Predictions on Movie Ratings

André Jakob

12/06/2019

Introduction

This report is part of the professional certificate program in Data Science offered by Edx and HarvardX. Since this is evaluated by other students, I would like to ask the evaluators to be kind with my grammatical errors, since english is not my native language. So I thank you in advance, both for the patience and the evaluation itself. The purpose of the exercise is to write a Mashine Learning algorithim that predicts the potential movie rating from the given ratings dataset.

Method

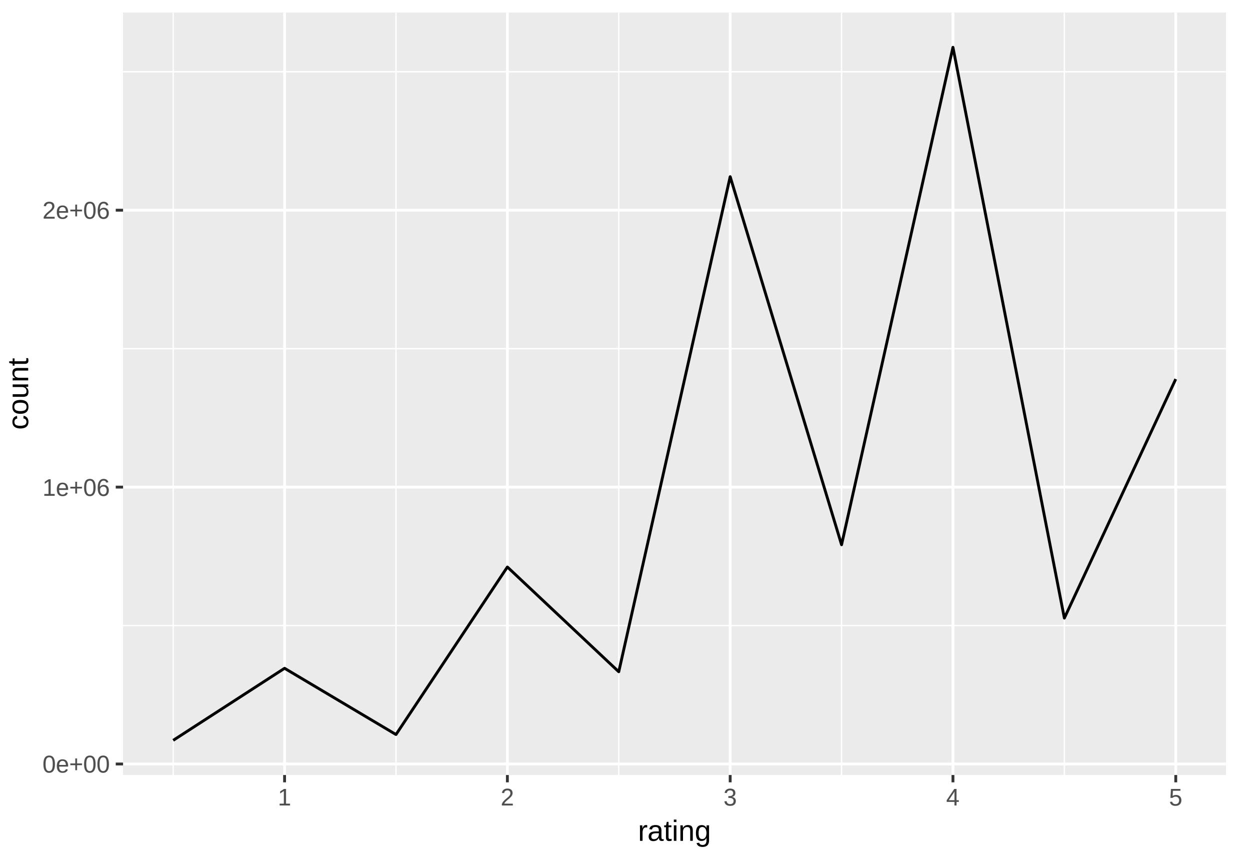
In this chapter I describe the step of the work done, so it has basically the same content as the code writen to acchieve the prediction. Obviously you can read both, or just one of those, what you think is better. Generally I tried to write each step done before the respective code, so if you prefer read the steps right among the codes, please go to the next chapter. We started from the movielens dataset, which has an enormous dimmension of 10,000,054 observations. After some previous analysis on the data to know how it was structured, we followed the lessons of the previous EdX courses, which teached that are some steps to be done to acchieve a good prediction from a great data set as these. First we defined a sample index with 10% of the size of the original data set. This sample was divided between test and training set. Below we see how many users and movies the training set has.

```r
k1<-edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
kable(k1)
```

