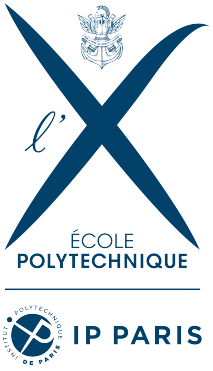


École Polytechnique

BACHELOR THESIS IN COMPUTER SCIENCE

Algorithm Design for Explainable Anomaly Detection for Data Streams

Author:

Mija Pilkaite, École Polytechnique

Advisor:

Yanlei Diao, CEDAR, LIX

Academic year 2023/2024

Abstract

In the growing use of Spark applications and data streams in business, financial markets, and healthcare, human power is not sufficient for observing anomalies within the data. Moreover, detected anomalies require quick solutions to avoid losses. For that to be achieved, explanation discovery, a field of Explainable AI, becomes a crucial tool for the industry. In this study, we explore a framework for explanation detection called ExStream, which aims to provide features that could be the reason for the anomaly as an explanation and returns predicates for the range of values that could be the cause. We improve the algorithm to fit high-dimensional time series and improve it to be more generalizable and applicable for different domains of problems. After the mentioned improvements, the model correctly provides explanations and the anomalous segments, successfully ignores the labelling noise within the segments. False positive filtering filters out the features that act abnormally through more than just anomalous segments and thus are not correlated to the simulated anomaly. The work extends ExStream to successfully explain anomalies on high-dimensional time series with suggestions for future improvements and uses different evaluation metrics to assess the performance.

Contents

1 Introduction

The exponential growth of data nowadays has commanded the development of sophisticated tools for its analysis and management. In particular, anomaly detection and explanation generation for high-dimensional time series data have been shown to be critical [12]. Given the amount of data and processes, even minor performance anomalies or errors in these jobs could cause significant delays and have a major financial impact. This growing amount of data requires algorithms not just capable of detection but also providing valuable and interpretable explanations for anomalies detected. This is particularly important in environments where the timely detection of such anomalies can prevent potential financial losses, security breaches, or system failures [20, 7]. The relevance of anomaly detection is evident across multiple domains, including finance, where it aids in spotting fraudulent transactions [23]; healthcare [26], where it helps in the early detection of disease outbreaks or adverse patient conditions; the Internet of Things (IoT) [8], where it ensures the smooth operation of connected devices; and in application monitoring [14], where it assists in maintaining the integrity and performance of software applications. The critical role of anomaly detection is underscored by its use in large-scale data processing platforms such as Apache Spark, utilized by major organizations like Amazon, Alibaba, and eBay for running analytics jobs on massive datasets. These platforms require the rapid processing of data to ensure the relevance of results, making the detection of even minor performance anomalies or errors crucial to avoid significant delays and financial repercussions. The results are used for sales strategies, inventory management and to improve user experience [9].

Building on the previous advancements, this study focuses on ExStream [25] and proposes enhancements to the algorithm to improve its accuracy, make it adaptable to different datasets, and less susceptible to labeling noise. This paper aims to adapt ExStream on Apache Spark deployments, where we analyze metrics of Spark jobs and use the anomalies detected to generate explanations.

1.1 Challenges

So far, the data mining community has mostly focused on the anomaly detection part (presto, telegraph, spot), where an anomaly is defined to be an unexpected pattern within the data [6]. This field itself faces a set of challenges due to the complicated nature of this task and the limitations of the implementations. Especially in high-dimensional data, there is a high variation within the normal data, thus making it hard to define the normal and anomalous intervals. Feature ranges can change due to non-anomaly-related reasons, thus complicating the establishment of the benchmark. Furthermore, due to the aforementioned large volumes of data, there is a lack of human-labeled and human-evaluated data, thus complicating the establishment of the evaluation metrics and slowing down the advancements of the models. Finally, even if the anomaly is detected and recognized, the use cases of that are rather limited. The user cannot easily identify the reason for the anomaly, cannot easily come up with the solution nor gets any knowledge of how to prevent it in the future [6]. Thus, interpretable machine learning and explanation generation have been used to remedy some of these issues [25, 7].

Explanation generation is still underexplored and a new area of study, thus providing many challenges. This domain has been mostly explored as a tool for explanation generation for SQL queries, which is not quite applicable to arbitrary data, especially high-dimensional time series.

As the amount of data is growing exponentially, human power cannot keep up and provide data labeling, thus making it hard to evaluate the model's performance and accuracy. Also, the goal of every explanation generation model is to provide *good* explanations, which are defined as satisfying these requirements [25]:

- *Conciseness* - smaller, shorter explanations are favored so that the explanation would be easier to understand for humans.
- *Consistency* - initially, the criterion of consistency was interpreted as the alignment of the model's explanations with those provided by humans. However, large volumes of data were impossible to evaluate

by the human mind with current resources, thus making the metric unfeasible. Consequently, the definition of consistency has been refined to reflect the model's capacity to deliver uniform or analogous explanations for identical or comparable types of anomalies and behaviors.

- *Prediction power* - explanations should have predictive value for future anomalies.

1.2 Contributions

So far, ExStream has been applied to rather small datasets, based on SASE queries (ExStream). This study will focus on ExStream performance exploration on a larger and a different domain dataset and improvements to make it more extendable between the domains.

1. **ExStream's performance exploration on high dimensional time series dataset.** This study aims to discover its applications in different domains. Time series is one of the most commonly found types of data nowadays, thus it is crucial to check the adaptability of these models on this type of data. We check ExStream's capability to generate accurate segments and logically assess the magnitude of the reward attributed.

2. **Changing the implementation of segmentation.** As of today, ExStream has considered mixed segments to be those with values in both reference and anomalous intervals. However, for almost-continuous variables, such as high-dimensional time series, that is incredibly hard to achieve, when a feature has a big scale. Thus, we implement equi-width binning to avoid inaccuracies and control label noise.

3. **False-positive filtering implementation.** For some of the features that are chosen as an explanation, we see that anomalous values could be interpreted as simply growth over time or be dependent on other reasons that are not anomaly-related.

4. **Evaluation.** We evaluate the performance of a new model graphically and using two other techniques - comparison with MacroBase [2] and Exathlon [13] evaluation metrics.

2 Related Works

Outlier explanation in SQL Query Results. Scorpion [24] is a well-known framework that significantly contributes to anomaly detection in database systems by creating explanations for outliers in group-by-aggregate queries. It relies on users marking outliers within the results of group-by queries, after which Scorpion searches for predicates that could explain those outliers. The goal is that they should not disrupt the overall patterns or insights drawn from the reference data. The framework is particularly useful for analyzing reports and summaries, as it is well-fit for its domain. However, its application is limited to specific data structures and query types, making it ineffective for problems outside group-by-aggregation queries and failing at broader data context explanations. Manual user input makes the process not automated and rather subjective. Recent studies have aimed to improve Scorpion's explanatory capabilities. For example, Roy et al. [19] use precomputation and user-defined explanation templates to enable interactive explanation discovery, offering more detailed insights but facing challenges in flexibility and adaptability to new data scenarios. Thus, neither of the frameworks is suitable for the problems ExStream is addressing.

Outlier explanation in Data Streams. Besides ExStream, which will be discussed in more detail in section 3, MacroBase [2] is a model with both anomaly detection and explanation generation modules. For anomaly detection, it uses a density-based method MAD, which captures point outliers but cannot detect contextual or collective outliers, which are frequently found in time series and data streams [6]. Developed for efficiently prioritizing user attention towards unusual or significant data patterns, it acts as a search engine for fast data. For explanation discovery, MacroBase compares the features of outlier points against the norm (non-outlier

data points) to find which characteristics are the most indicative of this outlier behavior. It employs contrast analysis and identifies the most significant features (the ones with the most noticeable differences). Then, it aggregates and summarizes the identified significant features to return combinations of feature values as an explanation associated with the outlier behavior. MacroBase is also capable of prioritizing explanations based on their frequency and impact to help users identify the most important insights. It has been successfully deployed in real-life settings, including at a telematics company monitoring hundreds of thousands of vehicles [1]. Its use within the industry proves the necessity of such techniques in data stream analysis. The limitation of Macrobase is its limited capability to analyze a large number of features and a slow run time for larger datasets. More recent techniques [5] involve utilizing Deep Learning to better capture the possible explanations for data outliers. BALANCE [7] applies concepts from explainable AI (XAI) but does so to attribute changes in KPIs (key performance indicators) to specific features or components within the system's data. It uses Bayesian methods to select relevant features and compute attribution scores for these features, determining their contribution to the observed anomalies. This process helps to pinpoint the underlying causes of issues within the data, facilitating targeted actions for system recovery and maintenance.

