80245013 Advanced Network Management
Final Project

Tsinghua University
December 25, 2019

Mafia Team

Albert Millan – 2019280366
Matthieu Lin – 2019280208
Andrei Glinski – 2019280807

Contents

1 Problem Definition

The assigned project's aim is to design an algorithm that can accurately forecast anomalies within a dataset. Precisely, the goal is to determine whether a datapoint, given its timestamp and value, is classified as an anomaly. The dataset englobes a set of 26 different KPIs from various major internet companies such as Baidu, Alibaba and Tencent amongst others.

This report aims to explain the method employed to classify anomalies from the given dataset. The solution involves a mixture of three models, namely Donut [1], Facebook Prophet [2] and a naive implementation. The dataset is initially split by KPI, generating 26 datasets. These are then individually fed into the pipeline, applying the one model that yields the best performance from the aforementioned models. Lastly, the results are recombined into a single testing dataset file and formatted for submission purposes.

The report is structured in the following manner. The data analysis and pruning procedure is presented in section 2. It involves a definition of anomalies and our strategy to identify them. Section 3 It is followed by a description of the machine learning pipeline employed to classify outliers in section 4, outlining the properties of the three models employed. Results are presented in section 5, outlining how our models compare with each other. Finally, concluding remarks are provided in section 6.

2 Background Research & Data Analysis

Inferring the correctness of an algorithm using unlabelled data is an arduous task. Some approaches exists, however, not all of them can assure the correctness of the predictions with a high confidence degree. For example, traditional machine learning approaches often split the labelled training dataset into train, test and validation subsets, employing the train subset to train the model, test subset to tune the parameters, and validation subset to estimate the correctness of the models. While this approach can yield optimal results when the train and test datasets have a similar nature, it often fails when the values of the datapoints on the training and testing datasets differ sufficiently. This is evidenced in figure 1, showing that the predicted values for the KPI 'a40b1df87e3f1c87'

trained with the training set exhibit an increasing trend, while the predictions generated by the test set range decrease and do not exhibit any trend. In turn, the generated predictions for the value do not accurately map the actual values in the test set, hence yielding an incorrect proportion of anomalies.

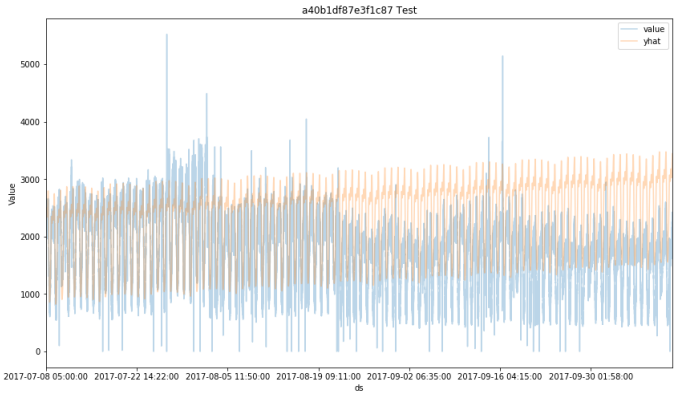

Figure 1: Values and predictions when trained with training dataset.

It is therefore argued that performance/score can be improved with a model that maps the test dataset closely but is also insensible to outliers. Another unconventional approach, is to use the test dataset as the training for our predictions for the values. This way, when the model makes predictions for the values at the same time interval it used during the training procedure, it generates predictions that map the test dataset closely, in turn, yielding more accurate predictions. This is illustrated in figure 2.

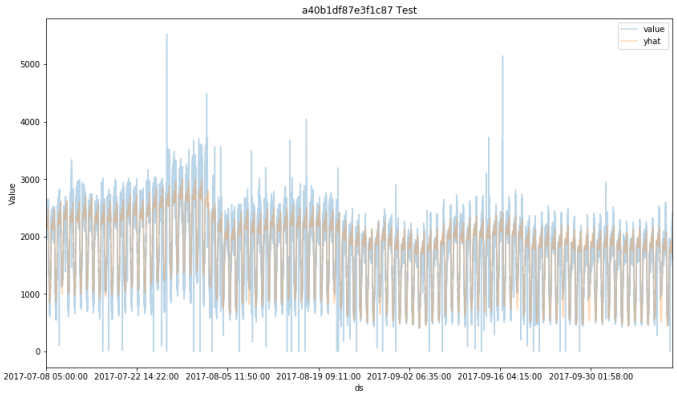

Figure 2: Values and predictions when trained with testing dataset.

