AVRO CASE STUDY

October 8th, 2020

Nico Biagioli

Summary

The analysis of Avro-issues dataset showed data affected by highly positive distribution skewness of both independent and depended variables. No feature seems to individually impact the target in a significant way. The unbalance and the presence of several outliers also characterize the data. After an investigation of the meaning and relevance of the available time variables, they were excluded from the dataset along with other features. The first approach was to use a Random Forest Regressor, mainly because of its ability to deal with skewed and high features dataset, which was combined with an accurate validation strategy able to cope with lack of data and unbalance. Accuracy results are very poor, not improving the simplest target mean benchmark. A trivial resampling to solve unbalance was performed. It improved the coefficient of determination but not the absolute error, and needs further understanding and consolidation. Additional features were extracted from the available json file, but did not enhance the model. Further improvements and mitigations were identified and proposed. Time did not allow their implementation.

Content:

Paragraphs	**Page**	**Assignment answers**
1. Introduction and business problem	1	
2. Data understanding and pre-processing	2	
3. Exploratory data analysis	3	
4. Model and results	4	Answers 1 and 2
5. Additional features	6	Answers 3 and 4
6. Conclusion and next steps	8	

1. Introduction

The present work analyses a dataset containing records of software development issues related to the Apache Avro environment. The collection of items refers to different types of alarms and incidents and relative solution tracking. The scope of the analysis is to be able to predict the resolution time given an alert with certain characteristics. The expected benefit of such prediction capability for the software management business would probably translate in improvement of the incident tracking, update deployments and overall faster resolutions.

The software used for the following analysis is Python, run on the Jupyter notebook platform. Scikit learn library was used for most of the functions and algorithms. This choice is due to my knowledge and experience with Python as main programming language and because it offers a large choice of updated statistical and machine learning libraries. It became in fact one of the most widely used program for this kind of purpose (and not only).

The Python notebook is included in the delivery folder, along with the presentation and updated csv file.

2. Data understanding and pre-processing

The ‘avro-issues’.csv dataset includes 1458 datapoints, each of whom corresponds to an issue alarm case, and 17 variables that will be shortly described below along with some major observations.

Numerical variables:

- ‘vote_count’: represents the number of votes that appear in the website discussion.
- ‘comment_count’: count of the comments added by operator and users onto the alarm resolution thread.
- ‘description_length’: indicates the length of the issue description.
- ‘summary_length’: length of the issue summary in the website.
- ‘days_in_current_status’: days passed between the latest status update and the time of observation.

The first three listed numerical variables are all heavily right skewed distributed. ‘summary_length’ is almost normally distributed.

‘days_in_current_status’ is not of interest for the Fixed cases (the only ones that will be used for training the regression) because it only depends on when the data are recorded. It could though give an idea of the solution time for the open cases. This information could be used for a simple model that makes a first attempt to predict the resolution time by checking, a trivial example, if the open case is in the same state for a time less than the average resolution time. And it then assigns that average, or a derived value, as estimated resolution time. With similar filtering approaches the ‘days_in_current_status’ variable could be made useful, but cannot be included in the type of model that will be initially implemented.

Time variables:

- ‘created’: date of creation of the alarm issue in the database.
- ‘updated’: date of the latest update in the issue thread. This can include any status change, assignment, added comment, added patch etc.
- ‘resolutiondate’: date of alert resolution.

Categorical variables:

- ‘status’: stage of the resolution process.
- ’priority’: urgency rating including 5 levels.
- ‘issue_type’: type of the alert.
- ‘reporter’: person who creates the alert.
- ‘assignee’: appointed responsible for alert resolution.
- ‘resolution’: specifies the resolution state. Since only cases with ‘resolution’=Fixed will be considered this variable is only used for dataset down selection.

‘key’ and ‘project’ are label type categorical variables that will be ignored for the model.

What is not explicitly included is the target variable. Since the assignment requires to predict the resolution time, the target is obviously calculated as ‘resolutiontime’ =‘ resolutiondate’ - ‘created’ and converted in hours (because more relevant in a business context).