Anomaly detection. Anomaly detection is a crucial step in explanation discovery. It has been researched more broadly than the explanation step and is adapted for a range of data with different characteristics and types [6, 5, 11]. One can use simple statistical methods [3], distance-based [22], density-based [4], and deep learning methods [5]. The statistical methods rely on assumptions that data follows a certain distribution and use the parameters of that distribution to detect anomalies. Distance-based methods detect anomalies by consideration of the distance or similarity between the points. For density-based detection, methods estimate the density around each data point, and points within a region of significantly lower density are considered to be anomalies. Deep Learning methods use neural networks to model complex, non-linear relationships in data and thus can be used to identify anomalies within high-dimensional datasets, such as ours. As the study is focused on explanation discovery rather than anomaly detection, it is only broadly discussed here as a crucial part of Explainable AI, however, the techniques used are outside of the scope of this paper.

Interpretable Machine Learning. With the increased use of Machine Learning models and Artificial Intelligence, there has been a growing need for explanations of why certain decisions are made by the model. This field has been gaining traction in recent years [16, 17]. Here, we focus on some classical and state-of-the-art model explainers that have been widely used within the Explainable AI community. These techniques have two broad families - interpretable models and model-agnostic approaches. Interpretable models are more understandable due to their structure and the process of decision-making. Some of these include linear regression, logistic regression, decision trees, and rule-based models. Decision-making is a transparent process, making it easy to understand why certain decisions were made. However, these models are quite limited and cannot capture complex patterns in the data. Model-agnostic approaches treat the model as a black box and hope to understand its behavior through the model's inputs and outputs [16]. One of the most used model-agnostic approaches is LIME (Local Interpretable Model-agnostic Explanations) [18]. It operates under the premise that while global model interpretability is often challenging, local interpretability—understanding why a model made a specific prediction—is more feasible and equally valuable. LIME generates explanations by approximating the original model's behavior in the vicinity of the prediction with a simpler, interpretable model (such as a linear model) that highlights the contributing factors for that particular decision. Also, some newer techniques emphasize the creation of intuitive and actionable insights into model decisions [10]. This model-agnostic approach transcends traditional explanation methods by not only revealing the importance of input features but also illustrating how alterations to these inputs could change the model's predictions. By focusing on counterfactual scenarios—hypothetical changes to the input that would lead to a different outcome—the technique offers a practical framework for understanding complex decision-making processes in machine learning models.

3 Project Setup

3.1 Dataset

The dataset was retrieved from real data traces from applications deployed in Apache Spark clusters. Then, 3 different types of anomalies were simulated on 6 different applications and recorded over a set time period. In the dataset, we are provided with the long data trace, each having over 11,000 entries, and timeframes for reference (normal) periods of data and the timestamps when the anomalies were simulated. We include a trace for one of the features as an example in 1. Spark jobs run on huge amounts of data daily and are used for business analysis, sales strategies, and user experience improvement. Timeless detection of anomalies is crucial in this setting to avoid monetary losses and meet project deadlines [9]. In this study, 6 data traces were used each representing a different type of anomaly. The traces were constructed by injecting anomalies within these applications and we have 4 different types represented:

- *Bursty input (type 1):* represents scenarios when the data input rate is much larger than usual.
- *Bursty input until crash (type 2):* larger data input until the executors crash due to usage of memory.
- *Stalled input (type 3):* represents a period when there is no data input due to some issues with a data source.
- *CPU contention (type 4):* a period when multiple programs are competing for the CPU resources. The congestion causes scheduling delays.

The goal of this study was to test ExStream's adaptability to high-dimensional time series. Moreover, as ExStream requires the user to provide labels for reference and anomalous periods to use as the ground truth, this technique of dataset generation proves to be a great fit. In figure 1, we can see a trace for feature 0 (*driver_BlockManager_memory_memUsed_MB_value*) for bursty input anomaly.

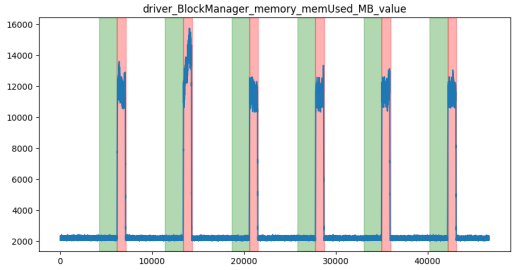


Figure 1: Feature 0 trace for bursty input

We will reference the whole period of collection of data as a *data trace*. Then, we have the trace divided into *intervals*. The *interval* consists of two *periods* - one normal (reference) and one anomalous. We are given the labels for intervals, as ExStream requires the user's input for good and anomalous data points. In figure 1, we have a whole trace depicted with normal (reference) and anomalous periods in green and red respectively, which form an interval. These are the definitions we will keep consistent throughout the report. In our dataset, as a general rule, the reference period is twice the length of the anomalous period.

3.2 ExStream's Architecture

The framework focuses on explaining anomalies in event stream monitoring and offers a sophisticated approach to explaining anomalies in real-time Complex Event processing systems (CEP). The model consists of first creating a sufficient feature space that must contain all necessary features to explain the detected anomaly.

Then, the importance of each feature is calculated through an entropy-based distance function, which computes each feature's contribution to the explanation and picks those that best explain the difference between normal and anomalous data. The inspiration for this technique stems from the concept of entropy in information theory, which measures the randomness and *chaos* of a system. We have two key concepts - class entropy and segmentation entropy. The first is calculated based on the proportions of normal and anomalous points within the anomaly and their mixture. The latter (segmentation entropy) represents the segments within the range of the feature (the distribution of normal, mixed, and anomalous segments). If the values are more mixed (a value appears both within the normal and anomalous range), it contributes to higher entropy thus punishing the features with a high segmentation rate. Finally, a penalty is added for every mixed segment to account for the lack of separation. ExStream computes the worst-case scenario of separation (points within the mixed segment being distributed uniformly) and adds the penalty. Then, we compute the single feature reward. Mathematically, the formula can be expressed as:

$$H_{\text{Class}} = p_A * \log(\frac{1}{p_A}) + p_N * \log(\frac{1}{p_N})$$

$$H_{\text{Segmentation}} = \sum_{i=1}^{n} p_i * \log(\frac{1}{p_i})$$

where p_A is a ratio of anomalous points within the segment, p_N is a ratio of normal points within the segment, p_i is a number of both normal and anomalous points within the segment and n represents the number of segments for a feature. The reward of the feature is computed by:

$$R(f) = \frac{H_{\text{class}}}{H_{\text{segmentation}} + H_{\text{worst-case penalty}}}$$

Grouping these features and picking k-top features is not enough, as they could be highly correlated, be flagged as anomalous for different reasons, and because of submodularity simply not add to the reward. Thus, ExStream solves the submodular optimization problem, where the goal is to pick an optimal amount of features that offer the best explanation. This symbolizes maximizing the information reward of the explanation, as the submodularity problem has a property of diminishing returns. As this problem is NP-Hard [15], ExStream uses different techniques to approximate the possible solution. For that, we employ 3 techniques:

1. Reward Leap Filtering - as the features are ranked in the order of their reward, we monitor for the sharp increase in reward value and eliminate features that are below a steep jump under the assumption that they most likely will not meaningfully contribute to the explanation.

2. False Positive Filtering - some features might have high rewards because of unrelated reasons to the anomaly. ExStream aims to use a self-supervised learning technique to apply the generated explanation to similar anomalies and hopes for the explanation to be consistent.

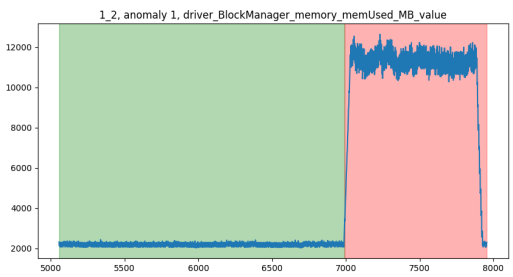
3. Filtering by Correlation Clustering - some features might be redundant even though they have high rewards, as they might provide the user with similar information. The model clusters correlated features and returns the one with the highest reward out of the cluster.

ExStream is a state-of-the-art technique in a newly developing field of interpretable machine learning. It can be a life-changing advancement in fields such as medicine, the Internet of Things, financial crime, and many more. The goal of the study is to make the model more robust, achieve more accurate results, and check its adaptability on different datasets.

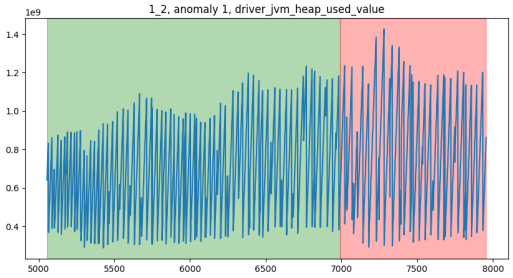
4 Model performance and evaluation

In this chapter, our focus is on ExStream's performance on high-dimensional time series and discovering its limitations and potential improvements. As per the dataset section, we have 6 traces of Spark application-generated data.

