Predicting Movie Rating

Sujatha Rajarathnam

6/12/2019

Predicting Movie Rating

Overview:

We have approximately 10 millions records on movies rated by users. I would like to build a model that helps to predict a movie rating by a specific user. To determine effectiveness of the model we would use Root Mean Square Error (RMSE) as the means to rate the success of the model. The goal is to achieve root mean square error of less than 0.8649

About the data set: To be exact the training dataset contains 9000055 rows and 6 columns. The test data set contains 999999 rows and 6 columns.

```
#################################################################
################# LOADING THE REQUIRED PACKAGES ##################
#################################################################
# If tidyverse package does not exist, load from the following repo and install it
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Warning: package 'dplyr' was built under R version 3.6.1
```

```
# If caret package does not exist, load from the following repo and install it
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")

# If data.table package does not exist, load from the following repo and install it
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")



#################################################################
########## LOADING THE TEST and VALIDATION DATA SETS ##############
#################################################################
# 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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
```

```r
                                           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")
```

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

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)
edx <- rbind(edx, removed)

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


# Checking the number of rows and columns in the edx dataset
dim(edx) # 9000055       6
```

```
## [1] 9000055       6
```

```r
# checking the number of rows and columns in the validation dataset
dim(validation) # 999999      6
```

```
## [1] 999999      6
```

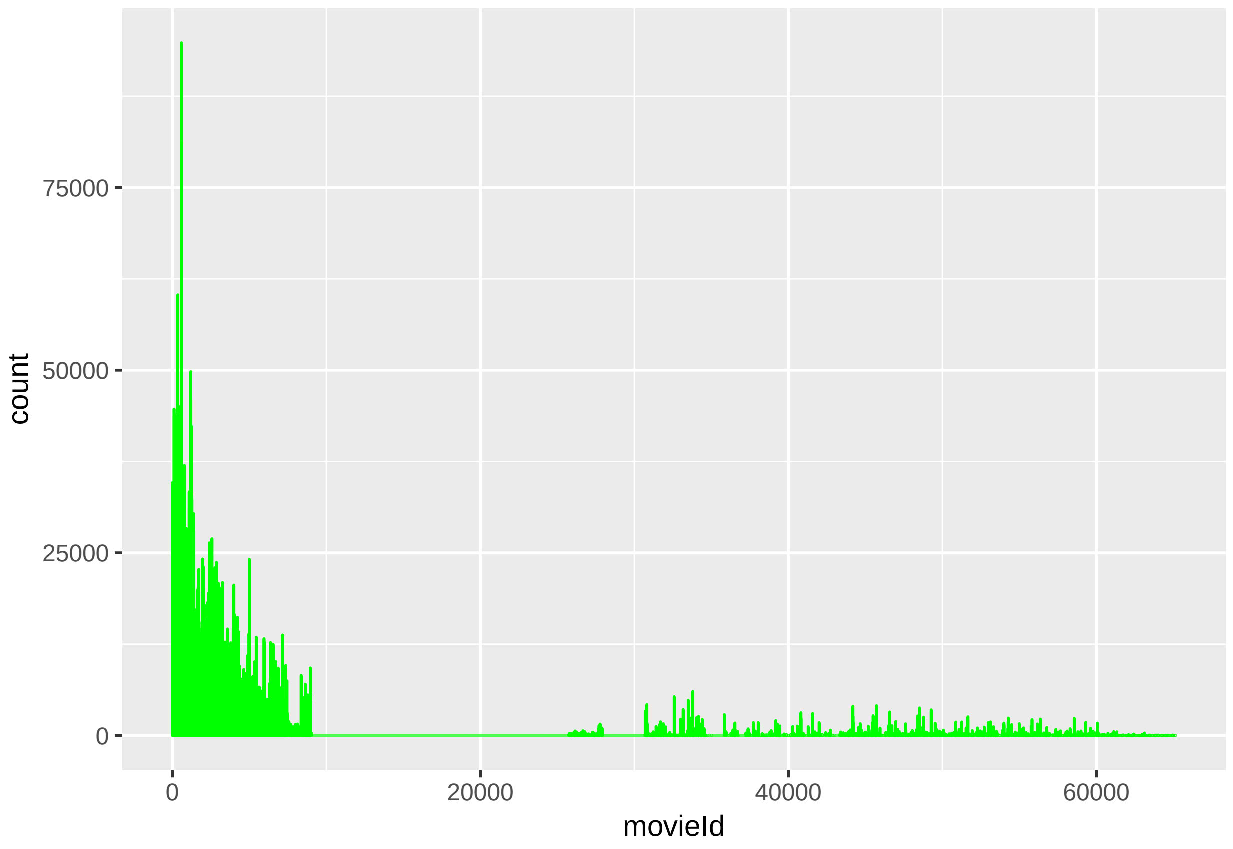
The dataset contains 6 columns, you can see their column names below.

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

Method and Analysis

I would like to build the model that is simple and yet effective. We will do that by calculating first the average rating of all the movies given by the user. We then will apply the movie effect because some are blockbusters and others are not. You can see this from the histogram below.

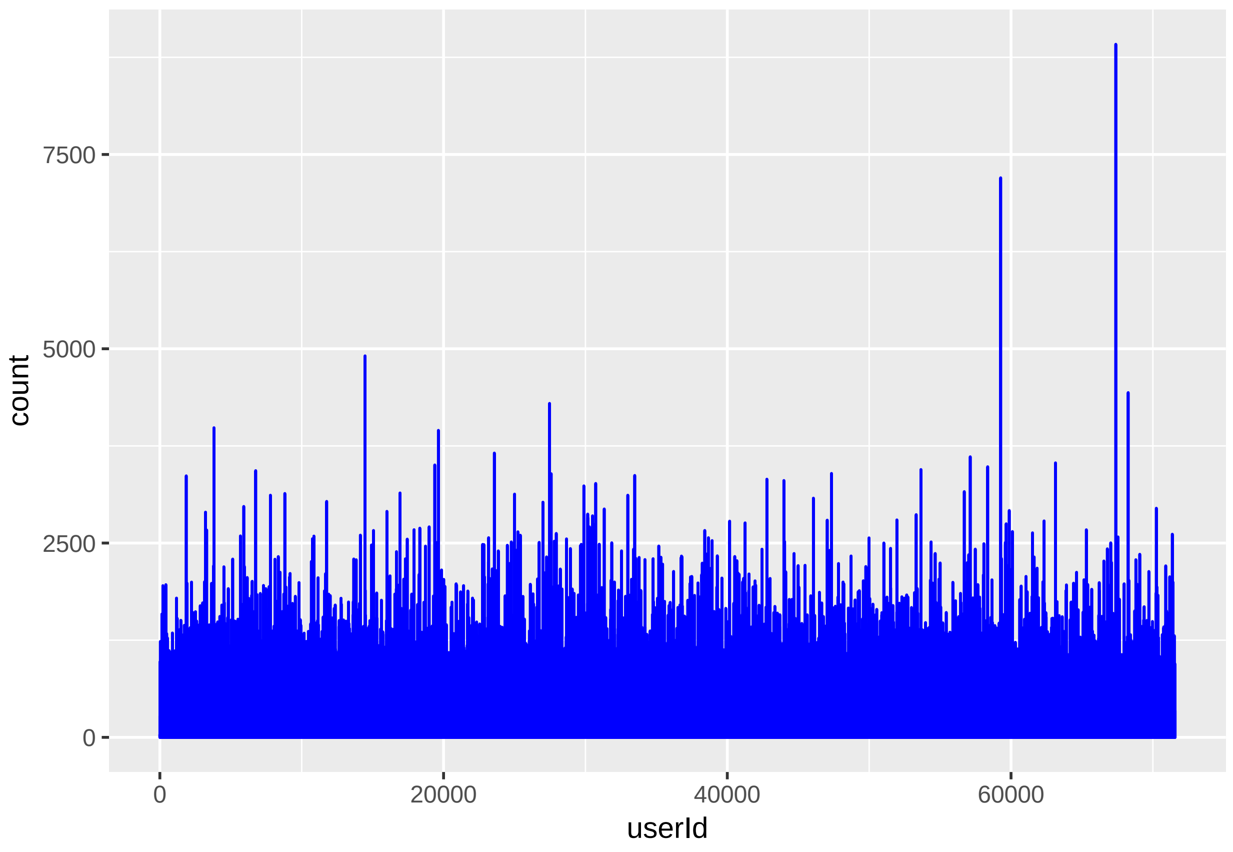
```
# You can see some movies are rated more than the others, because some are popular,
# blockbusters, critically acclaimed and etc. You can see them visually through the
# below plot
edx %>% ggplot(aes(movieId,color="movie ratings")) + geom_histogram(binwidth = 5, color="green")
```



## Method and Analysis - User effect

Similarly you can see the user effect it is clear from the below plot tat not all users rate the same level and some rate higher than the others

