Machine learning algorithms for datasets popularity prediction

Kipras Kančys and Valentin Kuznetsov

Abstract. This report represents continued study where ML algorithms were used to predict databases popularity. Three topics were covered. First of all, there was a discrepancy between old and new meta-data collection procedures, so a reason for that had to be found. Secondly, different parameters were analysed and dropped to make algorithms perform better. And third, it was decided to move modelling part on *Spark*.

Keywords. Machine learning algorithms, classification problem, parameter selection, correlation matrix, Spark, pyspark.

1 Introduction

CMS experiment at CERN constantly produces huge amount of data. Latter on the data is analysed all across the globe by many scientists. Some of datasets are copied more than once to be more accessible. This study is trying to predict dataset popularity that latter could lead to making more efficient data placement at CERN CMS experiment. Machine learning algorithms are used on historical usages of datasets and various meta-data information.

2 Earlier work

Two ways of meta-data collection process were implemented. One using *Python* scripts (old-data). A bit latter another way of meta-data collection was written in *Go* (new-data) that significantly speeds up the data collection process. However the results given by both of these methods differs a lot. So first task of this study was to find the reason for that. Multiple scripts for data transformation, preparation and prediction were already written. That implemented more than one ML algorithm. One of those scripts was perform rolling prediction (previous weeks data were used to predict next week datasets popularity). That script was analysed in this study.

3 Analysis

3.1 Discrepancy between old and new-data

Histograms were drawn for every parameter of both old and new-data. After investigation conclusion were drawn that all parameters belonging to relN_...

(number of releases associated with given dataset) group are biased. Instead of calculating rel1_1 parameter for each dataset, it was accumulating rel1_1 value as a sum of all releases for all datasets. Below there are two histograms of parameter rel1_1 taken from old and new-data. Similar distributions can be observed for each relN_... parameter.

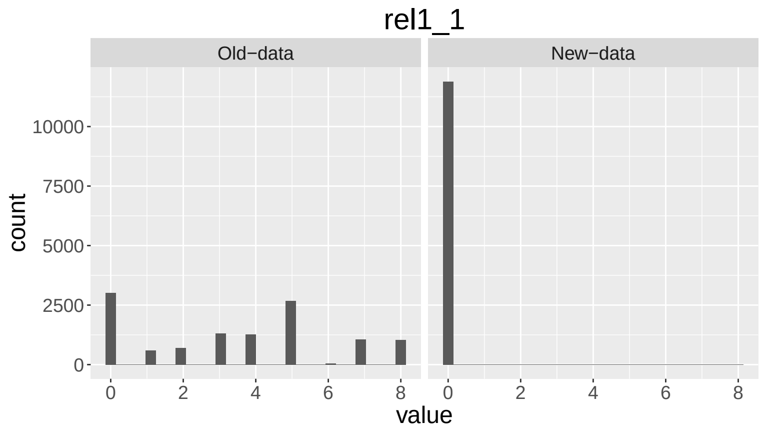

Fig. 1. Histogram of old and new-data rel1-1 parameter

Results before the bug and after a fix can be found in appendix A. Attention should be brought to TPR values in old and new-data.

After the bug was fixed there was no need to continue using old-data as the result of algorithms using old and new-data were similar. While data collection using *Go* is much faster.

3.2 Selection of parameters

There are around 100 parameters describing each dataset. Some of them are highly correlated (numberOfFiles and size) or simply uninformative (like dataset id). So to get the best results out of our models some parameters had to be dropped. Results before and after dropping parameters can be found in appendix C.

3.3 Spark

It was noticed that once a huge amount of data (6, 9 or 12 months data) is used the rolling script could take up to a day or two. So it was decided to move modelling part on *Spark*. Code was written to run rolling prediction in *Spark*. Three classifiers were available in pyspark library - Random Forest, Decision Tree and GBT classifier. Also different metrics were used to measure model accuracy - are under ROC and PR curves. Results can be found in appendix E. Even though it looks like running code on *Spark* takes more time, it should not be forgotten that *Spark* will perform much better in comparison with single machine when more data will be used.

4 Conclusions

The main outcomes of this study are:

- The bug in new-data collection procedure was found.
- Optimal set of parameters were found for roll script.
- The code to run rolling prediction on *Spark* was written.

A Results before and after the bug in roll script

The difference between old and new-data are very well visible in left graphs (TPR). After the bug fix old and new-data perform similar as it should.