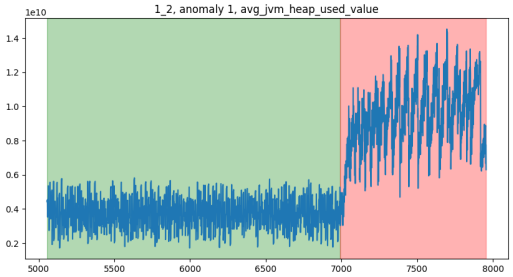
For reference, we will use 4 different features to represent the best and poor choices of the model and illustrate the data distribution. Initially, the analysis will be discussed for trace 1_2 as it is the most apparent one visually and the best fit for a data explainer, such as ExStream. In the following graphs, we have a normal period in green, an anomalous period in red, and the trace of the specific feature during a specified timeframe in blue. In figure 2 we can see some of the features for anomaly 1 for trace 1_2.



(a) Feature 0 for anomaly 1 of trace 1_2



(b) Feature 1 for anomaly 1 of trace 1_2



(c) Feature 2 for anomaly 1 of trace 1_2

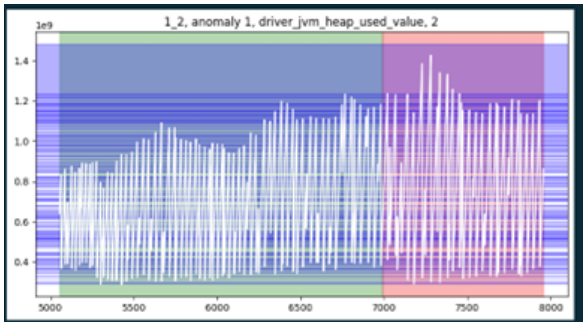
(d) Feature 4 for anomaly 1 of trace 1_2

Figure 2: Bursty input features

The initial ExStream model had an implementation for false-positive filtering that removed features with high standard deviation (the threshold given as one of the hyperparameters) and those that are constantly increasing or decreasing within the normal range. The first evaluation is done with this method, but it is removed for further analysis as it does not represent the desired technique for feature removal described in the paper.

After the initial run for anomaly 0 of a trace 1_2, ExStream provides us with an explanation: feature 1 as in figure 2b with a reward of 0.09 and feature 4 as in figure 2d with a reward of 0.06. The model returns the feature with the corresponding reward and the intervals that it considers to be anomalous or mixed (thus providing us with a predicate of the feature's range responsible for the anomaly), which due to the output length could not be included here. As we can see, we have one feature picked that visually looks anomalous (figure 2d feature 4) and another one that is the least anomalous out of the ones provided. As similar results were generated for all of the traces and the anomalies, it was apparent that some model improvements were needed to achieve good performance on this type of data.

The first step was to explore the returned segmentation of the models and as well determine the reasoning for these low rewards (for highly anomalous and minimally segmented features it should be close to 1.0). Figure 3 depicts the returned anomalous ranges (segments) for the explanation features of trace 1_2.



(a) Anomalous segments for feature 1 for anomaly 1 of trace 1_2

(b) Anomalous segments for feature 4 for anomaly 1 of trace 1_2

Figure 3: Provided anomalous segments by ExStream's explanation

In the provided graphs, we have a normal period in green, an anomalous period in red, and a trace in white. Blue bins are supposed to be read horizontally, as they depict the ranges of anomalous values for this feature. As we can see, we get a huge amount of segments returned for these features, which can account for the low reward for these explanations (as there is a lot of segmentation entropy). Important to notice is that the segments, especially for feature 1 do not seem to correlate with the visual truth. This could potentially be the reason for the poor performance of the model. Exploring the other features, the same pattern was persistent, where each feature had an excessive amount of segments and most had one value each. The solution to this issue will be discussed in Section 5.

Moreover, another important detail to notice is that ExStream does not perform well with features that are constructed as differences (otherwise known as *diff* features). For anomaly detection algorithms in general it is better not to have constantly increasing or decreasing features, thus some features get converted to depict the difference between two consecutive values. This is the desired approach for anomaly detection but for these features when the value remains constant, we have it drop to 0 at the graph. As this value becomes common throughout the graph, the segment is considered heavily mixed and thus gets penalized a lot and usually cannot be chosen as an explanation. This led us to the discovery of another limitation of ExStream, as it cannot process the *diff* features but they are a common occurrence in anomaly detection algorithms.

Furthermore, an interesting topic to explore was why *feature 0* was not chosen as an explanation for this anomaly as it looks highly anomalous. At the rightest end of the anomalous range, one can discover that the feature has gone back to normal. Therefore, these last points can be labeling noise, where the anomalous period should have ended a little earlier. As the standard deviation of the normal data is very small, most of the values within the normal range become mixed. This amounts to a significant amount of values considered mixed and the worst-case segmentation penalty brings down the reward. The problem of labeling noise can be found in more features throughout the dataset and the model is highly susceptible to this phenomenon. As ExStream takes the user's input for labeling normal and anomalous intervals, the labeling noise can be a common occurrence, thus this limitation has to be solved.

Another limitation of ExStream can be found depicted in Figure 4.

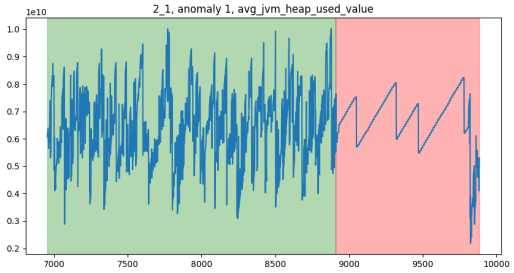
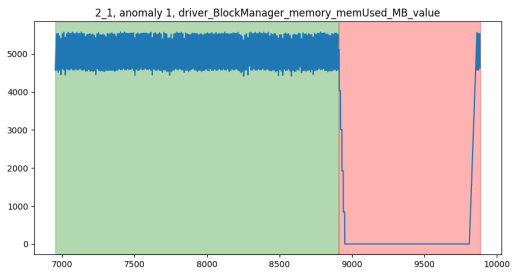


Figure 4: Feature 3 for anomaly 4 for trace 2_2

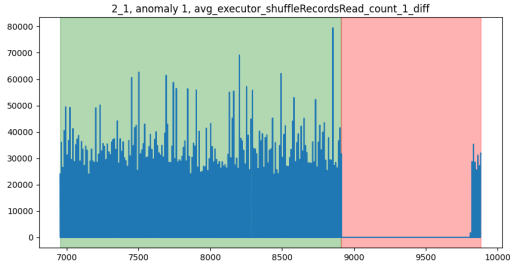
ExStream, due to its nature, cannot detect even very highly anomalous features if the anomaly of the data lies in the frequency and not the range itself. Not only is this feature not chosen, but also for this case its reward is *Feature 3 reward: 0.01730913941832158*. Most of the values within the anomalous range are the exact same ones as the normal values, thus making this feature undetectable for the explanation generation.

Similar model behavior can be seen throughout different traces for all different anomalies. For bursty input anomalies, the model always chooses feature 1 as in 2b as the only one or one of the explanations. For stalled input, as shown in figure 5, we get 3-4 features per explanation and they have very low rewards. In most cases, they are features 1, 7, 8. As we can see, feature 1 does not look graphically anomalous and features 7 and 8 are very similar. Other *diff* features within the dataset have also similar appearances but do not get chosen.

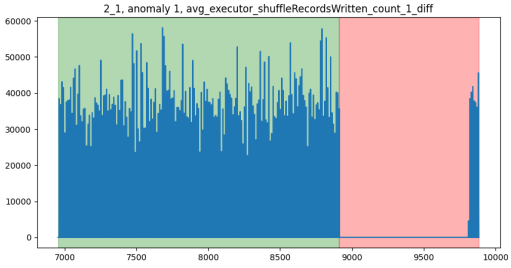


(a) Feature 0 for anomaly 1 of trace 2_1

(b) Feature 1 for anomaly 1 of trace 2_1



(c) Feature 6 for anomaly 1 of trace 2_1



(d) Feature 7 for anomaly 1 of trace 2_1

Figure 5: Stalled input features

This is because ExStream considers a value to be mixed if the exact same value appears in both normal and anomalous regions. As we have noticed for some features, the scale can be as large as 10^{10} making the probability of the same value appearing twice rather small. Thus, features with smaller scale and small standard deviation rarely get chosen thus limiting the model from making the best anomaly explanation.

Another drawback is that mixed segments within a feature get punished in the exact same way not taking into account the actual distribution of normal and anomalous values. This means that if we have two mixed segments of 10 points each, one with 5 normal and 5 anomalous values and one with 1 normal and 9 anomalous values, the punishment for this segment is the same. This accounts for the model's susceptibility to label noise

and could reduce the accuracy of picking the best features.

Finally, as mentioned above, false positive filtering that removes high standard deviation and constantly increasing or decreasing features does not achieve the goal of actually removing false positive features. In the case of trace 1_2, feature 2 gets removed due to its high standard deviation. If not removed, ExStream for the bursty input chooses features 1 and 2, where feature 2 has a higher reward than the previous explanations.