Identifying the value for the threshold that maximizes the evaluation metric of the predicted labels requires, again, a thorough understanding of the KPI dataset and what is considered an anomaly. The proportion of correctly and misclassified anomalies depends, to some extent, on the value assigned to the threshold, as well of the model employed. Hence having an estimate of the proportion of anomalies in the test dataset is paramount for obtain optimal performance[1]. This is illustrated in figure 3.

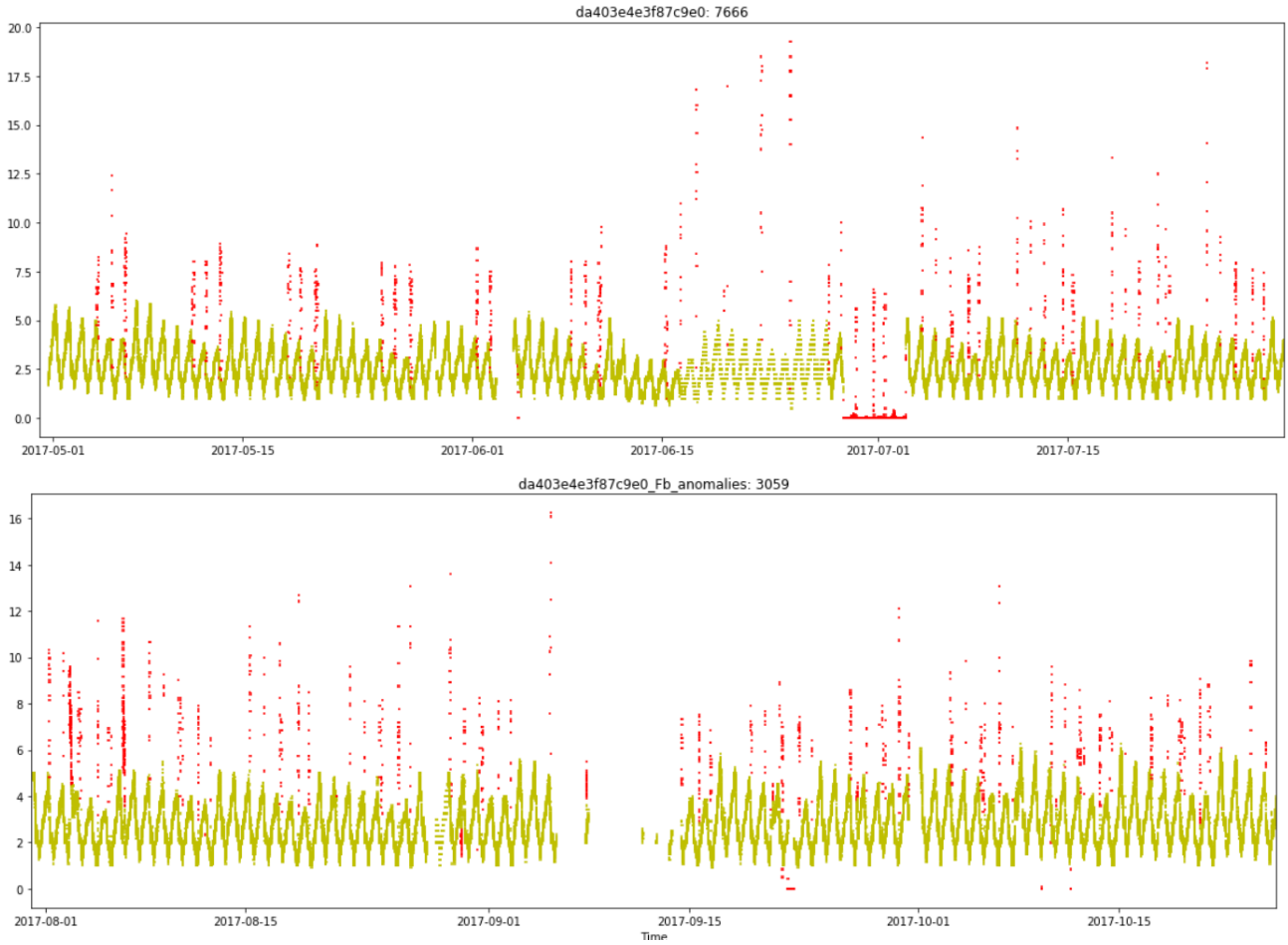

Figure 3: Anomalies in train dataset and predicted anomalies in test dataset. Notice the number of anomalies in the test dataset is reduced by a half.

