MovieLens

Mauro Berlanda

4/5/2020

Introduction/Overview/Executive Summary

The goal of this project is to develop an algorithm to predict movie ratings.

This analysis is performed on (MovieLens) dataset. It contains 10 millions ratings and 100 000 tags applications applied to 10 000 movies by 72 000 users.

After downloading the entire dataset, two dataframes are created:

- `edx` which is used to perform the analysis and develop the algorithm to predict ratings
- `validation` which contains the true values of the predictions

```
dim(edx)
```

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

```
dim(validation)
```

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

```
colnames(edx)
```

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

The key steps required to identify an optimal solution are:

- todo
- create a train and a validation set
- explore the content of the train set
- another step

Method/Analysis

The process of this analysis includes data wrangling, data exploration and the description of the modeling approach.

Data Wrangling/Cleaning

The `edx` and `validation` are the result of some manipulation on a downloaded dataset from `grouplens.org`.

1. download the zip file in a temporary directory and unzip it into the workspace directory
2. read the raw data from two `.dat` files creating `ratings` and `movies` dataframes
3. normalize column names and cast value as `numeric` and `character`, then join both datasets into the `movielens` dataframe
4. partition the data in `movielens` to create a train test used for developing the model (`edx`, 90%) and the true values used to calculate the RMSE of the final model (`validation`, 10%)
5. ensure that userId and movieId in validation set are also in edx set

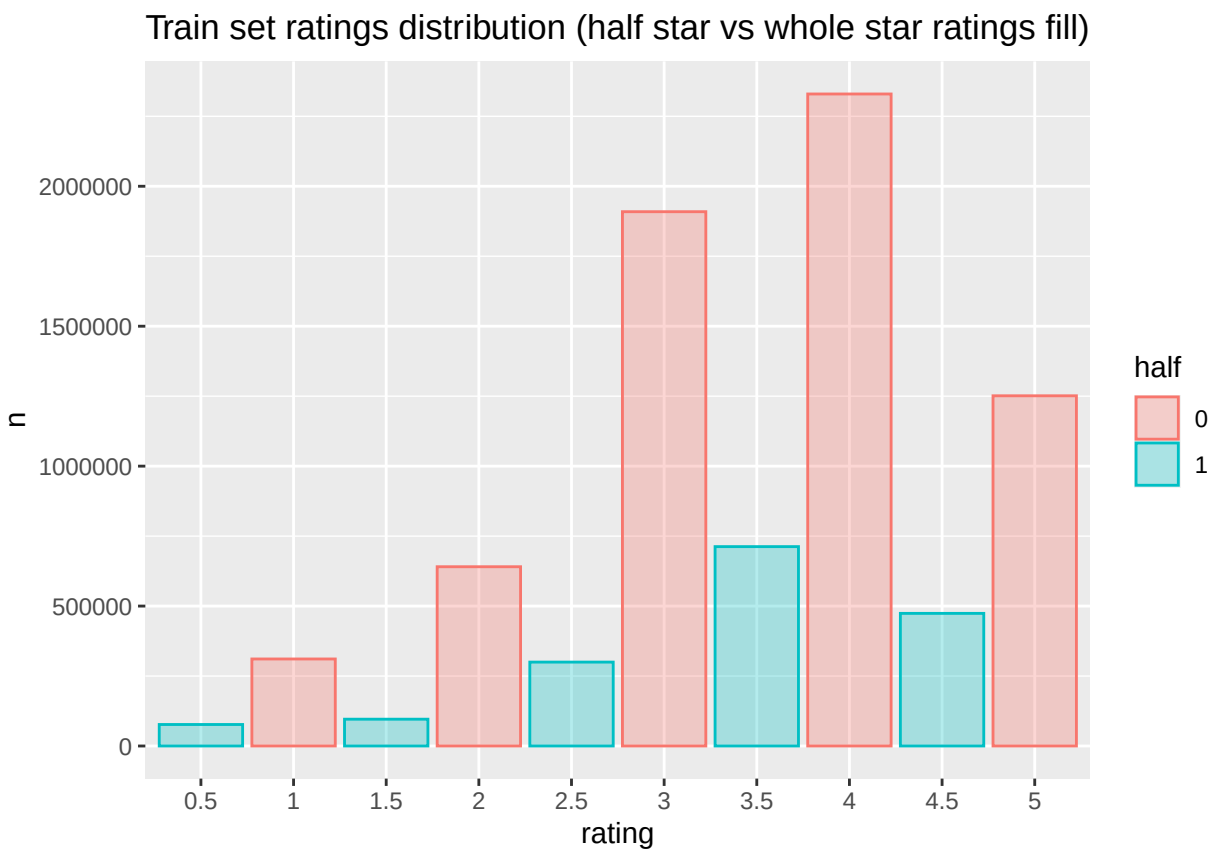
Instead of using all the known observations from `edx` dataset, we create a `train_set` and a `test_set` to continue this analysis as follows:

```
set.seed(17, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# Ensure that userId and movieId in the test_set are included in the train_set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
```

We perform our analysis unsing `train_set` and we validate it with `test_set`.