To sum up, these are the main limitations of the current ExStream implementation:

1. Susceptibility to label noise - in real-life data, we can expect some label noise, especially as ExStream relies on the user to provide the timeframe for the anomaly. The current implementation interprets all points as of equal weight, which will be fixed in the later sections.

2. Definition of a mixed value - ExStream considers a mixed value to be a point that appears in both reference and anomalous periods. This is not realistic for features with big scale that are nearly continuous, such as time series. Thus, we will redefine the notion of a mixed value.

3. Performance on high-dimensional time series - as of the initial run, ExStream could not provide the correct explanations for the simulated anomalies. With a set of changes applied to the algorithm, we aim to improve its performance.

4. False-positive filtering - some features might behave abnormally for reasons that are not connected to the anomaly. These are not correct explanations and as a rule, we aim for the explanations to be generalizable. In this study, a partial solution will be addressed for this and the implementation described in the original paper will be left for future works.

5. Frequency anomalies - as of today, ExStream is not capable of detecting frequency-based anomalies, only the value range ones. This limitation will be left to be addressed in future works.

In the next sessions, we will address these limitations of the model and propose possible solutions for labeling noise, high segmentation within the features, appropriately weighing the degree of mixing within mixed segments, and an alternative false positive filtering.

5 Improvements of the model

In this section, we will address previously discussed weaknesses of the framework and propose solutions that aim to address those issues. We will discuss the results and further possible improvements to adapt this technology to real-life problems.

5.1 Segmentation and the concept of a mixed value

As mentioned above, a value is considered to be mixed if it appears both in a normal and an anomalous interval (thus, we need to have the same value appear at least twice within the data trace). This is not the best approach for nearly continuous data such as time series. When features such as those present in Figure 2b and 5b appear, that have a large scale, the values never actually overlap. We have a penalty for segmentation, as most of the segments formed have 1 or 2 values each. However, in the model's ExStream reward, the penalty for a mixed segment is significantly larger than that for segmentation. To mitigate this problem, we introduce binning. The choice was to use equi-width bins and not equi-depth bins to discretize the space in a less sensitive way to differences in standard deviations. With this implementation, the range of every feature was divided into a certain number of bins (the number of bins was established as a hyperparameter for ExStream, thus making it dependent on the user for maximum adaptability of the model). The values from the user were still used as the ground truth. A bin was considered to be anomalous if all the values within the bin were anomalous in

the user's input, normal if all were normal, and mixed if there was any mixture between the values appearing within the bin.

Even though this approach improved the segmentation problem, bins were prone to be mixed as the values were relabelled in the case of the appearance of one different value. As an example, even though out of 10 points within that bin 9 would be normal but 1 anomalous, we would have 10 mixed values, even though in the user's input they are not considered to be mixed. This approach made the model even less resistant to label noise as each mixed segment has the same impact, not taking into account the degree of mixture within, the model did not experience significant improvements.

To address this, the solution of considering the ratio of the points within the bin was introduced. The first approach that was implemented was computing the percentage of anomalous points within the bin:

$$p_{\text{anomalous points}} = \frac{\text{Number of anomalous points in the bin}}{\text{Number of anomalous points in the bin + Number of normal points in the bin}}$$

Then, a threshold was established. It was chosen arbitrarily to test a smaller penalty for mixed segments with lower entropy. The rule chosen was as follows: if less than 20% of the points within the bin were anomalous, the bin is treated to be normal. If the ratio of anomalous points falls between 20% and 80%, we consider the bin to be mixed. Finally, if the bin is more than 80% anomalous, we attribute it to anomalous bins. This approach relaxed the notion of a mixed segment thus introducing more leniency for features with labeling noise. We can explore the new segmentation in the previously discussed trace 1_2. Now, for every anomaly in both bursty and stalled input types, feature 0 becomes our explanation. In figure 6, we can see that the segmentation also significantly improves as now the model successfully ignores the label noise, where the data has already come back to normal in the anomalous region, such as in feature 0 in figure 5a. As well, the algorithm focuses more on general patterns rather than individual points, thus generalizing much better.

However, with this approach, we encounter two main issues. First, the thresholds are set and are quite ambiguous making it not the best method to expand the adaptability of the model. Also, even though weighing helps, the rewards are still not quite accurate as the mixed segments are still punished the same not considering the entropy within. The second drawback of the threshold solution is that for some of the features, the reward now exceeds 1. This is because the entropy of the class is higher than that of the two segments we retrieve at the end. The last 28 *noisy* points get assigned as normal, thus making the division between the classes even stronger and thus giving the segmentation less entropy. Now, as there are no mixed segments, this causes the reward to go slightly above 1. While this issue might not seem as significant, the goal is to keep the initial assumptions of the framework, thus we address these limitations with further advancements. The important takeaway from this method is that the binning technique and reducing the segmentation should be retained in future approaches and simply generalized better as it solves most of the issues within the model.

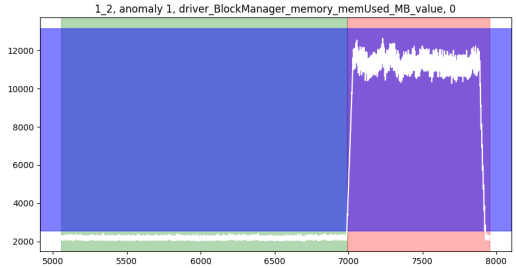


Figure 6: Improved segmentation

5.2 Penalty weight for mixed segments and label noise

This section will aim to address the equal treatment of the mixed segments and find a way to reduce the impact of labeling noise with a more generalizable method. We introduce weights for each bin based on the position

of points on the time axis.

With this technique, the goal is not only to base the penalty of the segments on the degree of their mixture but also to introduce weighing points based on where they are on the time axis. As we have explored before, most of the labeling noise stems from the edges of the intervals - at the beginning, the intersection of normal and anomalous data, and at the end of the anomalous period. Thus, mixed points in the middle of an interval and mixed points at the edges should have different impacts on the reward of the feature.

We explore a function, that would assign an appropriate weight to each point depending on its position on the time axis. We also want a hyperparameter so that a user can regulate the amount of expected label noise.

We chose two functions for this task and checked their adaptability.

Gaussian function with *sigma* as a hyperparameter. The *sigma* is given as a percentage of the total interval length so that we can adjust it for normal and anomalous periods, which have different lengths in our dataset. We use the expression below to compute normalized sigma.

$$\text{normalized\_sigma} = \frac{\text{total\_length} \times \sigma}{100}$$

We compute the center of the interval, which must be separate for anomalous and reference periods.

$$\text{center} = \frac{\text{total\_length}}{2}$$

The position of the sample is then normalized to be relative to this center by subtracting the center from the position.

$$\text{relative\_position} = \text{position} - \text{center}$$

Finally, we compute the Gaussian penalty on the time axis using the formula for a Gaussian function. We will denote this measure as w_t for the weight based on time.

$$w_t = e^{-\frac{\text{relative\_position}^2}{2 \times \text{normalized\_sigma}^2}}$$

This formula assigns the highest penalty score (1.0, since $e^0 = 1$) to positions at the center of the interval and lower scores to positions further away, with the rate of decrease determined by σ. Combining, we get:

$$w_t = e^{-\frac{(\text{position} - \frac{\text{total\_length}}{2})^2}{2\left(\frac{\text{total\_length} \times \sigma}{100}\right)^2}}$$

The distribution for different sigma values is depicted in figure 7. The graph represents how strong the penalty is depending on the point's placement on the time axis. The larger the σ value indicates a wider curve, and therefore less labeling noise should be expected within the dataset.

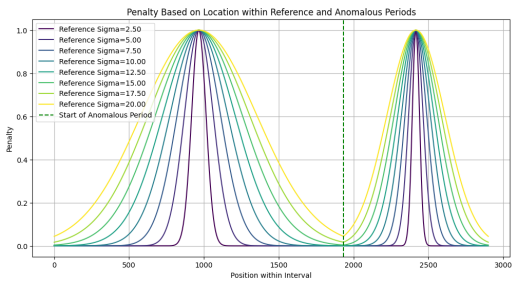


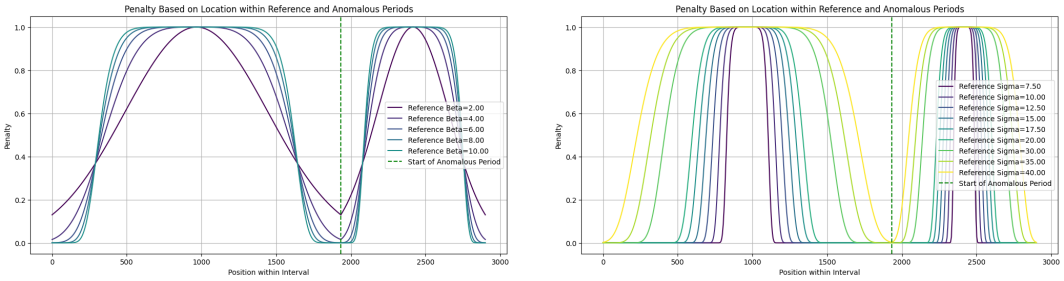
Figure 7: w_t distribution based on a sigma value

For every point within the bin, we assign a weight based on its position on the time axis. Then, the probability of a bin being anomalous p_a is computed.

