Project on Movielens Dataset

Sonam Bhadauria

5/31/2019

Project Overview

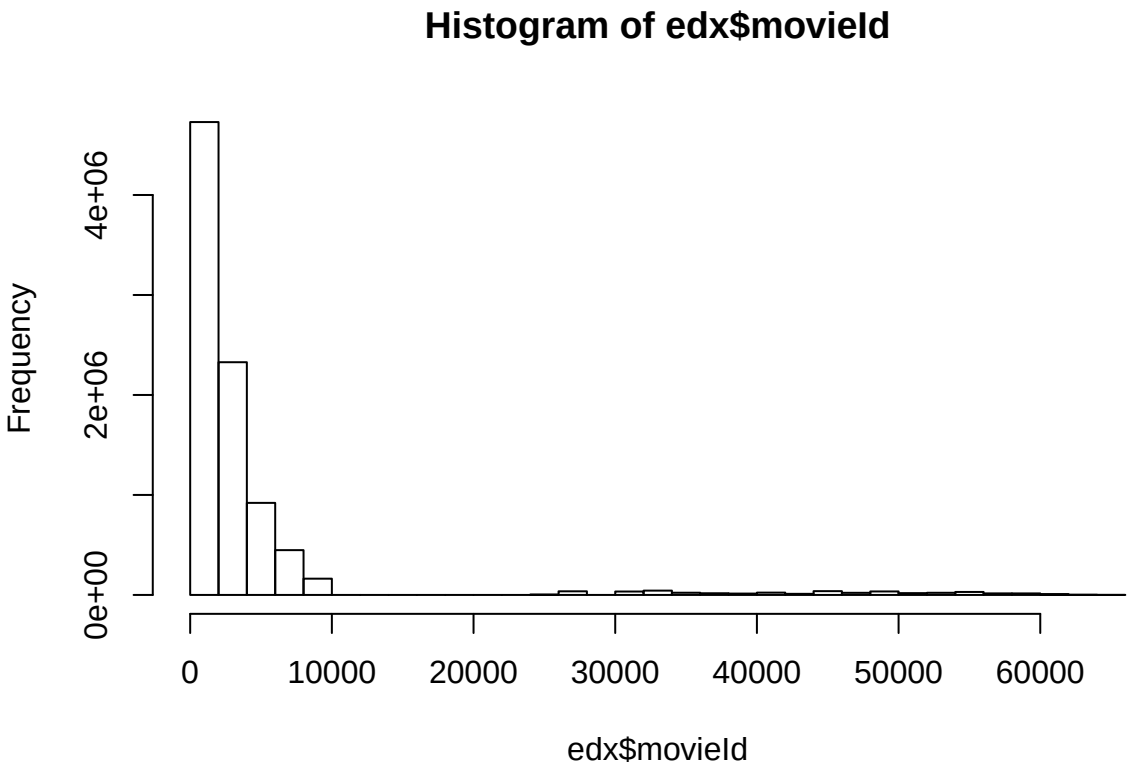
We have used Movielens dataset to create a Movie recommendation system using R language. Recommendation systems use ratings that users have given to items(in this case - movies) to make specific recommendations.

We had used the code provided in the "Data Science: CAPSTONE"course to create our datasets : edx & validation (90% - 10 % of movielens data respectively). Mainly, we had worked on edx dataset to develop our algorithm and then we used validation dataset once we had our final model ready to check its accuracy. RMSE is being used to evaluate how accurate our predictions are.

Analysis

At First, we ran few checks in our dataset to check its structure or if it has any kind of anomalies. After tidying the data, we can look at some of its general properties now :

- We can check the number of unique users that provided ratings and how many unique movies were rated.
- As per the distribution below, we can notice some movies get more rating than the others.



Since we had already prepped the data, we went ahead and split the edx dataset in order to create train & test datasets. We are going to build the algorithm with train set and use test set to evaluate the accuracy of the model.

RMSE

we can interpret the RMSE (Residual Mean Squared Error) similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If this number is larger than 1, it means our typical error is larger than one star, which is not good.

Let us write a function that computes the RMSE for vectors of ratings and their corresponding predictors:

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

Models

- Base RMSE - We started by building the simplest possible recommendation system: we predict the same rating for all movies regardless of user. In this case, is the average of all ratings.

