Comparative Evaluation of Streaming Deep Outlier Detection Techniques for Time Series Data

ADRIANO MUNDO ANDREA POZZOLI
mundo@kth.se | apozzoli@kth.se

January 15, 2021

Abstract

Outlier detection is the process of identifying unexpected items or events, which substantially differ from the normal. With the advent of big data, the need for robust outlier detection in streaming data has been raised and there exist a huge amount of algorithms that have been proposed in this research area. Some techniques can be applied to unlabeled data, which is known as unsupervised outlier detection and it is essential for many practical applications such as fraud detection. Recently, deep learning-based outlier detection techniques have also been proposed in this area but there is a lack of comparative studies of these techniques for univariate time-series in a streaming scenario. This knowledge gap is filled with this research, where we experimentally compare three different deep outlier detection algorithms, Telemanom, OED and Donut, on five publicly available datasets retrieved from Numenta Anomaly Benchmark (NAB), the standard for streaming univariate time-series data. The work aims at being the basis for future research on this topic and we can provide insights on the performance of the algorithms. Finally, our analysis gives advice on the feasibility of these techniques in a streaming fashion.

Keywords: outlier detection, univariate time-series, deep learning, streaming data

Contents

1 Introduction

Outlier detection has been a research topic for a long time [1] but in recent years has gained particular attention in different application domains thanks to a lot of technological advancements which allowed to collect large amounts of data from critical systems responsible for intrusion detection, fraud detection, fault diagnosis and financial markets. These huge data volumes require sophisticated extraction and analysis processes, which has led to a renewed interest for practitioners in the research fields of data science and data mining.

Data that are collected in an orderly fashion and correlated in time represent a *time-series*. One of the main tasks of time-series analysis is outlier detection since actionable insights can be derived from their discovery and they can be adapted for handling data in form of streams instead of batches as studied by Aggrawal et. al [2, 3], but many challenges and issues arise [4]. This work shows experimental results in both static and streaming scenarios. With the ever-growing computational power, deep learning approaches increased in popularity and have shown tremendous capabilities in learning representations of complex data. Thus, a large number of different deep outlier detection methods were introduced for learning feature representations or outlier scores through neural networks. They demonstrated better performance than conventional methods on addressing challenging detection problems in a variety of real-world applications [5]. This work aims at providing a comparison of algorithms within this area while deepening the case of univariate time-series.

To sum up, in this study we present a comparative evaluation of a variety of deep outlier detection techniques for univariate time-series data in a static and streaming scenario. The report is organized as follows: the remainder part of this first section introduces background concepts needed to understand the research, the algorithms included in the study and related work in this research area. At the end, a problem discussion is provided. Section 2 describes the research questions and hypothesis, while Section 3 and 4 explain the evaluation methodology and the results, respectively. Section 5 summarizes a discussion on the problem findings, limitations and possible future works.

1.1 Background

In general, an outlier has been described as a data point that is significantly different from the remaining data. More precisely, Hawkins [6] defined an outlier as an: *"observation which deviates so much from the other observations as to arouse suspicion that it was generated by a different mechanism"* while Chandola et al. [7] defined anomalies as: *"patterns in data that do not conform to a well-defined notion of normal behavior"*. Authors have proposed many definitions for the term outlier but there is still no universally accepted definition. An attempt to prove the equality between outliers and anomalies was done by Aggrawal et al. [2] with the following interpretation: *"outliers are also referred to as abnormalities, discordants, deviants, or anomalies in the data mining and statistics literature."* In this report, the two terms outlier and anomaly have been used interchangeably. An important aspect of an outlier detection technique is the **nature of the anomaly**, that the relative literature has classified into three categories [2, 4]:

- *Point Anomalies (or Type I Outliers):* if an individual data instance can be considered as anomalous with respect to the rest of data, then the instance has been termed as a point anomaly. This is the simplest type of anomaly and remains the focus of the majority of research on anomaly detection.

- *Contextual Anomalies (or Type II Outliers):* if a data instance is anomalous in a specific context (but not otherwise), then it has been termed as a contextual anomaly. The notion of a context is induced by the structure in the dataset and has to be specified as a part of the problem formulation.

- *Collective Anomalies (or Type III Outliers):* if a collection of related data instances is anomalous with respect to the entire dataset, it has been termed as a collective anomaly. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous.

Another key aspect is the nature of **input** data, which was represented by univariate (one-dimension) or multivariate (multiple dimensions) time-series and in this research, we focus on the former. Accordingly to Garcia et al. [8], a *univariate time-series* $X = \{x_t\}_{t \in T}$ is an ordered set of real-valued observations, where each observation is recorded at a specific time $t \in T \subseteq \mathbf{Z}^+$. Then, x_t is the point or observation collected at time t. For further investigation on anomaly detection for univariate time-series, the reader can refer to the extensive survey by Braei et al. [9], while for details regarding both univariate and multivariate time-series to Garcia et. al [8].

Typically, the **output** produced by an outlier detection technique was categorized in one of the following two types [2, 7]:

- *Outlier scores:* a score quantifying the level of "outlierness" of each data point. This score can be used to rank the data in order of their outlier tendency. This is a general form of output.

- *Binary labels:* a label that indicates whether a data point is an outlier or not. Some algorithms return directly binary labels, otherwise, scores can be converted into binary labels by imposing thresholds on the scores chosen on the statistical distribution. The final result is used for decision-making in real-life applications.

Based on the extent to which the labels are available, outlier detection techniques could operate in one of the following three modes [7]: *supervised* which assumes the availability of a training data set which has labeled instances for normal and anomaly classes. The standard approach in such cases is to build a predictive model for normal vs anomaly classes. Any unseen data instance is compared against the model to determine which class it belongs to; *semi-supervised* assume that the training data has labeled instances for only the normal class. Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques; *unsupervised* which do not require training data, and thus are most widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from a high false alarm rate.

The importance of study outlier detection techniques derives from the connection between outliers in data and significant actionable insights in a variety of real-world applications. Moreover, most of the data are streaming and time-series data, hence detecting outliers in big streaming data is a challenging and difficult task since there is a need of acquiring and processing data as soon as they occur, before they are stored and immediately notify about potential threats. In the extended background section, there are more details on the chosen algorithms.