Another approach is to use *an adapted generalized normal (Subbotin) distribution* [21]. It keeps the properties of the generalized normal distribution that is normalized such that its peak value is exactly 1 at the center of a specified interval and drops to 0 outside that interval. It retains the shape with the peak sharpness and tail heaviness controlled by the beta parameter, but it is not a probability distribution since it does not necessarily integrate to 1 over its entire range.

The formula we use to calculate the weight of the point w_t based on the time domain is the following:

$$w_t = \begin{cases} \exp\left(-\left(\frac{|position - \frac{interval\_length}{2}|}{\sigma \times \frac{interval\_length}{100}}\right)^{\beta}\right), & \text{if } 0 \leq position \leq interval\_length \\ 0, & \text{otherwise} \end{cases}$$



(a) Changes based on beta values

(b) Changes based on sigma

Figure 8: Adapted generalized normal distribution

The changes within distribution based on β andσ are depicted in figures 8a, 8b.

This plateau-like distribution fits the problem better, as by definition labeling noise is a small part of the data that affects the ends of the distribution. Thus, the Gaussian descent is rather too sharp to fit the problem.

As each point has its attributed weight based on its temporal placement, we compute the probability of a bin being anomalous p_a. It is also important to note the *class imbalance*. In general, anomalous data is much rarer than reference data in any real-life time series. We must address the class distribution in the computations. Thus, we compute the proportion of one class within one interval (prior probability).

$$pr_n(\text{normal proportion}) = \frac{\text{number of normal points within the interval}}{\text{number of total points within the interval}}$$

$$pr_a(\text{anomalous proportion}) = \frac{\text{number of anomalous points within the interval}}{\text{number of total points within the interval}}$$

We use the formula to address the class imbalance by multiplying each sum of weights by $1 -$ class prior:

$$p_a = \frac{sum(\text{anomalous weights}) * (1 - pr_a)}{sum(\text{anomalous weights}) * (1 - pr_a) + sum(\text{normal weights}) * (1 - pr_n)}$$

This method effectively accounts for the ratio of normal and anomalous points within the bin and takes into consideration the class imbalance. For the dataset from this study, there is not a large imbalance, however, if it was larger in different scenarios, this must be addressed in the new approach. Now the objective is for each bin to attribute a number that would signify a *probability of a segment being mixed*. Thus, using the probability of a segment being anomalous, we want to compute the probability of being mixed. The goal is to construct a function with a peak at 0.5 (as it would mean 50% anomalous, 50% normal, the biggest mixture within the bin) that converges to 0 for values close to 0 and 1 (as then the bin is nearly one type).

We introduce the binary entropy function in figure 9.

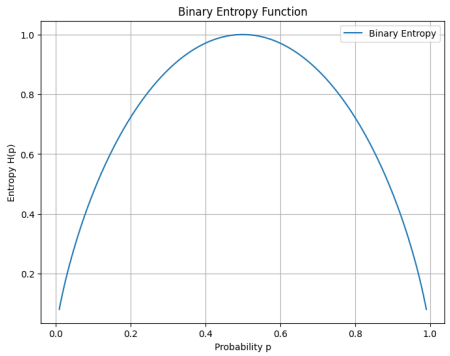


Figure 9: Binary entropy function

We compute the binary entropy using previously computed p_a by:

$$H(p_a) = \begin{cases} 0 & \text{if } p_a = 0 \text{ or } p_a = 1, \\ -p_a \log_2(p_a) - (1 - p_a) \log_2(1 - p_a) & \text{otherwise.} \end{cases}$$

Now, $H(p_a)$ is our probability of a segment being mixed and each bin is assigned its value with anomalous and normal bins having a value of 0. Then, the algorithm proceeds to create segments. If two consecutive bins are of the same type, we join them. In case the type is mixed, their probability of mixture is assigned to be the weighted average of their original scores $H(p_a)$. We compute the score by the following:

$$\text{Weighted Average Ratio} = \frac{\sum_{i=1}^{n} (\text{ratio}_i \times \text{num\_points}_i)}{\sum_{i=1}^{n} \text{num\_points}_i}$$

where $ratio_i$ is the ratio of the i-th interval within the mixed segment, num_points_i is the number of points in the interval within the mixed segment, n is the total number of segments combined into the mixed segment.

Finally, we compute the single-feature reward as in the original ExStream implementation. However, in the original implementation, the penalty for a mixed segment was assigned to be the worst-case ordering of the points computing the maximum entropy. Now, that penalty is multiplied by the segment's probability of being mixed $H(p_a)$. Single feature reward transforms into:

$$R(f) = \frac{H_{\text{class}}}{H_{\text{segmentation}} + H(p_a) * H_{\text{worst-case penalty}}}$$

Thus, segments with lower scores are punished less, whereas the highly mixed segments are punished more harshly.

To sum up, let us provide with some pseudocode to summarize the new approach. We will provide the reader with both more code based description and more wordly description.

Now, let us explore the results graphically. We will focus on a model with a σ value of 35 and divide our feature range into 100 bins and have a β value of 10. These values should accurately capture the distribution within the data and discriminate the edges just enough to exclude label noise. ExStream's framework should choose the feature with the highest deviations from normal to anomalous data. It also ranks the rewards of the features and returns those after the steepest growth in the reward values. Let us explore the returned features.

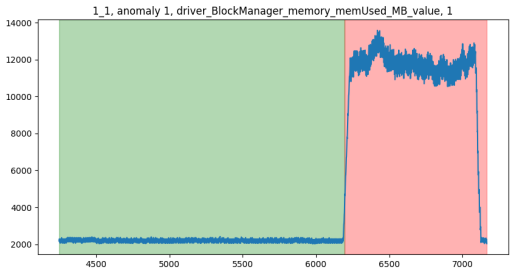
For traces 1_1, 1_2, 2_1, and 2_2, the model unanimously picks feature 0. The trace for the first anomaly is depicted for these four traces. Since anomalies within a trace are of the same type, features generally follow similar patterns. For all of the traces, feature 0 had a feature reward in the 70%-90% range for most of the anomalies, thus showing that the calculation of the rewards as well has improved significantly.

Algorithm 1 Feature Analysis for Anomaly Detection

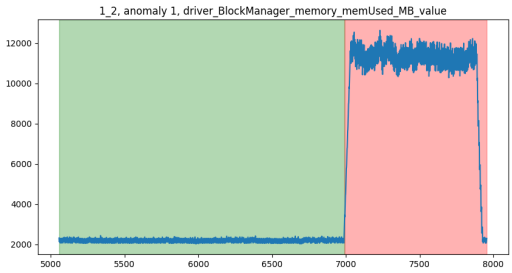
Input: Normal and anomalous records, number of features, binning parameters
Output: Adjusted segments for each feature
for each feature in the dataset **do**
    Extract series of normal and anomalous values for the feature
    Combine the series and determine the value range
    Create bins across the value range based on the binning parameters
    Initialize lists for normal, anomalous, and mixed segments
    for each bin **do**
        Classify the bin based on the presence of normal and/or anomalous values
        Calculate scores for samples if the bin is mixed
        Aggregate scores to define the bin's overall classification
        if current bin continues the classification of the previous bin **then**
            Extend the current segment with the bin's range
        else
            Finalize the current segment
            Start a new segment with the bin's classification and range
        end if
    end for
    Finalize the last segment if any
    Adjust segments by setting the start of the first to $-\infty$ and the end of the last to $+\infty$
    Merge adjacent segments if they have the same classification, adjusting their ranges accordingly
    Store the adjusted segments for the current feature
end for
return Adjusted segments for all features

Algorithm 2 Unified Procedure for Processing Features

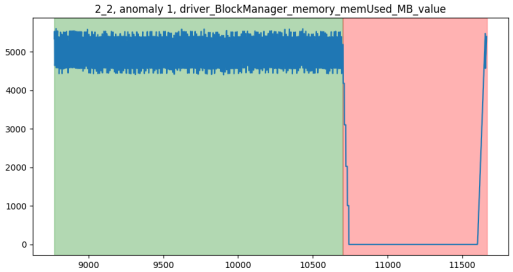
for $ft \leftarrow 0$ **to** $n\_features - 1$ **do**
  $normal\_values \leftarrow$ Extract from $normal\_records[:, ft]$
  $anomalous\_values \leftarrow$ Extract from $anomalous\_records[:, ft]$
  $combined\_samples \leftarrow$ Concatenate($normal\_values, anomalous\_values$)
  $bins \leftarrow$ Calculate bins for $combined\_samples$ based on $n\_bins, sigma, alpha$
  for bin **in** $bins$ **do**
    Determine $classification$ based on the presence of normal and anomalous samples in bin
    if $classification$ is mixed **then**
      Calculate $score$ for each sample in bin using $temporal\_weight$
      $classification \leftarrow$ Determine based on average $score$
    end if
    Add bin information to bin_info_list
  end for
  $current\_segment \leftarrow$ None
  $segment\_list \leftarrow []$
  for bin_info **in** bin_info_list **do**
    if $current\_segment$ is None **or** $classification$ changed **then**
      Close $current\_segment$ and add to $segment\_list$ if it exists
      $current\_segment \leftarrow$ New segment from bin_info
    else
      Extend $current\_segment$ with bin_info
    end if
  end for
  Add final $current\_segment$ to $segment\_list$ if it exists
  for each $segment$ **in** $segment\_list$ **do**
    if first $segment$ **then**
      $segment.start \leftarrow -\infty$
    end if
    if last $segment$ **then**
      $segment.end \leftarrow +\infty$
    end if
    Merge $segment$ with adjacent if they have the same classification
  end for
  Output adjusted segments for feature ft
end for



