Detecting univariate anomalies in multivariate web analytics time series

Ryno Marree
r.marree@tilburguniversity.edu

June 6th, 2018

Jheronimus
Academy
of Data Science

Master Thesis Data Science & Entrepreneurship

Academic Supervisors:
Dr. W. J. Liebregts
Dr. Ir. M. C. Willemsen
Prof. Dr. Ir. J. J. van Wijk

Company Supervisor:
T. Luijendijk

Abstract

In this study, anomaly detection on web metrics was performed using a recurrent autoencoder. Previous research suggests a multivariate model could be beneficial for anomaly detection performance. Therefore a multivariate detection algorithm is tested to see if it could indirectly detect multiple univariate anomalies in the data. To make these anomalies more interpretable, a sequential ensemble is used where the multivariate detector is combined with univariate detectors to find the variable where the anomaly originated. A synthetic dataset is produced to compare results of different configurations as the real dataset is unlabeled.

Keywords: Anomaly detection, unsupervised learning, recurrent autoencoder, multivariate time series, web metrics.

Acknowledgements

First of all I would like to thank my two main university supervisors, Martijn Willemsen and Werner Liebregts, for valuable support and feedback throughout the project and facilitating the making of this thesis. I would also like to thank some of my peers (Diederik Pondman, Simone Tornioli, Eline van de Ven and Daan van der Vleuten) for their time investments into the peer feedback process, which made for interesting discussions and great feedback. Last but not least a special thanks to Ties Luijendijk, my company supervisor, who always did a great job of facilitating me while writing this thesis at the company and went above and beyond in supporting me on a practical and a personal level.

Table of contents

1. Introduction

1.1 Research Motivation

The data era is coming upon us as the sheer amount of available information has been growing at an ever-increasing rate (Vieira, Leone Filho, & Semolini, 2018). Regardless of the industry, businesses are collecting data in order to make better informed decisions and gain a competitive edge (Hall, Witten, & Eibe, 2011). Considering that the size of datasets is getting bigger, analysts cannot keep up with this volume on their own (Vieira et al., 2018). Being able to automatically comb through this immense amount of data and extract value from it is therefore very worthwhile.

One of the methods that is used to alleviate this pressure on analysts is called anomaly detection (AD). Anomaly detection refers to a multitude of methods that aim to identify items, events or observations in a dataset which do not conform to an expected pattern or other datapoints (Chandola, Banerjee, & Kumar, 2009). Reviewing this unexpected data often translates to critical, actionable information (Cao, Nicolau, & McDermott, 2016). These abnormalities can arise either due to faulty data (e.g. sensor or network failure) or because of genuine sudden changes in values (Zhang, Wang, & Yu, 2016). Both cases are interesting, but for different reasons. Faulty data can have a big impact on drawn conclusions or the subsequent training of models. This data is therefore often removed (Liu, Shah, & Jiang, 2004). On the contrary, 'true' anomalies can contain a lot of new information about the underlying process and should warrant further investigation (Vieira et al., 2018). Use cases of anomaly detection are found in many industries. Examples are network monitoring, fraud detection and sensor failure, but also general-purpose data analysis and data mining (Chandola et al., 2009). Because of its many possible real-world applications, anomaly detection has been a popular research topic (e.g. Bronstein et al., 2001; Rajasegarar et al., 2008; Schaum, 2007).

For this study, the setting in which anomaly detection will be performed is web analytics data. When a company sets up a service like Google Analytics for their website, they gain access to all sorts of metrics pertaining to their website. A lot of these metrics have a time dependency, and significant unexpected changes in their behavior over time are extremely interesting to the website owner. One could think of examples in e-commerce when a wrong price on an item could lead to unexpectedly high or low sales numbers. A real example from the data is the abnormal high values for bounces occasionally, as shown later in this document in **Figure 7**. Obtaining this information in a timely manner is crucial. Besides sales volume, there are numerous interesting web metrics (e.g. page bounces or page load speeds) to track, which can all have a big business impact. Anomaly detection is therefore a strong tool used to monitor and maintain both technical and economic website performance (Cao et al., 2016).

1.2 Problem Statement

A key characteristic of web metrics is that they often come in the form of time series, which means all the variables have a time component. An important difference in time series analysis is the number of variables that are being considered. A time series is called univariate when there is only one variable of interest that changes its value over time. However, a single variable often does not tell the whole story. When trying to describe a more complex system, it often makes sense to look at multiple variables at once, to get a more complete picture. A time series where each point in time contains a tuple of values from multiple variables is called a multivariate series.

For anomaly detection in time series, the task is to find points in time where the value(s) differ(s) significantly from the norm (Gupta et al., 2013). This is where univariate and multivariate anomalies are fundamentally different. The conclusions (found anomalies) that can be drawn from multivariate anomaly detection do not say anything about any individual time series because multivariate anomalies concern the 'system' as a whole at that particular point in time. In other words, a multivariate anomaly detector decides if the combination of variables is anomalous at that point in time, combining several input into a singular output. In doing this, multivariate detection takes all variables and their interrelationships into account.

This means there are two main ways to design an anomaly detection system that controls multiple variables. Either every variables should have its own detector or one multivariate detector for all the variables simultaneously. There are various reasons why often times a multivariate detector is preferred. If there are multiple detectors, an anomaly in one could be caused by another, leading to overestimation of the number of anomalies. Secondly, a moderately sized anomaly affecting all variables might go unnoticed in univariate detection, as they do not combine information from different series. Thirdly, multiple univariate detection methods are often inferior as the collective dynamics of the series are not accounted for (Tsay, Peña, & Pankratz, 2000). This extra information packed in the multivariate time series can often help increase performance (Galeano, Peña, & Tsay, 2006). Researchers have tried to indirectly include this information, by using cross-correlation values as extra univariate features (Qiao, He, Cao, Huang, & Zhang, 2012). Using individual detectors give the users more control over detection in different variables. This can either be seen as adding undesired complexity or a desired level of customization, depending on the use case and whether or not different variables need to be treated differently. Another benefit of univariate detectors is that there is always a specific series or variable to link to an anomaly, aiding a great deal in interpretability and actionability.

However, a direct comparison between univariate and multivariate detectors has never been made in trying to detect multiple univariate anomalies. This is because the anomalies found in a multivariate system are different from that of a univariate system, the two outputs cannot be directly compared. In order to make a multivariate anomaly interpretable, the origin or type of anomaly needs to be known. Techniques that try to determine the causal events of an anomaly are called *root cause analysis* techniques (Lin et al., 2016). The root cause of an anomaly in this case could mean the univariate series where the anomaly is most significant, which could greatly help with the interpretability. Manual visual approaches (Stoffel, Fischer, & Keim, 2013) and also methods like Dynamic Time Warping (DTW) to match metrics to anomalies (Lee, 2017) have been suggested for this in literature, but these only work in a univariate case. It is yet to be determined if a combination of univariate and multivariate detectors (in a so-called *ensemble*) could perform this same task. Detecting multivariate anomalies until one is found to then aim to make it interpretable by using univariate detectors at that specific point in time. Reaping the benefits of both multivariate and univariate detectors while eliminating most of the drawbacks. This leads to the current problem statement:

All data streams in a multivariate system need to be monitored individually to detect interpretable anomalies.

1.3 Research Questions

Two research questions are formulated to address this problem statement. The first research question will quantify the observable differences in performance between the univariate and multivariate approaches on finding multiple univariate anomalies. Multiple univariate anomalies are used as the goal is to work from a multivariate anomalies towards interpretable univariate anomalies, thus that is what they shall be tested on. The second question warrants a look at the practical relevance of an anomaly detection ensemble system, in terms of both performance and interpretability.

Before even considering a more complicated setup involving multiple algorithms that deal with different dimensionalities, it is important to first ask whether or not it is even desirable to model time series in a multivariate manner. The impact on the ability to detect univariate anomalies is decisive for determining the viability of a multivariate detector. This leads to the following research question:

1. *To what extend does using a multivariate detector change the performance on multiple univariate anomaly detection?*

Besides looking at individual performance of different algorithms, it can be very interesting to look at combinations of them together in an ensemble. The aim is to balance the strengths and weaknesses of either and end up with a greater whole. A multivariate algorithm could be used as a base detector with possible univariate extensions after it once an anomaly is detected. Trying to identify the feasibility of this leads to the second research question:

2. *To what extend can an ensemble including univariate detectors be used to improve performance and interpretability of a multivariate anomaly detection system?*

1.4 Research Methods

To answer the posed research questions, an unlabeled dataset containing time series of web metrics is utilized, which was obtained through Google analytics. These time series consist of variables like number of page views and web sessions, which display clear double seasonality (daily and weekly). In order to compare a univariate and a multivariate approach, a state-of-the-art recurrent neural network architecture, called an autoencoder, will be used to model the behavior of the multivariate time series. It does this by creating a compressed and informationally dense lower dimensional representation of the input data, which is used to reconstruct the input as output. Univariate and multivariate anomalous instances will be derived from the reconstruction error (the difference between input and output). Either the reconstruction errors of multiple variables will then be combined into one distance metric in order to detect multivariate anomalies at that specific point in time or the variables will be assessed separately in order to detect univariate anomalies. The recurrent autoencoder was selected because recurrent networks are well able to deal with temporal dependencies in data and autoencoders are particularly well suited for anomaly detection, as will be further explained in section **2. Background**. Also, both univariate and multivariate distance metrics can be calculated for the reconstruction errors of the autoencoder, which makes it well suited to compare the two without using entirely different models.

The results from the autoencoder setup will be compared to a well-documented and easy to implement univariate statistical approach as a baseline. This algorithm is called Seasonal Hybrid ESD, which is made available by Twitter in an R package. Twitter's S-H ESD is a statistical algorithm that is able to deal with seasonality and contextual anomalies in time series. This algorithm has already been tried and tested on similar data and is therefore a strong candidate for an anomaly detection system in the setting of web metrics data. The algorithms will be assessed on a standard performance metric called the F1-score, which is a weighted combination of both recall and precision. Lastly, sequential and parallel combinations of univariate and multivariate detection methods will also be investigated to in an effort to improve an anomaly detection systems' performance and make it more interpretable.

2. Background

Abnormal or unexpected datapoints get labelled many different names, but the most common in literature are anomalies (e.g. Lin, 1994) or outliers (Knorr & Ng, 1998). It appears the term outlier is more often used for unwanted behavior (i.e. faulty data) that should be removed from a dataset before training a model (also-called *data cleansing* (Goldstein & Uchida, 2016)). On the other hand, anomalies are considered interesting in and of themselves and are the subject of study. Making the distinction between outliers and anomalies is non-trivial and requires knowledge of the origin of the observation. Despite their differences, the two terms are often used interchangeably and the techniques are the same (Chandola et al., 2009). The term 'anomalies' aligns better with the intent of this research and is therefore henceforth the descriptor of choice.

Detecting anomalies in time series is however no straightforward task and involves many decisions and components before a final output can be produced. The rest of this section intents to cover and explain the entire process from model selection to final output generation. It will do so by first discussing in section **2.1 Type of Anomalies** what type of anomalies there are and which are relevant for this research. Secondly, the choice of the final model will be impacted by the (non) presence of labels in the data. This will be discussed in section **2.2 Data Labels**. Furthermore the temporal aspect of the data also plays an important role in model selection, which will be discussed in section **2.3 Modelling Techniques**. Once it is clear what types of models are suited for this particular problem, relevant pre- and postprocessing steps can be discussed, along with the models of choice for this research, as shown in **Figure 1**. Section **2.4 Time series decomposition** will elaborate on how to deal with seasonality in the data and why STL decomposition is used to break down the raw input data into parts that are viable for further modelling. Once the seasonality and trend components are removed from the data, either an ESD test (**2.5 Seasonal Hybrid ESD**) or an autoencoder (**2.6 Autoencoders**) can be applied to detect anomalies in the residual (as shown in **Figure 1)**. The ESD test is applied directly to the output of the time series decomposition, but the autoencoder tried so model the remaining structure in the residual to improve performance. The output of an anomaly detection algorithm can come in three different forms (bottom of **Figure 1**) and post-processing steps might be required to get the output in the desired format. Considerations for this post-processing step will be discussed in section **2.7 Output of Anomaly Detection**. Section **2.8 Performance measures** will examine how to compare the effectiveness of different algorithms, and finally section **2.9 Ensembles** will talk about how combinations of different algorithms could possibly be used to improve performance or interpretability of an anomaly detection system.

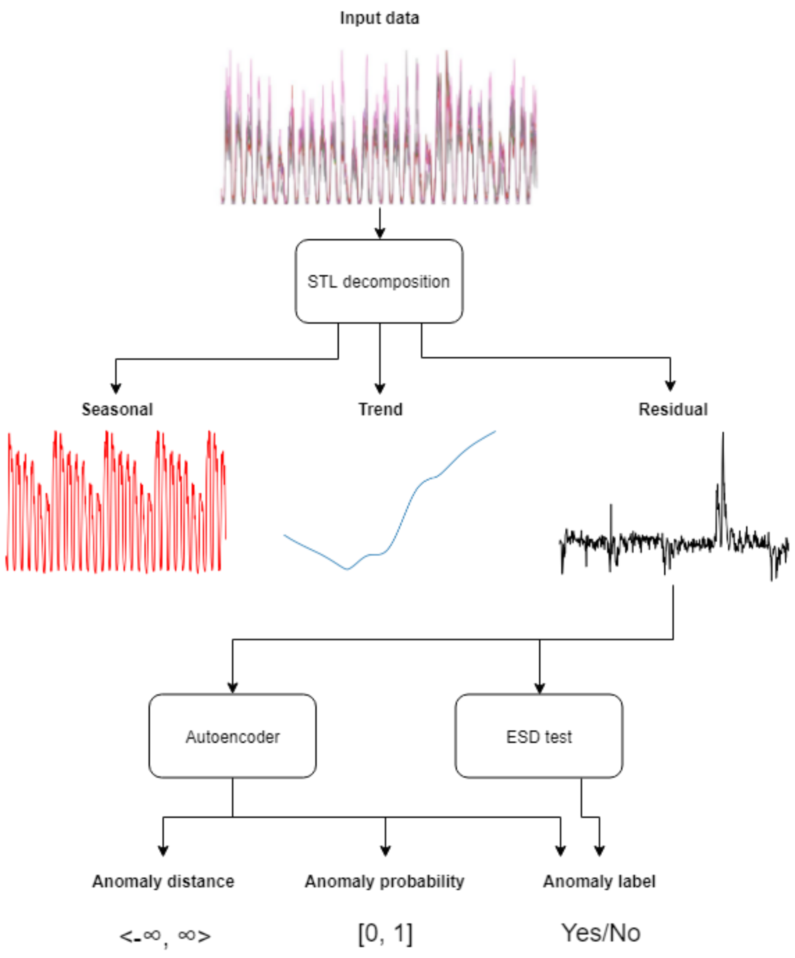


Figure 1. Time series anomaly detection.