1.2 Extended Background

In the following section, we shortly introduce the algorithms considered for the evaluation and their basic intuition but the interested reader should refer to the authors' original publication. In particular, the research focuses on the new flavor of well-established Artificial Neural Networks (ANN) algorithms which have shown substantial improvement with respect to traditional techniques [10, 5, 11]. We selected a subset of the most recent deep-learning outlier detection methods that can handle univariate time-series data and should be eligible candidates for working in a streaming scenario. The authors compared the performance of their algorithms with state-of-the-art techniques but there is a lack of comparison between them. All the implementations of the selected algorithms are publicly available.

Telemanom

Hundman et al. [12] demonstrated the effectiveness of Long-Short-Term-Memory (LSTMs) networks, a type of Recurrent Neural Networks (RNN) for detecting anomalies on the spacecraft monitoring system, which can deal with a real-world system that is usually highly non-stationary and dependent on the context. The work was presented through the lens of spacecraft anomaly detection, but it can be applied to many other cases involving anomaly detection for multivariate time series data. Specifically, they described their use of Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) to achieve high prediction performance while maintaining interpretability throughout the system. They used a non-parametric and unsupervised thresholding approach for evaluating residuals, which can handle thresholds non-stationarity and data streams. They demonstrated the viability of LSTMs for predicting spacecraft telemetry and key challenges in many anomaly detection scenarios.

OED

Kieu et al. [13] proposed two solutions for outlier detection in time series based on recurrent autoencoder ensembles. Their solution leveraged autoencoders with sparsely-connected recurrent neural networks (S-RNNs). These networks made it possible to generate multiple autoencoders with different neural network connection structures. The two solutions are ensemble frameworks, an independent framework IF and a shared framework SF, both of which incorporate multiple S-RNN autoencoders to enable outlier detection. Specifically, we used IF for our experiments which trains multiple autoencoders independently. It has been shown to work with real-world time series data sets, including both univariate and multivariate time series, outperforming state-of-the-art methods.

Donut

Xu et al. [14] presented Donut, an unsupervised anomaly detection algorithm based on VAE (Variational Auto Encoder). Donut outperforms state-of-the-art supervised ensemble approaches and a baseline VAE based anomaly detection algorithm. It was applied for analyzing seasonal KPIs (Key Performance Indicators) in web applications and its excellent performance is demonstrated through strong theoretical analysis, hence a new KDE interpretation and the discovery of the time gradient effect. It can deal with univariate and multivariate time-series data.

1.3 Related Work

In this section, an analysis of the existing literature is provided. Currently, with regards to outlier detection techniques two main approaches exist: on one hand, authors of new algorithms compare their results with state-of-the-art techniques and show that their implementation outperforms currently existing algorithms; on the other hand, several studies have been published which refer to a specific application scenario and not always with more than one datasets. Recently, deep learning has obtained enormous attention in different domains, hence there has been an increase in deep outlier detection methods. Chalapath et al. [11] proposed a survey on deep-learning for anomaly detection and similar work was done by Pang et al. [5] which gave a more comprehensive review of the literature and suggestions on open opportunities in the field. Although these studies provide a complete review of deep learning techniques there was no comparative analysis. For example, Ahmad et al. [15] proposed a real-time anomaly detection technique based on Hierarchical Temporal Memory (HTM), compared it with the state-of-the-art algorithms: TwitterADVec, Etsy Skyline, EXPoSe, etc. and used NAB for testing algorithms with univariate streaming time-series. Several papers followed the same methodology when presenting a new technique [8, 9, 3, 7, 5].

In the literature there exist comparisons on unsupervised anomaly detection but they lack some of the aspects we address in our work. Goldstein et al. [16] addressed the challenge of analyzing unsupervised anomaly detection by comparing 19 different algorithms on 10 different datasets from multiple application domains by analyzing multivariate tabular data. Domingues et al. [17] showed a comparative evaluation of unsupervised machine learning algorithms in the context of outlier detection belonging to a wide range of approaches such as probabilistic, distance-based, neighbor-based, domain-based, neural networks with both parametric and non-parametric algorithms. The algorithms are tested on 15 datasets. Campos et al. [18] provided an extensive experimental comparison on the performance of a representative set of standards k-nearest-neighborhood based methods for unsupervised outlier detection. The experiment was conducted on 12 methods over 23 datasets.

Other comparative focused on outlier detection techniques with time-series data, such as Däubener et al. [19] presents an empirical comparison of common machine learning and statistical methods applied to univariate time series with the purpose of detecting anomalies. The methods selected are residual-based approaches, hence the normal behavior is learned and modeled. Based on these models, deviations between observed and predicted values are used to decide whether a new observation is anomalous or not. Chandola et. al [20] have done a comprehensive evaluation of a large number of semi-supervised anomaly detection techniques for time series data on a large variety of datasets obtained from different application domains. They included kernel-based techniques, window-based techniques, predictive techniques and segmentation based.

Finally, we take into account previous studies that are most similar to our research. The papers have been published recently, hence it is a clear indication that there is an interest in this kind of analysis and research. Munir et. al [10] proposed a comparison between traditional and recently developed deep neural networks anomaly detection methods. Moreover, it takes into consideration the non-image streaming data which was missing in the literature. They compared 13 anomaly detection methods on two commonly used streaming datasets: NAB and Yahoo Webscope. The study revealed that deep learning-based methods are superior to traditional anomaly detection methods. Freeman et. al [21] presented a framework to choose the optimal anomaly detection methods based on the characteristics a time-series displays. They experimented with multiple methods over different data in an online fashion and evaluated their algorithms on NAB datasets. Hasani [22] compared several fast anomaly detection algorithms for time-series data on three NAB datasets in a streaming fashion with the aim of evaluating different categories of algorithms for which they expected fast and satisfactory recall and precision. Moreover, they evaluated algorithms execution time and CPU usage. They wanted to monitor the streaming log data that were generated in the e-dnevnk that acquire a massive number of online queries.

1.4 Problem & Research Question

In this section, we briefly discuss the connection of previous works with the knowledge gap and how them motivate the research. As stated in the related work section, there exist many algorithms for detecting anomalies in time-series [8, 9] but not many comparative studies on deep outlier detection do exist today [16]. Thus, it could be challenging for practitioners to keep up with the latest research and for newcomers to understand what could be the best tool and model for their specific needs.

