**Evaluation of anomaly detection models on time series with Exathlon**

Anomaly detection on timeseries

|  |
| --- |
| **Emmanouil Kalyvas**  School of Informatics  Aristotle University  Thessaloniki, Greece  kalyvase@csd.auth.gr |

**ABSTRACT**

Anomaly detection in time series data plays a critical role in various domains, including finance, cybersecurity and healthcare. With the growing complexity and volume of time series data, the need for accurate and robust anomaly detection models becomes paramount. This paper presents an evaluation of two anomaly detection models using the Exathlon framework [1][2], a comprehensive benchmarking platform specifically designed for time series analysis.

1**INTRODUCTION**

In this paper we try to evaluate and compare two anomaly detection models on time series data, using the Exathlon framework. The Exathlon framework is a benchmarking platform specifically designed for evaluating and comparing different anomaly detection techniques in the context of time series analysis. By providing a standardized environment, diverse datasets, and a wide array of evaluation metrics, Exathlon enables researchers and practitioners to assess the performance and suitability of various anomaly detection models consistently and objectively. The benchmark also introduces four levels of range-based evaluation metrics to evaluate the efficiency of an anomaly detection method. The range-based recall and precision are used to calculate an F-beta score (F1 in our case). The four levels for anomaly detection evaluation are the following:

* AD1 - Anomaly Existence
* AD2 - Range Detection
* AD3 - Early Detection
* AD4 - Exactly-Once Detection

Each one is stricter than the previous one. In our study we focus more on the level AD1, anomaly existence.

In our study, we used the two following models:

* A density-based isolation forest
* And a reconstruction based AutoEncoder

We used the isolation forest model as an unsupervised technique and the AutoEncoder model as a semi-supervised technique where we try to learn the distribution of the normal data and based on that find the outliers. For each of the models we used different techniques to train them in order to understand which performs better. In chapter 3 the training techniques are introduced in detail.

About the dataset we used, it contains data from runtime traces of spark applications, provided by the Exathlon framework. More information about the dataset is provided on chapter 2.

2**DATASET**

The data concerns records from traces recorded from repeated execution of Spark application. Our data consists of 9 traces with only 3 of them containing known anomalies. The rest of the 5 traces are considered normal with no outliers. The known anomalies are the following:

* Bursty Input (T1)
* Bursty Input Until Crash (T2)
* Stalled Input (T3)
* CPU Contention (T4)
* Process Failure (T5)
* Executor Failure (T6)
* Unknown anomalies

Our goal is to identify which time-steps in the time series are anomalous. The data we used contains only the anomalies T2, T4, T6 and some unknown anomalies.

2.1**Data smoothing**

Before we train our models, we apply some smoothing on our time series to get rid of the small noisy fluctuations that can case the model to often give big scores. To achieve that we apply a Savitzky-Golay filter [3] with a time window of ***9*** instances and polynomial degree of ***3***.

We also apply some smoothing on our final anomaly scores predicted by our models, to smooth out any small fluctuations on the predictions. For the smoothing of the final scores we used a fixed window of 5 and a polynomial degree of 3 for all of our instances.

3**TRAINING TECHNIQUES**

For the training of the models, we introduce two different training techniques:

* Single-Model – A single model is trained on the data
* Multi-Model – Multiple models of the same type are trained on different parts of the data

Apart from these two training approaches, we also tried to train our models on the rate of change of our data instead of the actual data values. More information about the mentioned approaches is described in the following chapters.

3.1**Single-Model Training**

In the single model training technique, we concatenate all our traces into a single big trace and try to train our model on it. In this approach the model has a wider view of the time series.

For the isolation forest model, we concatenated the 3 traces with known anomalies into one and tried to fit the model on it. Then based on the decision function of the model, we assign an anomaly score to each timestep of these 3 traces.

