Capstone_Report

David Brandtner

2/21/2022

Capstone: MovieLens rating prediction model

1. Introduction

The aim of the searched prediction model is to estimate the rating of a movie in order to make an user specific recommendation.

The *MovieLens dataset* has a huge row dimension reporting more than 10 million movie ratings made by more than 70 thousands unique users that have rated more than 10 thousands unique movies. Thus not every user has rated every movie and this make the dataset sparseness actually high, making the definition of useful predictors the hard part.

In addition to *rating*, the column containing values to estimate, the dataset is characterized by *movieId, userId, timestamp* (when rating has been made), *title* and *genres* columns. Each of them could be utilized as predictor or predictor source after a preprocessing step.

2. Analysis

2.1 Dataset train/test and validation splitting

MovieLens dataset has been splitted into an ultimate test set called *validation set* (counting 10% of original dataset) and a dataset called *edx* (counting for 90% of original dataset).

After a preprocessing step on *edx*, this is renamed as *edx_pr*, that it has been further partitioned in a train set called *edx_pr.train_set* (90% of *edx_pr*) and a test set called *edx_pr.test_set* (10% of *edx_pr*).

Same preprocessing step has been executed coherently on *validation set*, thus renaming it *validation_pr*.

validation_pr set has only an RMSE final estimate purpose, while model training and testing has to be operated on *edx_pr.train_set* and *edx_pr.test_set*.

2.2 Data Exploration and predictors hypothesis

A good starting point is to look at the distribution of the values to predict: *rating*.

```
summary(edx$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

It has a left skewed distribution with values ranging from 0.5 to 5. Its mean value is 3.512.

From a *MoviLens dataset* first compact display *title* has an interesting source of information, reporting along the movie title the year of its theatrical release.

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

With a string preprocessing step this information is extracted and a new *movieYr* column is created (*new preprocessed datasets created see § 2.1*).

```
str(edx_pr)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  7 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  $ movieYr  : num  1992 1995 1995 1994 1994 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

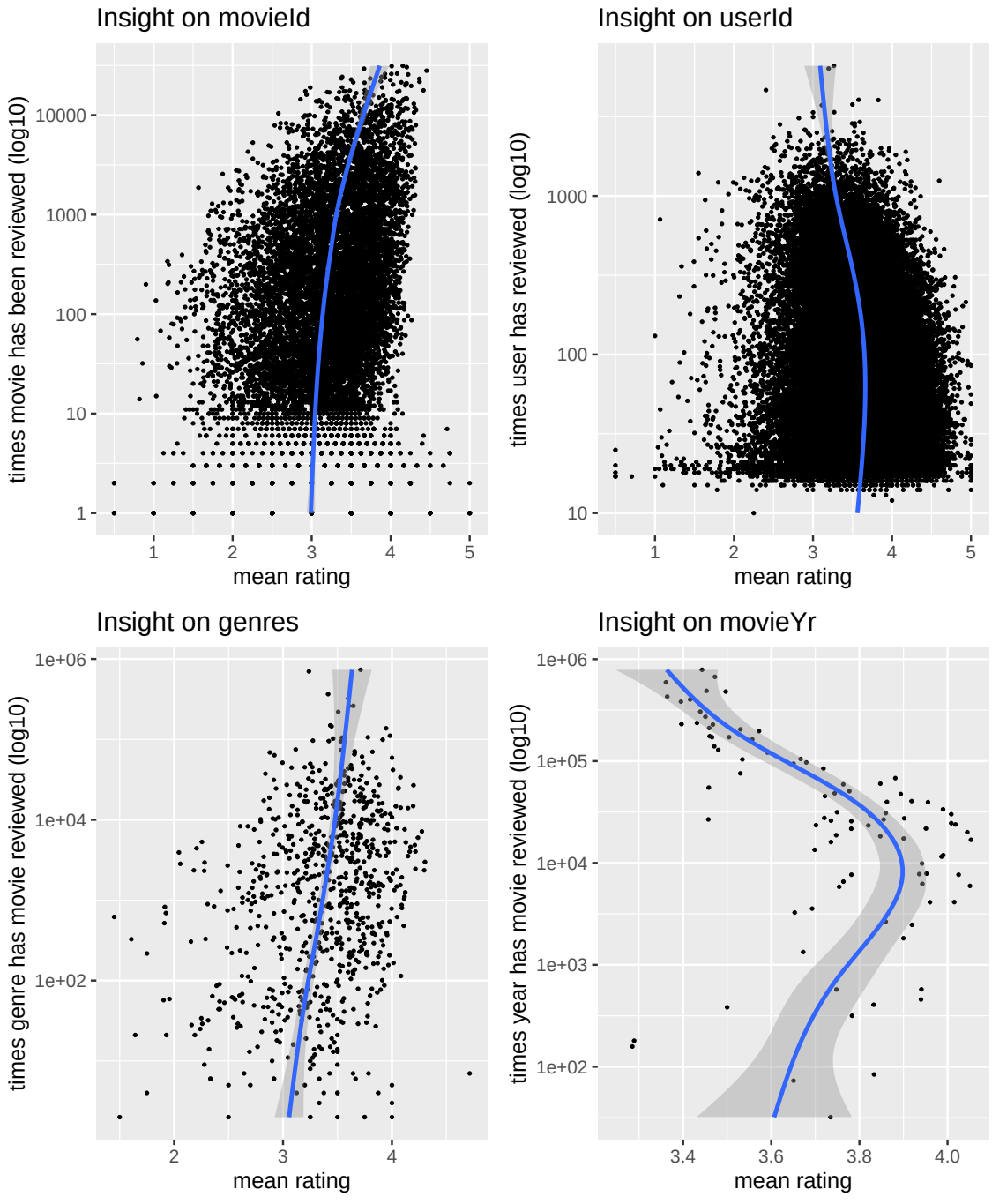
At this point for this analysis the hypothetical predictors that could influence rating values are the movie itself, the user taste, the genre and the year of theatrical release (*movieId, userId, genres, movieYr*). In attempt to obtain an insight of their behavior a mean rating for each distinct component of these predictors is calculated (mean rating grouping by a key) and plotted against their frequency count in four scatter plots (*see Fig at pag. 3*).