Previous findings did not explore deep outlier detection techniques with streaming time-series. Indeed, Goldstein et al. [16] comparison is between unsupervised algorithm but they lack a deep approach and do not consider streaming time-series, but multivariate tabular data. On the same line, Domingues et al. [17] did a pretty extensive comparison but an extension to deep learning algorithms is missing and time-series are not included in their test. They suggested

further research need in distributed and streaming settings. Also, Campos et al. [18] lack the deep approach and testing on streaming time-series. These comparative studies analyzed unsupervised techniques but without deep learning methods for streaming time-series. Instead, Däubener et al. [19] and Chandola et al. [20] focused their analysis on time series data but they did not consider deep learning algorithms even though they suggest future research line on a comparison between deep outlier techniques and exposed the idea of streaming algorithms. Recently, other comparisons emerged from Munir et al. [10], Freeman et al. [21] and Hasani et al. [22]. The focus is on streaming time-series but they focused mainly on the comparison between traditional and deep outlier techniques [10] or they analyzed time-series in an online fashion but they did not examine deep methods [21, 22].

The literature shows that exists a wide spectrum of outlier detection methods and as the number continues to grow, it becomes more difficult to keep track of all these techniques [9]. An evaluation of every new method is not feasible, hence to reduce the evaluation burden more comparisons are needed between deep outlier techniques [5, 11]. Therefore, due to the lack of previous works in this research area, this study aims at filling the gap and executing a comparative analysis of deep outlier detection techniques for streaming univariate time-series data.

The goals of the research are to show the ability of the selected algorithms to effectively work in a streaming scenario and how they perform. Thus, in the previously well-defined circumstances the **research question** of our study is: *How do outlier detection techniques perform and compare over different real-life datasets?*

2 Research Methodology

In this section, we provide details of the methodology and the datasets used for carrying out the research project. Different research methodologies have been taken into account. Between a quantitative or a qualitative study we opted for the first one because in order to compare the algorithms as required by the research question we needed some numerical and objective results. For what concerns the choice between a theoretical/descriptive or an experimental approach we chose to perform an experiment because it was impossible to compare the algorithms and verify their feasibility in a streaming scenario in a theoretical way. In fact, the selected algorithms were originally tested on different datasets and in a non-streaming scenario. Finally, we decided to use some existing datasets rather than creating our datasets in order to not bias the comparison and to have consistent data over different topics. Therefore the study uses an experimental quantitative research methodology on secondary datasets. The goal of the experiment is to make a comparative evaluation of the three selected techniques on different datasets using standard performance metrics. The experiment is divided into two phases. In the first phase, the algorithms are implemented, executed and compared using the datasets as historical to evaluate the feasibility and correctness of the algorithms (static scenario). In the second phase, the objective is to compare the algorithms with the same datasets in a streaming context, hence it is possible to evaluate if the selected algorithms can be applied in a streaming scenario and how they perform.
An important step of the experimental methodology is the selection of the datasets and the benchmarking for evaluating outlier detection methods.

2.1 Datasets

In this section, we briefly provide details on the dataset used for the study. We use publicly available outlier detection datasets as a benchmark, namely the Numenta Anomaly Benchmark (NAB) [23]. It consists of a collection of labeled datasets that allows to highlight and analyze point outliers. NAB was born in 2015 and rapidly became the standard for the analysis and comparison of outliers detection algorithms. It is completely open-source and each of the 58 available real-world datasets includes from 1000 to 22000 instances, that can be used in a streaming context. The time-series datasets are composed of pairs of two values, where the first value indicates a timestamp and the second value is scalar. The dataset files do not include labels therefore they can be used for unsupervised learning algorithms. However separate files contain the correct labels, hence they can be used for testing algorithms.

NAB offers also a tool for evaluating the performance of the algorithms under different features: a scoring system evaluates the algorithm capacity of detecting outliers, required time, absence of false-positives, streaming constraints and automation across datasets (no need for human intervention when changing dataset). The NAB scoring system does not use the traditional metrics such as precision and recall but computes a scoring function based on anomaly windows, which are used to weight true-positives, false-positives and false-negatives. Finally, the weighted features are analyzed with three different application profiles for evaluating the sensitivity of the detecting algorithms.

After having chosen the benchmark, the next step is the selection of the datasets. The 58 datasets that can be retrieved with NAB are divided into seven categories but to have not biased datasets and for comparing algorithms on different domains, the category selected is the *realKnowCause*, which contains seven different datasets that have

not been labeled by hand (differently from the other categories) and the anomaly causes are known. Among the seven datasets, two have been discarded (*rouge_agent_key_hold.csv* and *rouge_agent_key_updown.csv*) because they are composed mostly by zero values and the outliers (values different from zero) represent a change in the user of a computer, therefore it is not a real anomaly.

The remaining datasets are enough different from each other so that they should not bias the result of the experiment in favor of a particular algorithm. The outliers of these five datasets are real anomalies, hence the study is more meaningful and interesting. In the following list, we provide a brief description of the datasets. A representation of the datasets is provided in the figures below the list.

- *ambient_temperature_system_failure.csv :* it represents the temperature of an office where the anomaly consists of a sudden increase or decrease of the temperature of the ambient (Figure 1).
- *cpu_utilization_asg_misconfiguration.csv :* it is an AWS dataset that consists in monitoring the average CPU usage across a given cluster, and the anomaly happens when the usage is too high (a new machine is required) or too low (fewer machines can be used) (Figure 2).
- *ec2_request_latency_system_failure.csv :* it contains data from a server in Amazon's East Coast data center, and it ends with a complete system failure, therefore in this case the anomaly is a failure of the system (Figure 3).
- *machine_temperature_system_failure.csv :* it monitors through a sensor the temperature of a component of an industrial machine; three anomalies are present in the dataset, a planned shutdown, one that is difficult to be detected and finally a catastrophic failure of the machine due to the second anomaly (Figure 4).
- *nyc_taxi.csv :* each row represents the total number of taxi passengers in New York City into 30 minutes buckets, and the dataset contains five anomalies that occur during the marathon, Thanksgiving, Christmas, New Year's day and a snowstorm (Figure 5).

