



**École Polytechnique**

*BACHELOR THESIS IN COMPUTER SCIENCE*

# **A very long and informative title for my thesis work**

*Author:*

Remi Guillou, École Polytechnique

*Advisor:*

Yanlei Diao, École Polytechnique

*Academic year 2024/2025*

## Abstract

The increasing amount of data being created and processed in all sectors has led to the use of automatic methods for filtering and analysing this influx of information. These methods often rely on complex models whose methods are intractable and as such act as "Black Boxes". Such techniques are used for monitoring traffic on many different types of servers. Anomaly detection methods are important as they make it possible to know in real time when an issue has occurred. However, in order to efficiently resolve any issue, as important as detecting an anomaly is understanding why the flagged interval is anomalous. This is where the field of Explainable AI comes into play. In this paper, we will improve Exstream, which is a method relying on the single feature entropy measure between normal points and anomalous points to determine features that contribute the most to the anomaly and the feature intervals where anomalies would happen. Our contribution is twofold, first we explored the possibility of sampling normal points using the latent space from the detection method. These points would be more representative of the current anomaly and wouldn't be correlated with time. We found that the domain dependency of the data makes this technique unfeasible for our type of data. Secondly, we showed that the scoring for different features isn't a submodular function due to correlated features but is simply increasing. We also generalized the entropy measure over multiple features which would enable the explanation of anomalies on more complex dataset where the interaction of two or more features is responsible for the anomaly.

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Related works</b>	<b>4</b>
<b>3</b>	<b>The Exathlon project</b>	<b>4</b>
3.1	Dataset . . . . .	4
3.2	Anomaly detection . . . . .	4
3.2.1	Divad . . . . .	5
3.3	Anomaly Explanation . . . . .	5
3.3.1	Original Exstream . . . . .	5
3.3.2	Exstream using Bins . . . . .	7
3.4	Metrics . . . . .	7
<b>4</b>	<b>Improving the sample set</b>	<b>7</b>
<b>5</b>	<b>Generalizing Exstream to Higher dimensions</b>	<b>7</b>
5.1	Submodular optimization . . . . .	7
5.2	Methods for generalizing Entropy . . . . .	7
5.2.1	Boxes . . . . .	7
5.2.2	KNN . . . . .	7
5.2.3	Decision Trees . . . . .	7
<b>6</b>	<b>Conclusion</b>	<b>7</b>
<b>7</b>	<b>References</b>	<b>8</b>
<b>A</b>	<b>Appendix</b>	<b>9</b>

# 1 Introduction

The surge in data

[6]

## 2 Related works

## 3 The Exathlon project

The Exathlon project was started in 2021 as an anomaly detection and explanation benchmark [2]. The github repository implements a pipeline that implements all needed steps from preprocessing of the data to training and evaluating methods for detecting and explaining anomalies. The full pipeline can be seen in [Figure 2](#).

### 3.1 Dataset

The pipeline supports any dataset and custom preprocessing steps can be implemented. For the sake of this paper we will be using a time series datasets consisting of traces taken from Apache Spark servers. These traces consist of real data collected from 93 repeated executions of 10 distributed streaming applications on a 4-node Spark cluster over a period of 2.5 months. Each of these executions includes 5 randomly selected applications running concurrently. In total, 2,283 metrics were monitored once per second creating a dataset totaling more than 24GB in size. For the sake of our experiments, the dataset went through some preprocessing, cutting it down to 237 features. It can be noted that this preprocessing step is done automatically in the pipeline. The dataset consists of 59 undisturbed traces and 34 disturbed traces constituting 97 anomaly intervals. There are 6 types of anomalies:

- T1: bursty input
- T2: bursty input until crash
- T3: stalled input
- T4: CPU contention
- T5: driver failure
- T6: executor failure

The data is processed and then turned into a *window dataset* which transforms the data into a sliding window format in order to detect the anomalies.

### 3.2 Anomaly detection

The anomaly detection consists in three steps. First training the *window model*. This model gives an anomaly score to a window of a certain size. Then following the "Unifying anomaly detection method" introduced by Vincent Jacob [1], we obtain an "Online" anomaly scoring function which given the scoring for windows, attributes a score for each point. This scoring function is used to determine a threshold above which points will be labelled as anomalies in the anomaly detection step.