A model that assumes the same rating for all movies and users with all the differences explained by random variation would look like this:

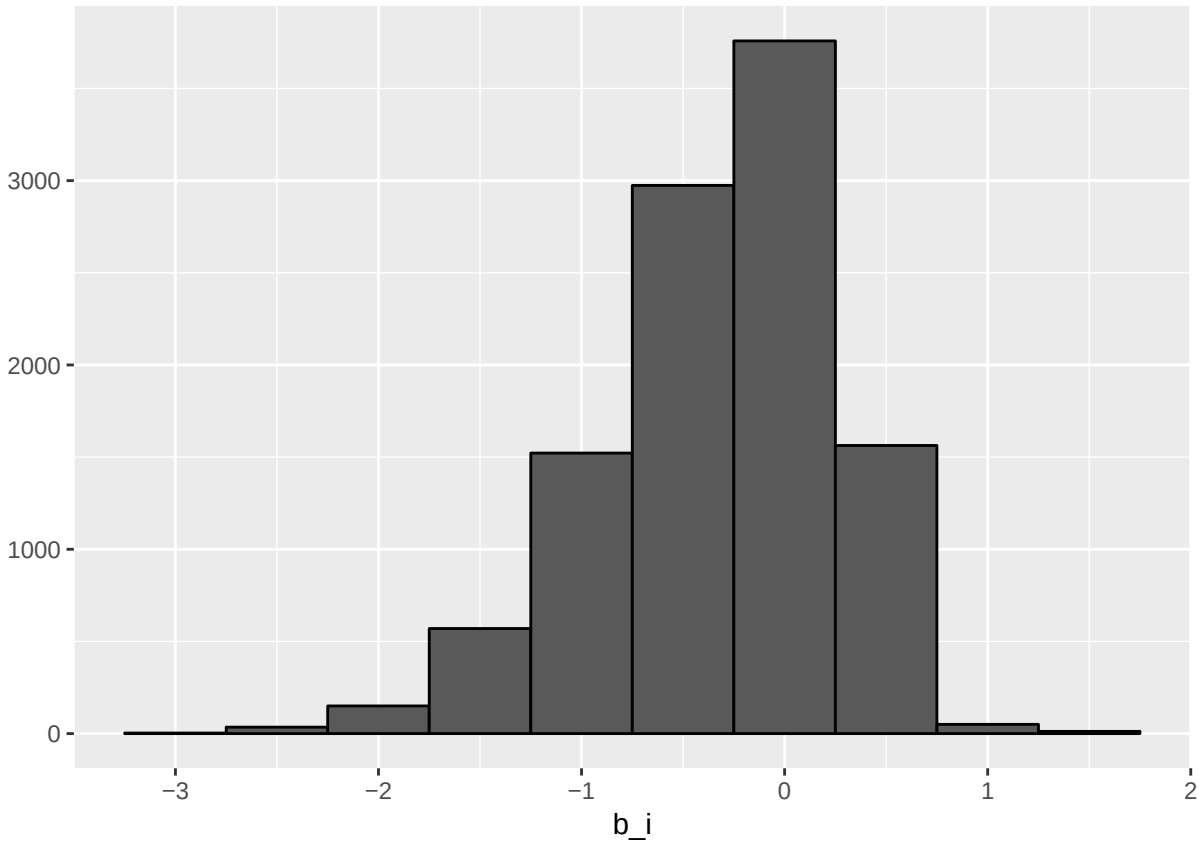
Y(u,i) = mu + e(u,i)

with e(u,i) independent errors sampled from the same distribution centered at 0 and mu the true rating for all movies.

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

- MOVIE EFFECTS MODEL - As per our data, we know different movies are rated differently. Hence, we added a term signifies movie bias/effect in our base rmse model which resulted in the improvement of our model.