<u>2.1 Type of Anomalies</u>

The context of a problem is always highly influential on the possible ways to tackle it, as different types of data require different approaches and different use cases require different solutions. The most studied context for anomaly detection has been cross-sectional data (e.g. Ben-gal, 2005; He, Xu, Huang, & Deng, 2004; Schaum, 2007). The focus of this study however, is specifically on anomaly detection in time series, which is a subset of the field. Even though this setting has been less studied, time series data is very prevalent, as most real-world data has a temporal component (Längkvist, Karlsson, & Loutfi, 2014). Time series analysis is often considered one of the top challenging problems in data mining (e.g. Yang & Wu, 2006). What makes time series different from cross-sectional data is that the features do not relate to different items, but instead to evenly spaced discrete time steps. This changes the problem of anomaly detection from looking at deviating items in a static set to deviating observations over time. The detection of these deviating observations is interesting in a one-dimensional time series (i.e. univariate) as well as in *n*-dimensional time series (i.e. multivariate). Suppose a multivariate time series *D* consisting of multiple (univariate) time series X_i. In this setting, multivariate anomaly detection aims to find points in time where the observed values of one or more variables in *D* differ significantly from the norm. It is very common to first model the normal behavior of the variables X_i (vectors). This model is then used to detect deviations from expected behavior or a change in the relationships between variables. These are then classified as anomalous. (Cheng et al., 2009). In univariate anomaly detection, the goal is similarly to find unexpected behavior, but instead, only one variable X_i is considered and a possible relationship with other variables is neglected.

In general, there are three possible types of anomalies; *global*, *contextual* and *collective* anomalies. Global and contextual anomalies are both point anomalies (i.e. concern one specific data point), in contrast to the grouped nature of collective anomalies. A data point is considered a global anomaly when it is outside the range of the entire dataset. On the other hand, if a data point is only anomalous given its specific context of surrounding data points, it is termed a contextual anomaly (Chandola et al., 2009). Another term for contextual anomalies is *conditional* anomalies, as they are only considered as such under certain conditions (Xiuyao, Mingxi, Jermaine, & Ranka, 2007). The last type of anomaly, the collective anomaly, is defined as a subset of the data where the collective values deviate significantly, but the individual points in the subset are not anomalous themselves. In a time series setting, these deviating subsequences are also referred to as *discords* (Keogh, Lin, & Fu, 2005).

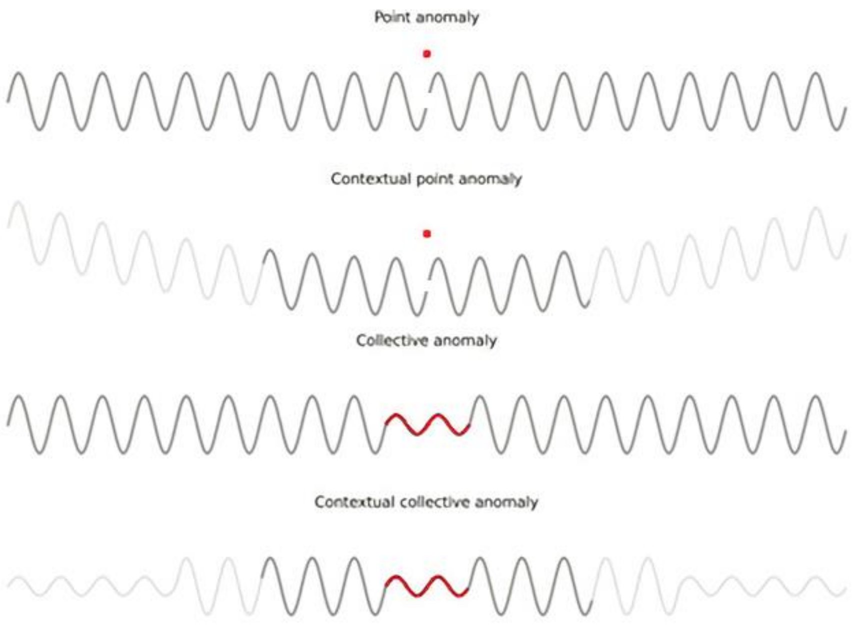


***Figure 2. Contextual versus non-contextual point and collective anomalies.*[1].**

Figure 2 demonstrates what point and collective anomalies could look like in a time series. The first time series contains a point anomaly, where one instance is outside the observed global range of normal values, therefore a global point anomaly. The other red dot in the second time series is however only considered anomalous as it occurs in the middle of the sequence, where the range of expected values has changed compared to the start and end of the sequence. The latter two graphs containing anomalous subsequences can be explained using similar reasoning. The first collective anomaly does not fall outside of the normal range of the dataset, but its behavior is unexpected in comparison to the rest of the series. The contextual collective anomaly is considered as such because it appears to be disrupting the longer alternation pattern in the series.

[1] Retrieved from: https://datascience.stackexchange.com/questions/11446/anomaly-detection-in-time-series-data-help-required

The techniques used for detecting collective anomalies are very different from the global and contextual anomaly detection techniques because it concerns anomalous sequences instead of points (Chandola et al., 2009). Collective anomalies are also close to treading into the realm of *breakout detection* because of their grouped nature. In breakout detection, the interest lies in trend changes (*ramp-up* or *mean-shift*) and not abnormal instances (Kejariwal, 2014). Besides requiring different techniques, the interpretation of collective anomalies is also different. They concern longer periods of what could be a (temporary) change in the normal behavior. The problem is that while multiple subsequent point anomalies could be interpreted as (or used as a proxy for) a collective anomaly, if one is specifically looking for collective anomalies, many point anomalies will be missed. To take a broad approach, this research will therefore focus on detecting global and contextual point anomalies.

2.2 Data Labels

Whether or not labels are available in the training data is a critical factor in determining the kinds of options that are available to detect new anomalies. Depending on this, the types of techniques are split in two: *signature-based* and *anomaly-based* detection (Patcha & Park, 2007). One is recognizing patterns (signatures) of known anomalous behavior (i.e. known labels) and the other is modelling normal behavior to label strong deviations as abnormal (Bhuyan, Bhattacharyya, & Kalita, 2016).

Unlike its counterpart, anomaly-based detection can be responsive to any type of anomalous behavior and label it as such (García-Teodoro et al., 2009). On the other hand, signature-based detection is better for classifying different known types of anomalies and will output a class label (Fernandes, Rodrigues, & Proença, 2015). From the perspective of a classification task, one can be viewed as a classification with only one class (normal behavior), while the other is a multi-class classification problem (different classes of anomalies or normal behavior) (Dau, Ciesielski, & Song, 2014). Because signature-based anomaly detection requires labels beforehand, it is necessarily a supervised learning task, since it is creating a mapping between input and target variables. In a lot of contexts claiming to exactly know the anomalous behavior is a strong assumption (Ma & Perkins, 2014). Thus anomaly-based detection techniques are often better suited, unless the scenario lends itself particularly well to a signature-based classification task.

A prerequisite for a supervised learning task is that labelled instances are available, as shown in **Figure 3**. Gathering these labels however, is often either non-trivial or practically impossible. Often times gathering labels that cover all possible types of anomalous instances is many times harder than getting labels for normal behavior. Furthermore, in many instances the anomalous behavior is dynamic in nature, meaning that new types of anomalies might arise, for which labels were never collected in the training data (Chandola et al., 2009). Unsupervised adaptations on the other hand, assume that the test data contains very few anomalies and that the model training process is robust to these few anomalies. Preferably, a rather 'clean' dataset is needed for this as including anomalies in the training data runs the risk of modelling this behavior and thereby worsening performance. When the dataset is known or made to be clean, learning normal behavior is called *semi-supervised learning* (***Figure 3*** (*b*)). It tries to combine the best of both worlds as it uses only negative instances to train (i.e. model only 'normal behavior') (Song, Jiang, Men, & Yang, 2017). This idea is also referred to as *one-class classification*

(Moya & Hush, 1996) and is essentially the inverse problem of anomaly detection as it tried to classify only a single class to determine if new data belongs to the same (normal) class. Even though semi-supervised learning tries to deal with the general issues of collecting labeled data of anomalies, gathering training data with only negative (non-anomalous) instances is might still be very difficult in time series data if there is no clear definition of an anomaly. In the case of web-based data from Google Analytics, it is very unlikely labels are already present unless the data owner already has an anomaly detection system running. This limits the choice of algorithms to unsupervised ones.

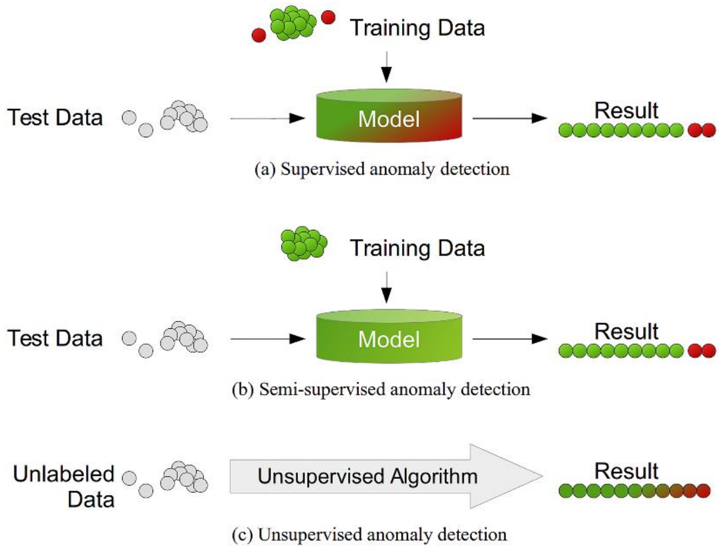


Figure 3. Different anomaly detection modes (Goldstein & Uchida, 2016).
(a) Supervised anomaly detection uses a completely labeled dataset for training. It classifies both normal and anomalous behavior. (b) Semi-supervised anomaly detection only uses normal behavior for training. Strong deviations from the model in the test data are classified as anomalous. (c) Unsupervised anomaly detection algorithms only make use of intrinsic information in the dataset to detect points that are dissimilar to the majority.

2.3 Modelling Techniques

Since the algorithm needs to be unsupervised, the options for the types of models are limited. In an unsupervised setting, anomaly detection can be done by trying to learn the general behavior of the model in the face of noisy data. Learning this normal behavior is then a one-class classification problem. Since the data has a temporal aspect, the model needs to be able to deal with this. The choice here is between statistical models and machine learning models.

Statistical models can not inherently deal with temporally structured data. These models therefore require pre-processing steps to turn the time series into a standard distribution that is testable, which will be further discussed in section **2.4 Time series decomposition**. Both a statistical model and a machine learning model will be used for anomaly detection in this research. Either choice has its advantages and disadvantages over the other and will be discussed more in detail later.

Recently in the world of machine learning, neural networks have become very prominent because of their great performance on a variety of tasks (Ng, 2006). A major issue with standard neural networks however, is that they make the assumption that all the datapoints are independent (Kwon et al., 2017). To solve this issue and be able to deal with time-dependent data points, sequential neural networks have emerged. These special types of neural networks are called recurrent neural networks, or RNN's (Bao, Yue, & Rao, 2017). Recurrent neural networks are known to be strong sequence modelers (Aldosari, 2016), and have become very popular for this task. These networks work by using their hidden state at the last timestep as input for the next timestep hereby 'remembering' what happened in the past. State-of-the-art models like LSTM (Long-Short Term Memory) network and its successor the GRU (Gated Recurrent Unit) (Cho et al., 2014) network show promising performance on sequence modelling tasks (Ergen, Mirza, & Kozat, 2017). By having the memory unit retain time-related information for an arbitrary amount of time they can solve the problem of vanishing gradients earlier RNN structures faced, which prevented learning longer time dependencies (How, Loo, & Sahari, 2016). LSTM and GRU networks have shown fairly similar performance (Chung et al., 2014), but the training time of GRU networks is considerably less, as it has fewer parameters because of the lack of an output gate (Ergen et al., 2017). Therefore a recurrent neural network based model using GRU nodes will be selected as the algorithm of choice to model the time series.

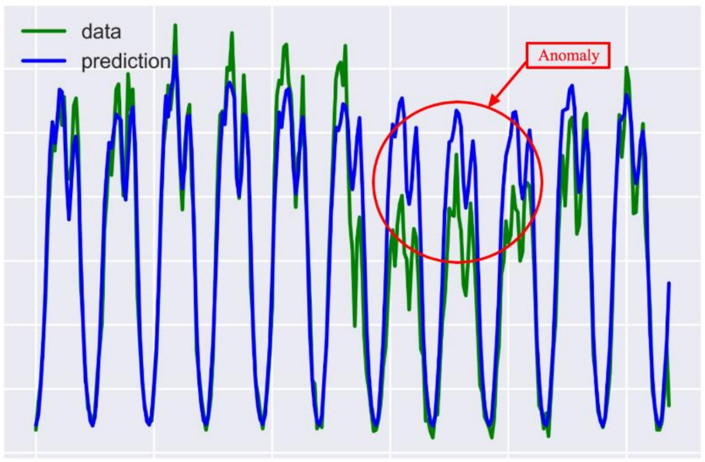


Figure 4. Prediction based anomaly detection (Tiunov, 2017).

After choosing the preferred type of model to fit on the data, prediction based methods are used for finding anomalies. There are three steps to this process. First, a prediction is made for the observed values, as demonstrated in **Figure 4**. Next, this predicted value is compared to the observed value and assigned some distance metric. Lastly, this distance metric is evaluated to determine whether or not the observed point is anomalous. Even though this might not be precisely what is happening in the algorithms, it is still conceptually extremely similar.

2.4 Time series decomposition

Time series that relate to consumer behavior often portray a cyclic nature, as they contain variations that occur at specific regular time intervals. Being people, we naturally have certain periodic behavior, due to for instance our circadian rhythm, a workweek and weekend cycle and many other short or long-term cyclical behavior. Time-dependent metrics that pertain to human behavior therefore often display so-called *full-periodic patterns*. This is the case when every point in time is a part of the cyclic behavior of the time series (Han, Dong, & Yin, 1999). In the context of consumer data, this periodic behavior is commonly referred to as seasonality.

Real life time series are however never perfectly cyclical. Often there are also long-term trends at play and the data is inherently noisy. This idea is also conveyed in the concept of *time series decomposition*. A time series is viewed as a composite of a seasonal element, a trend element and a noise element (also referred to as the residual or remainder) (West, 1997). These three parts fit together through either additive or multiplicative relationships (Agrawal & Adhikari, 2013). Using these models of their respective elements, efforts have also been made to decompose these series into their respective ingredients. After successful decomposition, the residuals can be regarded as an estimate of the errors of the model fit (Box & Pierce, 1970). An example of such decomposition is shown in **Figure 5**, where a well-known dataset[2] of atmospheric CO_2 concentration is decomposed.