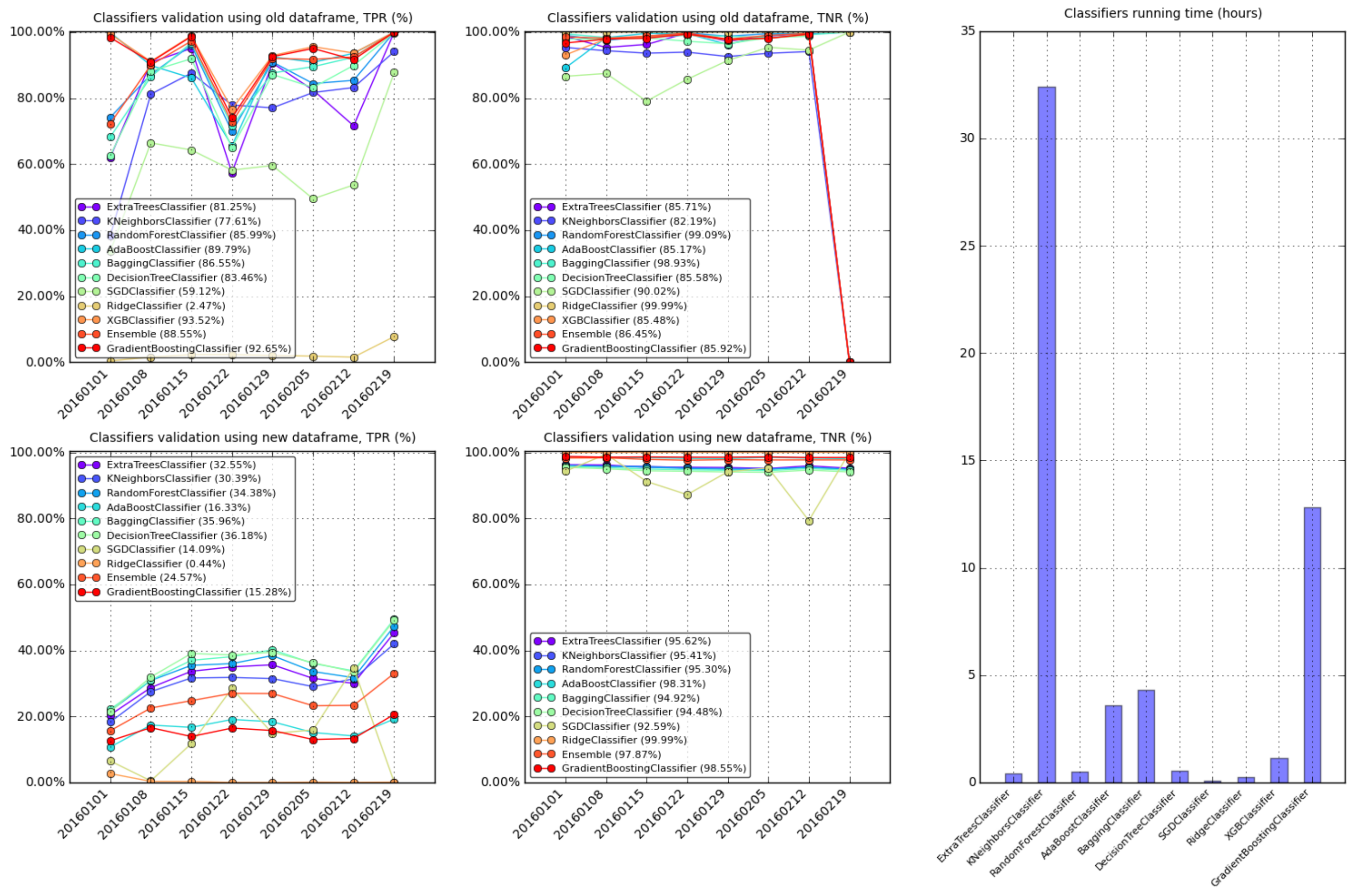

Fig. 2. Roll script results with data before the bug fix

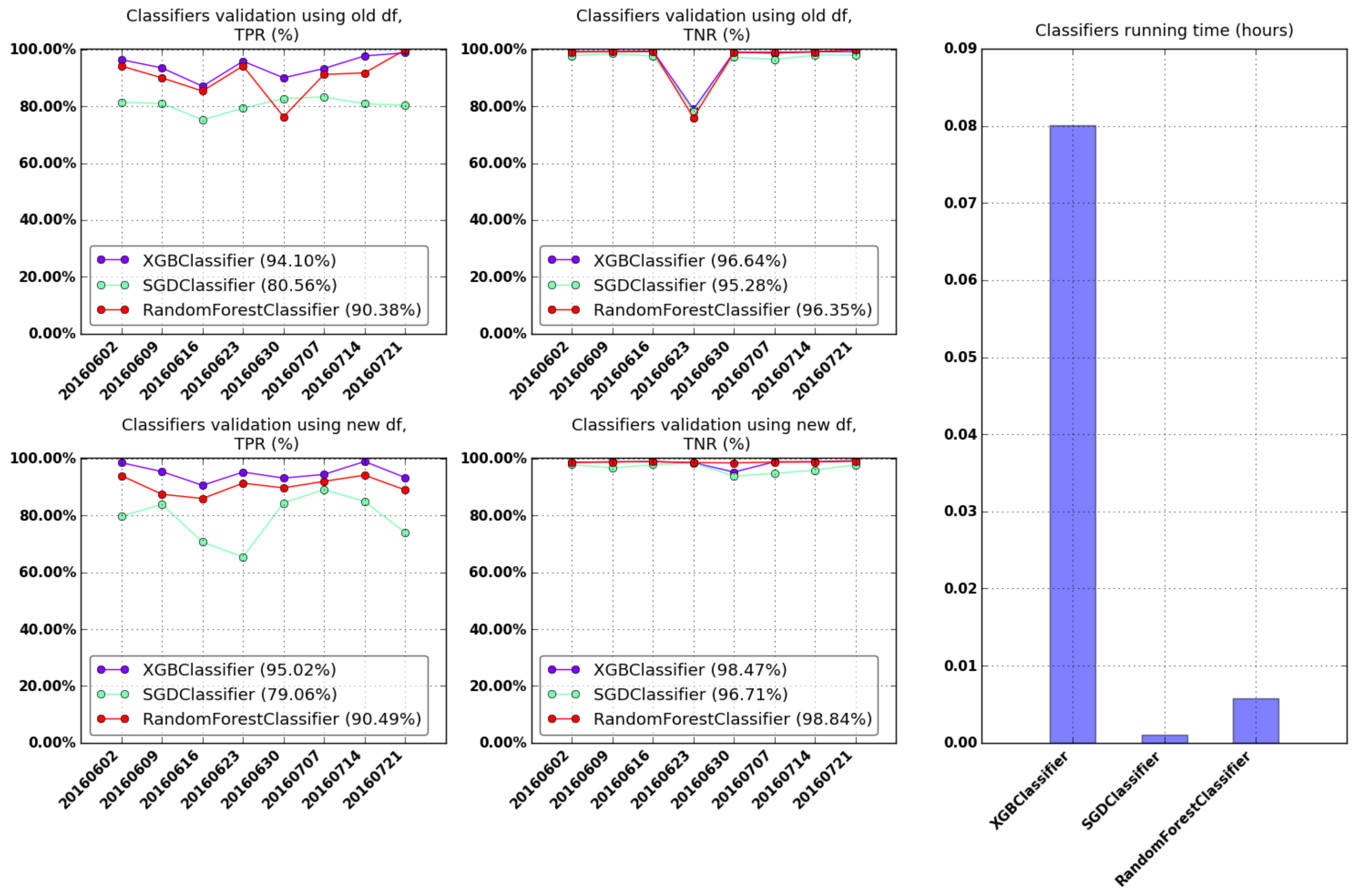

Fig. 3. Roll script results with data after the bug fix

B Correlation matrix

Parameters that are correlated more than 0.6 are show in the matrix.