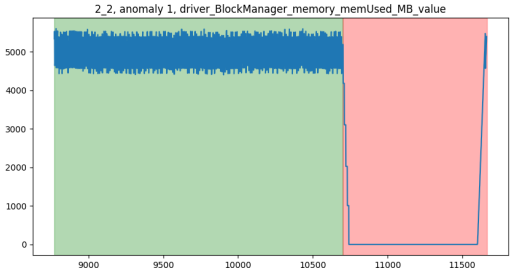
(a) Feature 0 for anomaly 1 of trace 1_1



(b) Feature 0 for anomaly 1 of trace 1_2



(c) Feature 0 for anomaly 1 of trace 2_1

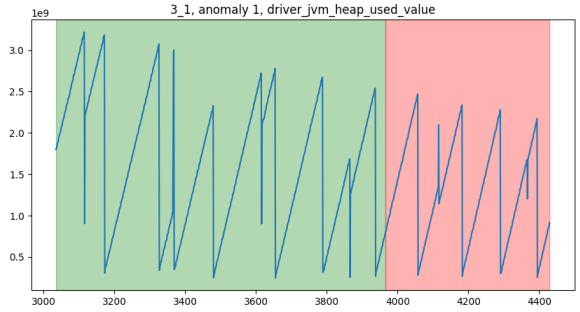


(d) Feature 0 for anomaly 1 of trace 2_2

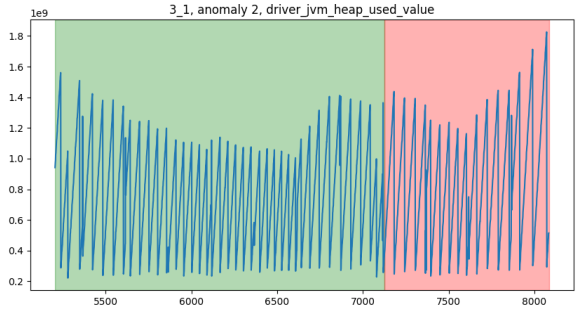
Figure 10: Explanations for bursty and stalled input

Other features have more mixed segments and are not as clearly separated as these, thus the model constantly picks the best explanation. For CPU contention anomalies, it was more complicated to find consistent explanations. We will explore two anomalies of each trace to keep clarity and conciseness. This will be sufficient to show the low rewards and the lack of consistency within the same type of anomalies. However, the traces of intervals will prove that the CPU contention anomalies are not good types of data to apply ExStream on.

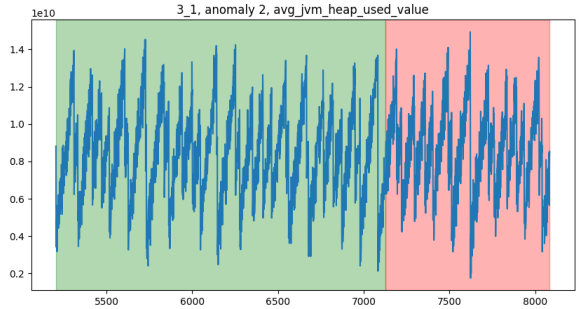
For trace 3_1, for the first two anomalies, we have an output of *feature 1 with the importance of 0.09 for anomaly 1; for anomaly 2: feature 1 with the importance of 0.08, feature 2 with the importance of 0.06 and feature 5 with the importance of 0.09.* The rewards are significantly lower and the length is inconsistent. Let us explore these features in 11.



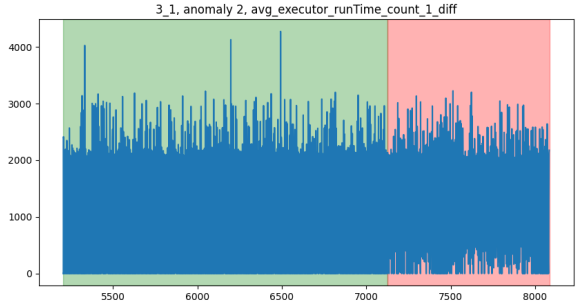
(a) Feature 1 for anomaly 1 of trace 3_1



(b) Feature 1 for anomaly 2 of trace 3_1



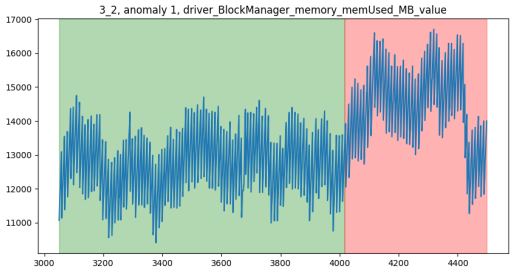
(c) Feature 2 for anomaly 2 of trace 3_1



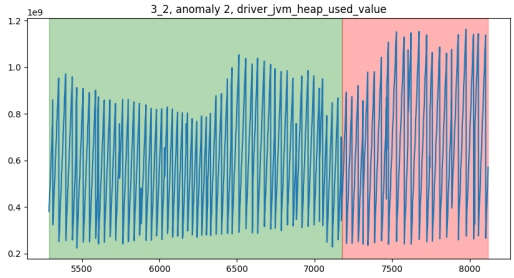
(d) Feature 5 for anomaly 2 of trace 3_1

Figure 11: Explanations for CPU contention trace 3_1

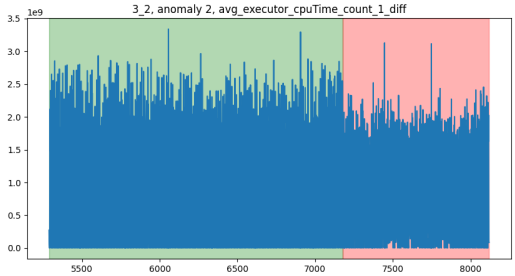
For trace 3_2, the predictions for the first two anomalies are the following: *feature 0 with the importance of 0.1 for anomaly 1. For anomaly 2: feature 1 with the importance of 0.08, and feature 5 with the importance of 0.08*. Graphically, the following can be depicted as:



(a) Feature 0 for anomaly 1 of trace 3_2



(b) Feature 1 for anomaly 2 of trace 3_2



(c) Feature 5 for anomaly 2 of trace 3_2

Figure 12: Features for anomaly 1 of trace 3_2

The features are far more mixed and have less clear division between normal and anomalous data. More-

over, the features are less consistent with each other, thus making the explanations less consistent.

Finally, the rest of the results can be explored in the tables below. For each table, we have a trace with its anomalies. Horizontally, for each anomaly, we depict the reward of a feature if it is considered to be an explanation for that particular anomaly.

Table 1: Explanations for trace 1_1

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	0.84	-	-	-	-	-	-	-
2	0.86	-	-	-	-	-	-	-
3	0.83	-	-	-	-	-	-	-
4	0.82	-	-	-	-	-	-	-
5	0.84	-	-	-	-	-	-	-
6	0.74	-	-	-	-	-	-	-

For bursty input, the model consistently provides feature 0 as an explanation. It is consistent with the graphical observations and has consistent scores throughout the rewards for the same type of anomaly.

Table 2: Explanations for trace 1_2

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	0.84	-	-	-	-	-	-	-
2	0.77	-	-	-	-	-	-	-
3	0.83	-	-	-	-	-	-	-
4	0.84	-	-	-	-	-	-	-
5	0.83	-	-	-	-	-	-	-

Similarly, as above, the model consistently picks feature 0 for an explanation. As we recall, both traces illustrate anomalies of the same type on different applications, thus consistency is a good measure.

Table 3: Explanations for trace 2_1

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	0.78	-	-	-	-	-	-	-
2	0.81	-	-	-	-	-	-	-
3	0.81	-	-	-	-	-	-	-
4	0.76	-	-	-	-	-	-	-

For stalled input anomalies, the model also chooses feature 0 as the best and only explanation. Graphically, feature 0 is the most anomalous feature, thus providing us with the intuition of a correct guess.