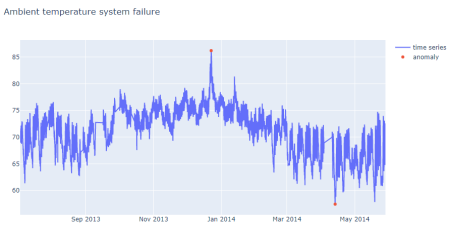

Figure 1: Ambient temperature

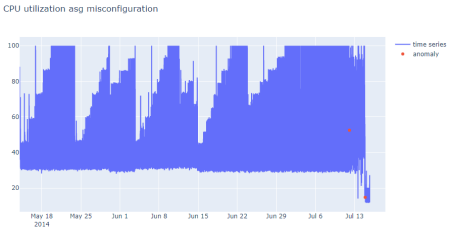

Figure 2: CPU utilization

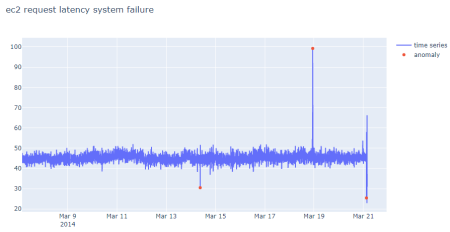

Figure 3: EC2 latency

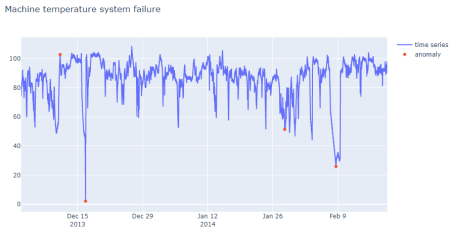

Figure 4: Machine temperature

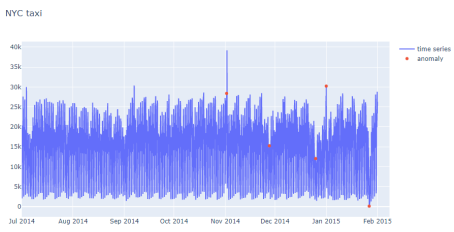

Figure 5: NYC taxi

2.2 Evaluation Metrics

The two phases of the experiment use different evaluation metrics in order to evaluate the algorithms. In fact, for comparison on historical data, it is better to use canonical evaluation metrics, while for the comparison in streaming context the NAB scoring system is more adequate.

To evaluate and compare the three algorithms on non-streaming data, the metrics taken into account are:

- **TP:** true positives, which correspond to the number of outliers that have been detected and labeled correctly.
- **FP:** false positives, therefore the number of points that have been labeled as an outlier but are normal data points.
- **FN:** false negatives, the total number of real anomalies that have not been detected by the algorithm.
- **Precision:** the precision is the ratio between the true positives on the total number of anomalies that the algorithm detected, i.e.
$$precision = \frac{TP}{TP+FP}$$
- **Recall:** the recall is computed by dividing the correctly detected outliers by the total number of anomalies contained in the dataset, i.e.
$$recall = \frac{TP}{TP+FN}$$
- **F1-score:** the F1 score is one of the most used metrics in outlier detection algorithm comparisons because it is a harmonic mean between precision and recall and so it tries to evaluate the algorithms avoiding to focus only on one of the two metrics, i.e.
$$F1score = 2 * \frac{precision * recall}{precision + recall}$$

We decided to focus on these evaluation metrics because they are the standard for comparing outlier detection techniques. In fact, we noticed that most of the outlier detection techniques comparison papers use these metrics in order to evaluate their studies. Alternatively, the AUC score could be a valid evaluation metrics due to the unbalanced nature of the datasets.

For what concerns the streaming detection of the outlier points, the algorithms are compared using different metrics. As explained in the NAB paper [23], the usual metrics for comparing the outlier detection algorithms are not efficient in evaluating algorithms in a streaming context. Therefore, for comparing algorithms in a streaming scenario, the Numenta scoring system will be used. This system runs the algorithm for outlier detection and then computes the TP, FP, and FN. To these three metrics, a weight is assigned based on three different application profiles [24]. If the weight is equal to 1 the profile uses the default value of the metric; if it is greater than 1 the importance of the metric is emphasized; if it is lower than 1 the importance of the metric is deemphasized. In this way, the algorithm is evaluated giving importance to a particular metric. The three application profiles that are described in the Numenta Benchmark, and that are used for evaluating algorithms, are:

- *Standard Application Profile:* the weights are attributed equally across the scoring metrics, therefore none of the metrics have more importance than the others
$$TP = 1$$
$$FP = 1$$
$$FN = .11$$
- *Alternate application profile 1:* also named *reward low fp rate*, it rewards a detector that has a very low FP rate, so the weights that Numenta uses for this scope are the following
$$TP = 1 \text{ (give full credit for properly detected anomalies)}$$
$$FP = 0.22 \text{ (decrement the score more for any false positives)}$$
$$FN = .5 \text{ (decrement the score less for any missed anomalies)}$$
- *Alternate application profile 2:* also named *reward low fn rate*, it rewards a detector that does not miss any true anomalies, and this is performed by Numenta using
$$TP = 1 \text{ (give full credit for properly detected anomalies)}$$
$$FP = .055 \text{ (decrement the score less for any false positives)}$$
$$FN = 2 \text{ (decrement the score more for any missed anomalies)}$$

The final NAB score is normalized so that the baseline value is 0 (value computed using a null detector) while the maximum one is 100.

2.3 Procedure & Participants

During the deployment of the project, it is not required to contact other subjects that can provide data or algorithms. Both data and algorithms are in the public domain. Therefore, they can be used without the necessity of contacting organizations or people to get the utilization rights.

The research project in the analysis does not require any other participants except the authors. The project presents neither surveys nor interviews, and there are not experiments conducted on animals or persons. Therefore, from an ethical point of view, the project does not need to collect and store personal or sensitive data. The authors declare that there is no conflict of interest while conducting the research project.

The first phase of the experiment has been conducted according to the following step:

1. The datasets have been retrieved from the NAB repository [25].
2. The implementations of the three algorithms have been retrieved.
3. The implementations of the three algorithms have been modified in order to work on the selected datasets.
4. Each algorithm has been run on each dataset with the aim of discovering outliers, therefore to predict for each point in the dataset if it is an outlier or not.
5. For each run TP, FP, FN metrics have been computed in order to obtain precision, recall and F1-score.

