Report of ANM 2019 Fall Project - AIOps Challenge

Shubh Vachher
Department of Computer Science
Tsinghua University
Beijing, China
vachhers11@mails.tsinghua.edu.cn

Davide Liu
Department of Computer Science
Tsinghua University
Beijing, China
liud20@mails.tsinghua.edu.cn

Abstract

This is the report of the project assigned during the course of Advanced Network Management during the first semester of the year 2019/2020. Our team, composed by the students Davide Liu and Shubh Vachher, successfully completed the task achieving a final score of 0.677 on the project website (2) on the test dataset. The project consisted of an anomaly detection competition on a dataset composed of 26 different KPIs from large Internet companies whose names have been omitted to respect their privacy. KPIs (key performance indicators) are time series data, measuring metrics such as number of page views, online users, and number of orders monitored by large internet-based companies. These time series have a very simple structure including a timestamp, a value associated to the timestamp and, in a supervised context, also a label indicating whether the current timestamp represents an anomaly or not.

1 Our work

In this paragraph we are going to give an overview of our work explaining the methods we used to pre-process the data and finally to predict the anomalies. In the next few sections we are going to analyze these methods one by one and are going to show our results in the last section of this report. The first model we used was Donut which is an autoencoder, such as VAE, but specialized for time series anomaly detection. We made many adjustments to the thresholding for anomaly prediction for the donut model output scores. Our next method used a Deep Feedforward network with simple feature engineering inputs since these kind of models have been recognized as a powerful technique to represent the complex relationships between input and output. The final submission file was made with best predictions out of the two models on separate KPIs and this is the one that achieved the 0.677 score.

2 Data preprocessing

The first step consists of preprocessing the data so as to make it more suitable for the models to make their predictions. In particular, it is possible that in the KPI datasets some timestamps are missing. This can occur for several reasons such as during server maintenance times or logging errors. Common solutions consist of replacing these missing points with 0 or computing their values by some interpolation techniques. We tried both the techniques and found no visible improvement in final test set scores. Thus, to keep our preprocessing simple, the solution we adopted is just filling the missing points with 0 value and let our models learn about and managing those. Finally, all values have been standardized in individual KPIs, which means subtracting the mean and dividing by their standard deviation. It is important to say that during each phase, all the rows belonging to the same KPIs have been considered as disjoint datasets.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

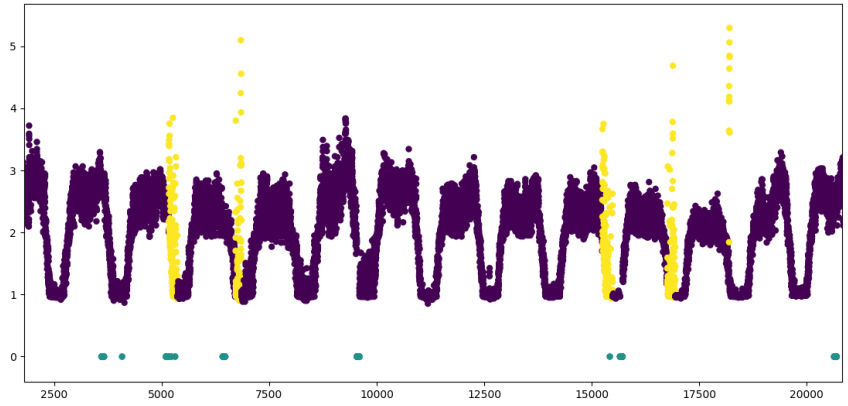


Figure 1: Anomalies are in Yellow and missing points in Green.

3 Donut

Donut (1) is an unsupervised anomaly detection algorithm based on Variational Autoencoder which can work when the data is unlabeled but can also take advantage of the occasional labels when available. The main structural difference between Donut and standard VAE is the presence of a sliding window to get a vector of input rows instead than just one as it is in VAE. Besides its structure, the are three key techniques that allow Donut to greatly outperform state-of-art supervised and VAE based anomaly detection algorithms: Modified ELBO and Missing Data Injection during training and MCMC Imputation during detection.