The required supervised model needs to know resolution time, therefore only the rows with ‘status’=Resolved and ‘status’=Closed, for which ‘resolution’=Fixed, will be considered for the regression dataset. The difference between the two is that the former, whilst resolved, is still waiting for resolution acceptance whereas the latter has already received the acceptance and therefore the issue and thread are closed. In this second case the update date corresponds to the latest change in status from Resolved to Closed. For Resolved (but not closed) cases the update date corresponds to the latest change and can included anything (status, comment, attachment etc). Thus ‘updated’ tells nothing on the resolution time and will be excluded from the features. It would clearly help to fit the regression dataset, but that’s not the scope as the aim is to predict resolution time for new open cases. For them, update is mostly corresponding to minor actions shortly subsequent to the issue creation, therefore completely unrelated to the update times in a resolved or closed issue. 96% of the selected regression data set corresponds to Closed cases and 4% to Open ones thus configuring an unbalanced dataset, the total count of fixed issues is 1025. The reason of the unbalance is not the variable ‘status’ itself, that is exclude from the final features, but the fact that Resolved and Closed have different target value ranges and also respond differently to the variations of some of the other features. The datapoints with status different from the above two values were excluded.

3. Exploratory Data Analysis

The main initial observation is that both the numerical variables and the target are positively skewed. In addition to that, as discussed in the previous paragraph, differences are present between Closed and Resolved cases, that configure the two main clusters. Therefore they were separately investigated trying to understand consequences for the model. Because of that, the following plots and observations among features and target are presented for the above two distinct cases. Major findings include:

- The mean and std of ‘solutiontime’ for Resolved are roughly double than those for Closed, table 1.
- Resolved only includes 3 ‘issue_type’ values and we see a slight tendency to higher solutiontime for New Feature. Closed includes all 7 values, the distributions of which show more outliers and no defined relation with the target. See figure 3.
- Priority box plots show some higher dependency between Major value and target for Resolved and again no relation and more outliers for Closed, figure 2.
- Pearson correlation coefficients were calculated for the numerical features, even though the requirement of linearity is not met, only as rough indication of possible correlations to be further investigated (see notebook).
- ‘Reporter’ and ‘Assignee’ distributions are also skewed. Most of the value counts are very low. Roughly, only 10% of the reporter/assignee counts are 10 or larger. With the same top 2 persons at the top of both count lists.

Additional plots are included in the Python notebook.

Resolution time			
	Resolved	Closed	All
count	40	985	1025
Mean (hours)	2273.86	922	975.25
Std (hours)	4862	2571	2705.70