For the AutoEncoder model, we concatenate the 5 normal traces into one big trace and then for every ***n*** timesteps, we try to reconstruct the features of the timesteps. Then on the 3 traces with known anomalies, for every ***n*** timesteps, we try to reconstruct the features of the timesteps using the AutoEncoder that was trained on the normal data. The reconstruction error is what we consider for the anomaly score, the higher the more anomalous. The parameter ***n*** is the width of the time window we use to go through our data.

3.2**Multi-Model Training**

In the multi model training technique, we try to train multiple models on our time series. The way we do that, is to concatenate first all the traces into a single big trace and then split it in ***m*** parts, one for each one of the ***m*** models. In order to score our time steps, we evaluate them with all the models and get the mean score as the final value.

For the isolation forest model, we use again the 3 traces with known anomalies for the training. When the parameter ***m*** is equal to 3, then we train one model for each trace.

For the AutoEncoder model we use once more the 5 normal traces for the training and then evaluate the 3 traces with known anomalies using the trained models. When the parameter ***m*** is equal to 5 then we train one model for each normal trace.

3.3**Train on the rate of change**

In addition to training our models on the actual dataset values, we tried to train them also on the rate of change of our data. To achieve that, we subtracted the values of each timestep with the values of the next timestep. This gives us the different between the two timesteps with can contain useful information. We combined this approach with each of the two training techniques mentioned above and compared our findings.

4**RESULTS**

In the next chapters we present the results of our study, for each of the models used.

4.1**Isolation Forest**

We trained the isolation forest model using both the single model technique and the multi model technique described in the previous chapters. Below are all the best results we managed to achieve using both techniques.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num. of models | Train on rate of change | Global  AUPRC | T2 AUPRC | T4  AUPRC | T6 AUPRC | Unknown Anomaly AUPRC |
| 1 | FALSE | 0.688 | 0.864 | 0.670 | 0.435 | 0.596 |
| 1 | TRUE | 0.603 | 0.375 | 0.678 | 0.287 | 0.375 |
| 7 | FALSE | 0.674 | 0.889 | 0.683 | 0.491 | 0.583 |
| 14 | TRUE | 0.694 | 0.857 | 0.623 | 0.596 | 0.857 |

Table 1 - Best Isolation Forest Scores

We can see that the multi model approach with 14 models trained on the rate of change of the data, seems to produce the best results. Despite that fact, if we plot the AUPRC (area under precision and recall curve) versus the number of models, we can see that there is a big fluctuation of the performance, especially when training on the rate of change.

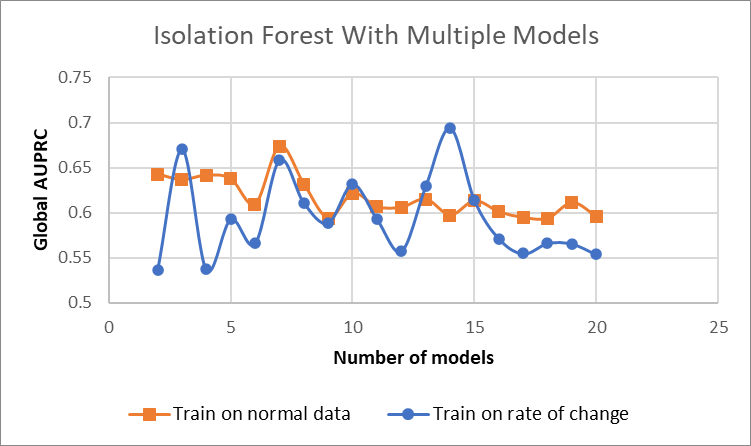


Figure 1 - Global AUPRC vs Number of models for isolation forest

This could happen due to the split of our traces in multiple parts for our models. If the split happens between a series of anomalous points, then it’s possible we ruin the flow of the data and miss such anomalous points. Training on the rates of change is more sensitive to this because it is calculated based on two consecutive points. For certain splits this situation is possibly not happening that often or not at all and we end up with better results. So, splitting the traces into parts of variant size depending on the data distribution might be an improvement to this approach.

The multi model technique seems to do consistently better than the single model one on the executor failure anomaly and especially when using big number of models.