For what concerns the second phase of the experiment the steps to be followed has been:

1. Install NAB as specified in the NAB repository [25].
2. Modify the implementations of the algorithms in order to meet the requirements of NAB for streaming detection.
3. Run each algorithm on each dataset with the aim of obtaining for each point a label that states if the point is an outlier or not and a probability of being an outlier.
4. For each run, compute the NAB score using the three different profiles.

Following this procedure is possible to answer the research question. In fact, after having completed all the steps, it is possible to evaluate the feasibility of the implementations of the algorithms, how the three algorithms perform in term of precision, recall and F1-score (for the first phase), if the algorithms can work in a streaming context and how they perform in term of the NAB's three different profiles (for the second phase). Moreover, after having collected the results, it is possible to compare them in order to evaluate if an algorithm performs better than the others.

3 Results & Analysis

Before talking about the results, a premise must be done. In order to conduct the experiments and obtain the desired requirements we used a MacBook Pro (2016) with a processor 2.7 GHz Quad-Core Intel Core i7, a RAM memory of 16 GB and a Machintosh Hard Disk. As operative system on the computer is installed macOS Big Sur, version 11.1. The choice of the computer should not affect the results of the analysis, due to the fact that we do not consider time of execution and CPU usage as evaluation metrics. Therefore, on a similar computer but also on a different one, the results of the experiment can be replicated. We are glad to make public our implementations and algorithms with the aim of making people able to replicate our experiment.

3.1 First phase results

As said in the previous chapter, the first phase of the experiment aimed to verify the correctness of the implementations of the three algorithms and to make a first comparison among them on historical datasets. After having retrieved the datasets from the Numenta repository, we retrieved the implementations of the three algorithms. OED implementation [26] has been the easiest to be set but it presented some errors in the computation of the evaluation metrics. After having solved them, the test of the algorithm on the datasets proceeded rapidly. For what concern Donut [27], we faced some problems related to the Tensorflow dependencies and the dimensionality of the model, but the code is quite easy to be understood therefore we managed to solve them. Finally, Telemanom [28] gave us some problems with the

required format for the datasets that is not clear in the repository instructions. However, we managed to solve also this issue and therefore all three implementations were able to provide us results. The next step has been running the three algorithms on the five datasets, collecting for each run the TP, FP, FN metrics. From these values, we computed the precision, recall and F1-score of each run. In Table 1 it is possible to see the results for each algorithm on each dataset in terms of precision (P), recall (R) and F1-score (F1). In the last column, we reported also the mean value on the five datasets for each metric. The results obtained are quite similar, but OED seems to have better performances in this phase. In fact, for each dataset, OED presents the best value for both precision, recall and F1-score. Also the average value shows that OED obtained better results with respect to the others. Interesting results are obtained with the Ambient temperature and the EC2 request latency datasets. The first one is interesting because all the three algorithms achieved the same result, with a recall equal to 1, which means that there are no false negatives, but a precision lower than one due to some points that the three algorithms thought being outliers, but they were not. Even more interesting is the EC2 dataset, in which OED and Donut achieved the perfect result having precision and recall equal to 1, while Telemanom did not find the right outliers achieving precision and recall equal to 0. These results led also to lower average performances for Telemanom.

Table 1: First phase result comparison

Detector	**Amb. temp.**			**CPU util.**			**EC2 lat.**			**Mac. temp.**			**NYC taxi**			**Average**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Telemanom	0.67	1	0.80	0.33	0.50	0.40	0	0	0	1	0.50	0.67	1	0.60	0.75	0.60	0.52	0.52
OED	0.67	1	0.80	0.50	0.50	0.50	1	1	1	1	0.50	0.67	0.60	0.60	0.60	0.75	0.72	0.71
Donut	0.67	1	0.80	0.50	0.50	0.50	1	1	1	0.50	0.50	0.50	0.50	0.40	0.44	0.63	0.68	0.65

3.2 Second phase results

The second phase of the experiment has been more important because it allowed us to answer our research question. In fact, it aimed to evaluate the performances of the three deep learning unsupervised outlier detection techniques in a streaming context and to compare them. In order to conduct the experiment, the first thing we did was the installation of the Numenta Benchmark. This step has been quite simple thanks to the clear instructions reported in the NAB repository. After that, we applied a few modifications to the implementations so that they were able to meet the Numenta requirements. In particular for each point of each dataset that is being processed by an algorithm, a label that indicates if the point is an outlier or not and a probability of being an outlier must be predicted. After having collected this information, for each run a resulting dataset composed of timestamp, value, label and probability has been processed using Numenta Benchmark. The benchmark evaluated the results over the three different profiles for each algorithm on each dataset. In Table 2 it is possible to see the result on the several executions in terms of Standard Profile (SP), Alternative Profile 1 (A1) and Alternative Profile 2 (A2). What emerges, at first sight, is that the A2 score is always greater than SP and A1. Due to the fact that A2 rewards detectors with a low FN rate, the three implementations achieve a low number of false negatives. Another observation is that the results of the three algorithms are really similar and it is difficult to understand if an algorithm is better than the other. Looking carefully at the numbers it is possible to see that Donut and Telemanom achieve better results than OED, but the difference between the three algorithms is very low. Again we can see that Telemanom has some problems with the EC2 request latency algorithm, while this time OED and Donut do not have the perfect results. In general, for the Standard Profile and the Alternative Profile 1, the score obtained by the three algorithms is a bit low, remembering that the maximum score is 100. For what concerns Alternative Profile 2 the results are quite good, considering that the maximum Numenta score with A2 that has been obtained is equal to 75.2. The similarity among the results can be due to the choice of the datasets. In fact, each dataset presents a low number of outliers and therefore the TP can assume few values.

In the second appendix, it is possible to see the graphs resulting from the execution of the different runs.