Table1

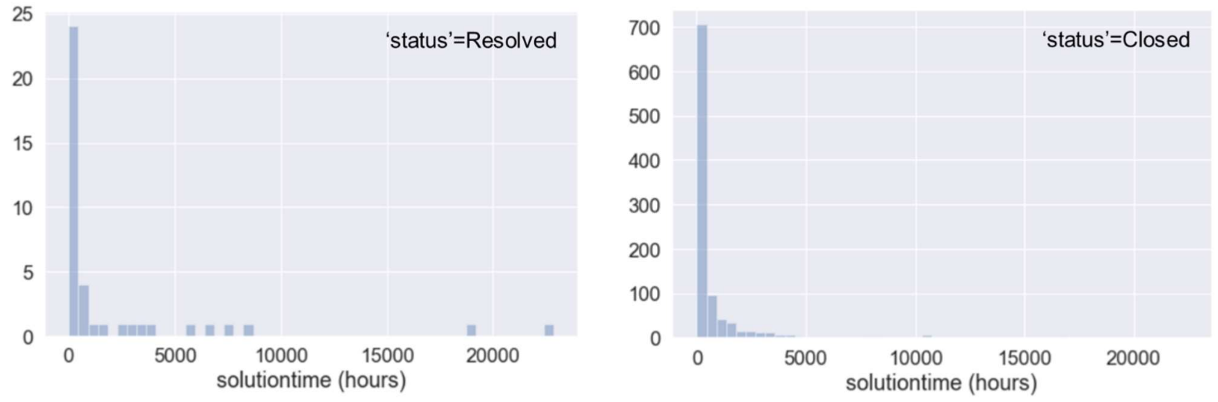

Figure 1

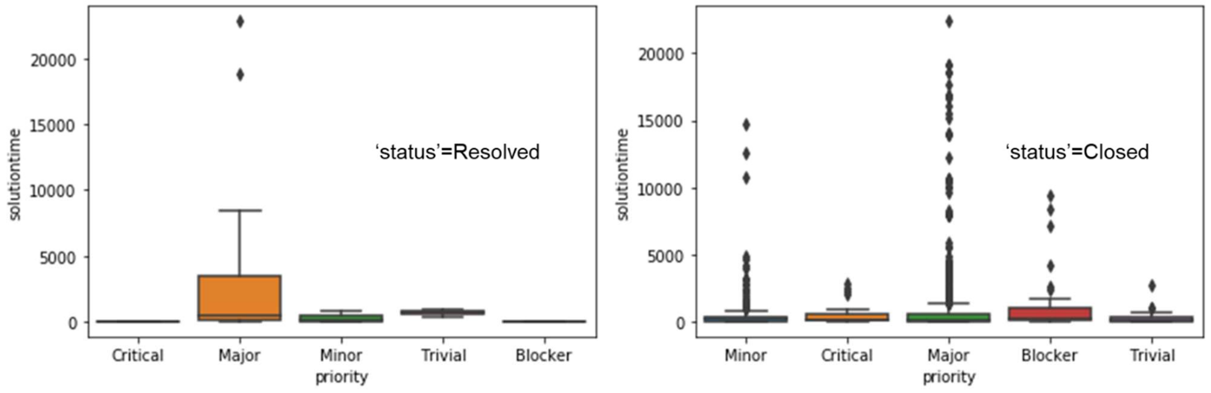

Figure 2

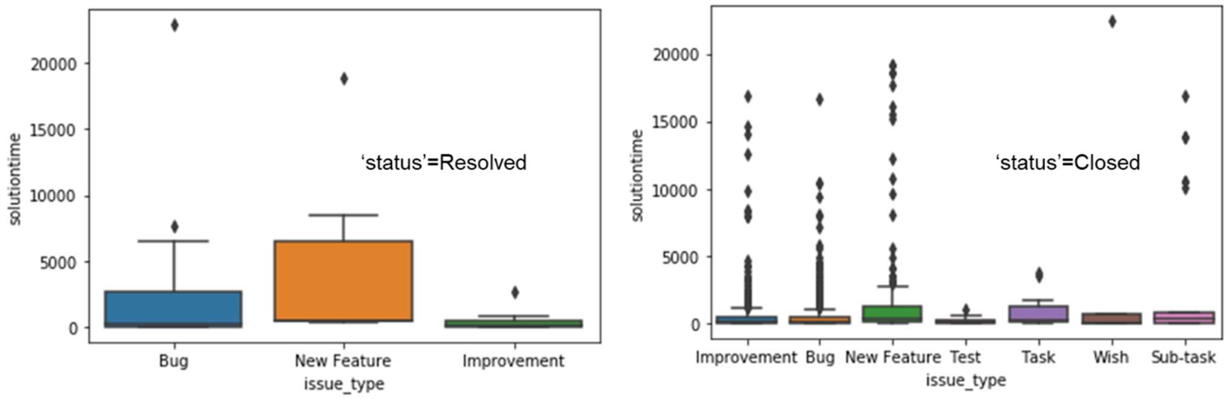

Figure 3

4. Model and results

Feature selection

In paragraph 3 the reasons for ignoring 'days_in_current_status' and 'updated' variables were already given. Considering their count distribution, 'assignee' and 'reporter' would intuitively not be of great influence for the target. Nevertheless, after short iterative trials, using a model-based approach the inclusion of those two features showed an improvement of the test set accuracy, without increasing the overfitting and therefore were kept. The rest of the categorical predictors are all included except for ‘status’. Among the numerical variables, only ‘summary_lenght’ and ‘description_length’ were discarded; both intuition and model-based approach supported that. The final list of features included in the model is the following:

- ‘priority’
- ‘issue_type’
- ‘reporter’
- ‘vote_count’
- ‘assignee’
- ‘comment_count’
- ‘watch_count’

For all the categorical variables one-hot-encoding was performed.