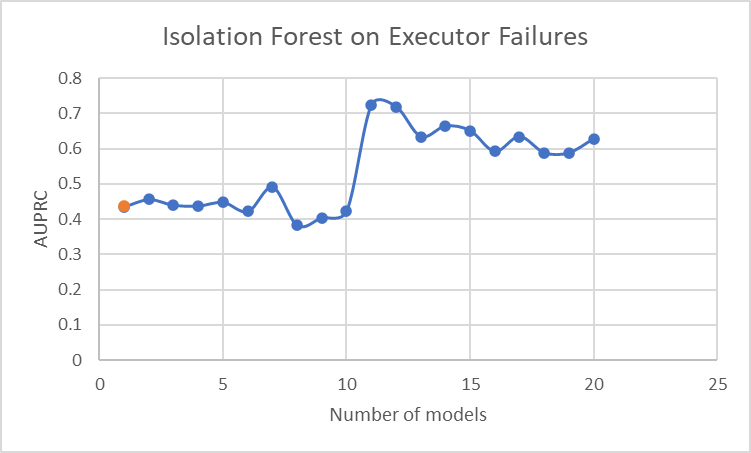


Figure 2 - Number of models versus Executor Failures when training on the normal data

The maximum AUPRC it achieves is 0.72 when using 11 models, trained on the normal data.

4.2**AutoEncoder**

For the Auto Encoder, we used 1 hidden layer of 128 Nodes for both the encoder and the decoder and 256 nodes for the encoded representation of the input. For the input of the encoder and output of the decoder, we used 19·***n*** nodes, where ***n*** is the width of the time window we used.

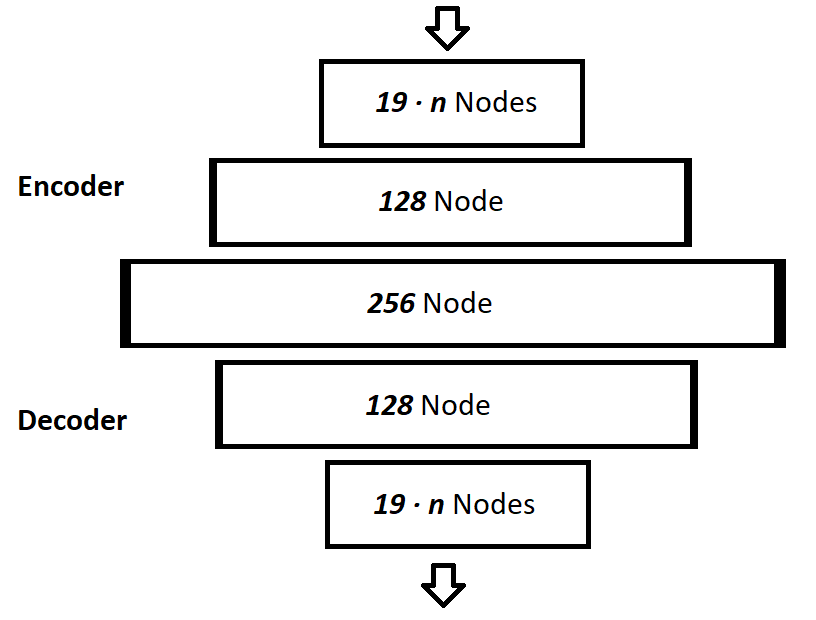


Figure 3 - AutoEncoder architecture

We trained once more this model with the techniques described in the previous chapters using different values for the time window ***n***. Below are the best results we gathered.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Num. of models | Train on rate of change | Global  AUPRC | T2 AUPRC | T4  AUPRC | T6 AUPRC | Unknown Anomaly AUPRC |
| 1 | FALSE | 0.632 | 0.200 | 0.718 | 0.719 | 0.250 |
| 1 | TRUE | 0.679 | 0.875 | 0.629 | 0.632 | 0.875 |
| 5 | FALSE | 0.627 | 0.167 | 0.693 | 0.445 | 0.600 |
| 5 | TRUE | 0.668 | 0.833 | 0.650 | 0.676 | 0.833 |
| 10 | FALSE | 0.630 | 0.167 | 0.708 | 0.445 | 0.530 |
| 10 | TRUE | 0.639 | 0.833 | 0.647 | 0.469 | 0.500 |