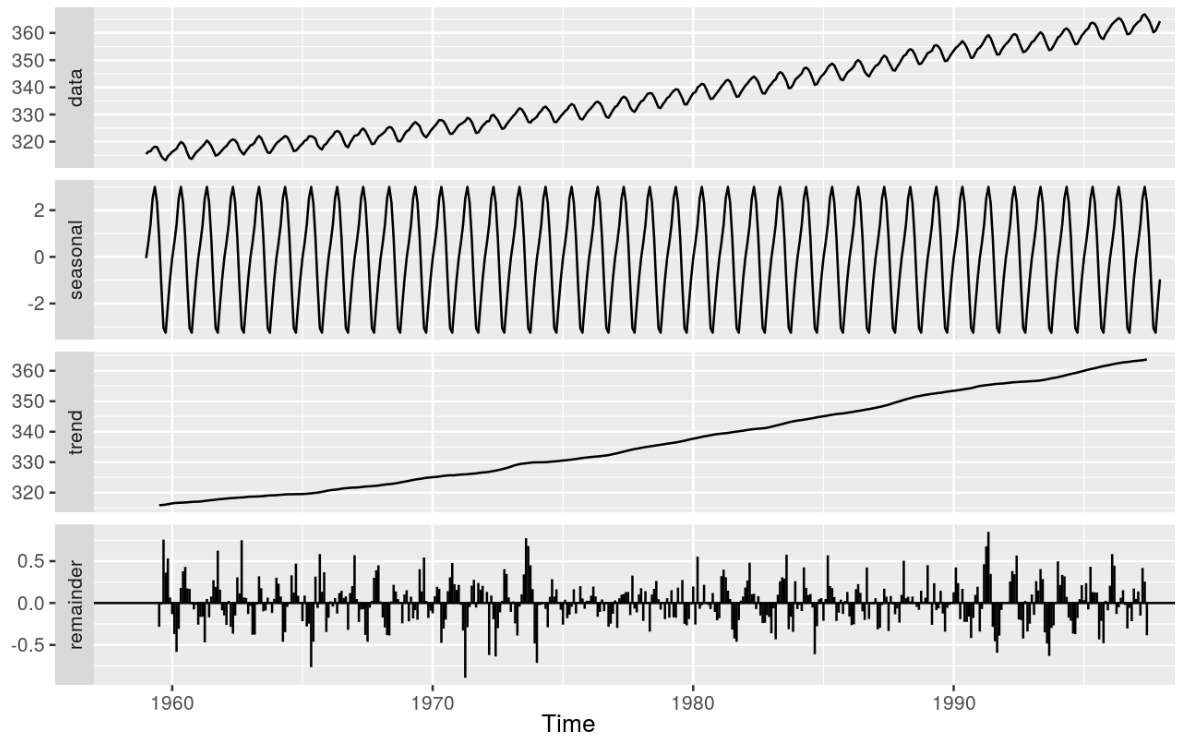


Figure 5. Seasonal decomposition of the CO_2 concentration time series.[3]

The goal of time series decomposition is often to study their individual components and gain knowledge and understanding about them separately. Removing seasonality from a time series leaves it with a simpler pattern that can be studied for various use cases (Cleveland & Tiao, 1976). Seasonality itself is also often not the most interesting part, as the long-term trends can be more important indicators of overall shifts or behavior. Detrending and deseasonalizing (i.e. removing those components from the original series) can also be critical as it prevents multiple time series from displaying correlation if in reality these correlations are not present (Horvatic, Stanley, & Podobnik, 2011).

Besides explorative use cases into the time series behavior, time series decomposition has also been demonstrated to be useful for improving performance in forecasting. For instance, Zhang & Qi (2005) noted that decomposing their monthly input data[4] and removing the seasonal and trend elements improved their neural network performance. After successfully decomposing a time series, the seasonal effects can be completely retained in the final result for a forecast,

[2] https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/co2.html

[3] Picture from: http://pkg.robjhyndman.com/forecast/reference/autoplot.seas.html

[4] The data contained a dozen monthly series containing retail sales amounts, industrial production numbers and privately owned housing numbers.

simply by re-adding it to the predicted residual (together with the trend, if any). Often the assumption is made that the seasonal component does not change over time (otherwise it will be reflected in the residual), under this assumption the variance and bias of the seasonal forecast can be considered zero. Additionally, once the seasonal component is removed, the trend curve becomes very smooth and is approximated more easily, with smaller bias and variance as well (Zhang, Shen, Zhao, & Yang, 2017). Strong and invariant estimates for the seasonal and trend components mean the variance is captured in the residuals and can be modelled directly. The residuals of time series decomposition are also likely to be stationary (i.e. statistical characteristics do not change over time), as most regular variance over time is captured in either the seasonal or trend components. This is important because when the input data is non-stationary, during training, a machine learning model will try to learn different input distributions as time changes and will therefore eventually learn less successfully (Lee, 2017). In conclusion, preprocessing steps like seasonal and trend decomposition are essential for assuring decent performance on time series modelling tasks.

The specific algorithm used for series decomposition in this research is STL, which stands for Seasonal and Trend decomposition using LOESS. STL was designed specifically with robustness in mind (R. B. Cleveland, Cleveland, McRae, & Terpenning, 1990). It is more robust to outliers than conventional decomposition algorithms (Hochenbaum et al., 2017), which is very important for a use case in anomaly detection. LOESS ('locally-weighted scatterplot smoothing') uses local regression to smooth the data in a robust fashion (Cleveland, 1979). Once the seasonal and trend elements are removed using LOESS, the remaining residuals will be close to normally distributed (Jensen, Van Do, Nguyen, & Arnes, 2016). These residuals will be used as input for the neural network algorithm to detect anomalies in.

2.5 Seasonal Hybrid ESD

Before the advent of machine learning, top of the line statistical models were the go-to for modelling and predicting time series. Statistical models have some advantages over more complicated machine learning models. Firstly, they are often less computationally expensive and secondly, conceptually more intuitive. For less complex problems, simpler statistical models also perform fine. One of the latest advancements in statistical modelling for anomaly detection is the new Seasonal Hybrid Extreme Studentized Deviate (S-H ESD) algorithm (Hochenbaum et al., 2017). The algorithm works well with seasonal time series data and can detect point anomalies in an unsupervised manner. It was invented and popularized by Twitter, who made available it in a package[5] for the popular statistical programming language R.

S-H ESD is an extension of the well-known Extreme Studentized Deviate test, which on its own can be used to detect outliers in normally distributed data (Rosner, 1975). The test is used to detect a single outlier by locating the point furthest away from the mean (Rosner, 1983). Under the assumption there are *n* outliers, this test is then repeated *n* times, where after each iteration the outlier is removed and critical values are recalculated. This is called the Generalized ESD algorithm (Jensen et al., 2016). Real-world data however, is often not normally distributed and can have a multimodal distribution. To remedy this, STL decomposition is used, which produces a residual which has a unimodal distribution that ESD can test. The authors also suggest that a more stable trend in the form of the median can be used in order to combat extreme anomalies from corrupting the residual,. Additionally, for data with a high fraction of anomalies, the paper

[5] https://github.com/twitter/AnomalyDetection

suggests to use the median and the Median Absolute Deviation (MAD) rather than the mean and the standard deviation for the ESD tests, as the latter are more sensitive to outliers.

The S-H ESD algorithm contains a few parameters that allow for customization of its usage. A required parameter is a value for the maximum amount of anomalies that should be detected, here rewritten as a percentage of the total dataset. The second most important parameter is the statistical significance level alpha which is used to accept or reject an anomaly.

There have not been many attempts at an open source general benchmark for anomaly detection, but recent work from Ahmad et al. (2017) changed this, with their Numenta Anomaly Benchmark (NAB)[6]. Algorithms are judged on three key aspects in the NAB: anomaly windows, the scoring function, and application profiles. On a benchmark for an artificial dataset with temporal anomalies, twitter AdVec (the version that works with indexless vectors) surprisingly performed the best out of the 7 tested algorithms. This was also the only area where it outshined its competition as the overall score on the benchmark made it end up in the middle of the overall rankings (Ahmad et al., 2017). Other works have also noted some shortcomings of the S-H ESD algorithms as it lacked the ability to detect longer term trend changes (Bodrog, Kajo, Kocsis, & Schultz, 2016). Upon release, users subjected the algorithm to tests which were made public on GitHub[7], where the ease of use combined with decent performance was hailed, but also some drawbacks were exposed. For one, it failed to detect 'negative seasonal anomalies' and R was said to be unsuitable for real-time detection. Real-time detection is however not a requirement in this research.

Very recently an improvement over the Twitter algorithm was suggested in the literature, called Enhanced S-H ESD (or S-H ESD+) (Vieira et al., 2018). This proposed solution aimed to address some of the weak points of the original base algorithm, whereby it minimized the number of false positives, automatically identifies optimal parameters and runs with faster response times. It was also tested on the NAB benchmark and indeed saw a rather significant performance increase of the overall score on the key aspects of around 25% (Vieira et al., 2018). The original S-H ESD algorithm was however selected for this research as there is an open source implementation readily available, it has been more thoroughly tested and provides a great balance between ease of use and performance.

2.6 Autoencoders

Machine learning (more specifically neural network) models have become increasingly popular for time series analysis (Aras & Kocakoç, 2016). They are easy to work with because they make no assumptions about the underlying data (Dau et al., 2014), a stark contrast to most statistical models. Neural networks also started competing with statistical methods on time series problems quite a while ago (Tang, de Almeida, & Fishwick, 1991). These neural network techniques have become very successful on a wide variety of tasks because they often produce more accurate results than other computational learning models (Agrawal & Agrawal, 2015). Deep learning techniques (i.e. neural networks with many layers) often achieve better results than other techniques when the dimensionality increases (Shipmon, Gurevitch, Piselli, & Edwards, 2017). This is because neural networks are great at capturing the hidden relationship between variables (Hinton & Salakhutdinov, 2006), which can increase performance. Even

[6] https://github.com/numenta/NAB
[7] https://github.com/martin-magakian/Anomaly-Detection-test

though there is a lot of research recently being conducted in the field of deep learning, research applying it to the anomaly detection problem in time series is still sparse (Gamboa, 2017).

As discussed before, a common way to implement anomaly detection is to produce predictions or expectations through a model of the time series and compare these to the observed values. Often these predictions for observations are done one or more steps ahead, which are then used as a forecast. Many forecasting methods suffer if the input data contains outliers, but another (more robust) way of making 'predictions' for anomaly detection is through *Autoencoders (AE)* (Japkowicz et al., 1995). Autoencoders are a special type of neural network, where the input equals the output. The difference between the input and reconstructed output is called the *reconstruction error,* minimizing this error through training is the objective function of this network (Kwon et al., 2017). The use of weight matrices corresponding to inputs and hidden layers with the addition of non-linear activation functions allows for the network to learn complex behavior (Maass, 1997).

The analogy to forecasts is that implicitly a 'prediction' for zero steps ahead is made, predicting what the current batch of timesteps should look like, given the learned past. Reconstructing input sounds like a trivial task, but to make it a useful exercise, some additional constraints are placed on the network. To stop the network from learning the identity function, the size of the hidden layers is made smaller than the input, forcing the network to come up with lower dimensional representation of the data, as illustrated in **Figure 6**. This is also often called a *bottleneck* filter (Khan & Madden, 2014). The intention is that while the bottleneck prevents learning the full identity map, the compressed information in the bottleneck can still properly reconstruct the input (Vincent & Larochelle, 2008). The time series input gets encoded to latent variables in the bottleneck layer, which is equivalent to feature extraction (Hsu, 2017). Hidden encoding of the AE can also be used as feature representation of the multivariate time series (Wong & Zhiyuan, 2018).

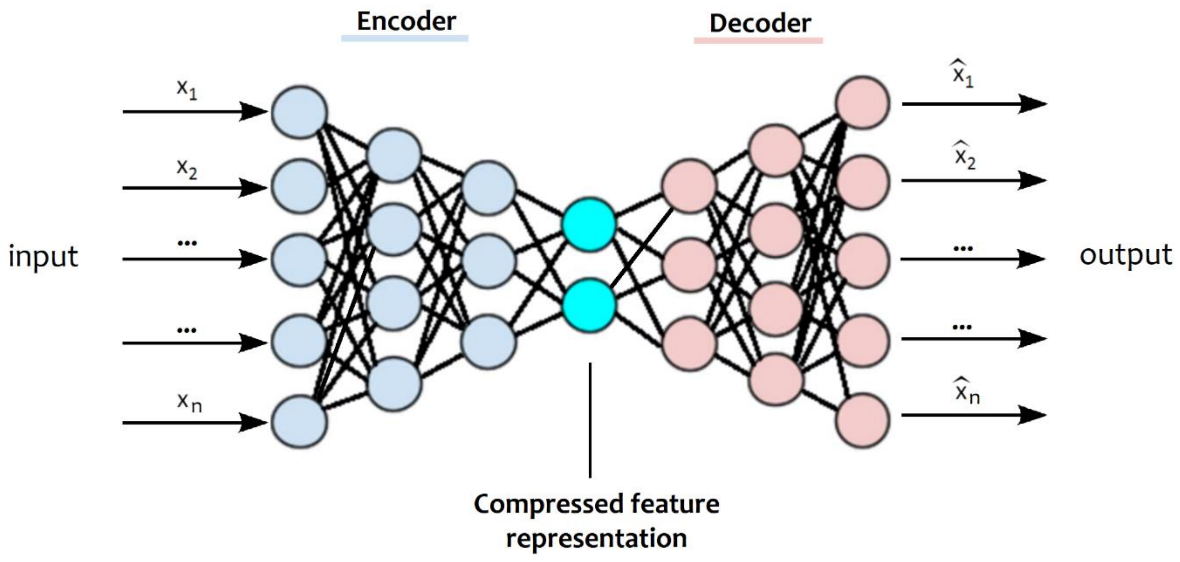


Figure 6. Deep autoencoder network structure.

The compression or *encoder* part of the network forces it to learn the most important features which accurately capture the normal behavior of the time series. Typical applications of autoencoders are dimensionality reduction, compression, denoising (Marchi et al., 2015) and even generating new inputs (Doersch, 2016). Recently, autoencoders have also successfully been applied to the anomaly detection problem (Song et al., 2017; Zhou & Paffenroth, 2017). Back in 2006, Galeano et al. improved their anomaly detection performance when univariate methods were applied to an optimal projection of multivariate data. Later, Li et al., (2017) concluded the same. This indicates that latent space analysis can capture useful information and enhance anomaly detection performance. The downside to doing this is that the anomalies become harder to interpret as they no longer directly relate to the input space. Using an AE however, this latent space is converted back to its non-unitless input space by the *decoder* part of the network. This is good for interpretability as now input values can be compared to output values to see exactly where the anomaly came from. However, the older idea behind using latent space analysis for anomaly detection as used in Galeano et al. (2006) is similarly applied in autoencoders through compression. An autoencoder works well in anomaly detection because if it is trained on predominantly normal behavior, outliers do not conform to this learned behavior and therefore require more information to reconstruct correctly. Because of the bottleneck in the hidden layer, reconstruction of outliers will be poorer as they are essentially incompressible (Zhou & Paffenroth, 2017). This will result in a higher reconstruction error, which can be used as an anomaly score (Cao et al., 2016). The intuition here is that an anomaly is an event that can be poorly predicted from past events (Ergen et al., 2017). Having the objective of learning normal behavior implicitly turns the autoencoder into a one-class classification algorithm.