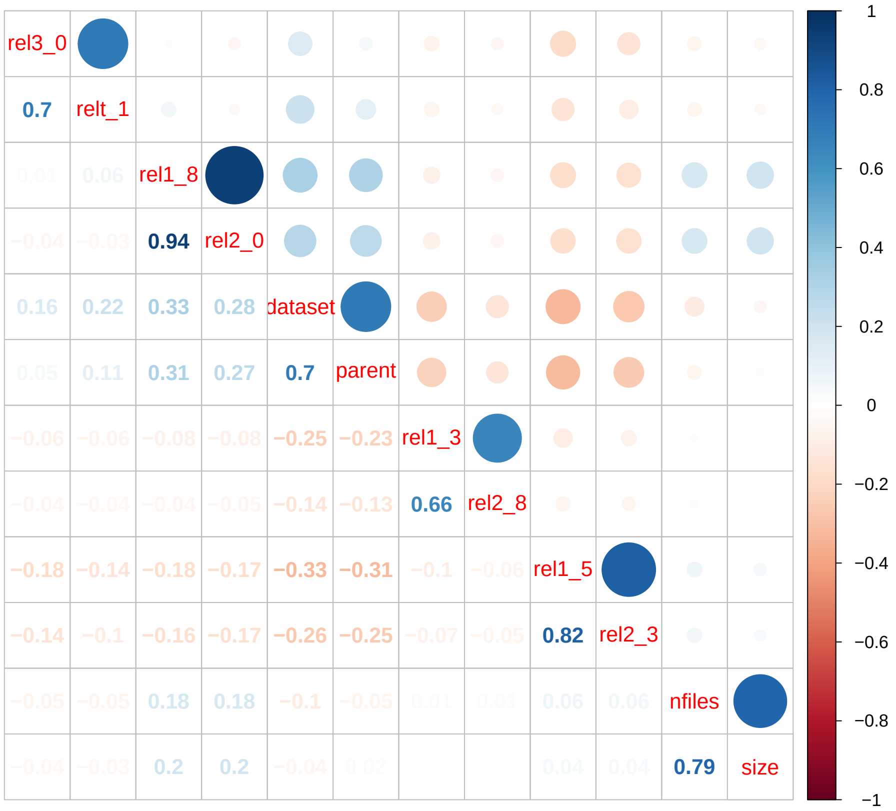

Fig. 4. Correlation matrix

C Results before dropping parameters

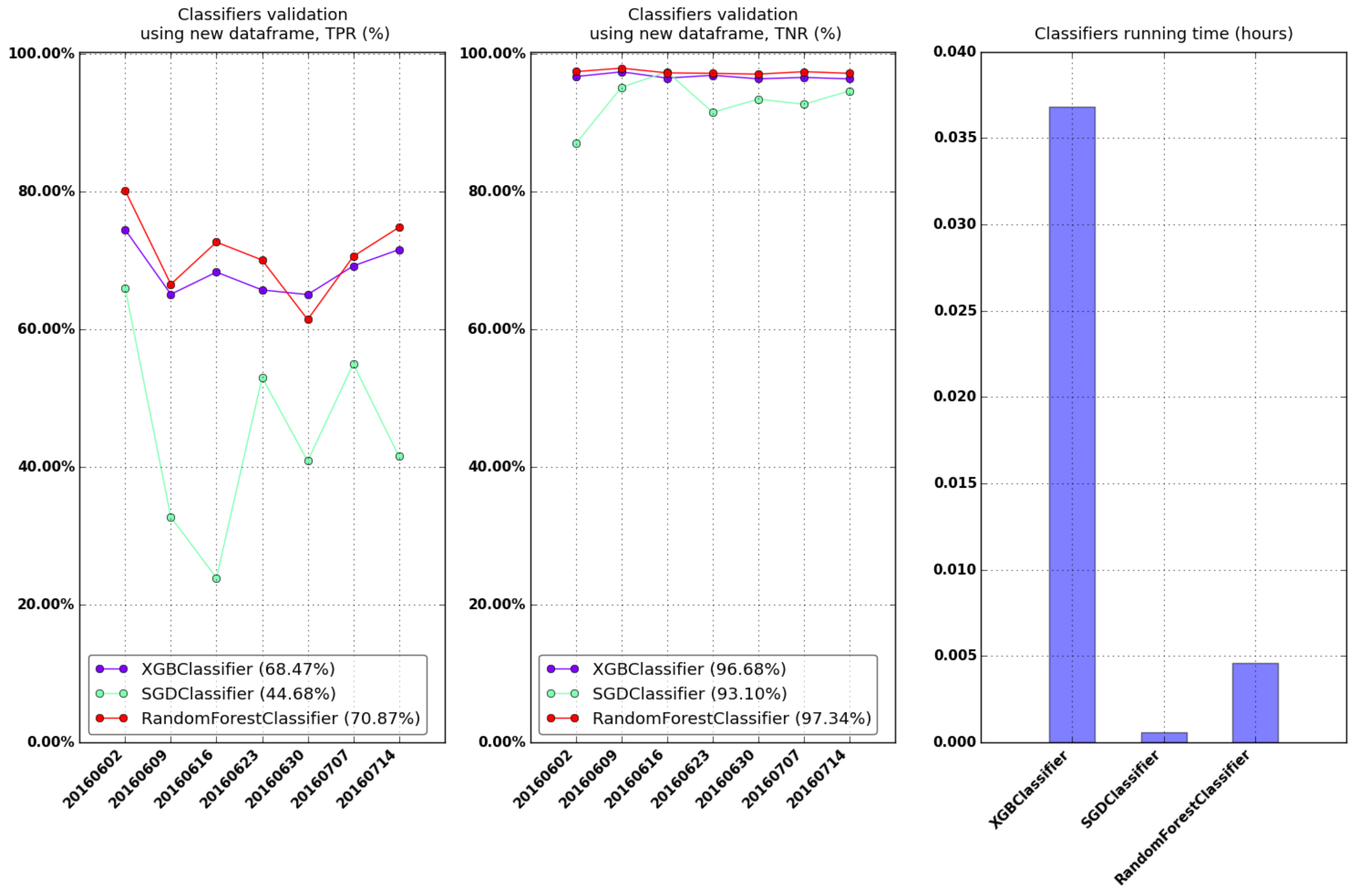