From this panel of plots it could be seen that there are movies and genres rated more than others, release years in which ratings are more frequent than others and there are users that rate more than others. In general the range in which frequencies span is between some orders of magnitude. Regarding movies and genres mean rating it is observed their increase with frequency – in this case actual "popularity". Instead for users and release year mean rating is observed to decrease with frequency. **The standard deviation of mean rating values is higher at low frequency** in all four scatterplot, while it has a reduction at higher frequency values. The low frequency areas contains therefore less reliable mean rating values. Mean rating values are in general left skewed according to the distribution of *rating*.

As results of this data insights it is advisable to preprocess as described above *movieId, userId, genres, and movieYr* since they still capture a pattern in the data and ameliorates heavy calculus burden linked to dataset sparseness.

```
plot_movieId + plot_userId + plot_genres + plot_movieYr
```

```
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
```



2.3 Modeling approach

The algorithm at first taken in consideration to make prediction of the outcome values is linear regression. Since two predictors count for over ten thousands unique items (*movieId, userId*) use of *lm()* function of R is deprecated to not occur in run time and/or memory allocation issue on an average Desktop PC.

In this condition a simplified linear model is chosen. It is composed by μ a constant value that represents the average of all outcome values y and four effect terms called b, for correction over μ, one for each predictor taken in consideration.

$$\mathbf{Y_{u,i}} = \mu + \underbrace{b_{1i}}_{\textbf{movie effect}} + \underbrace{b_{2i}}_{\textbf{user effect}} + \underbrace{b_{3i}}_{\textbf{genre effect}} + \underbrace{b_{4i}}_{\textbf{year effect}} + \epsilon_{\mathbf{u,i}}$$

where : $b_{ni} = \frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_{(n-1)i})$

When the sample size N is small the resulting average is endowed of a larger error than big sample size (an average movie rating based on thousands of user review has less uncertainty into respect of one rated by few users - according to the low frequency area of the above scatterplots). In order to reduce uncertainty into the model a penalty factor λ is added to improve fitness, operating a regularization step.

Effect term regularized : $b_{ni}(\lambda) = \frac{1}{\lambda + n_i}\sum_{u,i}(y_{u,i} - \mu - b_{(n-1)i})$

Penalty factor λ is chosen empirically using a tuning grid.

3. Results

The linear model with correction effects improves accuracy each *Effect* added. *Movie Effect* gives the greater gain in RMSE reduction, starting from the simple rating average, which is the base of the model. *User Effect* gives a good gain in RMSE too, and *Genres Effect* and *Year Effect* substantially carry the equivalent small improvement.

On the best performing model formulation, "*Movie + User + Genres + Year Effect Model*" with all four effects, regularization has been then calculated (penalty factor *lambda* = 4.5) showing a significant contribution on RMSE further reduction.

Model_Names	Model_Id	lambda	Dataset	RMSE
Just the average	model0	NA	edx_pr.test_set	1.06005
Movie Effect Model	model1	NA	edx_pr.test_set	0.94296
Movie + User Effect Model	model2	NA	edx_pr.test_set	0.86468
Movie + User + Genres Effect Model	model3	NA	edx_pr.test_set	0.86432
Movie + User + Genres + Year Effect Model	model4	NA	edx_pr.test_set	0.86413
Movie + User + Genres + Year Effect Model + Reg	model4_reg	4.5	edx_pr.test_set	0.86365
Movie + User + Genres + Year Effect Model + Reg	model4_reg	4.5	validation_pr	0.86470

At last "*Movie + User + Genres + Year Effect Model + Reg*" is used to make prediction of ratings on *validation_pr* set to asses its validity.

Final **RMSE is *0.86470***.

It is observable that RMSE has a pretty higher value in *validation_pr* set than the RMSE calculated on *edx_pr.test_set*, that is coherent to an *overfitting* on the data of test set.

4. Conclusion

After the assessment of prediction model aim (user specific recommendation) and *MovieLens* dataset critical issue (huge row dimension, sparseness, subsequent Desktop PC specs limit) an approach based on a simplified linear model with correction Effects has been adopted. Predictors have been created using mean of multiple items after grouping by key.

MovieLens dataset has been splitted in train set and a validation set. The train set has been further splitted in a proper train and test sets pair, leaving *validation set* only for the final RMSE calculation, that is $\boldsymbol{RMSE = 0.86470}$.

Model running on test set report a better RMSE value, indicating an *overfitting* on testing data. To limit this issue extent an adoption of a cross validation technique have to be integrated in the future.

Model improvement could be reached working on predictors too. Like future perspectives it is worth to try a simplification in basic genres of *genres*, and investigate if there is a time pattern that influences user ratings analyzing *timestamp*.