Table 2: Second phase result comparison

Detector	**Amb. temp.**			**CPU util.**			**EC2 lat.**			**Mac. temp.**			**NYC taxi**		
	SP	**A1**	**A2**	**SP**	**A1**	**A2**	**SP**	**A1**	**A2**	**SP**	**A1**	**A2**	**SP**	**A1**	**A2**
Telemanom	49.6	49.6	66.1	50.0	50.0	66.7	0	0	0	49.1	49.1	65.5	49.1	49.1	65.5
OED	49.1	49.1	65.5	49.6	49.6	65.1	49.8	49.8	66.5	48.3	48.3	64.4	49.3	48.1	65.9
Donut	49.6	49.6	66.2	50.0	50.0	66.7	50.2	50.1	66.8	49.3	49.1	66.2	49.1	49.1	65.7

4 Discussion

In this section, some insights from the results are analyzed. The first aspect to be outlined is that the three algorithms can work in a streaming context. Moreover, they are able to detect outliers of time series that belong to different real-life aspects and not only to the area for which they were studied. For what concerns our research question, in this project has been possible to evaluate the performance of the three algorithms through different metrics and also to compare them on five very different datasets. Therefore the project has been able to answer the initial question. However, the comparison of the three algorithms produced a result that is not definitive. In fact, the results we obtained for the three different techniques are too similar in order to state if an algorithm is better than the others. As we have seen, OED performed better in the historical scenario, while Telemanom and Donut in the streaming one. In general, the results obtained with the different metrics, are very similar, and none of the techniques outperform the others. Further studies must be conducted in order to make a better comparison of the three techniques. Another problem could be the selection of datasets. Despite we tried to select datasets such that the results were not biased and from different areas, the number of datasets and the number of outliers in each dataset could lead to incomplete results. In fact, with only five datasets and with few outliers for each of them, the differences between the three techniques could have some difficulties to emerge clearly.

4.1 Study Limitations

Once having presented the results of our comparative evaluation and discussed the main findings of the research, it is important to highlight the limitations of this study. This allows the reader to position the research in the context of the current outlier detection fields and understand how the findings can support their own research. Finally, the discussion of the limitations lays the foundations for the final sub-section which is dedicated to the exploration of potential future developments, as well as some of the variations and improvements that can be made on both methods and outlier detection techniques.

The first and obvious limitation stems from the limited availability of both time and computing resources. Those restrictions are the most typical for algorithms evaluations that include huge training time for the models, like in Donut or Telemanom. As a result, both training and testing have been reduced to perform the evaluation. In the current approach, we did not do any hyperparameter tuning and also did not use any prior information about the types and characteristics of the datasets in order to achieve an unbiased comparison for situations where no knowledge about data is available and real-world data are the target. This study was restricted to one main dataset, selected to show the most known use cases of outlier detection applications. The evaluation for this study has been restricted for univariate time series and provides a standard methodology for this kind of research setup, but alternative approaches are possible.

4.2 Future Work

Based on the limitations previously described, a set of potential future works can be defined. There are many routes but, the most noticeable is an extension on both the number and type of algorithms analyzed; indeed, as soon as new deep outlier detection algorithms are published and implemented, or become of public domain, they can included on a similar kind of research. Extra types and tuning of hyperparameters can also be added to the new algorithms and those already used in the evaluation. Further types of time series datasets and benchmarks can contribute to the generalization, stress-testing, versatility and flexibility of the study: other evaluation metrics such as AUROC, AUPRC can also be evaluated or the scoring introduce further built upon; integrated of another standard benchmark, such as Yahoo Webscope, can be used to evaluate the behaviour in a streaming scenario with both synthetic and real data. Additionally, more kind outliers and time-series can be included in the evaluation: an extension of the research with multivariate time series datasets is worth to be investigated and the detection of type II, or contextual outliers is missing in the literature. These options are left for further research studies.

5 Appendix

5.1 Abbreviations

- **A1:** Alternative Profile 1
- **A2:** Alternative Profile 2
- **ANN:** Artificial Neural Networks
- **AWS:** Amazon Web Services
- **F1:** F1 score
- **FN:** False Negatives
- **FP:** False Positives
- **NAB:** Numenta Anomaly Benchmark
- **P:** Precision
- **R:** Recall
- **SP:** Standard Profile
- **TP:** True Positive

5.2 Graphs

In this appendix the graphs obtained from the execution of the three algorithms on the five datasets are reported. Figures from 6 to 10 shows the outliers discovered by Telemanom, figures from 11 to 15 the ones discovered by OED and finally figures from 16 to 20 the ones related to Donut.