n_users	n_movies
69878	10677

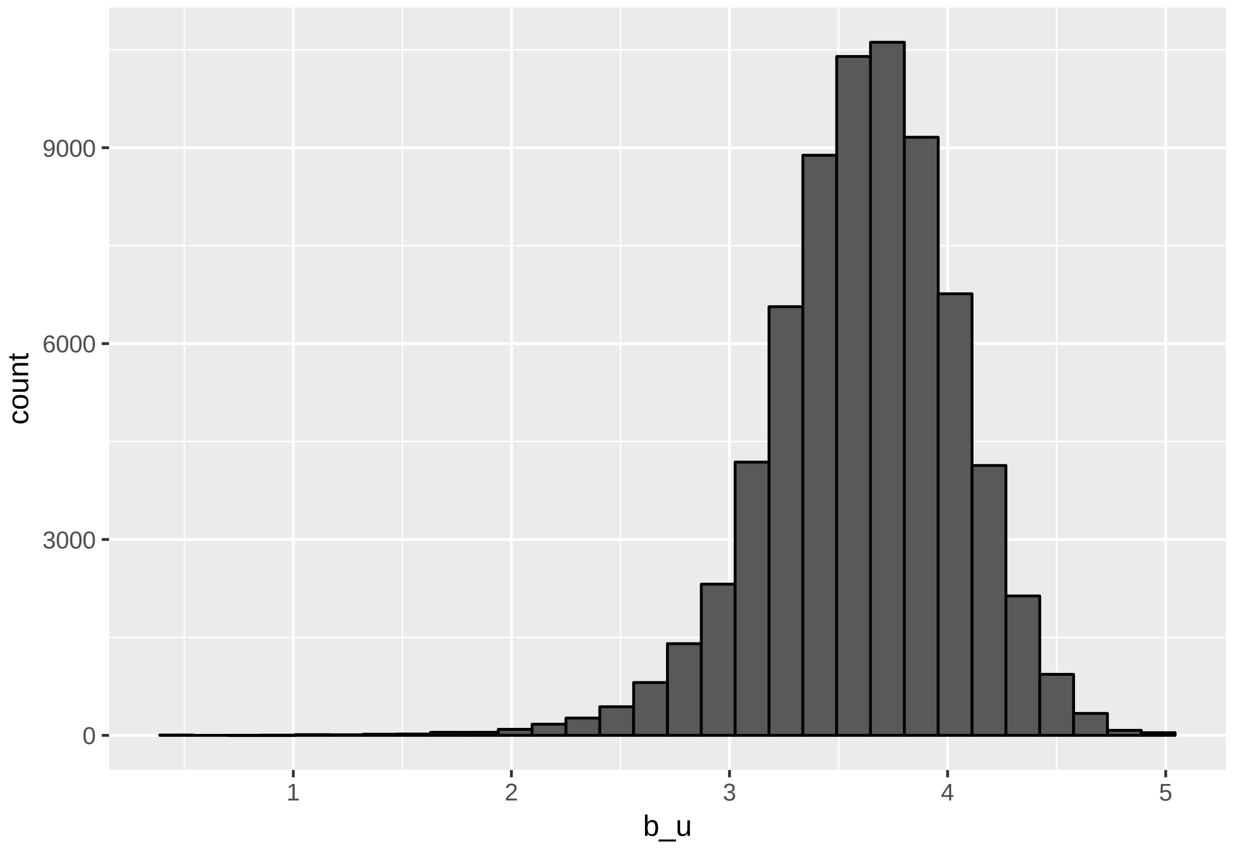
By the way, an important background to this exercise is to know that each user can rate the movies from 0.5 to 5, in 0.5 intervals. Even so, we see in the graphic below that a fractional rating is more rare than a rounded one.

```r
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()
```