Table 2 - Best AutoEncoder Scores

The auto encoder model seems to perform better when trained the rate of change of the data. The multi model technique does not seem to provide any important advantage over the single model this time. If we plot the global AUPRC versus the time window width, we see that we have once more big fluctuations especially when training on the rate of change with small time window. This is possibly due to the issue explained already on the isolation forest.

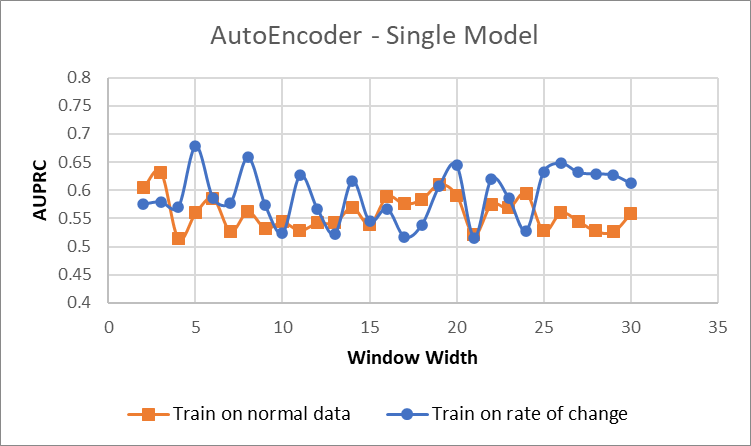


Figure 4 - Global AUPRC vs Window Width for single model

When using the multi model approach, it seems that training on the rate of change doesn’t help with the performance and this is probably due to the fact that training on the rate of change is more sensitive on the information loss of possible split of anomalies when splitting the traces for our models.