Regressor

Due to the nature of the problem - which is characterized by high positively skewed distributions for both the dependent and independent variables, non-linearity and high number of features (due to one-hot-encoding) – the most appropriate choice for a first attempt was to use a Random Forest Regressor, which is among the most capable algorithm to handle this context. In addition it offers the advantage of a quick implementation and can output feature importance coefficients, that are useful to refine the predictor selection.

Validation strategy

There are two main factors influencing the choice of a validation strategy for this case. The first is the limited amount of datapoints, 1025 in the selected dataset, combined with unbalance. A classical train/test split would produce the risk of evaluating the model relying mostly on the random split, this can be overcome by adopting a cross-validation. The second factor is the need of determining the best model parameters for the Random Forest Regressor. We have limited our options to ‘max_depth’ and ‘n-estimators’ because they typically are the most influential. A proper evaluation of model parameters should include a split of the train set in train and evaluation and, to avoid the split bias, in this case the Grid Search strategy comes very well at hands. To best optimize the two requirements above the solution is to implement a so called ‘nested’ cross validation which outputs the score resulting from the best model based on two nested for algorithms, but that has the drawback of not making available the best model parameters. Based on that, the choice has to be for one of the two above approaches. It was decided to perform a train/set split, determining the best model parameter with a Grid search on the train set and calculating the score on the test set. To mitigate the risk of depending too much on one split, the model was ran over 5 split random variables and the mean result was taken. Ideally the number of tested cases should be higher, but the computational cost was higher. Considering the reasonable standard deviation of the scorings among the 5 cases, it can be considered a good compromise. The use of cross validation would have come at the cost of evaluating the model parameters on the same set used for cross validation, therefore configuring risk of test data leakage.

Results

Scoring of the model run over the above described dataset shows very poor results, with low r2 and mean absolute error in the same range of the target mean. Overfitting is clear on one hand, on the other hand unbalance between Resolved and Closed status plays a major role. The most fast-to-implement solution is to resample the dataset by oversampling the minority class, status=Resolved. This was done simply by multiplying its datapoints to match the count of the majority class (Closed). R2 score improved significantly, but mean absolute error did not change. Further considerations and possible improvements are discussed in the last paragraph. Result comparison is shown in table 2.

	r2 score		Mean absolute error (hours)		Target mean (hours)
	Original dataset	Resampled	Original dataset	Resampled	
Train set	0.60	0.86			975
Test set	0.25	0.81	988.5	854.2	

Table 2

In general the model predicts better when the target magnitudes are low, see plot in the presentation and notebook.

The following three cases were selected from the test set to check model predictions. The model trained on the original dataset was used for the following predictions, because the improved one needs additional investigations.

Case ID	Description	target (h)	predicted (h)	error (h)
575	Status=Resolved, max target value	22854	965	21889
1199	Status=Closed, median target value	146	767	-416
107	Status=closed, min target value	0.064	508.064	-508

Table 3

The choice is intended to check the predictions on the extremes and on the median of the target values. For the max target the error is huge, in the other two cases the errors are around 50% of target mean and hence better (although very high compared to the target). The first impression is that the predictions are better when the status is Closed, but this doesn't surprise considering that the model is trained with a large majority of this cases. Looking at the other variables, there are no significant observations that could make sense of the errors. The main take away for further improvements is to investigate the role of the outliers and to refine strategies to solve the dataset unbalance, as discussed in the last paragraph.

5. Additional features

The assignment requires to identify other data from the Avro website, and contained in the original json file, that could be useful to improve the model. The following variables were identified and considered.