To conclude this section, we emphasize the importance of data visualization in the accomplishment of our final performance score. Realizing about the nature of the data, identification

[1]Recall that the proportion of anomalies in the train and test dataset might vary significantly.

of appropriate models and the generation of estimates was plausible due to the plotting of each training and test datasets KPI datasets. Visualizing training datasets with the assigned labels aided to understand what are anomalies within the datasets.

3 Data Processing

This section highlights operations performed on the training or test datasets before inputting them into the Prophet and donut models. Initially, the train and test datasets where split into 26 KPIs, and the following operations were performed on each subset.

Data used to train prophet had to be formatted before training. Timestamp was converted into datetime format ('YYYY-MM-DD HH:MM:SS'). Although the authors recommend removing outliers to prevent slight shifts on the generated predictions, the predictions were relatively insensitive to outliers in most models. Prophet is also capable of performing training despite having missing values with minor changes in the accuracy. The model is therefore appropriate for this type of data.

The Donut model was implemented under the same conditions specified in the original paper [2]. Pre-processing steps involved generating the missing entries, storing the indices of the missing entries in a temporary variable. The values were normalized and inputted to the model. The temporary variable was then used to remove the missing entries in the output array. In addition, the first 120 values, which correspond to the size of the sliding window, were filled with zeros and pre-pended to the aforementioned output array.

4 Methodology

This section describes the characteristics and properties of the donut and Prophet models, as well as the conditions and parameters employed to train them.

4.1 Prophet

Prophet is a python library developed by Facebook to generate time series forecasts at scale with ease. It provides intuitive parameters to adjust models in a rapid and trivial manner. It provides similar functionality to autoregressive moving average (ARIMA) models, but with an added simplicity of easy to understand API.

Training was performed by inputting timestamp and the values associated for each KPI in the test dataset. The predictions were generated inputting the time interval for which we wanted to make predictions for the values[2]. The predicted values were plotted against each value in the KPI, as well as the actual training dataset with the labels for that KPI to understand the nature of the anomalies. Based on this visualization, we derived a threshold value. We computed the difference between the predictions and actual values[3], and then compared each datapoint with the threshold. The case where the difference lied above the threshold were classified as anomalies, while the case where the difference lied below the threshold were not classified as anomalies.

4.2 Donut

The input data includes the timestamp and the associated values in the training dataset. The parameters employed to train the model are outlined in table 1 and are in accordance with the ones proposed in [1]. It outputs the 'reconstruction probability' stored in an array, indicating the likelihood that a given value is an anomaly. The larger the likelihood, the higher the probability that a given value is an anomaly. Again, by visualizing the data, we derive an estimation for the proportion of anomalies in the test dataset. The threshold of those values that have the highest reconstruction probabilities are classified as anomalies.

Parameter	**Value**
Sliding Window	120
Z Dimensions	5

Table 1: Donut Parameters

[2]Train and test dataset were the same in this model.
[3]Predictions minus actual datapoints

4.3 Pipeline

For each KPI, we visualized the correctly classified and missclassified anomalies[4] identified by each of the models, and select the one model that appears to capture the largest volume of anomalies while also minimizing the errors. Upon identifying the best model, the threshold values are manually and recurrently updated based on the submission scores (see figure 5):

- Perform an operation, an increase or a decrease, in the threshold for a given KPI.
- Visualize the updated plot with more/less anomalies and ensure it is appropriate by visualizing the anomalies in the training dataset.
- Submit to obtain a score.
- Perform the previous steps using the same operation on the same KPI until the f1-score does not improve anymore.

Sample predictions for KPI '02e99bd4f6cfb33f' shown in figure 4.