The training set was used to train an algorithm and respectively predictions. So those predictions were applied in the test set. The general idea was to start with the simplest possible prediction, and then adding complexions. Each improvement went along with the measurement of its Root-mean-square deviation - a measurement analog as the standard deviation. The simplest possible prediction was using just the average, regardless of the users and movies. As expected,we had a too high RMSE for about to one, which is obviously not enough, since the possible ratings go from 0.5 to 5. Using just the average to predict the ratings, regardless of the criteria, is almost good enough as guessing blindly. The next step was then add our first bias: the movies. First we used the least squares method to estimate this bias. Since the whole data-set is enormous, we therefore estimated this value from our train data set. With one bias the prediction was a little better, not good enough though. So we added another bias for users. Each user rates the movies according to their own criteria. Some users tend to give better ratings while others are more rigorous. Below I show a histogram for the average of stars each user gives. There is a substantial variability among them.

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



For the same reason said above, we could not run a prediction for the whole data set, so we estimated the bias from the training set instead and ran the prediction, achieving a prediction up to 0.865. Even so, we took a look at the greatest mistakes to learn what could be done to make the algorithm even better. We learned that some obscure movies can mess the prediction. So we saw the top and worse 10 movies according to the prediction and how often they were rated. Below both tables.

```r
k2<-movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10)
kable(k2)
```

title	b_i
Hellhounds on My Trail (1999)	1.487535
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487535
Shadows of Forgotten Ancestors (1964)	1.487535
Fighting Elegy (Kenka erejii) (1966)	1.487535
Sun Alley (Sonnenallee) (1999)	1.487535
Blue Light, The (Das Blaue Licht) (1932)	1.487535
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535
Human Condition II, The (Ningen no joken II) (1959)	1.237535
Human Condition III, The (Ningen no joken III) (1961)	1.237535
Constantine's Sword (2007)	1.237535

```r
k3<-movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i) %>%
  slice(1:10)
kable(k3)
```

title	b_i
Besotted (2001)	-3.012465
Hi-Line, The (1999)	-3.012465
Accused (Anklaget) (2005)	-3.012465
Confessions of a Superhero (2007)	-3.012465
War of the Worlds 2: The Next Wave (2008)	-3.012465
SuperBabies: Baby Geniuses 2 (2004)	-2.717822
Hip Hop Witch, Da (2000)	-2.691037
Disaster Movie (2008)	-2.653090
From Justin to Kelly (2003)	-2.610455
Criminals (1996)	-2.512465

To avoid obscure movies influence our prediction, we checked how many ratings those movies became.Below both tables.

```r
#Lets see how often the were rated.
k4<-edx %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10)
```

```
## Joining, by = "movieId"
```

```r
kable(k4)
```

title	b_i	n
Hellhounds on My Trail (1999)	1.487535	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487535	2
Shadows of Forgotten Ancestors (1964)	1.487535	1
Fighting Elegy (Kenka erejii) (1966)	1.487535	1
Sun Alley (Sonnenallee) (1999)	1.487535	1
Blue Light, The (Das Blaue Licht) (1932)	1.487535	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535	4
Human Condition II, The (Ningen no joken II) (1959)	1.237535	4
Human Condition III, The (Ningen no joken III) (1961)	1.237535	4
Constantine's Sword (2007)	1.237535	2

```r
k5<-edx %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10)
```