Many different models belonging to various families of methods [5] are implemented and ready to be evaluated. Such methods include among others: pca, xgboost, Autoencoder, Variational Autoencoder, LSTM, deep SVDD, deep SAD, iForest...

The most recent and best model on this dataset is Divad.

### 3.2.1 Divad

Divad is an anomaly detection method based around a VAE architecture [1]. This method was create to address the difference in the behaviour of traces based on the parameters and properties of the Spark application. We will call "Domain" the context in which the application is run. This context is characterised by the following elements: processing period, number of active executors, memory profile and input rate.

The different domains pose a real problem when trying to identify anomalies. A certain behavior can be considered anomalous in one domain but not in another. ‘add plot’. The limited amount of data also makes it impossible to train a model for each domain. Therefore we need a method that can detect anomalies and generalize to new domains. This is Divad.

The way it works is by assuming two independent priors that define the data  $z$ . The first prior is the class  $y$ , either anomaly or normal. The second prior is one that encodes the domain of the point. Therefore we are assuming that the points can be generated from their class and their domain and that these two priors are independent.

‘insert image’ The objective is thus to approximate these prior spaces from the data. In order to do so, Divad employs a Variational Autoencoder structure with two independent encoders and one decoder. The first encoder is meant to capture the Class information and the second the Domain information. This is done using well chosen error functions.

We therefore end up with two latent spaces:  $z_y$  and  $z_d$  that contain respectively information about the class and the domain of the data. We then use these two latent space to reconstruct the data.

We get an anomaly score by comparing the class latent space with what we would expect from the normal class. This is done by computing the KL divergence between the two distributions.

## 3.3 Anomaly Explanation

The anomaly explanation step will be the main focus of this paper. The objective of this step is to give a human readable and actionable explanation of the anomaly. This can be done in many ways such as lists of feature importance or plots as we have seen in the related works. In this paper, the focus will be on the Exstream method. For each anomaly, this method will output the features that cause the anomaly as well as the intervals where the anomaly is happening.

The explanation is given as a boolean expression in Conjunctive Normal Form. It consists of a conjunction of clauses, where each clause is a disjunction of predicates. Each predicate follows the form  $\{v \circ c\}$ , where  $v$  represents a feature,  $c$  is a constant, and  $\circ$  is one of the five operators:  $\circ \in \{>, <, \geq, \leq, =\}$ .

For example an explanation for an anomaly could be the following:  $(\text{feature1} > 0.5) \vee (\text{feature1} < 0.8) \wedge (\text{feature2} = 0.7)$ .

There previously existed two versions of Exstream. The original version was introduced by Haopeng Zhang, Yanlei Diao and Alexandra Meliou in 2017 [7]. It was then work uppon by Mija Pilkaite during her Bachelor thesis [4] and finally refined and improved by Clement Martineau [3].

### 3.3.1 Original Exstream

Originally, Exstream was created to provide a rigorous and formal method for providing short and human readable explanations for anomalies. This method presents the challenge of finding the optimal explanation to an anomaly as a submodular optimization problem. The intuition behind this model is that adding features to our explanation has a diminishing return. This is because the more features we add, the less information new features will provide.

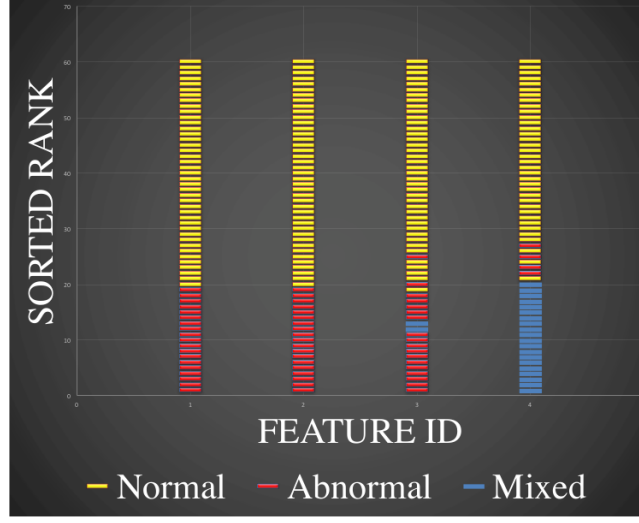


Figure 1: Visualization of the segments of 4 features. The red points are the anomalous points, the yellow points are the normal points and the blue are mixed. Segments are continuous points of the same colour.