Table 4: Explanations for trace 2_2

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	0.66	-	-	-	-	-	-	-
2	0.38	-	-	-	-	-	-	-
3	0.70	-	-	-	-	-	-	-
4	0.48	-	-	-	-	-	-	-

As the traces 2_1 and 2_2 are of the same type, the explanation stays consistent all throughout. Some lower reward scores can be explained by smaller values for the anomaly and more labeling noise.

Table 5: Explanations for trace 3_1

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	-	0.09	-	-	-	-	-	-
2	-	0.08	0.06	-	-	0.09	-	-
3	-	0.09	-	-	-	-	-	-
4	0.09	0.08	-	-	-	-	-	0.09
5	-	0.07	0.05	0.06	-	0.08	-	-

For CPU contention anomalies, the model struggles to pick consistent explanations. It is important to note that differently as for previous anomalies, here features do not keep up the same patterns and behaviors through the anomalies within the trace, thus less consistency should be expected.

Table 6: Explanations for trace 3_2

Anomaly	Feature 0	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
1	0.10	-	-	-	-	-	-	-
2	-	0.08	-	-	-	0.08	-	-
3	0.08	0.07	0.08	-	0.08	-	-	-
4	-	0.09	-	-	0.09	-	-	-
5	-	0.07	0.06	0.08	0.08	-	-	-

Explanations vary in length and even though there are some dominant ones, the CPU contention anomalies are not the best fit for data that ExStream is equipped to handle. As mentioned before, many of the features are differences (diffs) and thus have many overlapping values at 0 (when the feature is constant). Thus, one of the techniques to explore the CPU contention anomalies in greater detail could be by preparing the data differently.

5.3 False Positive Feature Filtering

Before the start of this study, false positive filtering was defined as features that have unusually high standard deviation and constantly increase or decrease through the normal period. The paper proposes a different approach. The proposal is to implement a self-supervised learning method where by comparing similar patterns in KPI (key performance indicators) features. These features are those that depict processing times or delays within the application - they indicate the performance of the application but do not provide us with a reason for the anomaly. We apply the retrieved explanation for the original anomaly and evaluate whether it is relevant for similar patterns within the data retrieved by comparing the KPIs. The goal of this approach is to filter out

the features that have different explanations at different time intervals. The best example here is if we have a dataset defining temperature in the summer and winter seasons. For the sake of this example, let us define the average winter temperature as that between 0-10°C and anomalous temperatures as those above 10°C. The average temperature in summer is 20-25°C. Thus, when winter's definition of anomaly is applied to the summer temperatures, it does not hold. Thus, the explanation is not generalizable and should be discarded.

Due to the time constraints, the complexity of the model, and the nature of the dataset, we propose a different definition of false positive filtering. As mentioned above, the features can act anomalous due to reasons not associated with the anomaly. Thus, in this approach, we define false positive features as those that act anomalously during normal periods.

The method compares feature rewards between normal and anomalous periods and between two consecutive normal periods. If the single-feature reward for a returned explanation is relatively close (in the 35% range), we consider the feature to be false positive.

In this dataset, we have examples of false positive features for CPU contention anomalies. To examine this technique, we will look at the feature's whole trace to see more reference periods. For trace 3_1, for 4 out of 5 anomalies, feature 1 gets removed.

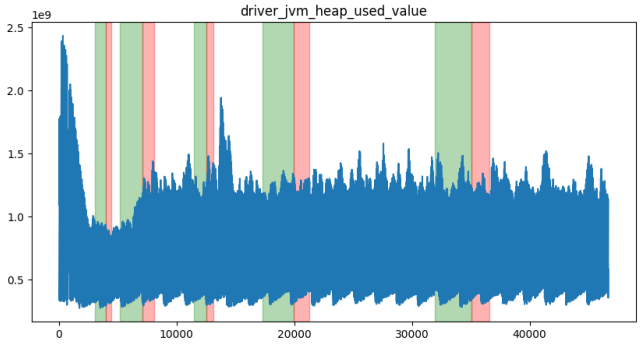


Figure 13: Trace of feature 1 for trace 3_1

As can be observed, this feature acts inconsistently all throughout the trace and therefore gets removed as a false positive feature that does not have an impact on the anomaly. The difference in ranges of values within two consecutive reference (green) periods seems as abnormal as that between normal and anomalous periods. As one can observe, the technique successfully implements false positive filtering. However, the approach described in the original paper has a slightly different definition of false positive features, thus we leave the implementation of it and the evaluation of this technique for future works.

6 Evaluation

Besides the graphical evaluation in previous sections, we also employ two different techniques to evaluate the performance of the model and the changes implemented.

1. Comparison with *Macrobase*: Macrobase is another popular data-explainer model. Thus, running this model on the dataset could let us compare the returned explanations by both models.

2. *Exathlon* metrics: Exathlon was a framework constructed to evaluate Anomaly Detection and Explanation Generation techniques. We use three metrics described in the paper.

6.1 Comparison with Macrobase

As mentioned above, Macrobase is another data-explainer model, thus we will apply it to the same dataset and compare the explanations retrieved. The goal is to compare the explanations returned by three implementations - old ExStream, Macrobase, and the new modified ExStream (with generalized normal distribution and sum to compute the weights). In the following tables, E1 stands for the first feature returned as an explanation, E2 stands for the second feature returned as an explanation, etc.

Table 7: Model comparison for Trace 1 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	2	-	-	-	2	-	-	-	0	-	-	-
2	1	2	-	-	0	-	-	-	0	-	-	-
3	1	2	-	-	0	-	-	-	0	-	-	-
4	1	2	-	-	2	-	-	-	0	-	-	-
5	1	2	-	-	0	-	-	-	0	-	-	-
6	1	2	-	-	0	-	-	-	0	-	-	-

Table 8: Model comparison for Trace 2 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	2	-	-	-	0	-	-	-	0	-	-	-
2	1	2	4	-	0	-	-	-	0	-	-	-
3	2	-	-	-	0	-	-	-	0	-	-	-
4	2	-	-	-	0	-	-	-	0	-	-	-
5	1	2	4	-	0	-	-	-	0	-	-	-

Table 9: Model comparison for Trace 3 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	1	2	6	7	0	-	-	-	0	-	-	-
2	1	2	6	7	0	-	-	-	0	-	-	-
3	1	2	6	7	0	-	-	-	0	-	-	-
4	2	-	-	-	0	-	-	-	0	-	-	-

Table 10: Model comparison for Trace 4 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	1	2	6	7	0	-	-	-	0	-	-	-
2	1	2	6	7	0	-	-	-	0	-	-	-
3	1	2	4	6	0	-	-	-	0	-	-	-
4	0	-	-	-	0	4	7	-	0	-	-	-

Table 11: Model comparison for Trace 5 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	1	2	4	6	Not found	-	-	-	1	-	-	-
2	1	2	-	-	Not found	-	-	-	1	2	5	-
3	1	2	-	-	Not found	-	-	-	1	-	-	-
4	1	2	-	-	0	-	-	-	0	1	7	-
5	1	2	3	4	Not found	-	-	-	1	2	3	5

Table 12: Model comparison for Trace 6 anomalies

Anomaly	**Old ExStream**				**Macrobase**				**New ExStream**			
	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
1	1	2	-	-	0	-	-	-	0	-	-	-
2	1	2	4	6	Not found	-	-	-	1	5	-	-
3	1	2	-	-	0	-	-	-	0	1	2	4
4	1	2	-	-	Not found	-	-	-	1	4	-	-
5	1	2	4	6	Not found	-	-	-	1	2	3	4

As one can observe, the original ExStream and Macrobase do not have many explanations in common. As discussed above, the original model does not choose the appropriate explanations, therefore the results attained are very different. The new model matches most of Macrobase's explanations. Important to note that ExStream chooses the best explanation and as feature 0 is a nearly perfect fit for an explanation (no mixed segments), usually others are discarded as they have significantly lower rewards.

For CPU contention anomalies (those in traces 3_1 and 3_2), Macrobase fails to find an explanation for most of the anomalies. This can be explained as above, that the data is not fit for data explainers. However, for those features that Macrobase recognizes as anomalous, the new ExStream also recognizes them as part of the explanation.

After analyzing the comparison results, we can conclude both graphically and with a comparison with another model, that the new framework is better adapted to handle these types of datasets.

6.2 Exathlon framework

Another framework made for Anomaly Detection and Explanation Generation evaluation is Exathlon. Three metrics will be used to evaluate the new approach, such as ED1 stability, ED1 accuracy, and ED2 accuracy. ED1 types are measurements that are checked on an individual anomaly by splitting up the data into subsets and thus are local evaluators. ED2 types are global, where the explanations retrieved are applied to the anomalies within the same trace and thus are global evaluators. One can compute the ED1 instability by randomly subsampling 80% of normal and anomalous data, then getting an explanation for each sample, and then computing the entropy of features coming from all explanations. The table shows that our model performs slightly worse in terms of stability but provides more concise explanations. However, this metric has some limitations. If the model constantly picks the wrong explanation (as seen with the original ExStream for this dataset), constantly picking the wrong explanation can provide a good stability measure but this does not imply the model would perform well with different data. Thus, even if the model is slightly worse in terms of stability, 1 being the perfect score, it can still be concluded to be rather stable for data perturbations. In the following table, we can compare the length of the explanations returned by both old and new implementations and their stability for each trace.