2.7 Output of Anomaly Detection

In the case of the autoencoder, once a prediction or reconstruction is made, the distance to the observed value can be calculated (i.e. a proxy for abnormality). In a univariate case, this is simply done by subtracting the two numbers from each other. While in the multivariate case, Mahalanobis distance is the preferred choice, as it scales the contribution of each variable according to its variability (Penny, 1996). The errors of different features do not have to be similarly distributed, thus treating them equally according to their variability is preferred as there is no reason to believe any of the variables play a bigger role in finding anomalies within the system. A possible limitation however is that Mahalanobis distance assumes normality (Tiku, Islam, & Qumsiyeh, 2010). The impact of non-normality remains to be seen in this application. Advances have been made to relax this assumption and create more robust distance metrics (Ekstrom, 2011), but no implementations are readily available to test this.

Once the distance is computed (real-valued with theoretically infinite range), there are several options for using it to label anomalies. The simplest method would be applying a numeric threshold, but this is not scalable to new data or a new model, as absolute values may differ. An alternative is to calculate the characteristics of the error distribution and base the final decision on the tail probabilities (Malhotra, Vig, Shroff, & Agarwal, 2015; Qiao et al., 2012). This can be done using either the Euclidian distance or the Mahalanobis distance. Modelling it in this way gives it a probabilistic interpretation as more extreme values have a lower probability of occurrence in the normal distribution (Aggarwal & Sathe, 2017).

The final output of this methodology is a value for every observed point describing the extent to which it exhibits 'non-normal' behavior. This output can come in the form of a binary label or a real-valued score within a range of 0 to 1. A label is implicitly also a score but instead, processed further and the cutoff is adjustable though hyperparameters in the algorithm. The model assigns a label to an internal score after processing and evaluating it against some decision boundary. This means less direct influence after training and the system becomes more like a black box for those not intimate with the inner workings of the algorithm. A raw score is however often not readily interpretable, as its ranges can vary wildly between different models (Zimek, Schubert, & Kriegel, 2012). This is why there are efforts to for instance convert raw scores to probability estimates, these can then work better with Bayesian risk models and are easier to aggregate in an ensemble (Gao & Tan, 2006). The choice of what output to use is entirely dependent on the use case and they can be interchanged easily in an ad-hoc fashion, as labels and probability estimates are both based on the raw score that is produced anyways. Labels are of course the easiest to interpret but real-valued scores have more granularity as they turn it into a top-*n* ranking problem. Since the twitter algorithm directly outputs labels, the same will be done to the reconstruction error of autoencoders by using the tail probabilities of the errors and finding the optimal cutoff for binary labels.

2.8 Performance measures

Measuring the performance of an algorithm is not always straightforward. One needs to be aware of what a performance metric is actually measuring. This could depend on what the output of your algorithm is, what your definition of success is and how different aspects of performance weigh towards the final assessment. The available data does not contain ground truth labels, which has a big impact on the assessment of performance. Since there is no ground truth in this anomaly detection problem, solutions cannot be evaluated in the true sense of the word. Benchmarks like the NAB (as discussed in **2.5 Seasonal Hybrid ESD**), assess models based on their predicted labels, which are therefore not an option. Not having a ground truth complicates multiple aspects of the model building and selecting procedure. Not only the choice of algorithm but also finding an appropriate configuration of hyper-parameters is made more difficult.

A different way of evaluating the performance of an algorithm is through *internal* measures. Discussed so far, have been *external* validity measures, which assess how well an algorithm performs given an external ground truth. Internal measures on the other hand, evaluate the quality of some result according to assumptions made about what constitutes a good result. In that case the metric is only based on the internal structure of the data (Goldstein & Uchida, 2016). Many of these internal measures can also be used to compare different algorithms relative to each other, which makes them 'relative validity measures' (Vendramin, Campello, & Hruschka, 2010).

Assessing the quality of an unsupervised learning problem is always hard as the measures are often tied to the criteria used by the algorithms to learn (Marques, Campello, Zimek, & Sander, 2015), which is not the best indicator for quality. Although these problems have been known for a while, they are rarely mentioned in the context of anomaly detection (Zimek, Campello, & Sander, 2014b). Requiring labels for assessing performance on an unsupervised learning task is also very counterintuitive as the whole premise of unsupervised learning assumes these labels are not present (Marques et al., 2015). There is however very little research on internal validation measures of unsupervised anomaly detection (Zimek et al., 2014b), there are just two suggestions in the literature: E-M and IREOS (Campos et al., 2016; Goix, Sabourin, & Clémençon, 2015), which are both not suited for time series as they do not take into account the sequential nature of the data (i.e. unable to function with contextual anomalies). They are both (cluster) distance based and work well on cross-sectional data.

A final possibility for an external measure would have been to subject real-world users to the results of the different algorithms and gather a human judgment about which is preferred. The algorithms can then learn from the 'ground truth' provided by the users. This is also called *active learning* (Stokes, Platt, & Kravis, 2008). This however, brings along some problems, as preferences do not have to be consistent between different individuals and it becomes extremely difficult to define what exactly one would be measuring. It would require meticulous experimental design and resources beyond the scope of this project.

This leaves an unsolved problem of how to meaningfully assess the performance of the anomaly detection algorithms. With no other options left, the choice was made to create a synthetic dataset with (ideally) identical characteristics to the real dataset. This process is not uncommon and occurs frequently in anomaly detection as there is often a lack of real world data (Ahmed, Mahmood, & Islam, 2016). Anomalies can be injected in a controlled manner which can later serve as labels for a ground truth (Jakkula & Cook, 2008). Artificial datasets work so well that they are also used to add training data to improve performance, even when labels are present (Fan et al., 2004). Using a synthetic dataset opens up options for external validity measures, common ones are the receiver operator curve (ROC-curve), where the tradeoff between FPR (false positive rate) and TPR (true positive rate) is made. From this, the area under curve (AUC) can be calculated and used as a performance metric (Goldstein & Uchida, 2016). Another popular performance metric is the F1-score, which is often used to assess classification results, as also used in the original paper for the S-H ESD algorithm (Hochenbaum et al., 2017). F1-score makes a tradeoff between the algorithms' *recall* (percentage of relevant items that were found) and *precision* (percentage of found items that were relevant) as it calculates the harmonic mean of the two. The F1-score was selected as it and its components are more easily interpretable and is better able to deal with unbalanced class problems (Goldstein & Uchida, 2016). The formula for the F1-score is displayed below.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Where:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

and

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

2.9 Ensembles

Since there are still many available models, one strategy is to put together multiple algorithms in some way to leverage the 'strength of the many' in so-called *ensembles*. Ensembles are methods that combine the results of multiple algorithms in order to increase performance, robustness or generalizability (Aggarwal & Sathe, 2017). Even though ensembles have been studied for decades in the classification setting, anomaly detection ensembles only became more popular recently (Rayana, Zhong, & Akoglu, 2017; Zimek, Campello, & Sander, 2014a).

There are two ways of combining multiple models together, it can be done in either a *sequential* or a *parallel/independent* fashion. This categorization distinguishes ensembles on the basis of whether individual model execution is independent of each other (i.e. can happen in parallel) or whether the input of one model depends on the output of a previous model (i.e. sequentially) (Aggarwal & Sathe, 2017). In a sequential ensemble, previous models have an impact on following models in terms of modifying the input data or choice of the algorithm that follows. The final result of such an ensemble is often the result of the last algorithm that was applied. In independent ensembles, multiple different algorithms are applied to (parts of) the dataset simultaneously and the input of one does not depend on the output of another. The results of the independent algorithms are later combined together in the hopes of increasing performance beyond the performance of any of its individual parts. This combination can happen in a naïve manner, by for instance taking a weighted average, but a more popular approach is to use so-called *meta-learning* algorithms. This is a technique that deals with combining a multitude of outputs from different 'base learners', to combine them through another learning process (Chan, 1999). The outputs used in combinations of multiple models are their labels, to limit the scope of this project and not tread into the realm of the meta-learning algorithms.

Normally ensembles are used to gain a performance increase, but in this research, this will not be the sole objective. If the dataset contains high correlation amongst the variables, it is very plausible that multiple variables will display abnormal behavior at once. An automated univariate system might in that case generate multiple alerts at once, which can be overwhelming and undesirable (Ahmad & Purdy, 2016). Since an anomaly output score can be computed for both multivariate or univariate anomalies, these can be used together so that the nature of the anomaly can be determined by comparing the results. If a multivariate anomaly is found at time *t*, sequentially performed univariate detection can indicate where this multivariate anomaly originated. Tsay et al., (2000) also suggested that this comparison (here through an ensemble) might be useful for detecting univariate outliers. This is very similar to subspace outlier detection as outlined in Zimek et al., (2012). There the outlier is explained by referring to specific attributes where the outlier shows a high deviation. An explanation is at least as important as the identification of outliers/anomalies (Knorr & Ng, 1999). Therefore it is interesting to model all of the variables together and only perform univariate detection when a multivariate anomaly is detected, to find the cause of the anomaly (if it is related to a specific variable). If no univariate anomaly is found afterwards, the multivariate anomaly could either be ignored (possibly resulting in increased performance too) or investigated further as the relationship between more than one variable might be abnormal.

3. Methodology

3.1 Dataset

The dataset made available for answering the research questions belongs to a digital marketing intelligence agency based in the Netherlands. The data itself concerns the websites of the company's clients. The client companies range all the way from small to medium and large businesses. The analysis will take place on the firms largest clients, as those data are the largest in volume (timespan), which is an important factor in training a neural network. Interesting features were selected based on expert knowledge from online marketeers within the company. All of these features are either standard web metric KPI's that are already often monitored or heavily tied to other KPI's and therefore interesting to track. The decision was made to use the following variables; number of users, number of new users, number of sessions, number of bounces[8], total duration of the sessions (in seconds), number of page views, number of transactions and total transaction revenue. All of these variables represent the aggregate values of the whole website and all its pages. The data is available in bins of one hour, as this is the most granular standard time step in Google Analytics.

It is important to define some characteristics of the dataset as these can have an impact on the choice of technique for anomaly detection (Cheboli, 2010). One interesting aspect is the relationship between the different variables. Measuring how similar these variables are is done using the normalized cross-correlation (example in **Figure 9**), which is a standard method of estimating the degree to which two series are correlated (Quenouille, 1949). Before calculating the correlations, the input time series was first decomposed (using STL), after which the seasonal and trend components were removed and its residuals were further used in the analysis. This is because, as discussed before, autocorrelation in a non-stationary time series is inappropriate (Horvatic et al., 2011).

After analyzing the correlations between all the variables, both the total number of transactions and the transaction revenue were removed from the input, as they were too unrelated[9] to the rest to work well in the same model (as visible in **Table 3** in the Appendix). After the removal, the lowest correlation coefficients between the remaining variables is 0.54 (between bounces and session duration) on an hourly scale, and the lowest on a daily scale is 0.82 (again, bounces and session duration). These values indicate really strong positive relationships in the data. Full tables of all cross-correlation coefficients from the sample are available in the appendix, see **Table 3** and **Table 4**.

In **Figure 7** and **Figure 8** the used data is plotted using feature scaling such that the different variables are all on the same scale. The provided data only contains values of variables over time, no labels for anomalous behavior.

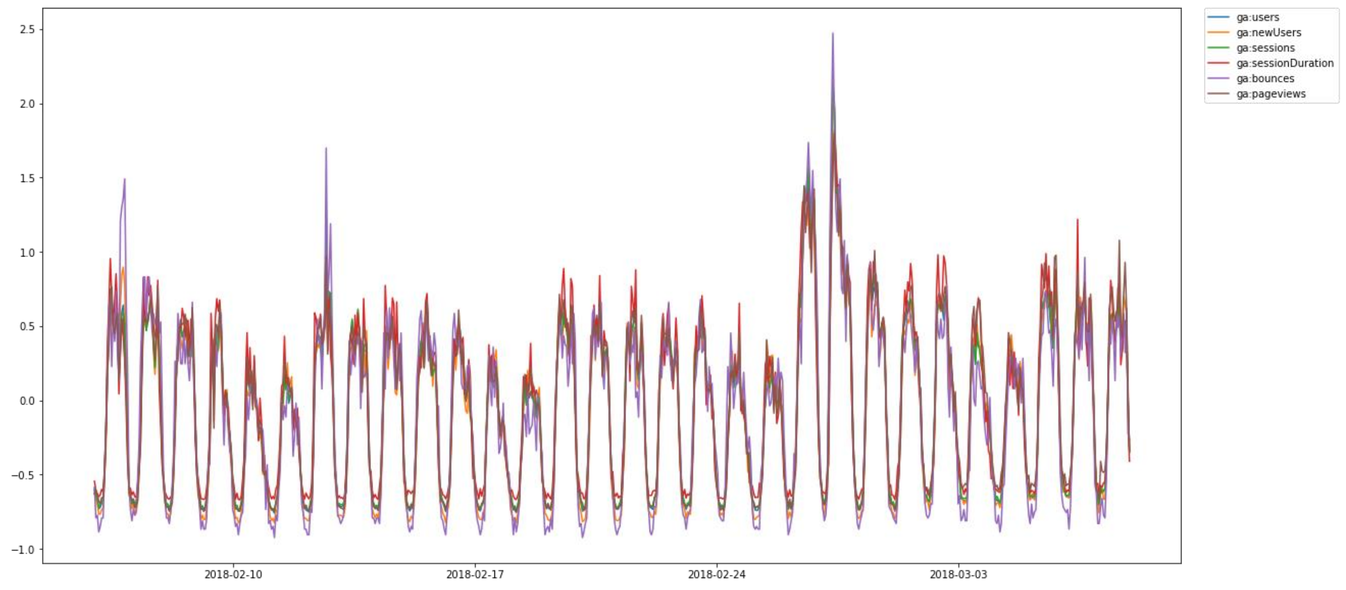


Figure 7. Rescaled time series on an hourly level.

[8] Visitors who after entering the site directly leave again, instead of continuing to view other pages.
[9] Transactions and transactionRevenue have very little to no autocorrelation in their residue, and are therefore already white noise process, meaning they would have no remaining structure left to model for the autoencoder anyways.