Data Exploration

Since our goal is to predict ratings, we start looking at the overall ratings distribution in the `train_set` dataset.




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.513   4.000   5.000
```

We notice that whole star ratings are predominant compares to half star rating.

The features we may use in our analysis:

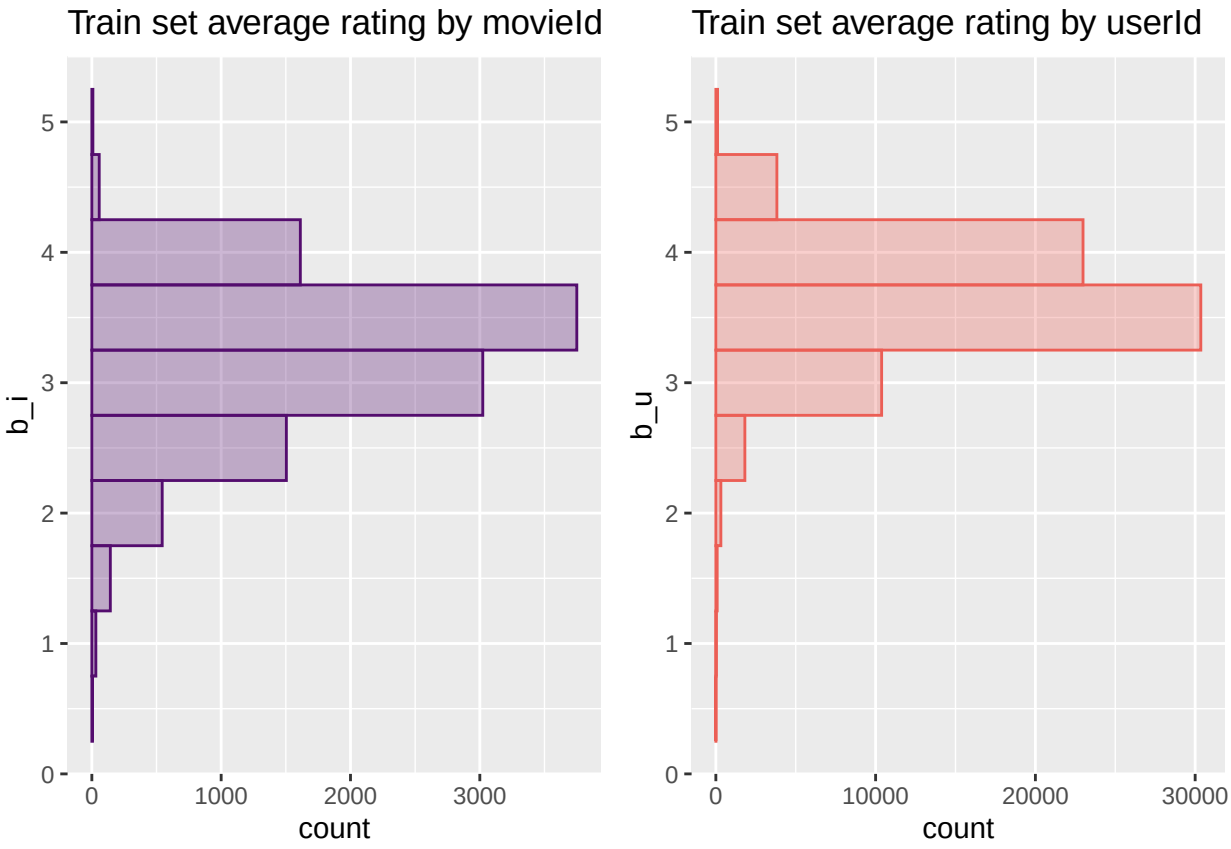
```
setdiff(colnames(edx), c("rating", "title"))
```

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

The ratings breakdown by top 10 `genres` is:

genres	count
Drama	3518593
Comedy	3187624
Action	2304788
Thriller	2093364
Adventure	1718289
Romance	1540803
Sci-Fi	1206731
Crime	1194688
Fantasy	833106
Children	664169

Comparing the distribution of rating by `movieId` and by `userId`



Investigating patterns based on timestamp seems to be too much expensive for my machine CPU. If the analysis will require it, it may be interesting to analyse some patterns like: year, week of year, day of year, day of week

Out of curiosity, the 10 most rated movies are:

movieId	title	count	avg_rating
296	Pulp Fiction (1994)	28201	4.158611
356	Forrest Gump (1994)	27912	4.014528
593	Silence of the Lambs, The (1991)	27386	4.202585
480	Jurassic Park (1993)	26434	3.662821
318	Shawshank Redemption, The (1994)	25207	4.452731
110	Braveheart (1995)	23631	4.081715
589	Terminator 2: Judgment Day (1991)	23423	3.930432
457	Fugitive, The (1993)	23390	4.008743
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	23075	4.221105
150	Apollo 13 (1995)	21842	3.886205

Results

Conclusion