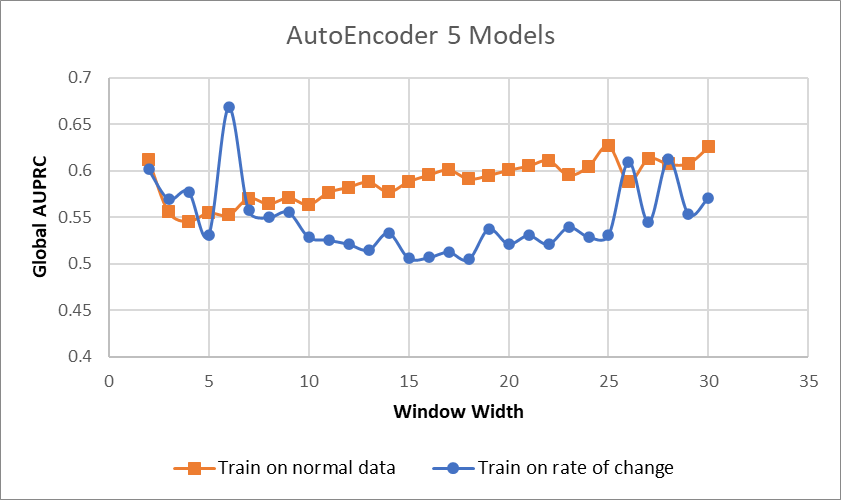


Figure 5 - Global AUPRC vs Window Width using 5 models

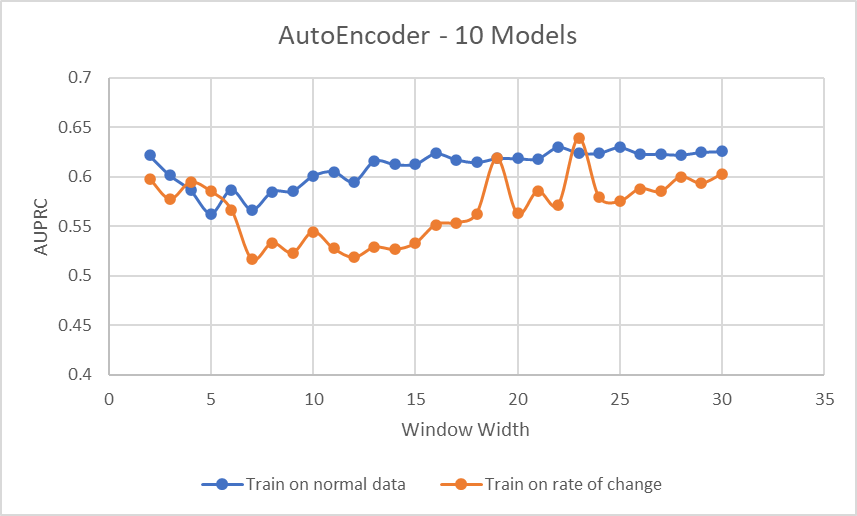


Figure 6 - Global AUPRC vs Window Width using 10 models

4.3**AD2 –** **Range Detection Scores**

None of our approaches did particularly good on the AD2 score of the Exathlon benchmark. The best scores we achieved was with the AutoEncoder model using the single model approach.

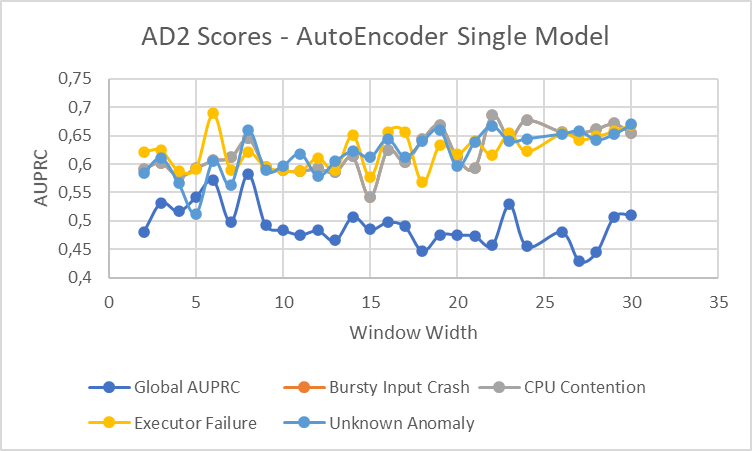


Figure 7 - AD2 score for AutoEncoder using single model approach

**CONCLUSIONS**

The multi model technique paired with the isolation forest, seems to do well on detecting the executor failure anomaly so it might work well for anomalies with similar characteristics. Despite that, it seems that splitting our traces into smaller parts, to use the multiple models, introduces some uncertainty on the performance of the approach. To understand better the cause of this issue and improve the multi model technique different trace splitting approaches should be tried. Also, training on the rate of change, does seem to bring some improvements, especially when using AutoEncoders.

All in all, anomaly detection on time series is a very complicated problem with no “one size fits all” solution.

**ACKNOWLEDGMENTS**

This paper was built for educational purposes for the lesson “Mining From Massive Datasets” of the MSc program “Data & Web Science” of Department of Informatics at Aristotle University of Thessaloniki.

**REFERENCES**

[1] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, “Exathlon,” Proceedings of the VLDB Endowment, vol. 14, no. 11, pp. 2613–2626, Jul. 2021, doi: <https://doi.org/10.14778/3476249.3476307>.

[2] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, “A demonstration of the exathlon benchmarking platform for explainable anomaly detection,” Proceedings of the VLDB Endowment, vol. 14, no. 12, pp. 2827–2830, Jul. 2021, doi: <https://doi.org/10.14778/3476311.3476355>.

[3] R. Schafer, “What Is a Savitzky-Golay Filter? [Lecture Notes],” IEEE Signal Processing Magazine, vol. 28, no. 4, pp. 111–117, Jul. 2011, doi: <https://doi.org/10.1109/msp.2011.941097>.