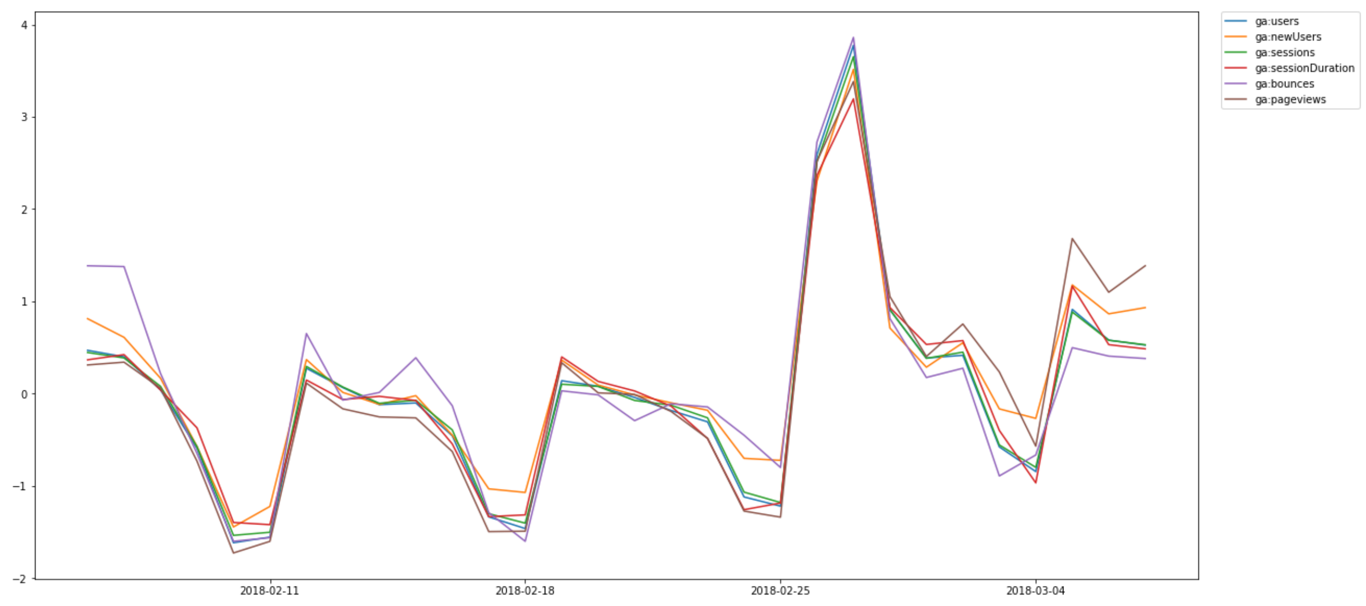


Figure 8. Rescaled time series on a daily level.

Both graphs above visually illustrate the heavily periodic nature of the data, using different time bins. Looking at **Figure 7**, the daily periodicity becomes very clear, the values tend to go (close to) their minimum during the night time and peak during waking hours. **Figure 8** shows the second level of periodicity on the weekly level, Sundays have valleys (e.g. 25th of February), whereas weekdays see the most activity. Also from both of the figures it becomes obvious that the variables are highly related to each other, this falls in line with our intuition, new users create new sessions, which mostly last for similar amounts over time with a certain amount of page views, etc. Finding a change in the relationships between these variables would be interesting as this could indicate a change in user behavior. This is where multivariate anomalies would be more easily detected. From the hourly figure, it also becomes obvious that there are some univariate outliers that do not influence the rest of the variables (e.g. bounces in **Figure 7** multiple times).

Using the normalized cross-correlation analysis, one can also numerically determine the delay or *lag* between different variables. This is done by sliding either of the two time series in discrete steps relative to the other and performing a correlation analysis at every step. The time shift where the cross-correlation is the highest is called the lag. This lag measure can be a proxy for dependency or causality between the two variables. **Figure 9** presents the result of this analysis for one pair of variables[10], as a visual example. From this graph, one can deduce that the two variables correlate together the strongest at a time difference of zero, also some periodicity is visible as there are spikes at multiples of approximately 24. The rest of the lag values corresponding to the highest found cross-correlation coefficient can be found in **Table 5** in the Appendix.

[10] The used estimator of the cross-correlation is shown to decays to zero at the tails of the time series, indicating it is biased. Practically this is not relevant, because out of the large number of possible lag values (here approximately 1500), only a small fraction around zero is of interest.

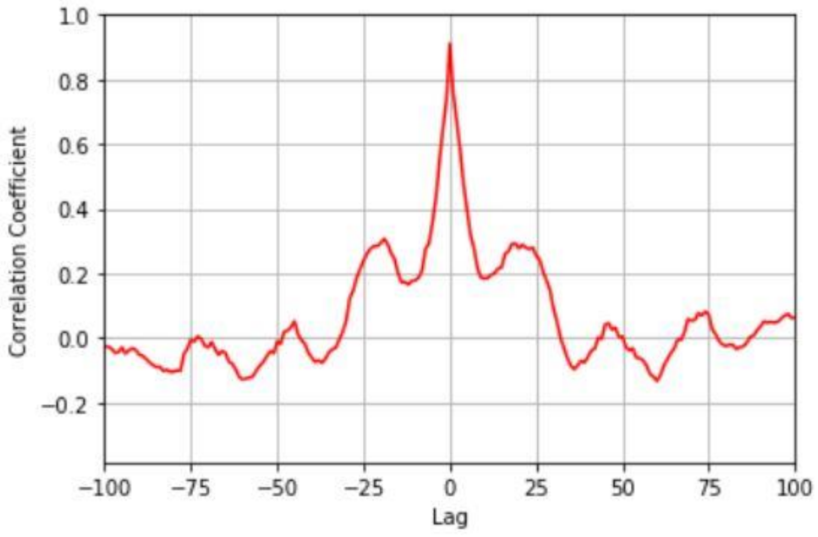


Figure 9. Normalized Cross-Correlation between sessions and page views on an hourly level.

Performing this analysis for every pair of variables shows there are no time lags present between the variables (correlation is highest at lag zero), at least within the time bins considered in this study (one hour). Having no lags present makes these time series *periodic and synchronous*. This setting is the simplest for analysis, every variable has a constant time period and all variables are aligned temporally (i.e. positively correlated synchronous behavior) (Cheboli, 2010).

3.2 Programming toolkit

During this research, multiple programming languages and packages were used to perform the analyses. The code for this research is available upon request. Most of the work was done in Python 3.6. using the following packages; Pandas and Numpy for general data transformations, Scikit-Learn for its MinCovDet and RobustScaler, Statsmodels for auto- and cross-correlation functions, stldecompose for obvious reasons, Keras to build the recurrent autoencoder and Scipy for the Kernel Density Estimation, ECDF distance functions and normality tests. Lastly, R 3.4.3 was used for its package AnomalyDetection (by Twitter).

3.3 Synthetic dataset generation

For the creation of the synthetic dataset, the characteristics will be taken from a 30-day sample of the data. A reverse seasonal decomposition will be performed, as those individual parts will be added up to create a non-stationary dataset containing seasonality. Seasonality was created according to the following formula:

$$y(t) = A \cdot sin(2\pi f t) + L$$

Where:

A = the amplitude of the oscillations,

f = the ordinary frequency, the number of oscillations that occur for each unit of time,

L = level of the sinewave on the y-axis.

A sinusoid was chosen as this is a regular pattern that most closely resembles the patters observable in the real data. The choice of exact pattern does not impact the analysis as STL decomposition is not biased towards certain shapes of seasonality. Two of these seasonal components where added together to create double seasonality, as in the original data (weekly and daily). Lastly, auto- and cross-correlated noise, with values close to the real sample set was added to the data. Autocorrelated random values were generated and cross-correlation is introduced by letting the additional error streams be dependent on the previous value of the first stream in the same way. The components were added according to the formula below, where S1 and S2 are seasonality components and ε is the noise term. Once the data had been generated, anomalies were added from a pseudo-random sample (RS) without replacement from the set of indexes in the synthetic dataset, in a way similar to Gottschlich (2018).

$$X(t) = S_1(t) * S_2(t) + \varepsilon(t)$$

Where:

$$S_{2_i} = \begin{cases} S_{2_i}, & if\ S_{1_i} > 0 \\ 1, & otherwise \end{cases}$$

$$\varepsilon_i = \begin{cases} A_i, & if\ i\ \in RS \\ \varepsilon_i, & otherwise \end{cases}$$

The A_i values are the anomaly values, these are values that are at least 1.68 standard deviations from the mean of the noise, up to 2.02 standard deviations. Only univariate anomalies are added to the data, as complex changes between the relationships of variables harder to model and not the focus of this study. To remove the change for artificial false positives, the noise was cut off at a maximum of 1.68 standard deviations, thus assuming that anomalies do not collide with normal data. What will make anomalies stand out more however, is the fact that they are uncorrelated to the other noise.

No trend component was added as the real dataset did not display significant trends. Because of the day-night cycle, the nights always come down to the same (almost) zero values. Thus the only possible changes are in the amplitude of the day values (hence the second seasonality is only added when the first is positive). **Figure 10** displays the individual components of the synthetic dataset where anomalies have yet to be added. Components *a* and *b* in **Figure 10** are the seasonal components and *c* is the noise component.

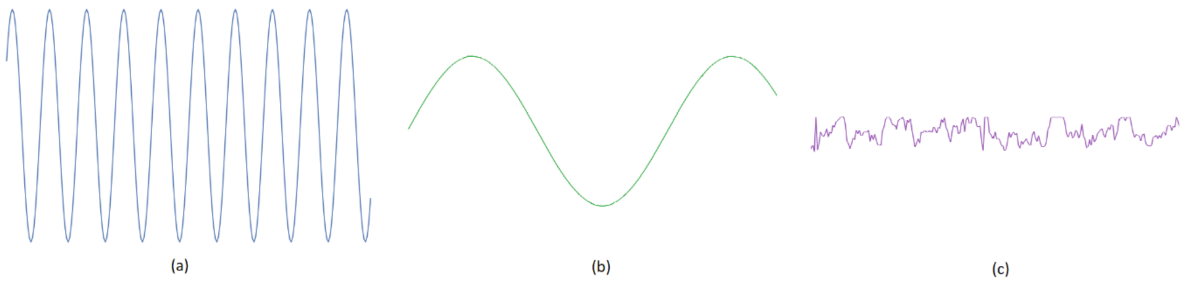


Figure 10. Components of the artificial dataset.
(a) First seasonal component, the daily cycles. (b) Second seasonal component, the weekly cycles. (c) The autocorrelated residual noise component.

This generation process was used to generate a train and a test set. The training set will be used to learn the optimal hyper-parameters of the models, such as alpha significance level, hidden layer size, etc. The test set will be used to assess performance on with the found hyper-parameters. A total equivalent of 2 months of data for both a train and test set was generated. Both the train and test set contain 78 anomalies spread evenly over their 6 data streams, which equals to approximately 1% of their data points, as anomalies typically constitute of 1% or less of the data (Das, Wong, Dietterich, Fern, & Emmott, 2017). A sample of one of these sets is shown in **Figure 11**. From the figure, it becomes clear that some anomalies became global anomalies while others remain contextual (i.e. some fall outside the global range, while others are only anomalous because of their position in the series). The exact shape of the anomalies does not matter as they are not being compared to other known anomalous instances. The most important aspect is that they differ significantly enough from normal behavior.

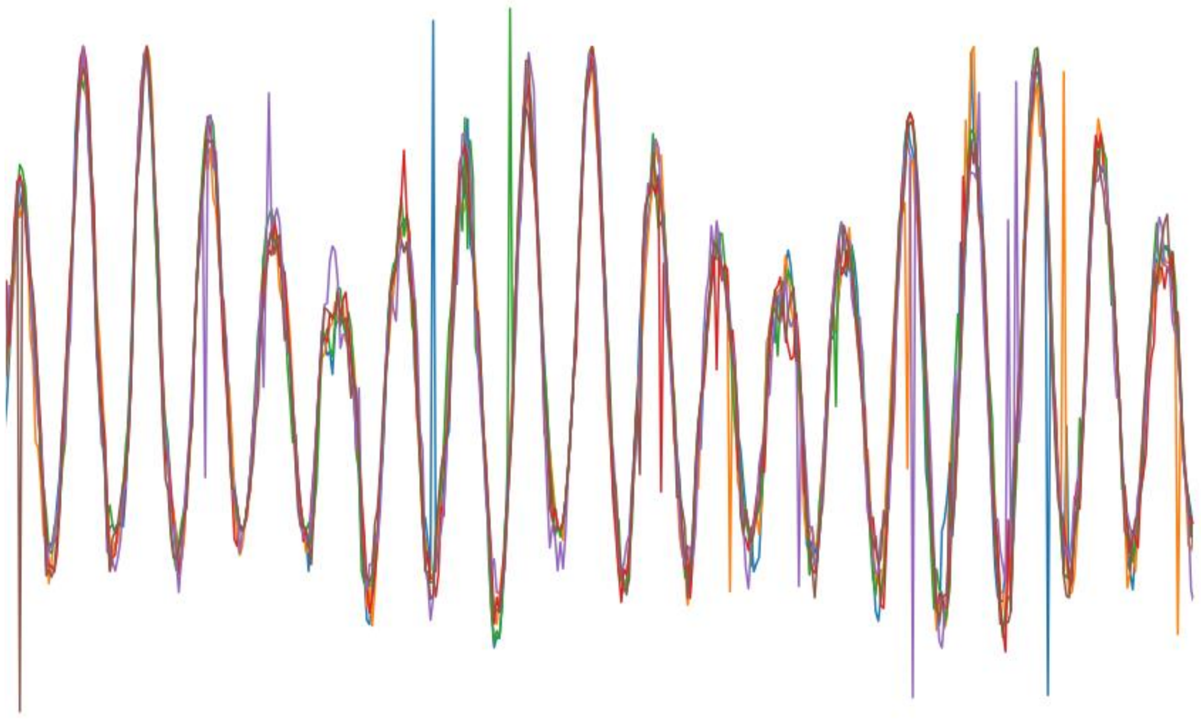


Figure 11. A sample of the synthetic dataset.

3.4 Twitter AD

The twitter S-H ESD algorithm only requires little work to setup correctly as everything comes bundled in an R package. There are a few parameters to be set before running the model, as discussed before in **2.5 Seasonal Hybrid ESD.** Assigning the correct anomaly percentage as a hyper-parameter is however making use of prior external knowledge about the data. Without knowing the distribution or amount of anomalies, this parameter could however also have been found by performing a grid search (a structured search through the parameter space, assessed on a performance metric). The remaining parameters (namely 'longterm' and 'alpha') will be optimized by means of a grid search on the test set with the goal of optimizing the F1-score.

3.5 Autoencoder AD

Shipmon et al., (2017) suggest training separate models is useful if there is no correlation. All the time series need to be related to each other otherwise it does not make sense to model them together. Since there is such strong correlation between all the variables in the dataset, they will all be modelled in the same autoencoder, which will likely improve performance. There is however a limitation in how big these multivariate systems should get. The increase in complexity of training and making inferences on larger multivariate models grown much more than linearly compared to the increase of the input dimensionality (Ahmad & Purdy, 2016), meaning a multivariate model with too many input variables, becomes impractical. Therefore the number of variables in this research is also limited to 6. This number makes it still practically relevant, but not too computationally expensive.

Before the data is fed into the model, it undergoes some preprocessing steps. As discussed before in section **2.4 Time series decomposition**, seasonal decomposition is an important step here and this is performed through STL. The residuals then need to be rescaled as not all the variables operate within the same range. Normally feature scaling can be done through min-max normalizing the features to a range of [0,1] (Dau et al., 2014), but our data is contaminated with anomalies, thus a more robust scaling method is needed. The decision was made to use Sklearn's RobustScaler algorithm for this, which removes the median and scales the data according to the quantile range. This means the rescaling is not bound to a fixed range, but the values of all the variables do behave similarly relative to each other. In comparisons in the documentation[11] it is illustrated how this robust scaling performs versus standard methods.