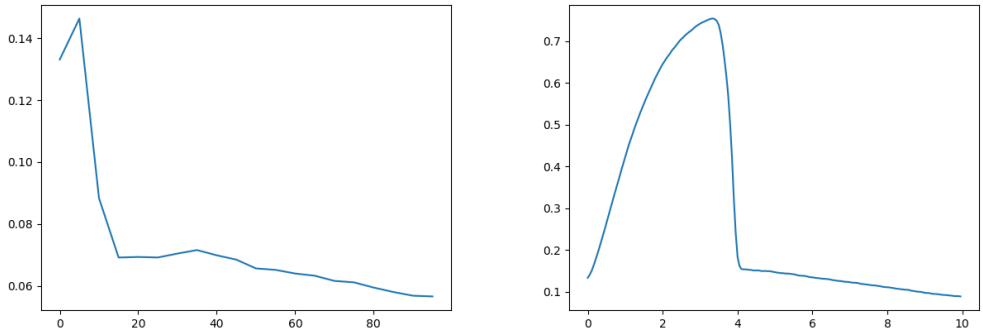
We first train the donut model from the open source code available on github (3) on our preprocessed dataset. The output of the donut model on our test dataset is the score that represents if the point that is being predicted is an anomaly or not and the more negative the score, the larger the probability of an anomalous point, according to the trained model. The next step is to assign a threshold beyond which this score represents an anomaly for us.

In our first experiment, we tried to find the optimal threshold for this score in the following way: we exploited the labeled train set to calculate the F1 score for each KPI testing different threshold values ranging from 0 to 100. Secondly, we repeated the process but this time testing smaller values along a range very close to the best threshold in order to fine tune our threshold. We can see from figure 2a and figure 2b that the fine tuning makes a large difference in the result in some cases. The final prediction of such custom threshold finding on all the KPIs gave us a score of 0.355

However, in order to make better predictions, we trained a random forest model on top of Donut scores to predict the label of each timestamp based on its row score given as input. We tried a random forest classifier with giving scores of 10 rows before it as input but just a single score as input outperformed the rest. We used cross validation set to find the best depth for random forest for different KPIs. The final prediction of random forest classifier gave us a score of 0.55 on the test set on the website of the challenge (2).

4 Deep Feedforward Network

Since we were not satisfied with the final score of 0.55 we decided to try out a deep feedforward neural network along with some data preprocessing steps. The network was simpler than the donut model but we were convinced that most of the KPIs did not need such a complicated model to make accurate predictions.



(a) Testing 0 to 100 values

(b) Fine Tuning Value

Figure 2: Threshold vs F1 score on one training KPI

We started off with a simple preprocessing step of feature engineering. Looking at our preprocessed data from figure 1 we could see that a sudden large fluctuation in values was a common occurrence near anomalous points. Thus we decided to take the first derivative of our time series KPI data as the input to our feedforward model. Since the first derivative fluctuated a lot we also took an average of the values for the first derivative each minute and assigned all rows in that minute the same first derivative value.

We used a simple feedforward network with dense layers followed by dropout with relu activation and mean squared error loss. The input to the model was the 100 previous values to the point being predicted and the output was directly classifying the point as an anomaly or not. The model can be visualized in the figure 3. The final predictions of this model on the test set gave a score of 0.61.

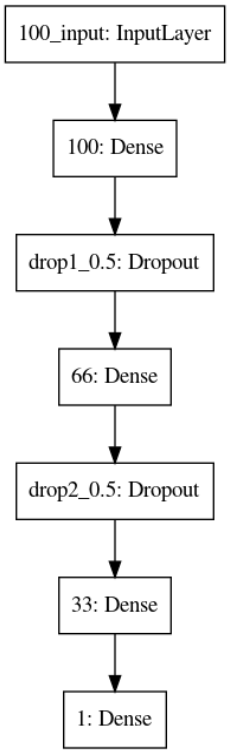


Figure 3: Layers in our Deep Feedforward network

5 Ensemble model

Since we had two models that performed similarly well we decided, based on how they performed individually on the f1 score in our training dataset that we would create a final submission file with the predictions of the deep feedforward network for the KPIs it performed well on and DONUT for the rest. We used DONUT finally for 8 KPIs and the deep feedforward network for the 18 rest and the final score we got from this ensemble model was 0.661.

6 Computational Requirements

The donut model and the deep feedforward model took approximately the same amount of time to train on our non-GPU systems with 4 i7 cpus. For both the models it took about 3 hours of training to train the models to a reasonable accuracy beyond which their metrics showed signs of overfitting. We found the KPI data to be large enough to allow us to imagine complex ideas and models while being small enough to not take ages to train on commercially available laptops.