- 'component': refers to the environment affected by the issue. That can be Java (larger portion), C, Python, etc. Intuitively it should have an interdependency with the target, since it can determine the severity and thus influencing the resolution time.
- 'version': this is the Avro release version to which the alert refers. A rapid look at the data shows that about 50% of the values are missing and therefore it won't be considered further.
- 'total logs': this parameter indicates the total number of times an issue thread has been updated. We want to verify whether it has any significant relationship with the target. Intuitively a higher amount of the updates shall potentially determine longer resolution times. However, the number of changes can be unrelated to their time frequencies.

A quick data visualization shows how both the first and third variables seem to have no significant impact on the target, figures 4 and 5, and also don't show clear dependency between themselves, figure 6. To confirm the above, Random Forest model was run with the additional features using the same Grid Search and evaluation procedure as in the original dataset. Result comparison is given in table 4. The modified .csv dataset, where two new columns corresponding to the first and third variables above were added, is attached (avro-issues_updated_NB.csv). A further step would be to retrieve the delta times between the update changes in order to better relate them to the resolution times.

R2 score	Original dataset	‘component’ and ‘logs’ added	Only ‘component’ added	Only ‘logs’ added
Train set	0.60	0.53	0.53	0.53
Test set	0.25	0.19	0.20	0.20

