} - \hat{b}_i - \hat{b}_u - \hat{\mu}\right)$$

Code:

```
lambdas <- seq(1, 7, 0.1)
RMSEs_lambda <- sapply(lambdas, function(l){
  b_m <- newTrainSet %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu)/(n()+l))
  b_u <- newTrainSet %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_m - mu)/(n()+l))

  b_g <- newTrainSet %>%
    left_join(b_m, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_m - b_u - mu)/(n()+l))
  predicted_ratings <- newTestSet %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by="genres") %>%
    mutate(prediction = mu + b_m + b_u + b_g) %>%
    mutate(roundedPrediction = round(prediction))
  return(RMSE(newTestSet$rating, predicted_ratings$prediction))
})
```

We run over a sequence of lambda to find the best one that give the lowest RMSE:



The lambda value that we need to find is 4.8. Then, we rebuild the model with that lambda:

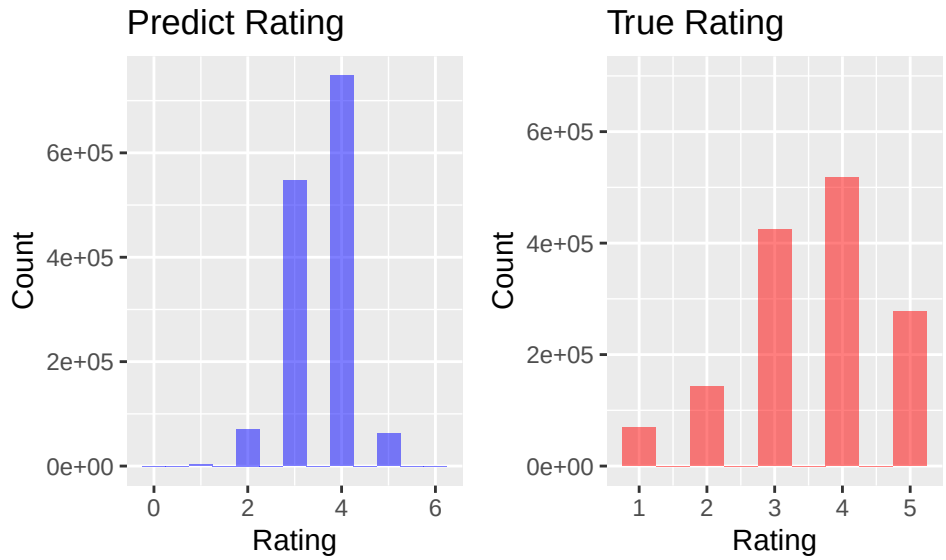
```
#Rebuild using max_lambda
b_m <- newTrainSet %>%
  group_by(movieId) %>%
  summarize(b_m = sum(rating - mu)/(n()+max_lambda))
b_u <- newTrainSet %>%
  left_join(b_m, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_m - mu)/(n()+max_lambda))

b_g <- newTrainSet %>%
  left_join(b_m, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - b_m - b_u - mu)/(n()+max_lambda))
predictRating <- newTestSet %>%
  left_join(b_m, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by="genres") %>%
  mutate(prediction = mu + b_m + b_u + b_g) %>%
  mutate(roundedPrediction = round(prediction))
```

We find out that regularization helps us reduce the RMSE:

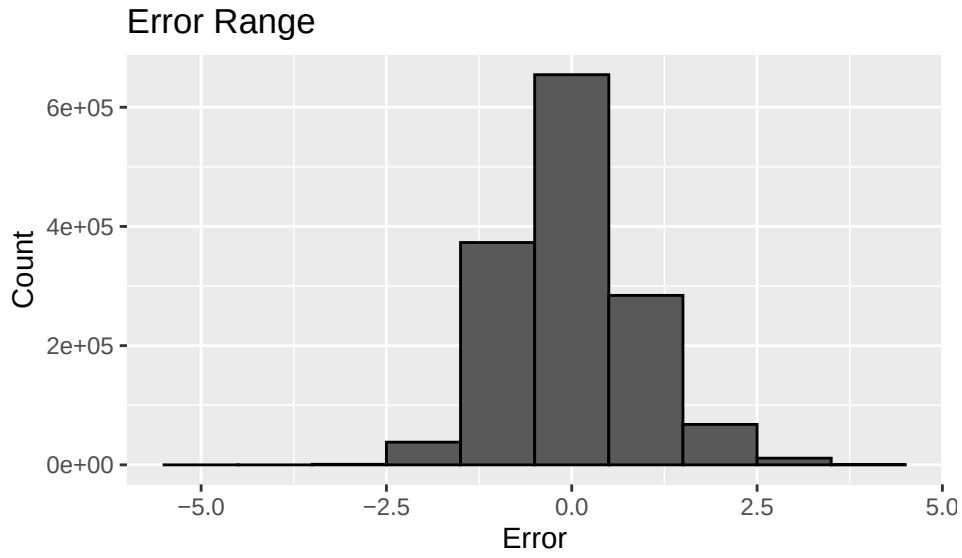
Type	Accuracy	RMSE
Mean	0.3616510	1.0607280
Median	0.3616510	1.1666776
Include biases	0.4583417	0.8702087
Include regularization	0.4576314	0.8692926

Conclusion

This is the graph of our predicted ratings versus the true rating:



This is the graph which shows the range of how many ratings that differs from their original values.



This is the accuracy and RMSE when I apply the model into the validation set:

Type	Accuracy	RMSE
Validation test	0.3633092	0.870801

As you can see, the major rating predictions fall into 3 and 4, and the error range lies most in +1/-1. Rating 5 is the rating that has most errors, since the reason is that we are only try to predict rounding rating, therefore a large number of predictions have been rounded down to 4. In addition, we see a reduction in accuracy when we apply the model into validation set. One reason is because in the model, we do not try to predict .5 ratings, thus lead to a significant loss of accuracy. We can improve it by adding some helper

functions that allow us to decide whether a predicted rating should be rounded up, down or stay in the middle (e.g 4.36 will become 4.5). In that way we can improve our data accuracy.