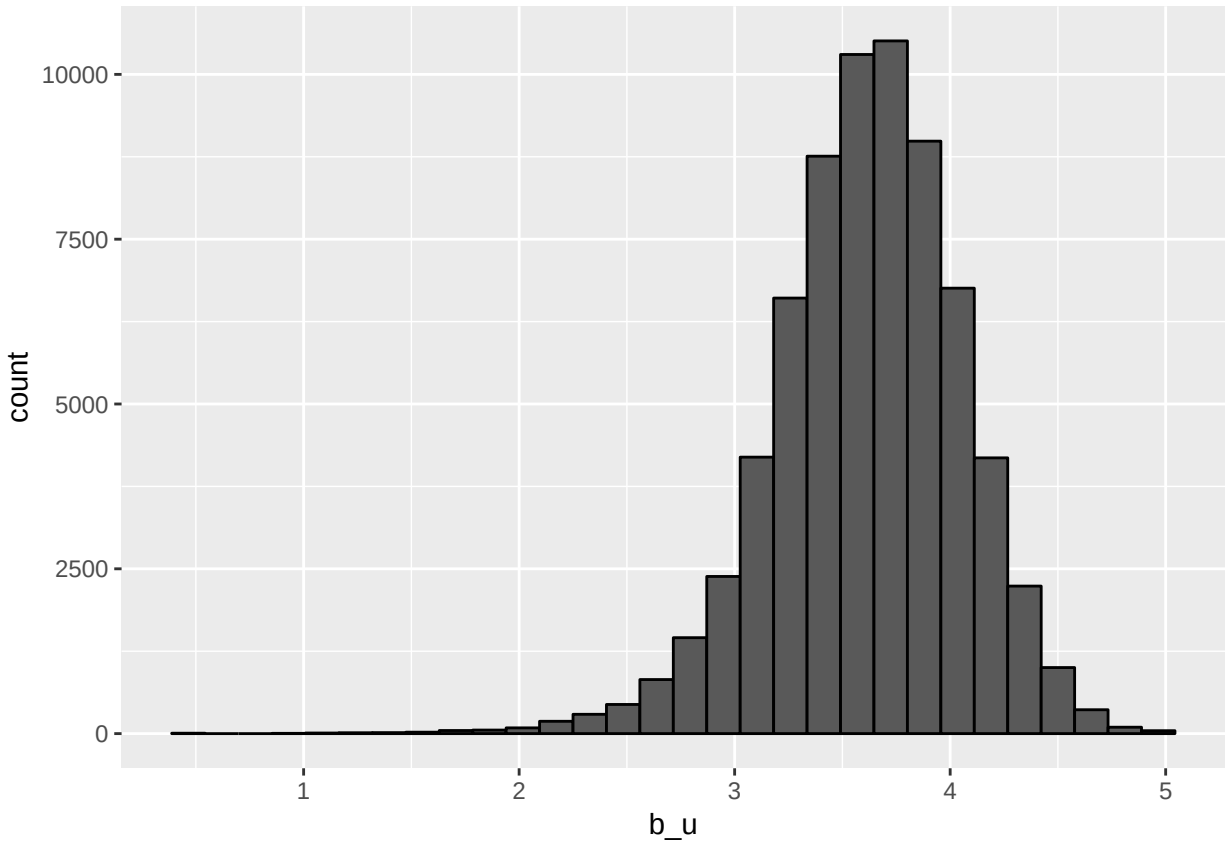
Y(u,i) = mu + b_i + e(u,i)




```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Base RMSE           1.06
## 2 Movie Effects Model 0.944
```

- USER EFFECTS MODEL - While computing the average rating for user, We noticed that there is substantial variability across users as well. This implies that a further improvement to our model can be made by adding user specific effect in the model. And yet again we see the improvement by constructing predictors.

Y(u,i) = mu + b_i + b_u + e(u,i)



