movieLens_edx_project

Siddhartha Sampath

6/10/2019

Executive Summary

The data used was the movieLens dataset provided by the edx team. Two main dataframes were provided, one called `edx` to be used for training and one called `validation` to be used to test predictions against for calculating the RMSE.

The goal of the project was to predict movie ratings for the validation dataset while minimizing the root mean square error. In order to do this, I tried many different models that I abandoned either because they did not produce the required RMSE or were unable to handle the size of the dataset. The closest one I found was the xgboost library in R that implements a gradeient boosted tree algorithm that is able to handle large data. Ultimately I implemented a linear model that assigned the rating of a movie to the mean of all ratings minus the mean of the ratings grouped by user and movie Id. The minimum RMSE achieved by this model on the validation dataset is 0.8648177

Methodology

The movie lens was divided into an `edx` (training) and `validation` (test) respectively. A summary of both dataframes show us what the look like.

```
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
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
summary(validation)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
dim(edx)
```

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

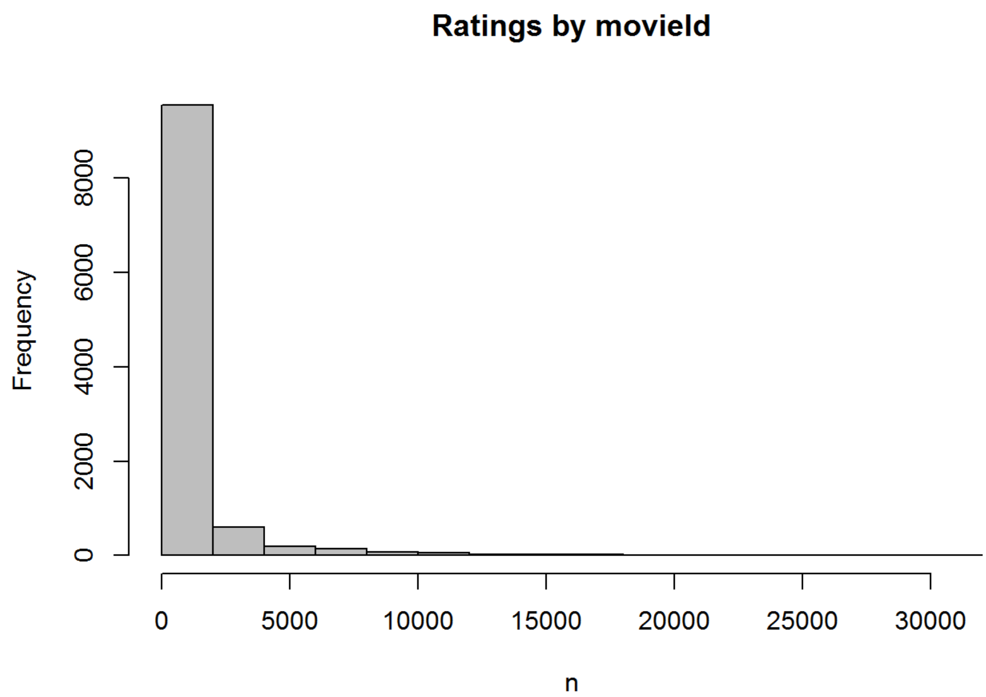
```
dim(validation)
```

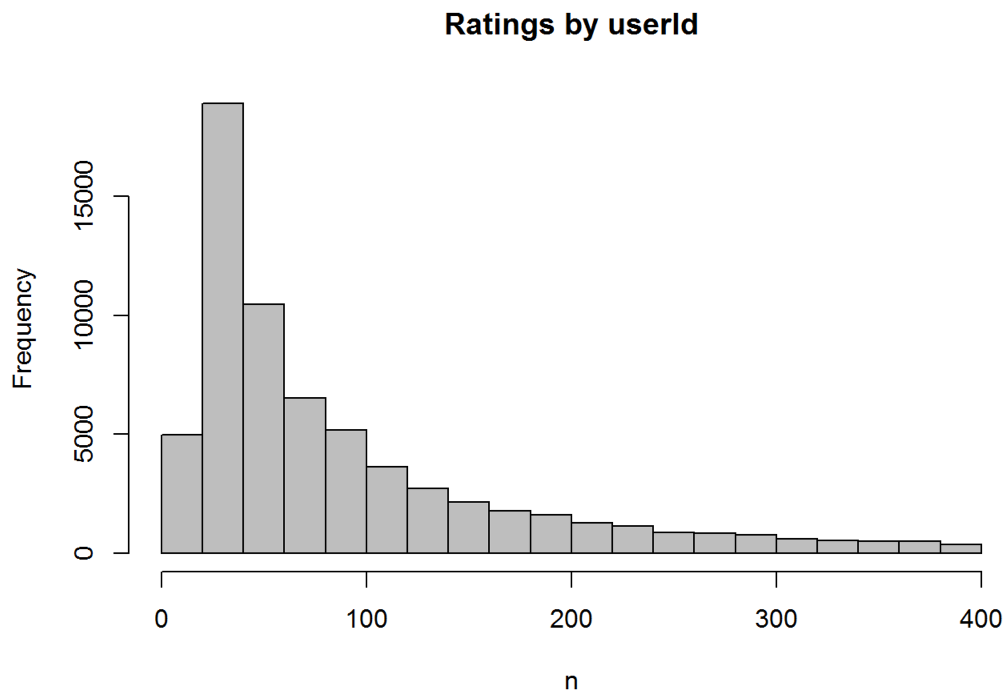
```
## [1] 999999      6
```

Code was added to ensure that movieIds and userIds present in the test dataset were included in the training dataset. As can be seen the validation dataset is roughly about 10% in size of the main dataset.