Table 4

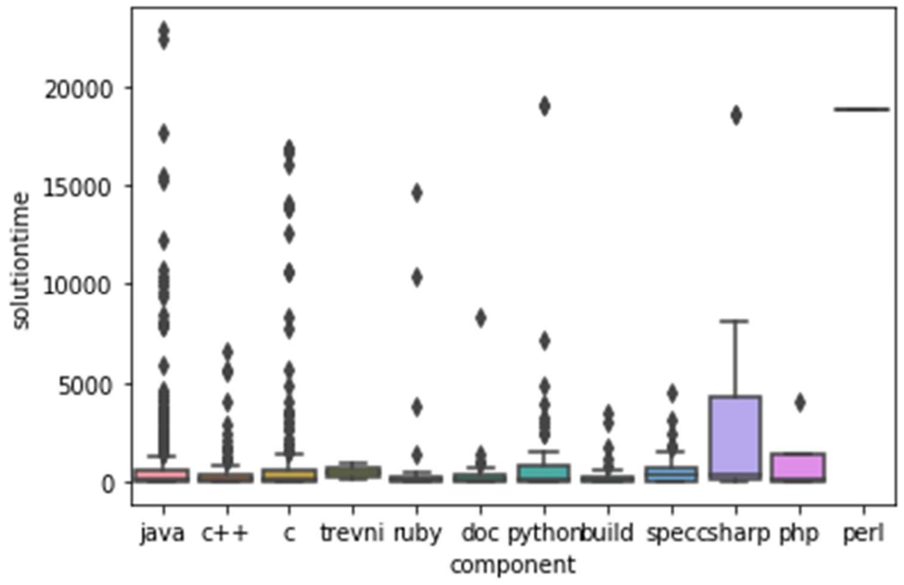

Figure 4

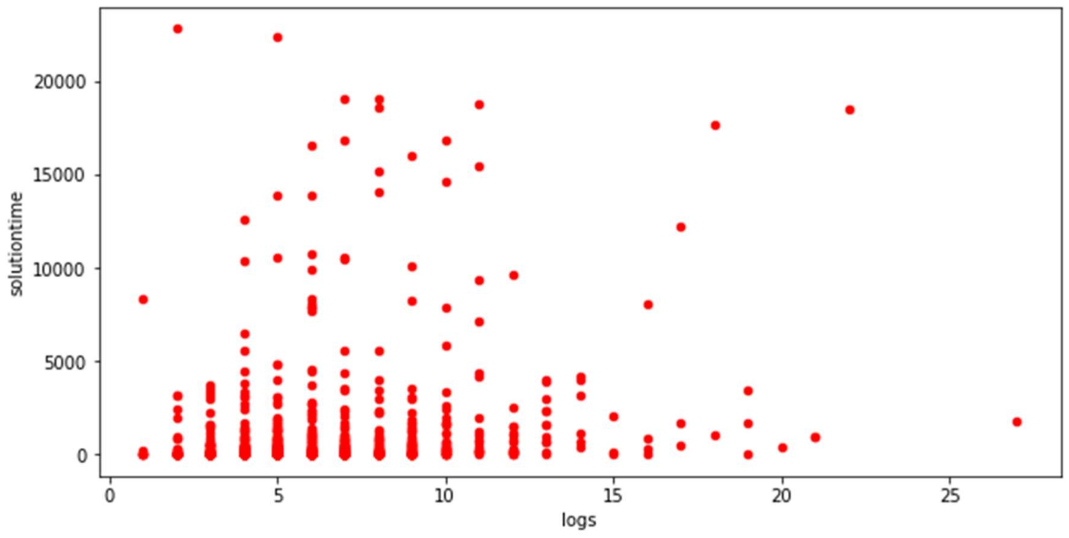

Figure 5

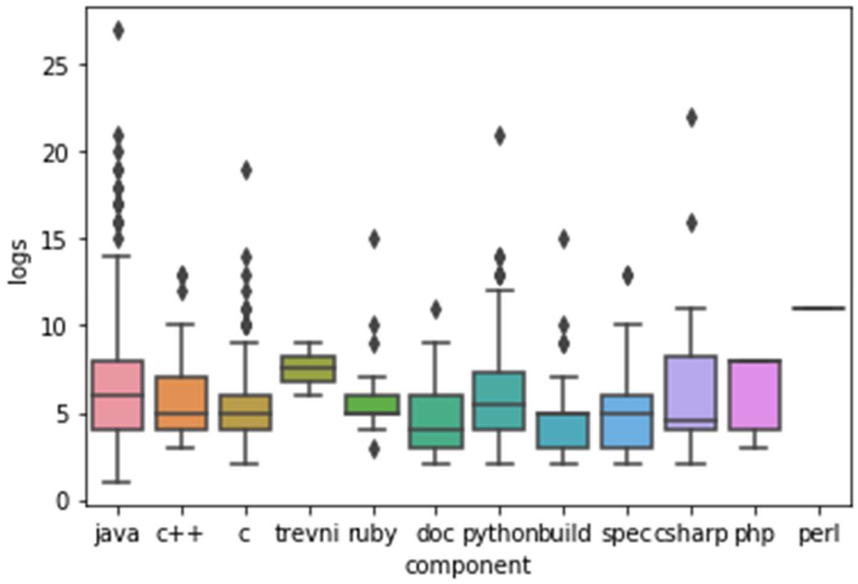

Figure 6

6. Conclusion and next steps

The problem presents a series of difficulties already discussed in the previous paragraphs and that are listed below, along with proposed mitigations and further analyses. Those could not be implemented in the present work because of time reason. The results obtained with the use of the Random Forest regressor show a reasonable starting point, especially with the data resampling, for a problem understanding. However, the model is still very inaccurate and instable.

Positive skewness: data transformations such as logarithmic should be applied both on the target and the numerical predictors in the attempt to obtain normally-like distributed values. This will also allow to fit a multivariate linear regressions, ideally variants like Ridge and Lasso that mitigate overfitting, to be used as a reference for more advanced regressors. Transformations will also make the dataset more digestible to a neural network regression that could be ran over different settings of dense layer parameters to see whether the accuracy can be improved.

Data unbalance: a quick solution was implemented by oversampling the status Resolved. That only improved the coefficient of determination and requires understanding of the consequences and consolidation. Another methodology to counteract unbalance is weighting the classes. Scikit learn includes some options for weighting at sampling or scoring levels. Additional and more advanced approaches exist in literature.

Lack of data: The datapoints considered in the regression dataset are only 1025, this probably also takes a role in the poor accuracy performance of the Random Forest regressor. Retrieving more data from the Avro website (that have been increasing considerably since ~ 2013 when the csv was extracted) would benefit any employed model against overfitting. Further investigation on useful predictor variables not included in the original csv file would also be beneficial.

Outliers: Data contains high rates of outliers. A simple approach for modelling would be to gradually start removing outliers values and see the effects. K-Means clustering algorithm could also be used to better investigate possible smaller data clusters.