Submodular optimisation being NP-hard, a heuristic was therefore introduced to solve the problem. This method relies on Entropy to compute single feature scores.

We start with two sets of points  $S_a$  and  $S_n$  containing respectively the anomalous and normal points. For each feature  $f_i$ , we will sort the points in  $S_a$  and  $S_n$  and merge them into one set. We will then define segments on this set as neighbouring points of the same class.

We will compute a segmentation entropy defined as follows: If there are  $n$  segmentations, and  $p_i$  represents the ratio of data points included in the  $i$ th segmentation, the segmentation entropy is:

$$H_{segmentation} = \sum_{i=1}^n p_i \log\left(\frac{1}{p_i}\right) \quad (1)$$

Exstream was originally aimed at categorical data. Therefore there can be features with values that belong both to anomaly and to normal points. We need to penalize these features. This is done by giving the worst score to those points of mixed class. Let  $c_i$  be a mixed interval and  $c_i^*$  be the segment rearranged in its worst ordering.

Then the maximum entropy is defined as:

$$H_{max} = \sum_{i=1}^n H_{segmentation}(c_i^*) \quad (2)$$

Finally, we normalize the score by the class entropy to get a value between 0 and 1. The class entropy is defined as follows: Let  $|S_a|$  and  $|S_n|$  be the number of points in  $S_a$  and  $S_n$  respectively.  $p_a = \frac{|S_a|}{|S_a| + |S_n|}$  and  $p_n = \frac{|S_n|}{|S_a| + |S_n|}$ . The class entropy is then:

$$H_{class} = p_a \log\left(\frac{1}{p_a}\right) + p_n \log\left(\frac{1}{p_n}\right) \quad (3)$$

The final score for a feature is then:

$$score(f_i) = \frac{H_{segmentation} + H_{max}}{H_{class}} \quad (4)$$

This score reflects the segmentation of the feature with respect to anomaly and normal classes. The higher the score, the more separated normal and anomaly classes are. For example a feature with a score of 1 would have all the anomaly points in one segment and all the normal points in another. As we can see on [Figure 1](#), the first two features have a score of 1. The third a score of 0.31 and the fourth 0.18.

Exstream selects the best features based on reward leap filtering. It also provides methods to filter out redundant features using correlation clusters. It also removes "false positive" features which are defined as those with excessively large standard deviation or that tend to only increase or decrease on both the normal and anomalous interval.

### **3.3.2 Exstream using Bins**

An attempt to improve Exstream was made by Mija Pilkaite during her Bachelor thesis [\[4\]](#). The first idea was to use bins to discretize the data. This would make the method more suited for continuous data. Each feature values would be divided into  $b$  bins. Each bin would be marked as either Normal, Anomalous or Mixed. The entropy would then be computed on these bins. The second idea was to remove uncertainty in the anomalous interval boundaries. Even when using ground truth segments with anomalous intervals marked by hand, there is always some uncertainty as to the data at the endpoints of the intervals.

In order to minimize this labeling uncertainty, the points would have different weights in the computation of the entropy depending on their position in the trace. This weighting would be done using a Gaussian distribution centered around the middle of the interval.

[\[4\]](#)

## **3.4 Metrics**

## **4 Improving the sample set**

## **5 Generalizing Exstream to Higher dimensions**

### **5.1 Submodular optimization**

### **5.2 Methods for generalizing Entropy**

#### **5.2.1 Boxes**

#### **5.2.2 KNN**

#### **5.2.3 Decision Trees**

## **6 Conclusion**

## 7 References

- [1] Vincent Jacob and Yanlei Diao. Unsupervised anomaly detection in multivariate time series across heterogeneous domains. *PVLDB*, 14(1):XXX–XXX, 2025.
- [2] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. Exathlon: A benchmark for explainable anomaly detection over time series. *PVLDB*, 14(11):2613–2626, 2021.
- [3] Clement Martineau. Advanced algorithm design for explainable anomaly detection on data streams. Internship report, Laboratoire d’Informatique de l’École Polytechnique (LIX), Team CEDAR (LIX - INRIA), July 2024.
- [4] Mija Pilkaite. Algorithm design for explainable anomaly detection for data streams, 2023/2024.
- [5] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *PVLDB*, 15(9):1779–1797, 2022.
- [6] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, s2-42(1):230–265, 1937.
- [7] Haopeng Zhang, Yanlei Diao, and Alexandra Meliou. Exstream: Explaining anomalies in event stream monitoring. pages 156–167, 2017.