Fig. 5. Roll script results without drops

D Results after dropping unnecessary parameters

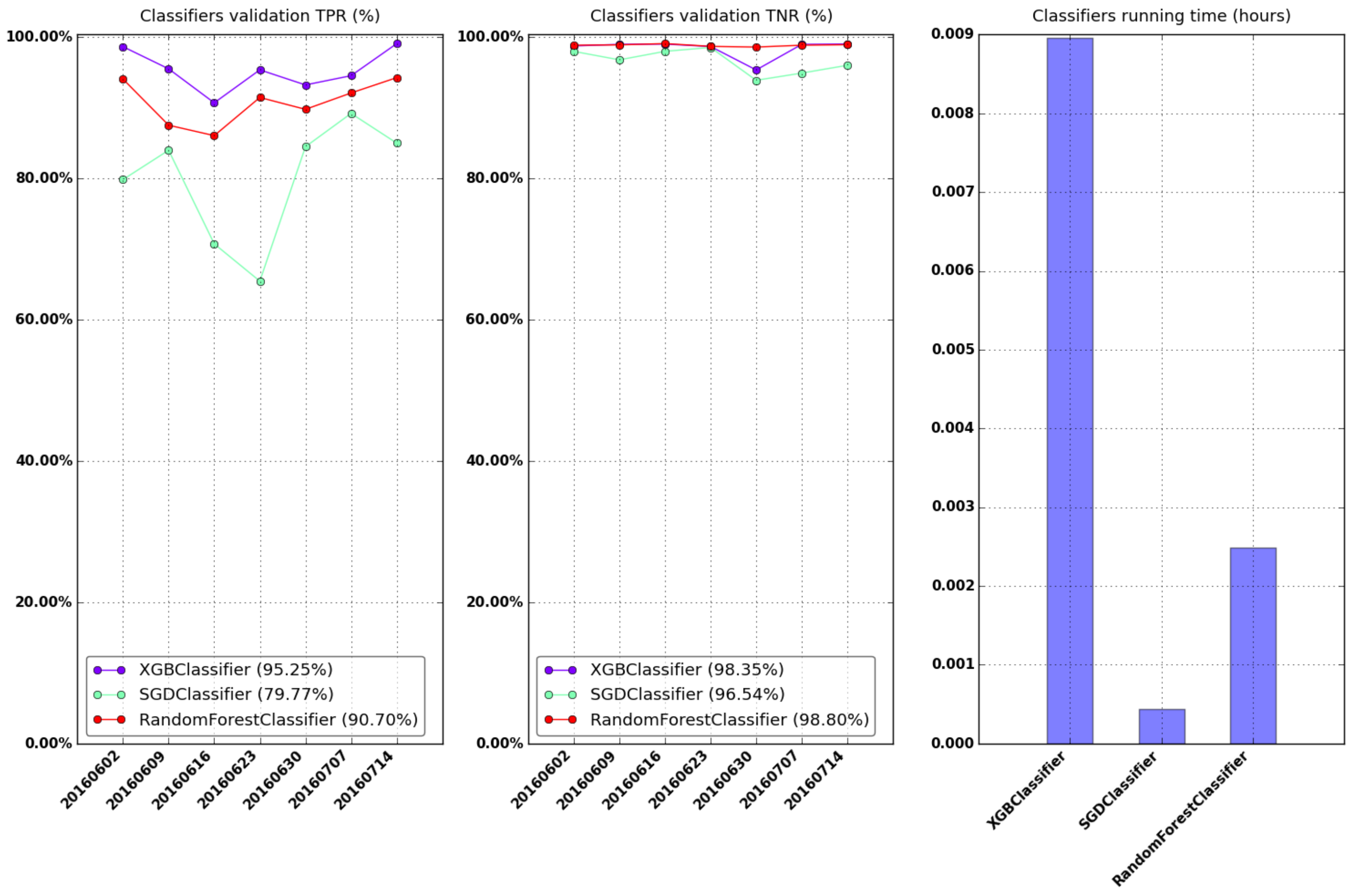