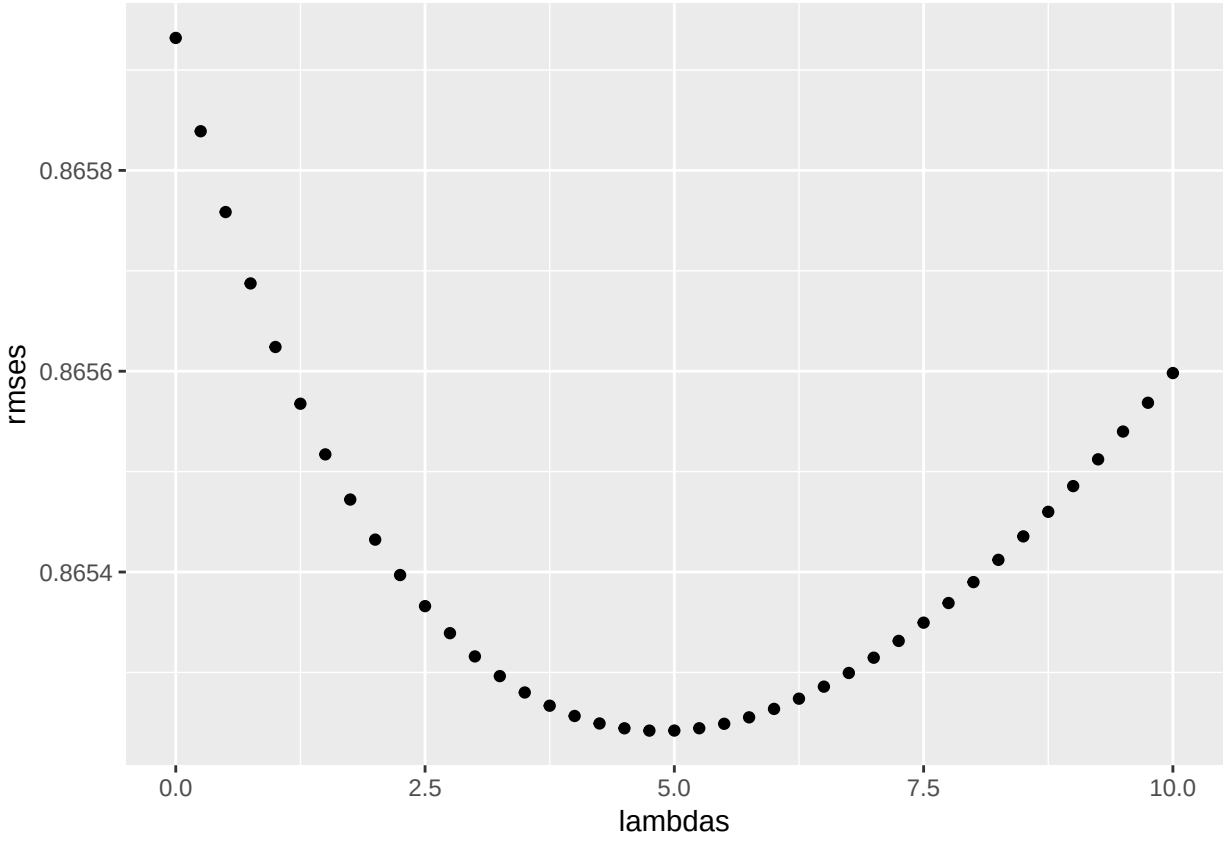

```
## # A tibble: 3 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Base RMSE                 1.06
## 2 Movie Effects Model       0.944
## 3 Movie + User Effects Model 0.866
```

Regularization

Regularization permits us to penalize large estimates that are formed using small sample sizes. As we could see in our data that the best & worst movies are rated by very few users.These are noisy estimates that we should not trust, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure. The penalized estimates provide a large improvement over the least squares estimates.

Here, lambda is tunning parameter, we have used cross validation (on train set) to pick its value.

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



For the full model, the optimum lambda is 4.75

```
## # A tibble: 4 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Base RMSE                             1.06
## 2 Movie Effects Model                   0.944
## 3 Movie + User Effects Model            0.866
## 4 Regularized Movie + User Effects Model 0.865
```

Time Concept

We used the time based concept in our model to see if it can further minimize the rmse. We tried to find the relationship factor (alpha) between the ratings and the time passed by between thefirst rating and all the subsequent ratings