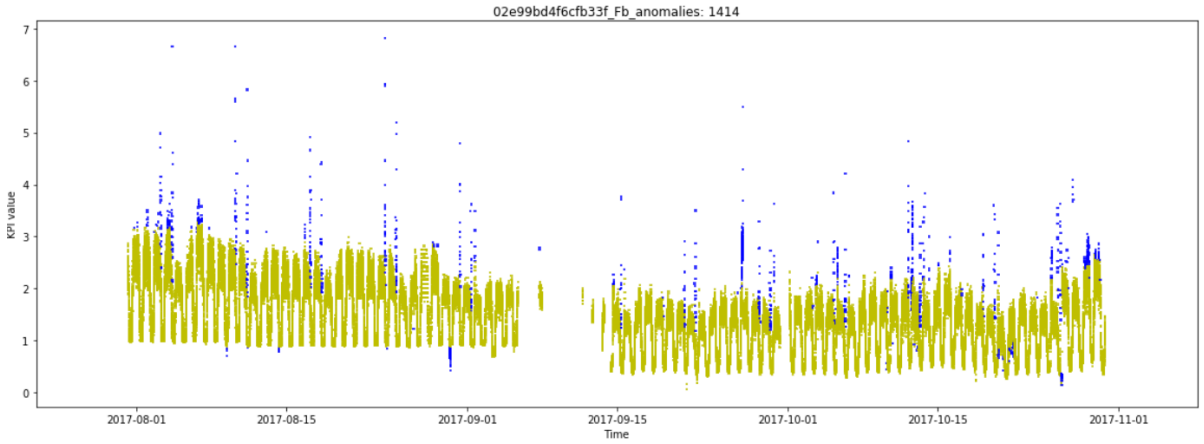

Figure 4: Anomaly classification result, blue colored points indicating anomalies.

[4]We estimate the number of anomalies by observing the graph.

KPI	Model	Threshold
02e99bd4f6cfb33f	Prophet	0.75000
046ec29ddf80d62e	Donut	0.01200
07927a9a18fa19ae	Donut	0.01400
09513ae3e75778a3	Donut	0.03000
18fbb1d5a5dc099d	Donut	0.03500
1c35dbf57f55f5e4	Donut	0.04500
40e25005ff8992bd	Donut	0.01000
54e8a140f6237526	Donut	0.01000
71595dd7171f4540	Donut	0.00430
769894baefea4e9e	Donut	0.00185
76f4550c43334374	Donut	0.03000
7c189dd36f048a6c	Donut	0.00420
88cf3a776ba00e7c	Donut	0.05600
8a20c229e9860d0c	Prophet	0.06000
8bef9af9a922e0b3	Donut	0.00335
8c892e5525f3e491	Donut	0.00400
9bd90500bfd11edb	Prophet	16.00000
9ee5879409dccef9	Donut	0.04500
a40b1df87e3f1c87	Donut	0.00750
a5bf5d65261d859a	Naive	4.00000
affb01ca2b4f0b45	Donut	0.00700
b3b2e6d1a791d63a	Prophet	2.20000
c58bfcbacb2822d1	Donut	0.00800
cff6d3c01e6a6bfa	Donut	0.01400
da403e4e3f87c9e0	Prophet	1.40000
e0770391decc44ce	Prophet	2750.00000

Figure 5: Pipeline.

5 Results

The proposed pipeline obtained a 78.35% f1-score, top score in the class competition. The list of scores and model employed to obtain them is provide in figure 6.

Models	F1-Score
Donut + Prophet	0.783587
Donut	0.719681
Facebook Prophet	0.540883
LSTM	0.390139
DecisionTreeClassifier	0.371066
ExtraTreesClassifier	0.378510
RandomForestClassifier	0.366923
BaggingClassifier	0.367624
GradientBoostingClassifier	0.319461

Figure 6: F1-Score for each of the models tested.

6 Conclusion

This project introduced a time series anomaly detection problem from several corporate organizations. Appropriate visualization of the data enabled us to comprehend the intrinsic nature of the data, distinguishing anomalies from non-anomalies. Such deep understanding of the data at hand was paramount to infer optimal thresholds to classify anomalies at later stages. Combining models such as Prophet and the Donut yielded outstanding results. While the thresholds can still be optimized further, the obtained results were sufficient to win the class competition. Overall, this project has taught us valuable time series analysis skills and about the importance of data visualization.

References

[1] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," *CoRR*, vol. abs/1802.03903, 2018.

[2] S. J. Taylor and B. Letham, "Forecasting at scale," *PeerJ Preprints*, vol. 5, p. e3190v2, Sept. 2017.