The size of the dataset also makes it difficult to use standard ml models in Rstudio. Therefore the best approach is to fit a linear model that predicts movie ratings based on movieId and userId and calculating the coefficients manually.

From observing a histogram of ratings by userId and movieId, we see that some movies are rated more than others and some users are more active than others. This motivates the use of regularization towards the model to ensure that movies with few ratings do not artificially show up as high or low rated movies but rather regress towardst the mean.





The model I fit is as follows (ref. Rafael Irizzary ML notes) $Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$ where the last ϵ is a random error term. Each of the parameters above can be estimated from the training set, the μ is the overall mean of all ratings across all users, the b_i is the movie specific mean and b_u the user specific mean.

With regularization, we penalize the estimates of the b_i (and similarly the b_u) by introducing a term λ for movies that have a low number of ratings and could thus be outliers instead of a reliable rating.

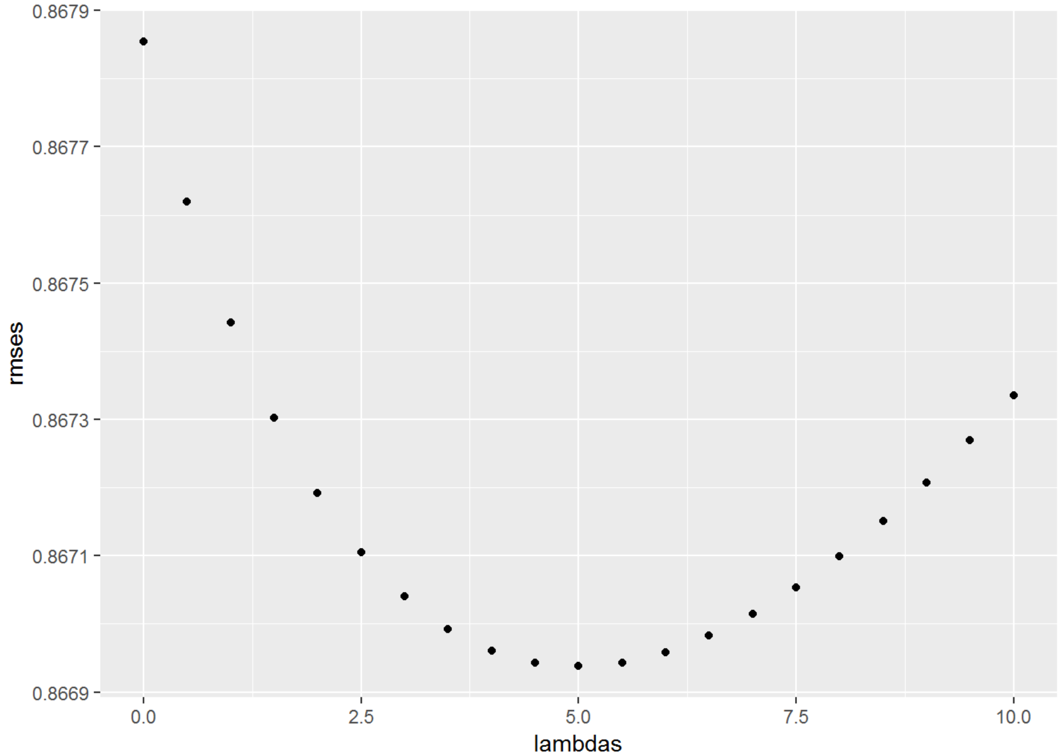
These estimates are calculated the following way: $\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$

To find the best value of lambda that minimizes the expression (ref. Rafael Irizzary ML notes)

1N∑u,i($y_{u,i}$ − b_u − b_i − μ)^2 + λ(∑ ib_{2i} = ∑ ub_{2u})

I further divided the training set `edx` into another training set `train_set` and a test set `test_set` and chose the value of lambda that minimized the RMSE.

```
qplot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
print(lambda)
```

```
## [1] 5
```

This value of 5 was then used to calculate the final model on all of `edx` and the estimates calculated were used to predict the ratings for `validation` and calculate the final RMSE.