Telemanom

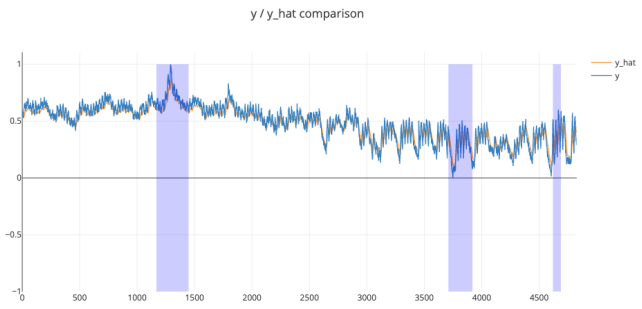

Figure 6: Telemanom - Ambient temperature

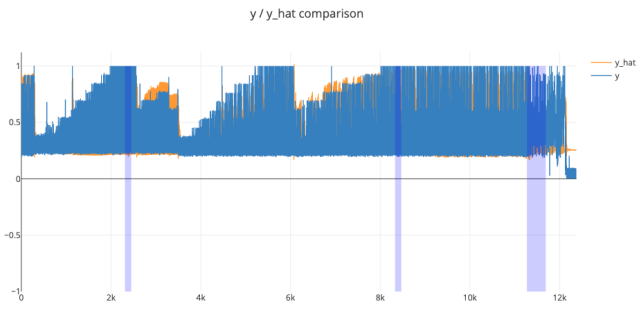

Figure 7: Telemanom - CPU utilization

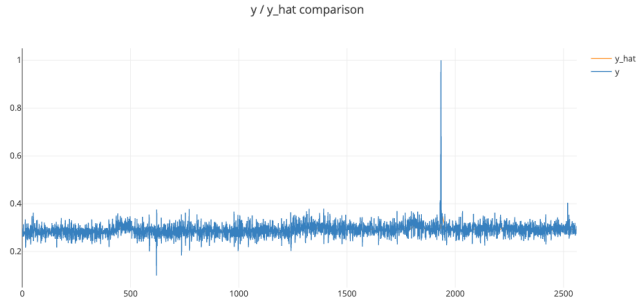

Figure 8: Telemanom - EC2 request latency

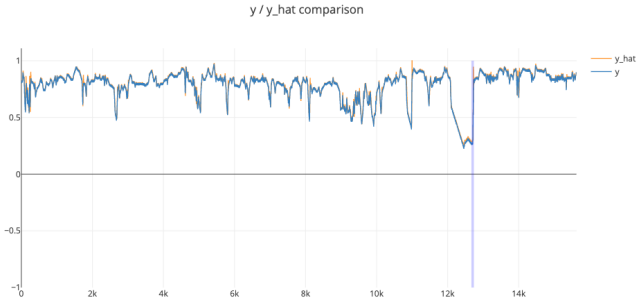

Figure 9: Telemanom - Machine temperature

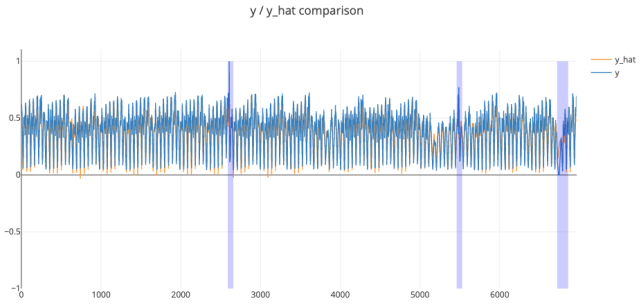

Figure 10: Telemanom - NYC taxi

OED

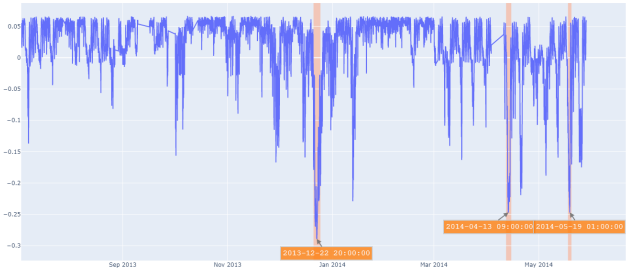

Figure 11: OED - Ambient temperature

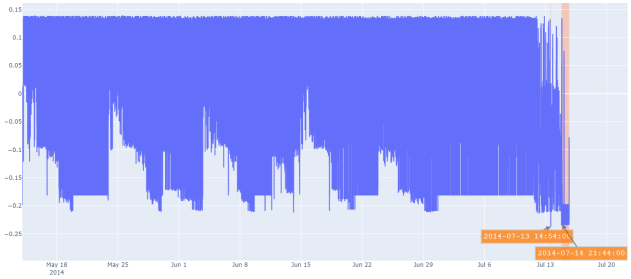

Figure 12: OED - CPU utilization

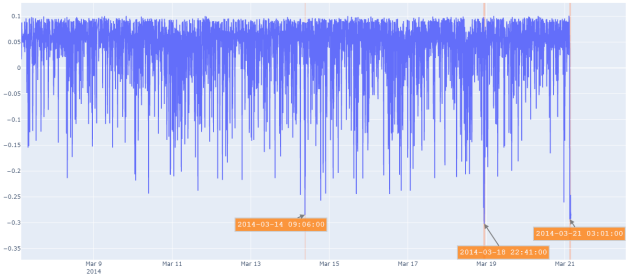

Figure 13: OED - EC2 request latency

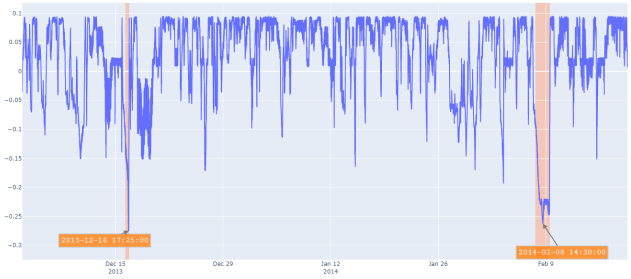

Figure 14: OED - Machine temperature

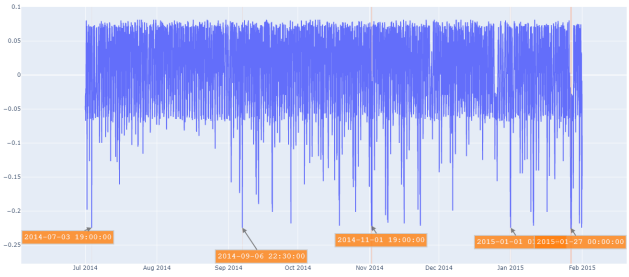

Figure 15: OED - NYC taxi

Donut

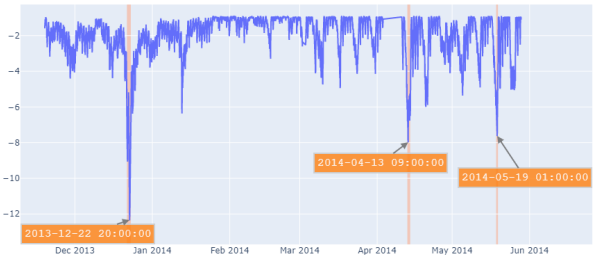

Figure 16: Donut - Ambient temperature

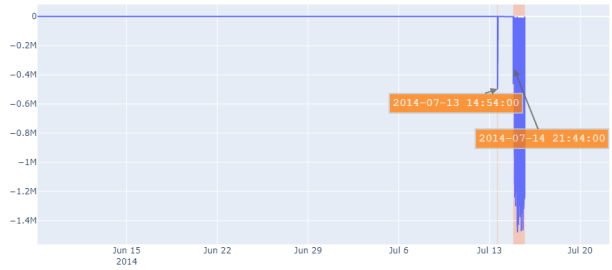

Figure 17: Donut - CPU utilization

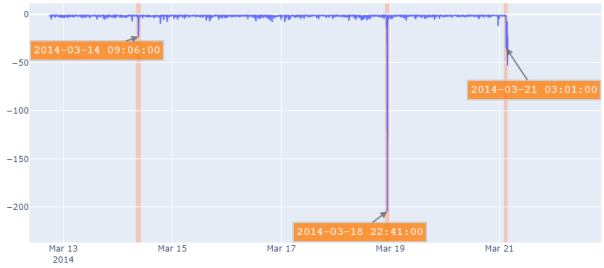

Figure 18: Donut - EC2 request latency

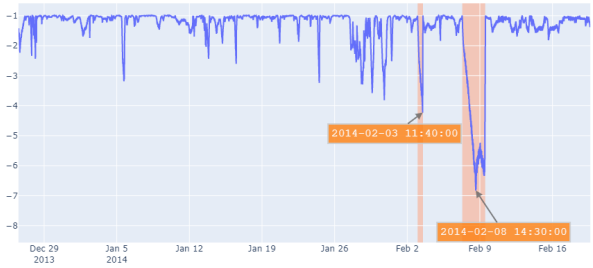

Figure 19: Donut - Machine temperature

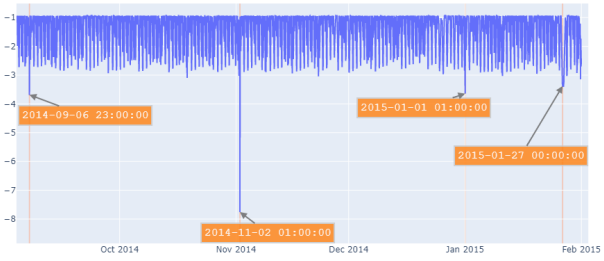

Figure 20: Donut - NYC taxi

References

[1] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977. ISBN 9780201076165

[2] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer Publishing Company, Incorporated, 2016. ISBN 9783319475770

[3] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014. doi: 10.1109/TKDE.2013.184. [Online]. Available: https://dl.acm.org/doi/book/10.5555/2621979

[4] S. Sadik and L. Gruenwald, "Research issues in outlier detection for data streams," *SIGKDD Explor. Newsl.*, vol. 15, no. 1, p. 33–40, Mar. 2014. doi: 10.1145/2594473.2594479. [Online]. Available: https://dl.acm.org/doi/10.1145/2594473.2594479

[5] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Survey*, 2020. doi: 10.1145/3439950. [Online]. Available: https://arxiv.org/abs/2007.02500

[6] D. Hawkins, *Identification of Outliers*. Chapman and Hall, 1980. ISBN 041221900X

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. doi: 10.1145/1541880.1541882. [Online]. Available: https://dl.acm.org/doi/10.1145/1541880.1541882

[8] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," 2020. [Online]. Available: https://arxiv.org/abs/2002.04236

[9] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art." *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2004.00433

[10] M. Munir, M. Chattha, A. Dengel, and S. Ahmed, "A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 12 2019. doi: 10.1109/ICMLA.2019.00105. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8999106

[11] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019. [Online]. Available: https://arxiv.org/abs/1901.03407

[12] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, Jul 2018. doi: 10.1145/3219819.3219845. [Online]. Available: https://dl.acm.org/doi/10.1145/3219819.3219845

[13] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," pp. 2725–2732, 7 2019. doi: 10.24963/ijcai.2019/378. [Online]. Available: https://www.ijcai.org/Proceedings/2019/378