Fig. 6. Roll script results after drops

E Spark results

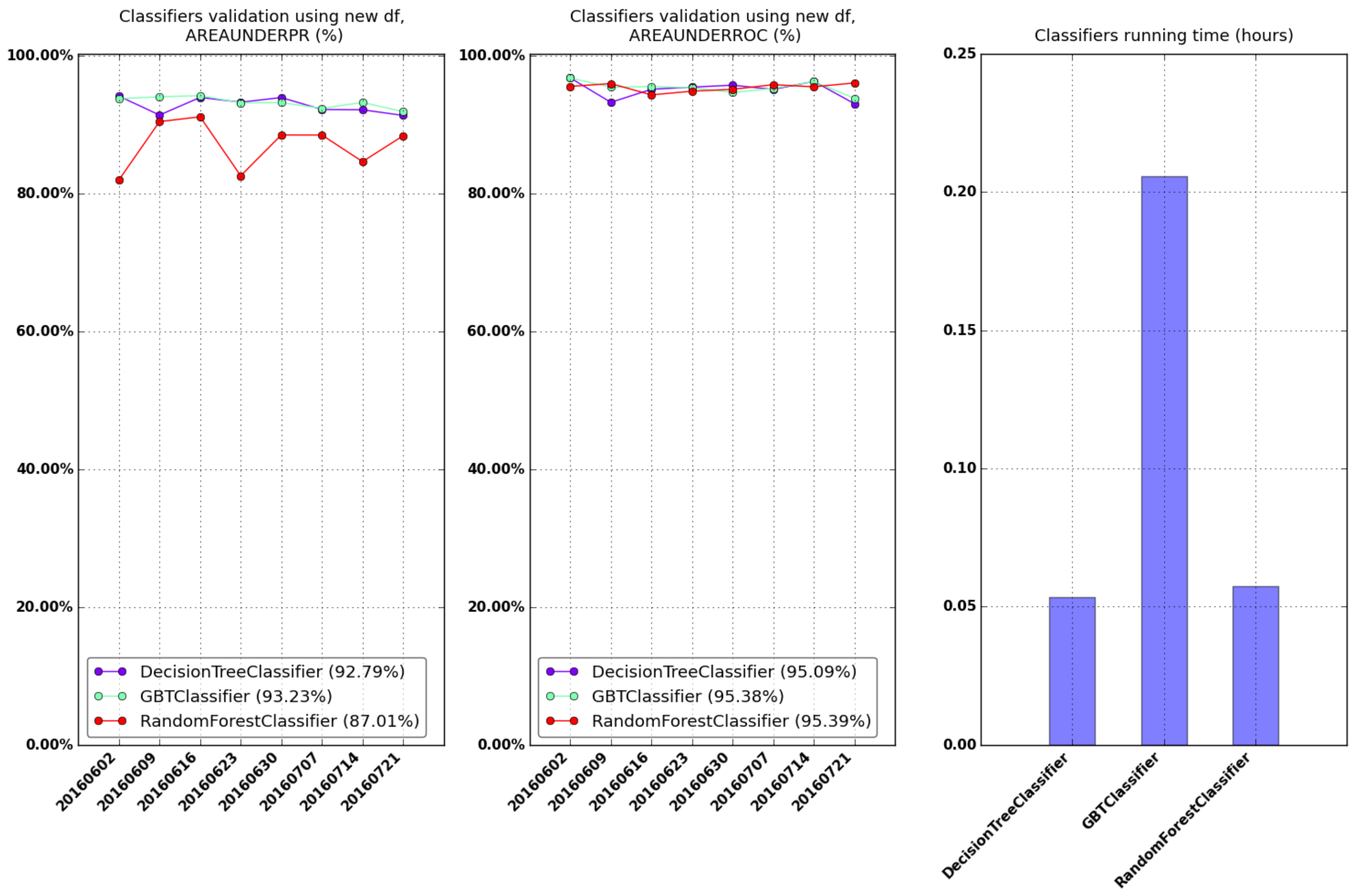

Fig. 7. Spark results