Results

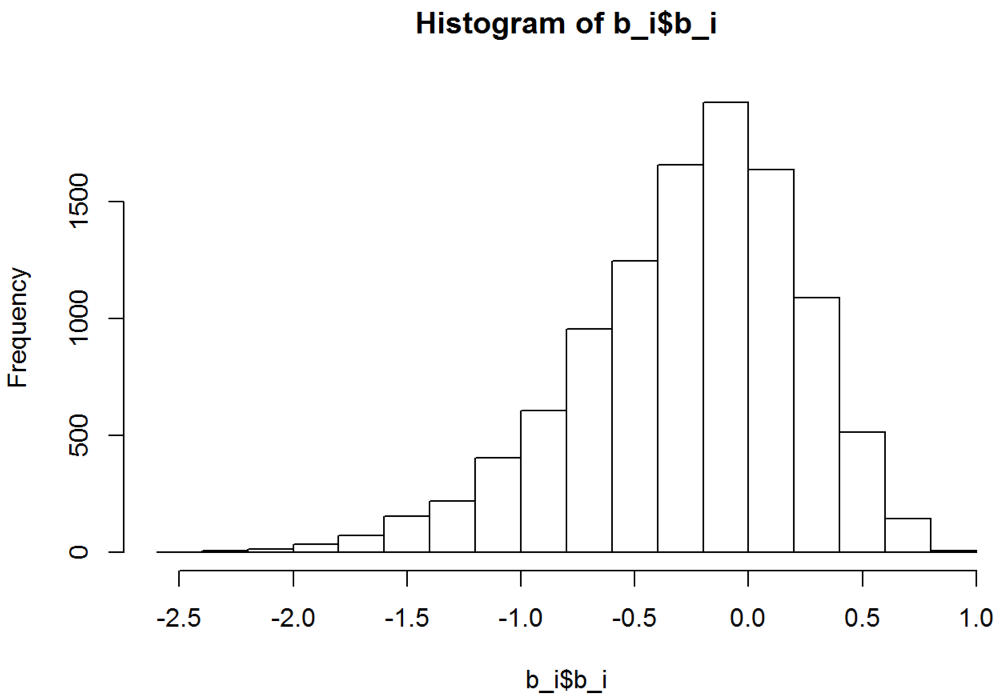
The final estimates were as follows

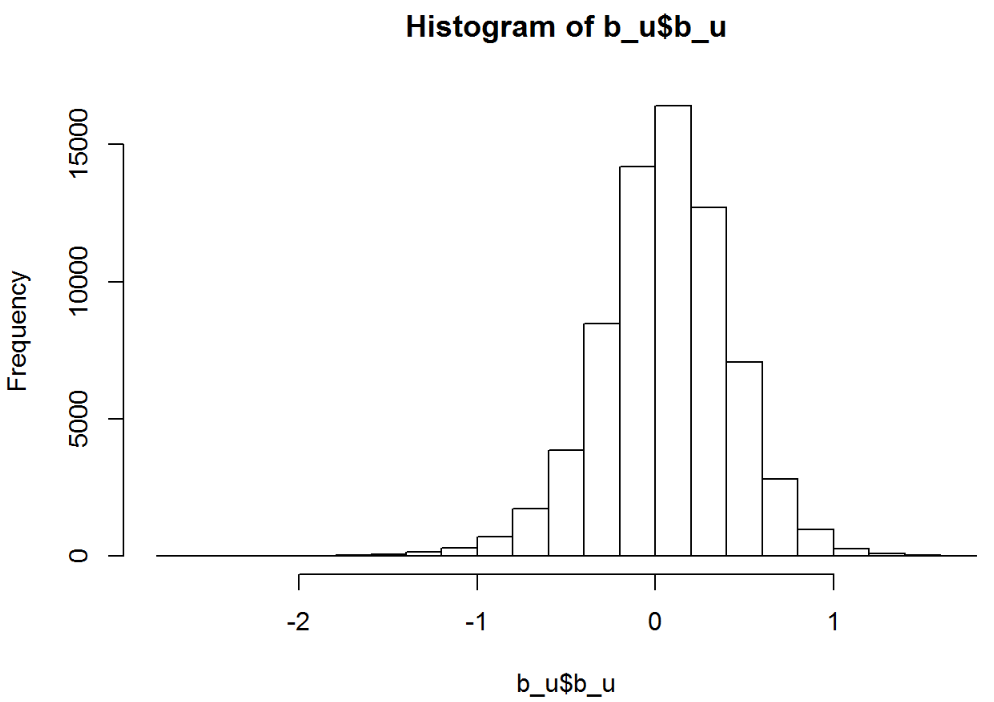
```
## [1] "mu = 3.512"
```

```
## [1] "lambda = 5"
```

```
## [1] "RMSE = 0.86482"
```

b_u and b_i were widely distributed as seen below





These movie and user estimates accordingly adjust the movie rating according to movie and user.

Conclusion

The recommendation model used whlie simple is quite accurate and able to scale well with large data. Other models I used such as xgboost (not shown here) while able to handle the large data weren't able to beat this model when it came to minimizing rmse. Further model enhancements could be done to improve the model by including timestamp and genre effects as well. For example it could be that users became more conservative with their ratings as time passed, or that certain genres are generally better rated than others (is this why very few purely comedic movies win the best film oscar?). Investigations into the effects of modeling these variables may help bring the RMSE of [illegible] the validation set further.