Table 13: Comparison of Two Models Across Six Traces

Trace	Size		Normalized Instability	
	Original ExStream	Modified ExStream	Original ExStream	Modified ExStream
1_1	1.67	1.00	1.00	1.14
1_2	1.80	1.00	0.87	1.00
2_1	3.25	1.00	1.00	2.27
2_2	3.25	1.00	1.03	1.72
3_1	2.80	2.40	1.29	0.99
3_2	2.80	2.80	1.40	1.13
Average	2.59	1.57	1.1	1.37

For the ED1 accuracy metric, again 80% of both anomalous and normal data is subsampled and the retrieved explanations are applied for the remaining 20% of the data. Three metrics are retrieved - precision (accuracy of the positive predictions made by the model), recall (sensitivity or true positive rate, measures the model's ability to identify all relevant cases within a dataset), and the F1 score (the harmonic mean of precision and recall). We explore the performance of both old and new models.

Trace	**Original ExStream**			**Modified ExStream**		
	F1-Score	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**
1_1	0.421	0.929	0.375	0.515	0.349	0.997
1_2	0.653	0.850	0.583	0.501	0.335	0.999
2_1	0.277	0.802	0.217	0.523	0.365	0.943
2_2	0.312	0.836	0.281	0.655	0.521	0.961
3_1	0.029	0.385	0.015	0.513	0.347	0.988
3_2	0.020	0.353	0.011	0.515	0.348	0.993

Table 14: ED1 comparison of F1-Score, Precision, and Recall between Original and New ExStream for each trace.

The last metric is the ED2 accuracy metric. Oppositely from ED1, ED2 is a global metric. The metric takes an explanation for one anomaly within the trace, it applies the explanation (retrieved anomalous segments) to the rest of the anomalies of the same type. Then, f1-score, recall, and precision are computed. The results are the following:

Trace	**Original Model**			**New Model**		
	F1-Score	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**
1_1	0.731	1.000	0.615	0.977	1.000	0.955
1_2	0.763	1.000	0.660	0.980	1.000	0.961
2_1	0.702	1.000	0.562	0.988	1.000	0.976
2_2	0.701	1.000	0.573	0.987	1.000	0.976
3_1	0.446	1.000	0.289	0.523	0.378	0.876
3_2	0.428	1.000	0.275	0.527	0.367	0.813

Table 15: ED2 comparison of F1-Score, Precision, and Recall between Original and New ExStream for each trace.

As we can see, the recall significantly improved whereas the precision dropped for most examples. F1-

score of ED1 improved quite significantly for all of the traces except 1_2. For ED2 the f1-score slightly dropped. However, it is important to note that the metric is not accurate for the new model as we have changed the approach of ExStream. We are using binning, thus changing the labels of some points to minimize the segmentation. We are relabelling points to construct the segments better. The framework returns the feature segments that are constructed as value segments, where the feature is *both anomalous and mixed*. Thus, we get a high recall rate as we have many segments that are mixed (due to binning) and we capture the entire anomaly within the mixed and anomalous segments, but precision is lost as constructing more mixed segments and the binning approach tends to relabel many points. If we change the explanation of ExStream to be only anomalous segments of values and no mixed, we have all three metrics (F1-score, recall, and precision) to be above 95% for all traces. Thus, this metric is here for reference, however, to effectively measure accuracy with the new method we propose a new method that could be implemented in the future.

To conclude, our model performs significantly better than the old ExStream and graphically and in comparison with Macrobase chooses better explanations. As well, it is more adaptable to different problems as variables such as β, σ, and number of bins can be adjusted.

7 Conclusion and Outlook

The new approach with binning and adaptation of penalties based on the entropy of mixed segments successfully improves the model's performance. Assigning weights to the points by their temporal placement well improves the model's susceptibility to labeling noise making it more applicable to different data streams. This study leaves room for further advancements in the field. The model fits better for this current dataset, but to ensure the adaptability of the model, the model should be run on more datasets of different natures to ensure that the approach has improved.

The model as of right now has a high number of hyperparameters, thus a hyper tuning of the parameters should be explored and perhaps even inserted into the framework to ensure that the user receives the best results for their anomaly detection. However, ExStream must be fast so that it can be applied in the situation for real-time data analysis, thus perhaps different approaches could be possible.

It is important to further explore the performance of the new implementation with the appropriate metrics. As mentioned above, the nature of explanations of ExStream could be changed to be only anomalous segments, however, this still would not yield the actual precision of a new model. A way to modify the accuracy metric for both ED1 and ED2 from the Exathlon framework would be to consider the ratio of anomalous points (as described before w_t). If each bin carries the weight of its anomality, one can more accurately capture the characteristics of a certain type of anomaly (the distribution of mixed, anomalous, and normal data with their ratios). Then, mixed and purely anomalous segments would not be regarded in the same way and the precision of a model would increase significantly. If proceeded, one could use the ROC curve or similar to measure the new metrics.

Finally, the false positive filtering should be implemented using the technology from the paper to ensure the highest performance and make the model generalizable for more problems. The concept of self-supervised learning would also provide the model with more data to provide general explanations making it more adaptable to real-time data.

Trace	Original ExStream	New Model
1_1	1.18	1.00
1_2	1.50	1.00
2_1	1.22	1.00
2_2	1.68	1.00
3_1	1.43	1.86
3_2	1.30	1.58
Average	1.38	1.24

8 References

[1] Firas Abuzaid, Peter D. Bailis, Jialin Ding, Edward Gan, S. Madden, D. Narayanan, Kexin Rong, and S. Suri. Macrobase. *ACM Transactions on Database Systems (TODS)*, 2018.

[2] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. Macrobase: Prioritizing attention in fast data. In *In Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.

[3] Ana Maria Bianco, Marta Garcia Ben, Eunie Jr. Martinez, and Victor J. Yohai. Outlier detection in regression models with arima errors using robust estimates. *Journal of Forecasting*, 20:565–579, 2001.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *ACM International Conference on Management of Data (SIGMOD)*, pages 93–104, 2000.

[5] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41:1–58, 2009.

[7] Chaoyu Chen, Hang Yu, Zhichao Lei, Jianguo Li, Shaokang Ren, Tingkai Zhang, Silin Hu, Jianchao Wang, and Wenhui Shi. Balance: Bayesian linear attribution for root cause localization. *Proceedings of the ACM on Management of Data 1*, 2023.

[8] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal 7.7*, pages 6481–6494, 2019.

[9] LevelUp Education. How are big companies using apache spark? https://medium.com/@tao_66792/how-are-big-companies-using-apache-spark-413743dbbbae, 2023.

[10] Zixuan Geng, Maximilian Schleich, and Dan Suciu. Computing rule-based explanations by leveraging counterfactuals. *arXiv preprint arXiv:2210.17071*, 2022.

[11] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 2250–2267, 2014.

[12] Thi Kieu Khanh Ho, Ali Karami, and N. Armanfard. Graph-based time-series anomaly detection: A survey. *ArXiv*, abs/2302.00058, 2023.

[13] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbu. Exathlon: A benchmark for explainable anomaly detection over time series. *arXiv preprint arXiv:2010.05073*, 2020.

[14] Hiranya Jayathilaka, C. Krintz, and R. Wolski. Detecting performance anomalies in cloud platform applications. *IEEE Transactions on Cloud Computing*, 8:764–777, 2020.

[15] Jon Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35:795–806, 2009.

[16] Cristoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2021.

[17] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Comput. Surv.*, feb 2023.

[18] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?": Explaining the predictions of any classifier. *ACM SIGKDD Conference*, pages 1135–1144, 2016.

[19] S. Roy, L. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *PVLDB*, 9:348–359, 2015.

[20] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE symposium on security and privacy*, 2010.

[21] M. T Subbotin. *On the law of frequency of error*, volume 31. Matematicheskii Sbornik, 1923.

[22] Luan Tran, Liyue Fan, and Cyrus Shahabi. Distance based outlier detection for data streams. *Proceedings of the VLDB Endowment (PVLDB)*, 9:1089–1100, 2015.

[23] Paolo Vanini, Sebastiano Rossi, Ermin Zvizdic, and Thomas Domenig. Online payment fraud: from anomaly detection to risk management. *Financ Innov 9*, 66, 2023.

[24] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6:553–564, 2013.

[25] Haopeng Zhang, Yanlei Diao, and Alexandra Meliou. Exstream: Explaining anomalies in event stream monitoring. In *20th International Conference on Extending Database Technology (EDBT)*, 2017.

[26] E. Šabić, D. Keeley, B. Henderson, and S. Nannemann. Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. *AI SOCIETY*, pages 149–158, 2023.