The input for an autoencoder is based on the idea of a *window*, this is the number of past points to be reconstructed at once, a fixed timeframe. One such window can consist of many time steps in the past and this window is suggested to be "*on the order of the anomalous behavior*" (Aldosari, 2016, p. 9). Helpful here is to think about the context of a contextual anomaly, in this case an anomaly can only be labelled as such if it is at deviating in regard to that day (thus here the window size equals 24 hours). Because inputs are always generated as windows, the so-called 'sliding window' is generated, as illustrated in **Figure 12**. Each time the window moves one timestep ahead and can recreate the new subsequence. A side effect of this is that a point ends up in many windows (amount equal to the size of the window). This means that there are many reconstructions and thus errors for each point in time, in different positions in different windows (Ahmad & Purdy, 2016).

[11] http://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

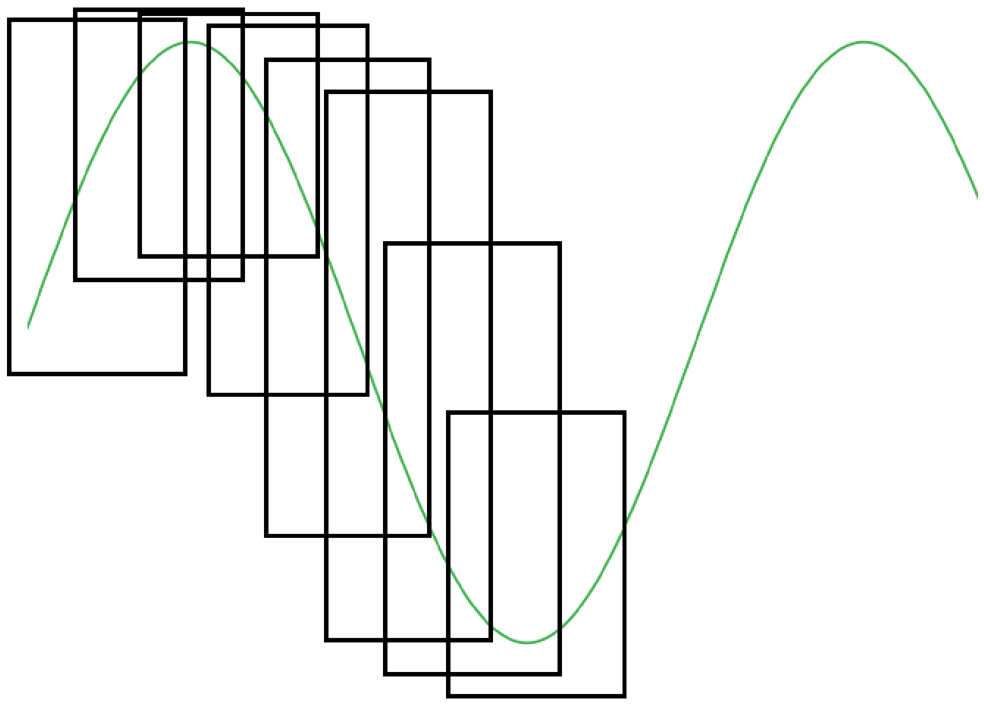


Figure 12. Sliding window approach.

Because there are many error values for the same point in time, a decision has to be made about which to use or how to combine them into one error value that can be transformed into an anomaly score. Standard combination methods that will be tested are; the median of the errors, their mean, the minimum observed value and maximum observed value. Another way of using all the errors is by modelling their Empirical Cumulative Distribution Function (ECDF) and comparing this function to the ECDF of all the observed errors. If the distribution of errors for this point is sufficiently different a higher score can be assigned to it, according to some distance metric. To calculate a distance between two CDF's, possible metrics are the Wasserstein distance (WD), the Kolmogorov-Smirnov (KS) statistic or the Energy distance (ED). Early analysis showed however that Energy distance[12] is the most viable option as KS is highly affected by local deformations, whereas WD barely changed. ED appears to be a good tradeoff between sensitivity and excessive responses. The last alternative is to only use the error of when it was the last point in the sequence of a window. This brings an advantage, as doing it this way means the algorithm could be used 'online', as new values come in. All of these methods for calculating the errors will be compared in section **4. Results**.

Regarding the autoencoder itself, because the input consists of sequential data, the neuron types in the neural network are RNN neurons. This makes the specific autoencoder used in this research a recurrent autoencoder, using GRU units in the hidden layers. The bottleneck layer is a single vector instead of a sequence because this is where the compression comes from. Because of the high correlations found in the dataset, compression in the bottleneck layer is likely to be very high. After the bottleneck, a RepeatVector layer (simply repeats its input n times) is added as the last hidden layer requires sequences equal to the window length as input size instead of a single vector. Cao et al. (2016a) proposed a rule of thumb for the hidden layer size of autoencoders, namely $1 + \sqrt{n}$ where n is the number of variables in the input data. This was however not for a recurrent autoencoder, which deals with t extra timesteps of past datapoints,

[12] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.energy_distance.html

making the total input points of size $t \cdot n$. Taking the square root of $24 \cdot 6$ (window times variables) and adding 1 gives an estimate of 13 units. Through trial and error (including many multiples/fractions of the initial estimate) the final hidden layer size that was settled upon was 12. Additional layers in the encoder and decoder did not improve performance, so both were kept to a single layer.

After constructing the network, training can be done on the train set using the RMSprop optimizer with a Mean Squared Error (MSE) loss function, which is recommended for recurrent autoencoders. RMSprop is designed to work with mini-batch learning, which works favorably as the input data from the sliding windows contains much redundancy computing gradients for mini-batches will be more efficient (Aldosari, 2016). A batch size larger than one is always recommended as it smoothens and expedites the learning process, by making it less variable. A whole daily period is passed through (batch size equals 24) before gradients are calculated. After around 20 epochs of training (whole passes through the dataset), the error function seemed to plateau, thus it was kept at 20 whenever training the AE. The learning rate was set to 0.012, which is larger than usual but is done to avoid getting stuck in local optima or overfitting.

After some initial testing, the reconstruction error on actual dataset turned out to be non-normally distributed. Meaning that a simple threshold based on the number of standard deviations might not work properly. Visual inspection made it seem fairly normally distributed, but when looking at test statistics, $p << 0.01$ (under the null hypothesis that the data is normally distributed). This means a non-parametric distribution estimation is needed. This is where the Kernel Density Estimation (KDE) function can be used (Cao et al., 2016b). KDE is an empirical distribution estimator and it works for anomaly detection by defining a threshold on density in the error distribution. The one drawback is however that performing KDE takes a relatively long time (Cao et al., 2016).

After constructing and training the network, reconstruction errors can be calculated by running the test set through the network. Univariate anomalies are then calculated by fitting all reconstruction errors of each variable in the train set to a KDE distribution, after which tail probabilities can be calculated for points in the test set. Multivariate anomales are detected by first combining the reconstruction errors of the variables in the test set together and calculating their respective Mahalanobis distances. These distances from the train set are then fitted to a distribution using KDE, where once again the tail probabilities can be calculated for all the points in the test set. These tail probabilites are used to find an optimal cutoff where the F1-score is maximized.

3.6 Measuring performance

As mentioned before, optimal parameters for the different models were found through grid search. Although grid search theoretically is computationally more expensive than other more advanced parameter optimization techniques, it does give the guaranteed optimal setting. Approximations or heuristic approaches cannot guarantee this and often provide little practical speed increase (Hsu, Chang, & Lin, 2016). Once the optimal parameters were found on the training set, anomaly scores and labels were generated using those parameters for the test set.

When predictions are made on the test set, detected univariate anomalies are only counted as true positives when the anomaly at that point in time also corresponds to that particular sequence. This is in contrast to multivariate anomalies, where it is counted as a true positive whenever a detected anomaly corresponds to an anomaly in any of the univariate series at that particular point in time. This implicitly assumes that there is a perfect univariate detector after the multivariate detector, but the performance of this combination will be tested later. This also can be seen as a univariate anomaly without knowing in which variable yet (not essential if one is simply assessing performance).

For univariate detection methods, once the confusion matrix (containing true/false positives/negatives) has been made for all the data streams, they are added together before calculating the overall precision, recall and F1-score. This is done because the performance of the system should be considered as a whole and individual parts should not be scored on their own.

Performance of possible sequential and parallel ensembles will be assessed by combining the output labels of their respective components. In parallel ensembles, union of both sets of detected anomalies is taken to assess performance on. The idea is that both algorithms could run independently from each other, thus output from both will be obtained at the same time and combined together. On the other hand in sequential ensembles, the point is that univariate detection methods would only need to be used once a multivariate anomaly is detected. Therefore the intersection of both sets is taken, as they need to appear in both in order to be registered. In order to test this though, all algorithms had to create outputs for every point, which is why the intersections and unions of the sets could simply be taken to simulate an ensemble, as labels for all points in time were available anyways.

To test how similar the sets of found anomalies from the different techniques are, the overlap coefficient is used (M.K & K, 2016). The overlap is interesting as a high similarity could indicate that one method is a suitable replacement for another. The measure is calculated by dividing the size of the intersection by the smallest size out of the two sets. Inspecting this metric can reveal something about how different two algorithms are in what anomalies they detect. The formula for the overlap coefficient is as follows:

$$Overlap(X, Y) = \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

4. Results

To get an intuitive feel for how well the autoencoder fits the data, the cumulative errors are inspected. This scalar value can give a crude indication of model performance. As discussed in section **2.4 Time series decomposition**, the residuals of decomposition can be considered as estimates of model fit. So the autoencoder is trying to improve the model fit even further, by explaining a larger portion of the errors. Therefore it is expected that there is a reduction in cumulative errors (reconstruction errors) compared to the residuals of the decomposition (the input data). After separately training autoencoders on both the synthetic and the real dataset, the error reduction can be compared to estimate if similar performance in anomaly detection can be expected. This is the case because improvements in model fit also mean improvements in anomaly detection, as anything that can be correctly modelled from learned past behavior must by definition not be anomalous.

The synthetic and real datasets are comparable because they undergo the same preprocessing steps and are on the same scale. The results are that the total error in the fake dataset (2 months) is reduced by around 14% (118.5k to 103.5K) while the real dataset (1-month sample) had its total errors reduced by approximately 22% (from 67.1k to 54.8k). Similarly, autocorrelation in the residuals was also significantly reduced in both real and fake datasets, meaning the errors were correctly transformed to a 'white noise' error process (i.e. most of the remaining structure in the errors left from the STL decomposition was explained by the AE). This is important as the autocorrelation in the reconstruction error seems to be roughly tied to performance, those with the lowest (to negligible) autocorrelation perform better than the others. These results indicate that performance of the autoencoder on the real dataset could be just as good or better. Optimal hyper-parameter settings for the algorithms can be found in

Table 7 in the Appendix. Also as predicted by literature, performance in tests without preprocessing using STL was indeed worse.

4.1 Anomaly detection results

			recall	precision	F1-score
	S-H ESD		**0,474**	**1,000**	**0,643**
AE (12,12)	UV rec	mean	**0,654**	0,761	0,703
		median	0,603	**0,870**	**0,712**
		min	0,603	0,810	0,691
		max	0,590	0,821	0,687
		last	0,551	0,741	0,632
	UV CDF	energy	**0,590**	**0,902**	**0,713**
	MV maha	mean	0,600	0,818	0,692
		median	**0,640**	**0,828**	**0,722**
		min	0,493	0,804	0,612
		max	0,587	0,772	0,667
		last	0,520	0,696	0,595

Table 1. Performance metrics of different anomaly detection algorithms.
UV rec means univariate recreation, UV CDF stands for the univariate Cumulative Distribution Function and MV maha stands for the multivariate Mahalanobis distance.

The results displayed in

Table 1 show the performance in terms of recall, precision and F1-score for the different autoencoder (AE) setups and the Seasonal-Hybrid ESD (S-H ESD) algorithm. It becomes clear that the Twitter algorithm heavily favors precision over recall, but this does hurt its overall F1-score. It also becomes clear that the multivariate (MV) method performs slightly better than its univariate (UV) counterparts. The performance of 'online' or real-time (depicted in

Table 1 as 'last') settings is clearly worse. If real-time detection would have been a hard constraint in this research, the best alternative would be the Twitter package, as it can run online in the exact same configuration.

Based on these results, the median was selected as the favored way to recombine multiple reconstruction errors of the same point. Further analysis will be performed using the median. Results of anomaly detection were also compared between training the model on the trainset and re-training the model on the test set. Model evaluation on the test set while trained on trainset was 0.70, but 0,664 with retraining (also slightly differs per training run). The complete results on anomaly detection in

Table 6 in the Appendix show that there are no significant differences between retraining and not retraining. This means there is no need to waste time retraining and that the learned model is generalizable from the train to the test set.

4.2 Ensemble results

	Sequential/intersection			Parallel/union		
	recall	precision	F1-score	recall	precision	F1-score
MV-ED	0,560	0,977	0,712	0,667	**0,794**	**0,725**
MV-UV	**0,587**	0,957	**0,727**	0,653	0,790	0,715
MW-TW	0,533	**1,000**	0,696	**0,707**	0,646	0,675

Table 2. Anomaly detection ensemble results.
MV = multivariate, ED = energy distance, UV = univariate recreation (median), TW = Twitter package

As shown in

Table 2, the results of a sequential ensemble heavily increase the precision performance of the overall system, while a parallel system combining all outputs together increases recall. This makes sense as taking the intersection of the sets applies stricter constraints on found anomalies, decreasing false positives, but thereby increasing false negatives. In a parallel ensemble, this is precisely the other way around and these tradeoffs do balance out the F1-score. A preference for either higher recall or precision could therefore influence the decision, but since there is currently no reason for a preference, the F1-score is used as it weights them both equally. The results from this table indicate that a sequential ensemble containing a multivariate detector and a several univariate detectors using tail probabilities of the median of the errors performs the best in terms of F1-score. When comparing these results to those of the single detectors in **Table 1**, there is even a slight increase in performance, from a max of 0.722 to 0.727 in F1-score. Other less practically relevant combinations were also tested in ensembles, for which the results are available in **Table 8** in the appendix.

The overlap coefficient is used to compare the found multivariate with univariate anomalies found through different methods. The overlap coefficients are as follows: 0.92 for the univariate AE using median recreation, 0.896 for the energy distance model and lastly 0.690 between the S-H ESD model and the MV AE. A coefficient of 0.92 indicates that there is extensive overlap between the sets of anomalies which the two algorithms detected. This is a strong result in the favor of using multivariate anomaly detection, as is means not many anomalies that would have otherwise been detected by multiple univariate algorithms are missed and a multivariate model could be a suitable replacement base detector.

5. Discussion

The findings of this study suggest that an anomaly detection ensemble can match or surpass the performance of its individual parts, because a multivariate detector can be a usable substitute base detector for multiple univariate detectors. Results also show that the multivariate anomaly detector can still clearly detect univariate anomalies in the face of highly cross-correlated noise. This could have been an issue as in the multivariate case, errors would add up, but this turned out not to be a problem. Furthermore, the autoencoder clearly outperformed Twitters standard anomaly detection package, even though it has been reported to perform extremely well on artificial data (as seen in the NAB benchmark). The performance of 'online' detection was however not up to par with the rest. In practice this means that one would have to wait the length of a window size before making more accurate predictions (in this case 24 hours). This restricts some more real-time use cases. Possibly a good middle ground can be found, but this is left for future research. Otherwise the Twitter package is a viable alternative as this can run real-time.