7 Evaluation trick

The model that we were finally happy with gave us a score of 0.661. Then, since we knew that only the beginning of an anomalous window of points was valuable to the final evaluation script to mark the whole window as an anomaly (4) we made a visualization of the average length of anomalous window sizes in the training dataset. You can see that from the figure 4 the window sizes 1 and 2 are very few in number. So, if we ignore the contributions of the same in our final score, we can also improve our precision by reducing false negatives. By removing every window of size two from our submission file, our score jumped from 0.661 to 0.677 and this was our final score in the challenge.

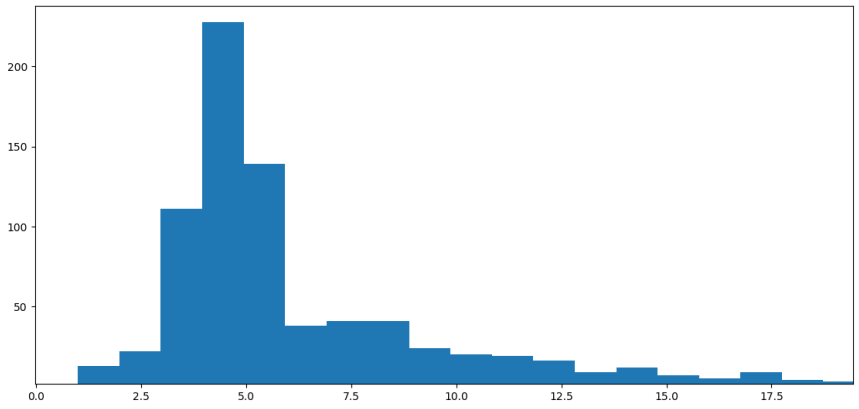


Figure 4: Average size of anomalous windows

8 Results

We can see the progess that the group made with the first submission being one with donut as well as custom thresholding followed by donut and random forest prediction, then followed by deep feedforward network and finally an ensemble model in figure 5. We believe that the models we have utilized are general purpose models that can further be used using the code we provide on kpi's of different types too. The 26 KPIs used represent a large variety of the time series data that is used in the industry. We have gained invaluable experience in working with time series data and also with working on data science problems in general.

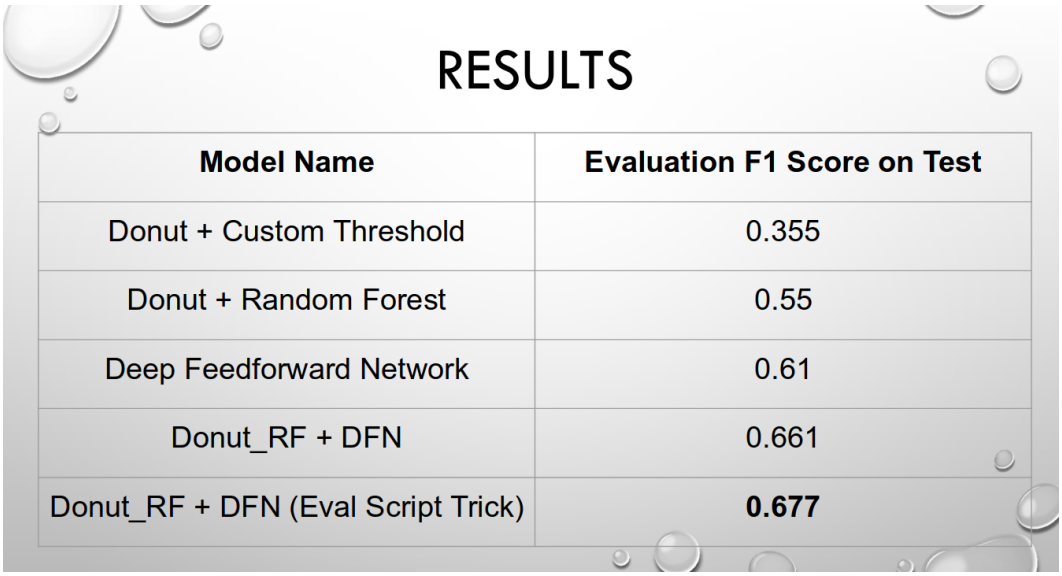


Figure 5: Average size of anomalous windows

References

[1] Haowen Xu & Wenxiao Chen & Nengwen Zhao & Zeyan Li & Jiahao Bu & Zhihan Li & Ying Liu & Youjian Zhao & Dan Pei (2018) *Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications.*

[2] AIOps Challenge: http://iops.ai/

[3] https://github.com/haowen-xu/donut

[4] https://github.com/iopsai/iops/blob/master/evaluation/evaluation.py#L19