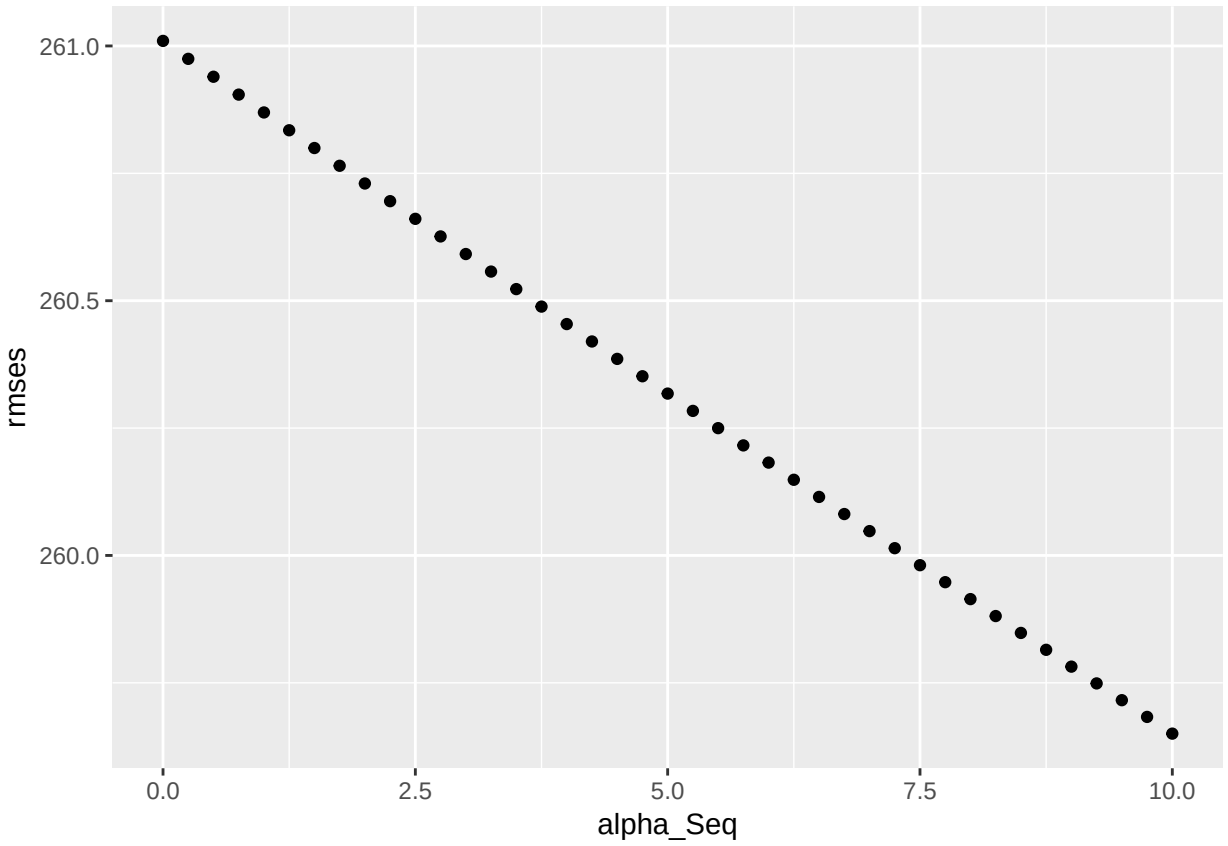
```
# Writing a function to bin the time deltas
get_bin <- function(max_timestamp, min_timestamp,curr_timestamp){
  max_timestamp
  min_timestamp
  ts_delta <- max_timestamp - min_timestamp
  ts_delta
  bin_num <- cut(ts_delta,breaks = 5,labels = c('1','2','3','4','5'))
  return(as.numeric(bin_num))
}

l = 4.75
alpha_Seq = seq(0,10,0.25)
```

```
rmses <- sapply(alpha_Seq, function(alpha_val){

  mu <- mean(train_set$rating)

  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  b_t <- train_set %>%
    left_join(b_i,by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    group_by(movieId) %>%
    mutate(bin = as.numeric(cut(timestamp, breaks = 5, labels = c(1:5)))) %>%
    group_by(movieId) %>%
    summarize(b_t = sum(rating - b_i - b_u - mu + (bin * alpha_val)/(n() * l)))

  predicted_ratings <-  test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_t, by = 'movieId') %>%
    mutate(pred = mu + b_i + b_u - b_t) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})
alpha_Seq[which.min(rmses)]
```

```
## [1] 10
```

```
which.min(rmses)
```

```
## [1] 41
```



Here, we can see that this approach does not help in reducing the RMSE. Hence we will skip adding this concept to our model and stick by the approach with lowest RMSE.

Result

```
## # A tibble: 4 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Base RMSE                            1.06
## 2 Movie Effects Model                  0.944
## 3 Movie + User Effects Model           0.866
## 4 Regularized Movie + User Effects Model 0.865
```

Conclusion

As we can see the Lowest RMSE is 0.865 for the "Regularized Movie + User Effects Model".

So we trained this model on the whole edx dataset and found the RMSE on validations dataset. There is a significant improvement in the RMSE i.e. 18.4 % (as comapared with the base RMSE).

We could also consider some other factors which might help in improving the RMSE even further, however our target is achieved and that is why we didnt go any further.

```
## [1] 0.8648201
```

```
## # A tibble: 5 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Base RMSE                            1.06
```

```
## 2 Movie Effects Model                    0.944
## 3 Movie + User Effects Model             0.866
## 4 Regularized Movie + User Effects Model 0.865
## 5 Final RMSE Validation                  0.865
```