5.1 Research questions

To answer the question whether or not there is a difference in performance between univariate and multivariate detection for detecting univariate anomalies,

Table 1 provides the necessary information. The best performing multivariate model (using the median recreation method) had an F1-score of 0.722 while the best performing univariate model (also using the median) had an F1-score of 0.712. Although this may not be a very large increase, it is at least evidence that a multivariate detection system does not perform any worse than multiple univariate ones. This is good news for anyone looking to use one overarching model instead of multiple independent models. Using one model also brings other advantages such as reducing the complexity of the dataflow, reduction in computation time and easier control.

Using multiple models together in an ensemble can also have some advantages.

Table 2 shows that in performance, using an ensemble does not improve the results by a big amount (if anything) compared to the results from **Table 1**. Using a parallel ensemble also removes the possibility of only having one model running normally, as this assumes outputs from both models are available for every point before making a final decision. Running a sequential model however can still give better interpretation to multivariate anomalies. First detecting multivariate anomalies to then perform a kind of root cause analysis on with univariate detection methods can greatly aid in how interpretable the final result is and where the anomaly originated. Overlap coefficients show that some univariate methods are very similar to the multivariate detection model, indicating that the multivariate model could perhaps be used as a replacement base detector without impacting what anomalies are found.

5.2 Limitations

If the ensemble were designed also detect actual multivariate anomalies (i.e. changing relationships between variables or multiple weaker univariate anomalies), the binary constrains of the univariate detector will need to be loosened (i.e. taking the intersection of the two sets). For instance to look at the most deviating variable might be better than assessing the values against a hard constraint and rejecting the anomaly. Doing this would still allow multivariate anomalies to be made more interpretable without discarding perhaps important coordinated anomalies. This was not done for this study as only univariate anomalies were injected into the synthetic dataset and the only binary labels were used as the input for the ensemble. Secondly, the use of synthetic data was, although unavoidable, a definite limitation. The characteristics of the fake dataset were however made to resemble the real dataset closely, which backs up the claim that performance on a real dataset would be similar. One can however never be sure without actually testing.

5.3 Future research

For future research it would be interesting to look at the performance of these techniques in finding other anomalies like multivariate anomalies or perhaps an adjusted adaptation to find collective anomalies. Additionally, different domains could be explored with similar characteristics in the data. The impact of a change in the characteristics of the data could yield some information about how generalizable the approach is and what factors drive performance. Here different aggregation levels of the data might also be interesting as this provides different use cases for different users. A lower level granularity with real-time (online) functionality might not be so interesting for a senior level employee, but very interesting for someone that needs speedy updates on the quality of the data.

There are quite some technical factors that may impact performance, which could warrant further investigation. One of those is empirically testing which seasonal decomposition method is most robust and best suited for an anomaly detection system. Seasonal decomposition lies at the foundation of all the algorithms tested in this research, thus any improvements here could be highly beneficial. Another factor that influences performance is the multivariate distance metric that was chosen. Mahalanobis distance was the most obvious choice and there seemed to be few alternatives worth considering, further research might be interesting in this area. Mahalanobis is in theory also not particularly well suited for non-normally distributed data, but this did not seem to impact performance. A third factor in the current setup is that KDE is a bottleneck for the speed of detection, KDE runs comparatively slow (order of seconds) and might not provide the most robust estimation of tail probabilities. Perhaps quantiles could be an interesting direction to look into as this also ties in well with formulating anomaly detection as a top-n ranking problem. Lastly, a meta-learning ensemble using the raw anomaly scores is an obvious extension of this research. A binary label on its own does not contain enough information to boost performance much.

There are also future possibilities to use the learnt representation of the autoencoder to improve forecasting results, as shown in Hsu (2017). This could allow for the autoencoder algorithm to be useful in other use cases outside of anomaly detection, like for instance forecasting. A last note is that if this autoencoder were to be used in practice (production), model performance would also need to be measured. Suggestions for this are to look at a running average of reconstruction errors, to catch long-term drifts, indicating that the model should be retrained. But investigating what works best for maintaining performance once to model is live, is also left for possible future research.

6. Conclusion

In this study, the effectiveness of a multi- and univariate anomaly detection ensemble was examined. A design was made to compare a recurrent autoencoder solution with a standard statistical anomaly detection package, open sourced by Twitter. A multivariate anomaly detection algorithm was used as a base detector, as its performance, according to literature, would be superior. In order to make the anomalies found in by the multivariate detector interpretable, univariate detectors are used to detect which variable was anomalous in the multivariate anomaly. A synthetic dataset was made using sinusoidal waves to create double seasonality and cross- and auto-correlated noise were added to mimic the conditions in the real dataset. Univariate anomalies were added to the dataset as those could be used as a benchmark for finding interpretable anomalies using a multivariate detector in combination with univariate detectors.

Results showed that a multivariate detector can be at least as (if not more) successful in detecting univariate anomalies as multiple univariate detectors can. Additionally, a sequential ensemble containing a multivariate and several univariate detectors can be used to improve the interpretability of multivariate anomaly detection, without sacrificing on performance. Using a multivariate detector in this way for detection in an ensemble has not been done before and has been proven to be fruitful. For illustration purposes, results from anomaly detection on the real dataset have been included in **Figure 13** (others can be found in **Figure 14** in the Appendix).

A limitation of the current study were the use of a synthetic dataset and the fact that multivariate anomalies in the real data were not considered. Future research could likely not easily address the former, but a deeper look into the latter would be very interesting.

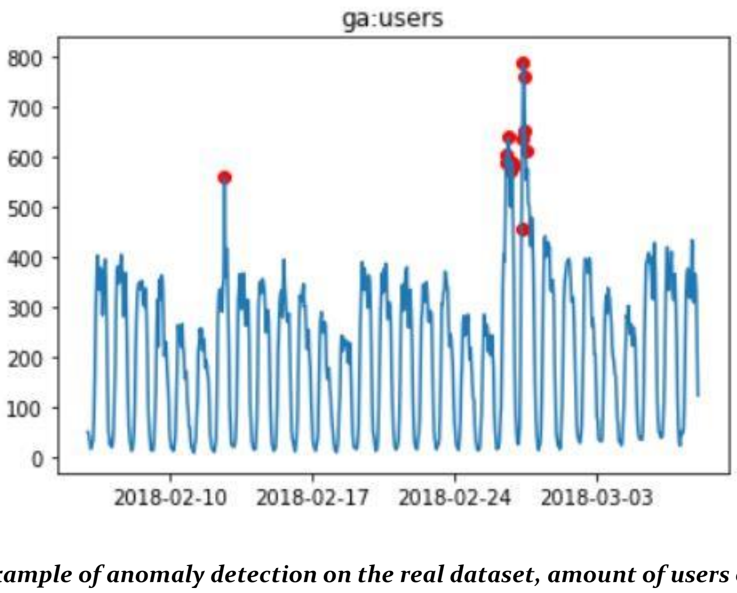


Figure 13. Example of anomaly detection on the real dataset, amount of users over a month.

7. References

Aggarwal, C. C., & Sathe, S. (2017). *Outlier Ensembles: An Introduction*. Springer International Publishing AG. https://doi.org/10.1007/978-3-319-47578-3_6

Agrawal, R. K., & Adhikari, R. (2013). An Introductory Study on Time Series Modeling and Forecasting. *ArXiv Preprint ArXiv:1302.6613*, *1302.6613*, 1–68. https://doi.org/10.13140/2.1.2771.8084

Agrawal, S., & Agrawal, J. (2015). Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, *60*(1), 708–713. https://doi.org/10.1016/j.procs.2015.08.220

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, 134–147. https://doi.org/10.1016/J.NEUCOM.2017.04.070

Ahmad, S., & Purdy, S. (2016). Real-Time Anomaly Detection for Streaming Analytics. https://doi.org/10.1109/ICDMW.2015.104

Ahmed, M., Mahmood, A. N., & Islam, M. R. (2016). A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, *55*, 278–288. https://doi.org/10.1016/j.future.2015.01.001

Aldosari, M. S. (2016). Unsupervised Anomaly Detection in Sequences Using Long Short Term Memory Recurrent Neural Networks, 98. https://doi.org/10.1017/CBO9781107415324.004

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long- short term memory. *Plos One*, 1–24. https://doi.org/10.6084/m9.figshare.5028110

Ben-gal, I. (2005). Outlier Detection. *Data Mining and Knowledge Discovery Handbook*, 131–146. https://doi.org/10.1007/0-387-25465-x_7

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2016). A multi-step outlier-based anomaly detection approach to network-wide traffic. *Information Sciences*, *348*, 243–271. https://doi.org/10.1016/j.ins.2016.02.023

Bodrog, L., Kajo, M., Kocsis, S., & Schultz, B. (2016). A robust algorithm for anomaly detection in mobile networks. *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 1–6. https://doi.org/10.1109/PIMRC.2016.7794573

Box, G. E. P., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, *65*(332), 1509–1526. https://doi.org/10.1080/01621459.1970.10481180

Bronstein, A., Das, J., Duro, M., Friedrich, R., Kleyner, G., Mueller, M., … Cohen, I. (2001). Self-aware services: Using Bayesian networks for detecting anomalies in Internet-based services. *2001 7th IEEE/IFIP International Symposium on Integrated Network Management Proceedings: Integrated Management Strategies for the New Millennium*, 623–638. https://doi.org/10.1109/INM.2001.918070

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., MicenkovÃ¡, B., Schubert, E., … Houle, M. E. (2016). *On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study*. *Data Mining and Knowledge Discovery* (Vol. 30). Springer US. https://doi.org/10.1007/s10618-015-0444-8

Cao, V. L., Nicolau, M., & McDermott, J. (2016). A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection. *PPSN XIV*, *9921*. https://doi.org/10.1007/978-3-319-45823-6_67

Chan, P. K. (1999). Meta-Learning in Distributed Data Mining Systems : Issues and Approaches Meta-learning in distributed data mining systems : Issues and approaches, (August).

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A Survey. *ACM Computing Surveys*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Cheboli, D. (2010). Anomaly detection of time series. *(Master Thesis)*. Retrieved from https://conservancy.umn.edu/handle/11299/92985

Cheng, H., Tan, P., Potter, C., Field, M., Klooster, S., & Bay, M. (2009). Detection and Characterization of Anomalies in Multivariate Time Series. *Proceedings of the SIAM International Conference on Data Mining*, 413–424. https://doi.org/10.1109/ICDMW.2008.48

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. https://doi.org/10.3115/v1/W14-4012

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 1–9. Retrieved from http://arxiv.org/abs/1412.3555

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*. https://doi.org/citeulike-article-id:1435502

Cleveland, W. P., & Tiao, G. C. (1976). Decomposition of Seasonal Time Series: a model for the X-11 Program. *Journal of the American Statistical Association*, *71*(355), 581–587. Retrieved from http://www.jstor.org/stable/2285586

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, *74*, 829–836.

Das, S., Wong, W. K., Dietterich, T., Fern, A., & Emmott, A. (2017). Incorporating expert feedback into active anomaly discovery. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 853–858. https://doi.org/10.1109/ICDM.2016.164

Dau, H. A., Ciesielski, V., & Song, A. (2014). Anomaly Detection Using Replicator Neural

Networks Trained on Examples of One Class. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8886*(March 2017), 287–298. https://doi.org/10.1007/978-3-319-13563-2

Doersch, C. (2016). Tutorial on Variational Autoencoders, 1–23. https://doi.org/10.3389/fphys.2016.00108

Ekstrom, J. (2011). Mahalanobis' distance beyond normal distributions. *Working Paper*, 1–16.

Ergen, T., Mirza, A. H., & Kozat, S. S. (2017). Unsupervised and Semi-supervised Anomaly Detection with LSTM Neural Networks, 1–12. Retrieved from http://arxiv.org/abs/1710.09207

Fan, W., Miller, M., Stolfo, S., Lee, W., & Chan, P. (2004). Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, *6*(5), 507–527. https://doi.org/10.1007/s10115-003-0132-7

Fernandes, G., Rodrigues, J. J. P. C., & Proença, M. L. (2015). Autonomous profile-based anomaly detection system using principal component analysis and flow analysis. *Applied Soft Computing Journal*, *34*, 513–525. https://doi.org/10.1016/j.asoc.2015.05.019

Galeano, P., Peña, D., & Tsay, R. S. (2006). Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, *101*(474), 654–669. https://doi.org/10.1198/016214505000001131

Gamboa, J. C. B. (2017). Deep Learning for Time-Series Analysis. Retrieved from http://arxiv.org/abs/1701.01887

Gao, J., & Tan, P.-N. (2006). Converting Output Scores from Outlier Detection Algorithms into Probability Estimates. Retrieved from http://www.cse.msu.edu/~ptan/papers/ICDM2.pdf

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, *28*(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Goix, N., Sabourin, A., & Clémençon, S. (2015). On Anomaly Ranking and Excess-Mass Curves. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, (5141), 287–295.

Goldstein, M., & Uchida, S. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE*, (April), 1–31. https://doi.org/10.7910/DVN/OPQMVF

Gottschlich, J. (2018). Paranom: A Parallel Anomaly Dataset Generator, 1–3. Retrieved from http://arxiv.org/abs/1801.03164

Gupta, M., Gao, J., Aggarwal, C., Han, J., & Gupta, Manish; Gao, Jing; Aggarwal, Charu C; Han, J. (2013). Outlier Detection for Temporal Data : A Survey. *Tkde*, *25*(1), 1–20.

https://doi.org/10.1109/TKDE.2013.184

Hall, M., Witten, I., & Eibe, F. (2011). *Data mining: Practical machine learning tools and techniques.*

Han, J. H. J., Dong, G. D. G., & Yin, Y. Y. Y. (1999). Efficient mining of partial periodic patterns in time series database. *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, 1–10. https://doi.org/10.1109/ICDE.1999.754913

He, Z., Xu, X., Huang, J. Z., & Deng, S. (2004). A frequent pattern discovery method for outlier detection. *Advances in Web-Age Information Management*, (January), 726–732. https://doi.org/10.1007/978-3-540-27772-9

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). Automatic Anomaly Detection in the Cloud Via Statistical Learning. Retrieved from http://arxiv.org/abs/1704.07706

Horvatic, D., Stanley, H. E., & Podobnik, B. (2011). Detrended cross-correlation analysis for non-stationary time series with periodic trends. *EPL (Europhysics Letters)*, *94*(1), 18007.