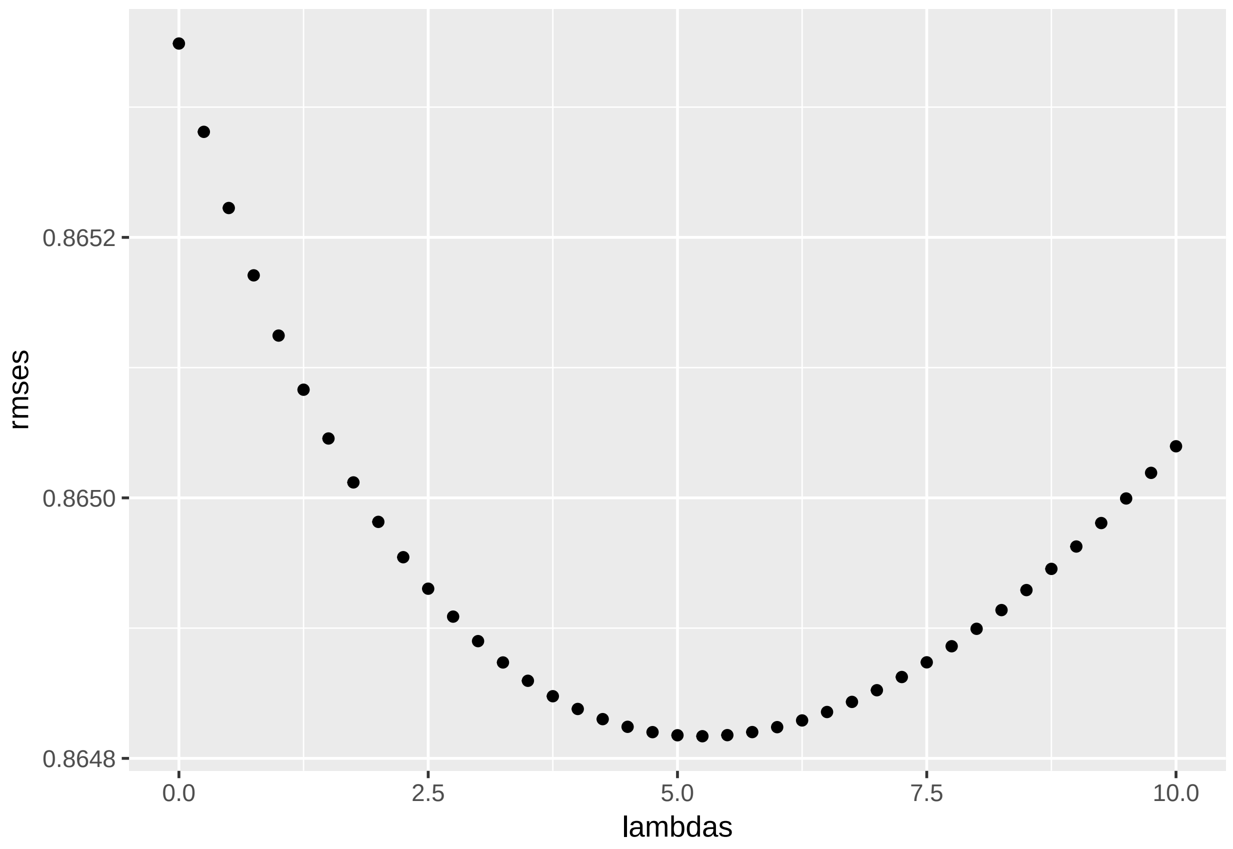
```
## Joining, by = "movieId"
```

```r
kable(k5)
```

title	b_i	n
Besotted (2001)	-3.012465	2
Hi-Line, The (1999)	-3.012465	1
Accused (Anklaget) (2005)	-3.012465	1
Confessions of a Superhero (2007)	-3.012465	1
War of the Worlds 2: The Next Wave (2008)	-3.012465	2
SuperBabies: Baby Geniuses 2 (2004)	-2.717822	56
Hip Hop Witch, Da (2000)	-2.691037	14
Disaster Movie (2008)	-2.653090	32
From Justin to Kelly (2003)	-2.610455	199
Criminals (1996)	-2.512465	2

As expected, most of them have been rated just a few times. To avoid this, it is common to use a method of regularization of the data. In this case a proper way is to penalize some of the ratings. The penalty therm is called lambda and before using it we needed to calculate the better value using cross validation. Below the plotted lambdas tested.

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



We founded the 5.25 as the better value and then ran the prediction using this lambda. We compared the last method with the other calculations and, as the table shows, after predicting the regularized Movie and user effect model we got the minimum RMSE of 0.86.

RMSE

The table below gives the RMSE founded using different complexions of calculations. The prediction with regularized movie and user effect model has got the minimum RMSE of 0.8648170.

```r
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438805
Regularized Movie + User Effect Model	0.8648170