[14] H. Xu, Y. Feng, J. Chen, Z. Wang, H. Qiao, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, and et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018. doi: 10.1145/3178876.3185996. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3178876.3185996

[15] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134 – 147, 2017. doi: 10.1016/j.neucom.2017.04.070 Online Real-Time Learning Strategies for Data Streams. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217309864

[16] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, p. e0152173, 04 2016. doi: 10.1371/journal.pone.0152173. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173

[17] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406 – 421, 2018. doi: 10.1016/j.patcog.2017.09.037. [Online]. Available: https://dl.acm.org/doi/10.1016/j.patcog.2017.09.037

[18] G. Campos, A. Zimek, J. Sander, R. Campello, B. Micenková, E. Schubert, I. Assent, and M. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, 07 2016. doi: 10.1007/s10618-015-0444-8. [Online]. Available: https://link.springer.com/article/10.1007/s10618-015-0444-8

[19] S. Däubener, S. Schmitt, H. Wang, P. Krause, and T. Bäck, "Anomaly detection in univariate time series: An empirical comparison of machine learning algorithms," 2019. [Online]. Available: https://www.semanticscholar.org/paper/Anomaly-Detection-in-Univariate-Time-Series%3A-An-of-D\unhbox\voidb@x\bgroup\accent127a\protect\penalty\@M\hskip\z@skip\egroupubener-Schmitt/1f38083d7874a73bca73d159bfd631be7c68939c

[20] V. Chandola, "Detecting anomalies in a time series database," vol. 09-004, 2009. doi: http://hdl.handle.net/11299/215791. [Online]. Available: https://conservancy.umn.edu/handle/11299/215791

[21] C. Freeman, J. Merriman, I. Beaver, and A. Mueen, "Experimental comparison of online anomaly detection algorithms," 05 2019. [Online]. Available: https://www.researchgate.net/publication/334691874_Experimental_Comparison_of_Online_Anomaly_Detection_Algorithms

[22] Z. Hasani, "Robust anomaly detection algorithms for real-time big data: Comparison of algorithms," pp. 1–6, 2017. doi: 10.1109/MECO.2017.7977130. [Online]. Available: https://ieeexplore.ieee.org/document/7977130

[23] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," pp. 38–44, 2015. doi: doi=10.1109/ICMLA.2015.141. [Online]. Available: https://ieeexplore.ieee.org/document/7424283

[24] A. Lavin and S. Ahmad, "The numenta anomaly benchmark [white paper]," *Redwood City, CA: Numenta, Inc.*, 2015. [Online]. Available: https://github.com/numenta/NAB/wiki

[25] Numenta, "The numenta anomaly benchmark," 2020, accessed on 2021/01/15. [Online]. Available: https://github.com/numenta/NAB

[26] T. Kieu, "Outlier detection for time series with recurrent autoencoder ensembles," 2019, accessed on 2021/01/15. [Online]. Available: https://github.com/tungk/OED

[27] H. Xu, "Www 2018: Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," 2019, accessed on 2021/01/15. [Online]. Available: https://github.com/NetManAIOps/donut

[28] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "A framework for using lstms to detect anomalies in multivariate time series data. includes spacecraft anomaly data and experiments from the mars science laboratory and smap missions," 2020, accessed on 2021/01/15. [Online]. Available: https://github.com/khundman/telemanom