## A Appendix

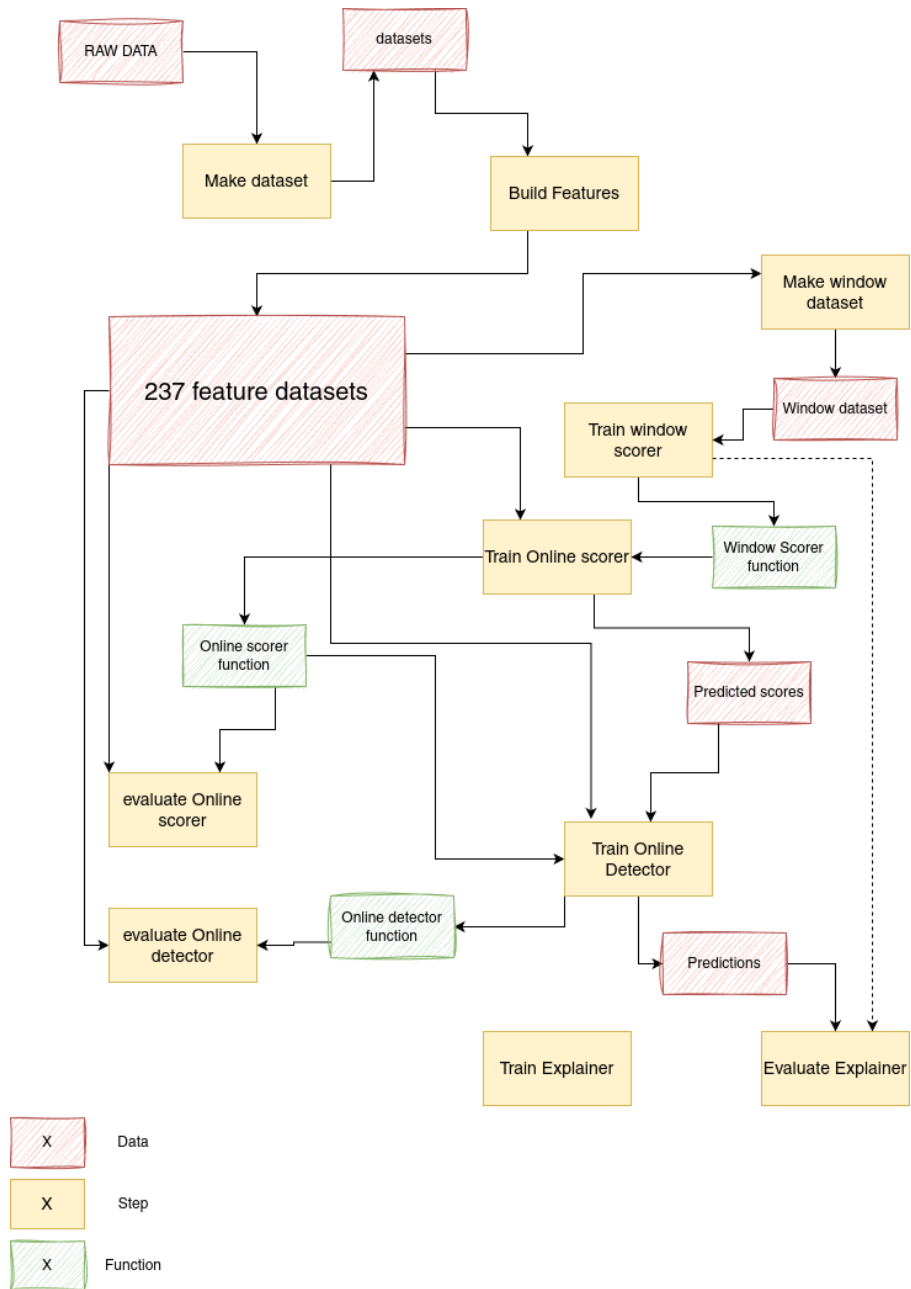


Figure 2: Pipeline of Exathlon