```
edx %>% ggplot(aes(userId,color="user ratings")) + geom_histogram(binwidth = 5, color="blue")
```

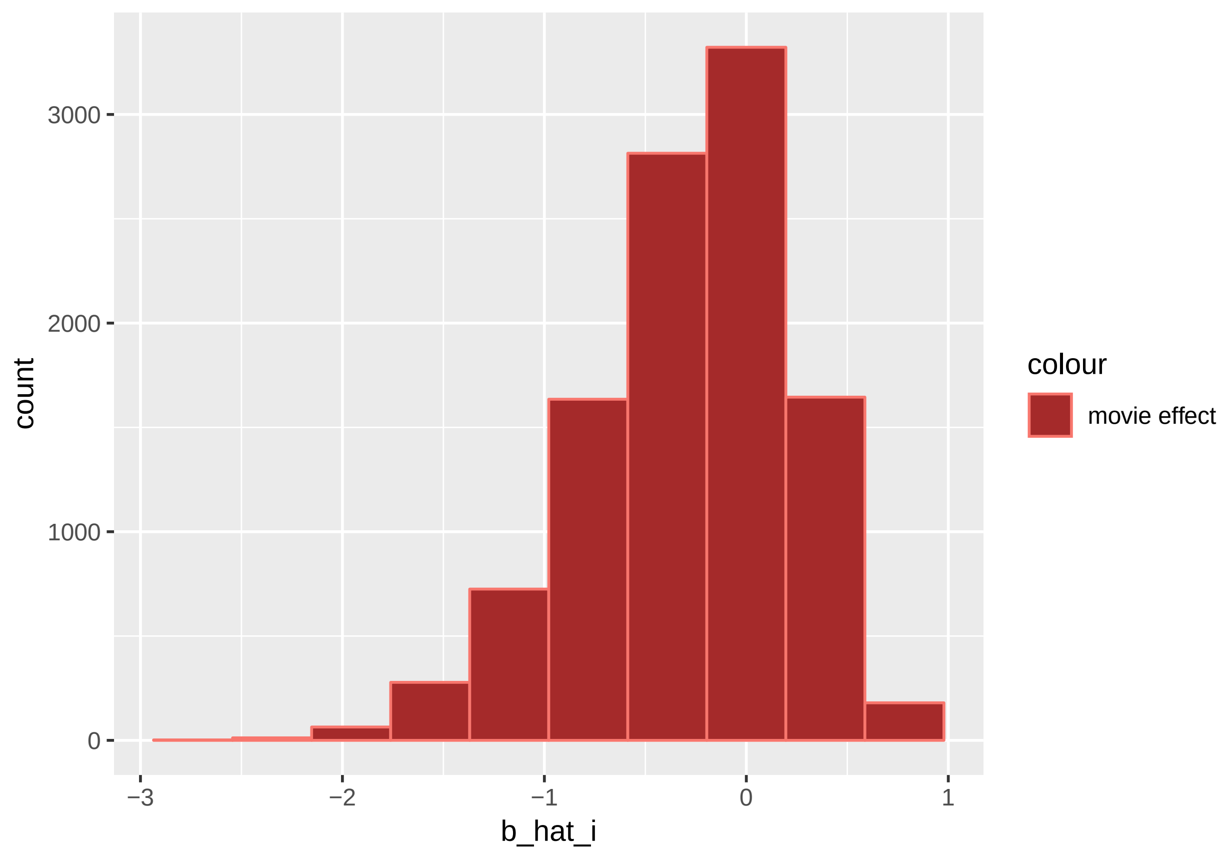


Method and Analysis - Building the equation

The above plots tell me that at the minimum to build a model that is effective to predict I need average of all movie ratins and add movie and user effect.So the equation is y__hat__iu = mu + bi + bu + epsilon, where mu is the average rating of all the movies, y__hat__iu is the predicted rating of a movie i by user u, bi is the average of movie effect of a movie i rated by all users and bu is the average of user effect on all movies rated by the user. Epsilon is the noise introduced, that is the residual obtained after determining all kinds of relationship we saw in the data. For simplicity we will assume that the epsilon is 0. Therefore the derived equation for the model would become: y__hat__iu = mu + bi + bu

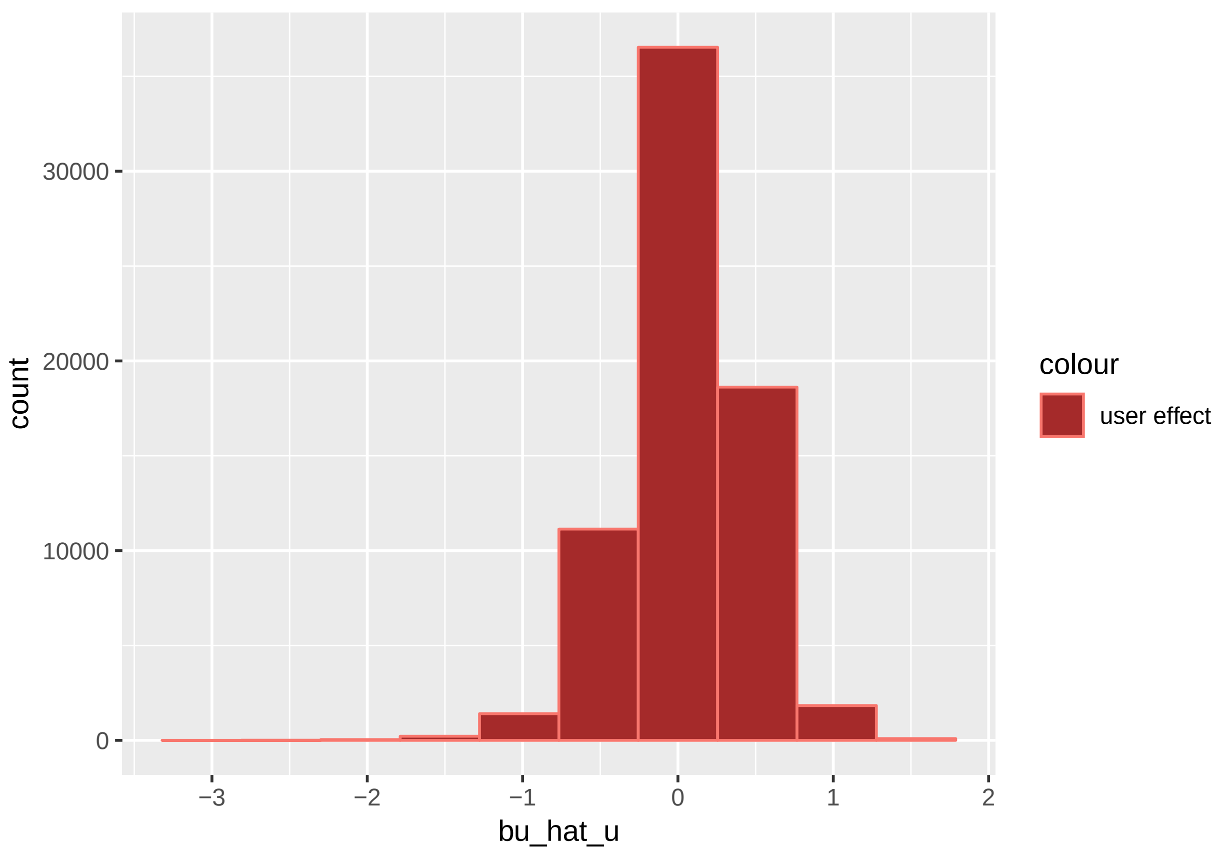
From the below histogram plot it is helpful to see the movie effect where we can see movie effect helping the good movies and discrediting the not so good movies.

```r
bi %>% ggplot(aes(b_hat_i, color="movie effect")) + geom_histogram(bins=10,fill="brown")
```



At the same time you can also see from the below histogram plot it is helpful to see the user effect, some users in general rate the movies they see optimistically than others. Some users more pessimistic than others.

```
bu %>% ggplot(aes(bu_hat_u, color="user effect")) + geom_histogram(bins=10,fill="brown")
```



Results

ROOT MEAN SQUARE ERROR (RMSE)

We will build a function for root mean square using the following code. Which we then call with predicted rating and actual rating in the test data set i.e validation dataset

```
RMSE <- function(predictedrating,actualrating)
{
  sqrt(mean((predictedrating-actualrating)^2))
}
```

Results - Contd

```
rmse_results <- RMSE(y_hat_iu,validation$rating)
rmse_results
```

```
## [1] 0.86489
```

Conclusion

We were able to build the model using the equation y_hat_iu = mu + bi + bu. We then were able to run this against the validation set and calculated the RMSE. We obtained RMSE of 0.86489 which tells us to we are able to predict the rating for the movie pretty close to the actual user's rating for any movie.

RMSE RESULT: 0.86489