How, D. N. T., Loo, C. K., & Sahari, K. S. M. (2016). Behavior recognition for humanoid robots using long short-term memory. *International Journal of Advanced Robotic Systems*, *13*(6), 1–14. https://doi.org/10.1177/1729881416663369

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2016). A Practical Guide to Support Vector Classification. *BJU International*, *101*(1), 1396–1400. https://doi.org/10.1177/02632760022050997

Hsu, D. (2017). Time Series Forecasting Based on Augmented Long Short-Term Memory, 1–14. Retrieved from http://arxiv.org/abs/1707.00666

Jakkula, V., & Cook, D. J. (2008). Anomaly detection using temporal data mining in a smart home environment. *Methods of Information in Medicine*, *47*(1), 70–75. https://doi.org/10.3414/ME9103

Japkowicz, N., Myers, C., & Gluck, M. (1995). A novelty detection approach to classification. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 518–523. Retrieved from http://www.ijcai.org/Past Proceedings/IJCAI-95-VOL 1/pdf/068.pdf

Jensen, K., Van Do, T., Nguyen, H. T., & Arnes, A. (2016). Better protection of SS7 networks with machine learning. *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*. https://doi.org/10.1109/ICITCS.2016.7740315

Kejariwal, A. (2014). Breakout detection in the wild. [Blog post]. Retrieved from https://blog.twitter.com/2014/breakout-detection-in-the-wild

Keogh, E., Lin, J., & Fu, A. (2005). HOT SAX: Finding the most Unusual Time Series Subsequences: Algorithms and Application. *Icdm*.

https://doi.org/10.1109/ICDM.2005.79

Khan, S. S., & Madden, M. G. (2014). One-class classification: Taxonomy of study and review of techniques. *Knowledge Engineering Review*, *29*(3), 345–374. https://doi.org/10.1017/S026988891300043X

Knorr, E. M., & Ng, R. T. (1998). Algorithms for Mining Distance-Based Outliers in Large Datasets. *24th International Conference on Very Large Data Bases*, 392–403.

Knorr, E. M., & Ng, R. T. (1999). Finding Intensional Knowledge of Distance-Based Outlliers. *Proceedings of the 25th VLDB Conference*.

Kwon, D., Kim, H., Kim, I., Kim, K. J., Kim, J., & Suh, S. C. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*. https://doi.org/10.1007/s10586-017-1117-8

Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, *42*(1), 11–24. https://doi.org/10.1016/j.patrec.2014.01.008

Lee, D. (2017). Anomaly Detection in Multivariate Non-stationary Time Series for Automatic DBMS Diagnosis. https://doi.org/10.1109/ICMLA.2017.0-126

Li, J., Pedrycz, W., & Jamal, I. (2017). Multivariate time series anomaly detection: A framework of Hidden Markov Models. *Applied Soft Computing Journal*, *60*, 229–240. https://doi.org/10.1016/j.asoc.2017.06.035

Lin, J., Zhang, Q., Bannazadeh, H., & Leon-Garcia, A. (2016). Automated anomaly detection and root cause analysis in virtualized cloud infrastructures. *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, (Noms), 550–556. https://doi.org/10.1109/NOMS.2016.7502857

Lin, T. Y. Anomaly Detection, 41 Proceedings New Security Paradigms Workshop § (1994). https://doi.org/10.1109/NSPW.1994.656226

Liu, H., Shah, S., & Jiang, W. (2004). On-line outlier detection and data cleaning. *Computers and Chemical Engineering*, *28*(9), 1635–1647. https://doi.org/10.1016/j.compchemeng.2004.01.009

M.K, V., & K, K. (2016). A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal*, *3*(1), 19–28. https://doi.org/10.5121/mlaij.2016.3103

Ma, J., & Perkins, S. (2014). Time-series novelty detection using one-class support vector machines. *Proceedings of the International Joint Conference on Neural Networks, 2003.*, *3*(August 2003), 1741–1745. https://doi.org/10.1109/IJCNN.2003.1223670

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series, (April), 22–24. https://doi.org/10.14722/ndss.2015.23268

Marchi, E., Vesperini, F., Eyben, F., & Squartini, S. (2015). A NOVEL APPROACH FOR AUTOMATIC ACOUSTIC NOVELTY DETECTION USING A DENOISING AUTOENCODER WITH BIDIRECTIONAL LSTM NEURAL NETWORKS, *289021*(289021), 1996–2000. https://doi.org/10.1109/ICASSP.2014.6854294

Marques, H. O., Campello, R. J. G. B., Zimek, A., & Sander, J. (2015). On the internal evaluation of unsupervised outlier detection. *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, *27*, 7:1-12. https://doi.org/10.1145/2791347.2791352.

Moya, M. M., & Hush, D. R. (1996). Network Constraints and Multiobjective Optimization for One Class Classification. *Neural Networks*, *9*(3), 463–474. https://doi.org/10.1016/0893-6080(95)00120-4

Ng, B. (2006). Survey of anomaly detection methods, 40. Retrieved from https://e-reports-ext.llnl.gov/pdf/339724.pdf

Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, *51*(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

Penny, K. (1996). Appropriate Critical Values When Testing for a Single Multivariate Outlier by Using the Mahalanobis Distance Author (s): Kay I . Penny Source : Journal of the Royal Statistical Society . Series C (Applied Statistics), Vol . 45 , No . 1 Published by : *Journal of the Royal Statistical Society*, *45*(1), 73–81.

Qiao, Z., He, J., Cao, J., Huang, G., & Zhang, P. (2012). Multiple Time Series Anomaly Detection Based on Compression and Correlation Analysis : A Medical Surveillance Case Study *, (71072172), 294–305.

Quenouille, M. H. (1949). Approximate Tests of Correlation in Time-Series. *Journal of the Royal Statistical Society*, *11*(2), 68–84.

Rajasegarar, S., Leckie, C., & Palaniswami, M. (2008). Anomaly detection in wireless sensor networks. *IEEE Wireless Communications*, *15*(4), 34–40. https://doi.org/10.1109/MWC.2008.4599219

Rayana, S., Zhong, W., & Akoglu, L. (2017). Sequential ensemble learning for outlier detection: A bias-variance perspective. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 1167–1172. https://doi.org/10.1109/ICDM.2016.117

Rosner, B. (1975). On the Detection of Many Outliers. *Technometrics*, *17*, 221–227.

Rosner, B. (1983). Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics*, *25*(2), 165–172.

Schaum, A. (2007). Advanced methods of multivariate anomaly detection. *IEEE Aerospace Conference Proceedings*, (1). https://doi.org/10.1109/AERO.2007.353061

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., & Edwards, S. T. (2017). Time Series Anomaly Detection; Detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. Retrieved from http://arxiv.org/abs/1708.03665

Song, H., Jiang, Z., Men, A., & Yang, B. (2017). A Hybrid Semi-Supervised Anomaly Detection Model for High-Dimensional Data. *Computational Intelligence and Neuroscience*, *2017*, 1–9. https://doi.org/10.1155/2017/8501683

Stoffel, F., Fischer, F., & Keim, D. A. (2013). Finding anomalies in time-series using visual correlation for interactive root cause analysis. *Proceedings of the Tenth Workshop on Visualization for Cyber Security - VizSec '13*, 65–72. https://doi.org/10.1145/2517957.2517966

Stokes, J. W., Platt, J. C., & Kravis, J. (2008). ALADIN : Active Learning of Anomalies to Detect Intrusion. *Microsoft*.

Tiku, M. L., Islam, M. Q., & Qumsiyeh, S. B. (2010). Mahalanobis distance under non-normality. *Statistics*, *1888*, 275–290. https://doi.org/10.1080/02331880903043223

Tiunov, P. (2017). [Untitled illustration of prediction based anomaly detection]. Retrieved from https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2

Tsay, R. S., Peña, D., & Pankratz, A. E. (2000). Outliers in multivariate time series. *Biometrika*, *87*(4), 789–804. https://doi.org/10.1093/biomet/87.4.789

Vendramin, L., Campello, R. J. G. B., & Hruschka, E. R. (2010). Relative Clustering Validity Criteria : A Comparative Overview. https://doi.org/10.1002/sam

Vieira, R. G., Leone Filho, M. A., & Semolini, R. (2018). An Enhanced Seasonal-Hybrid ESD Technique for Robust Anomaly Detection on Time Series, (i).

Vincent, P., & Larochelle, H. (2008). Extracting and Composing Robust Features with Denoising Autoencoders.

West, M. (1997). Time series Decomposition. *Biometrika*, *84*(2), 489–494.

Wong, T., & Zhiyuan, L. (2018). Recurrent Auto-Encoder Model For Multidimensional Time Series Representation. *Iclr*, 813–825.

Xiuyao, S., Mingxi, W., Jermaine, C., & Ranka, S. (2007). Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, *19*(5), 631–644. https://doi.org/10.1109/TKDE.2007.1009

Yang, Q., & Wu, X. (2006). 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology & Decision Making*, *05*(04), 597–604. https://doi.org/10.1142/S0219622006002258

Zhang, A., Wang, J., & Yu, P. S. (2016). Time Series Data Cleaning : From Anomaly Detection to Anomaly Repairing. *Pvldb*, *10*(4), 1046–1057. Retrieved from http://www.vldb.org/pvldb/vol10/p1046-song.pdf

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series.

European Journal of Operational Research, *160*(2), 501–514. https://doi.org/10.1016/j.ejor.2003.08.037

Zhang, X., Shen, F., Zhao, J., & Yang, G. (2017). Time Series Forecasting Using GRU Neural Network with Multi-lag After Decomposition. *Neural Information Processing*, *1*(1), 523–532. https://doi.org/https://doi.org/10.1007/978-3-319-70139-4_53

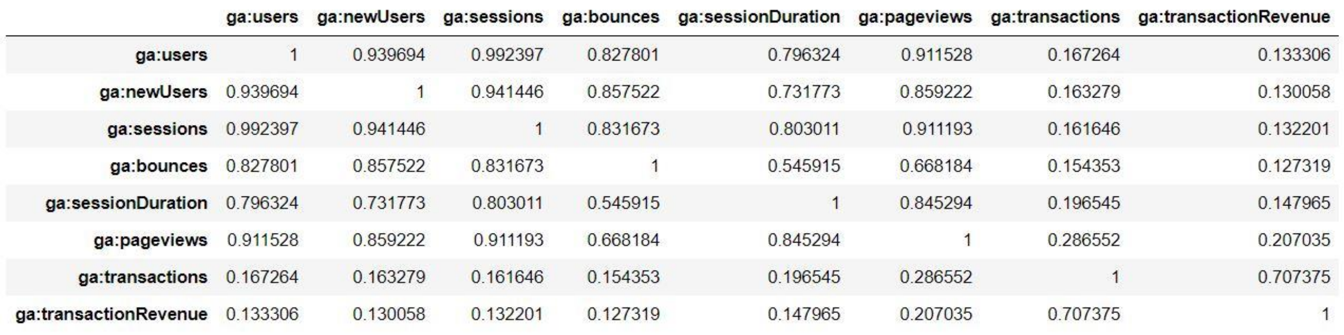
Zhou, C., & Paffenroth, R. C. (2017). Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17* (pp. 665–674). https://doi.org/10.1145/3097983.3098052

Zimek, A., Campello, R. J. G. B., & Sander, J. (2014a). Data perturbation for outlier detection ensembles. *Proceedings of the 26th International Conference on Scientific and Statistical Database Management - SSDBM '14*, 1–12. https://doi.org/10.1145/2618243.2618257

Zimek, A., Campello, R. J. G. B., & Sander, J. (2014b). Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations Newsletter*, *15*(1), 11–22. https://doi.org/10.1145/2594473.2594476

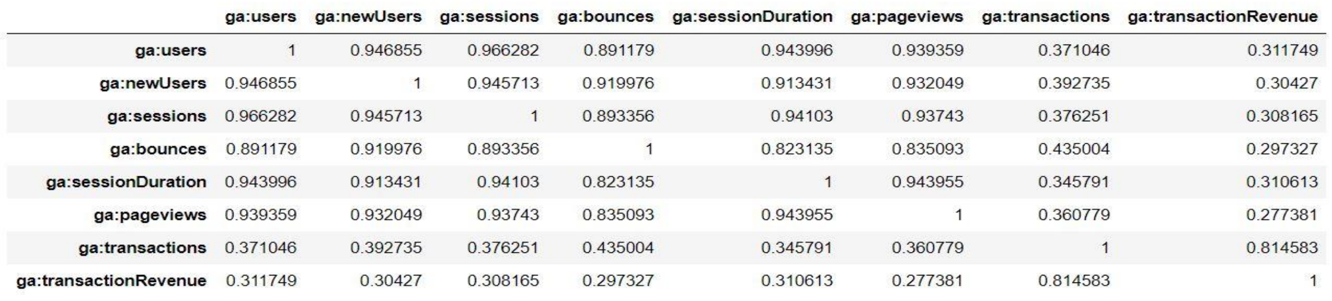
Zimek, A., Schubert, E., & Kriegel, H.-P. (2012). A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data. *Wiley Online Library*, *4*(5), 497–511. https://doi.org/DOI:10.1002/sam.11161

8. Appendix



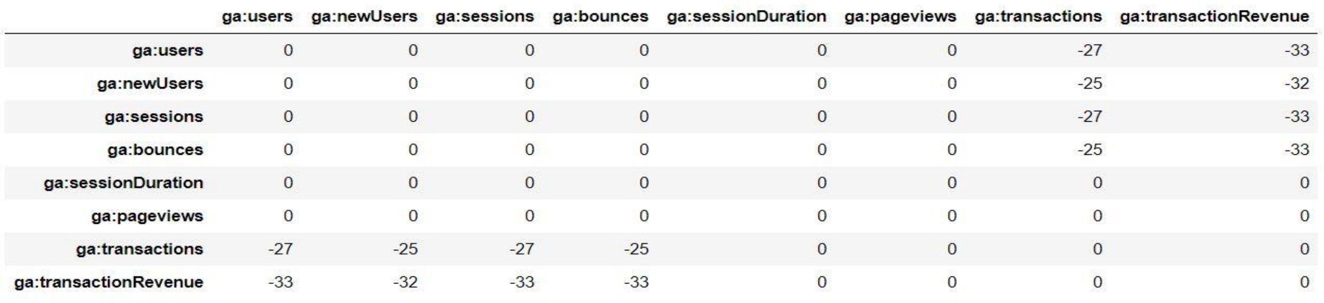
	ga:users	ga:newUsers	ga:sessions	ga:bounces	ga:sessionDuration	ga:pageviews	ga:transactions	ga:transactionRevenue
ga:users	1	0.939694	0.992397	0.827801	0.796324	0.911528	0.167264	0.133306
ga:newUsers	0.939694	1	0.941446	0.857522	0.731773	0.859222	0.163279	0.130058
ga:sessions	0.992397	0.941446	1	0.831673	0.803011	0.911193	0.161646	0.132201
ga:bounces	0.827801	0.857522	0.831673	1	0.545915	0.668184	0.154353	0.127319
ga:sessionDuration	0.796324	0.731773	0.803011	0.545915	1	0.845294	0.196545	0.147965
ga:pageviews	0.911528	0.859222	0.911193	0.668184	0.845294	1	0.286552	0.207035
ga:transactions	0.167264	0.163279	0.161646	0.154353	0.196545	0.286552	1	0.707375
ga:transactionRevenue	0.133306	0.130058	0.132201	0.127319	0.147965	0.207035	0.707375	1

Table 3. Normalized cross-correlation coefficients of hourly seasonal decomposition residuals.



	ga:users	ga:newUsers	ga:sessions	ga:bounces	ga:sessionDuration	ga:pageviews	ga:transactions	ga:transactionRevenue
ga:users	1	0.946855	0.966282	0.891179	0.943996	0.939359	0.371046	0.311749
ga:newUsers	0.946855	1	0.945713	0.919976	0.913431	0.932049	0.392735	0.30427
ga:sessions	0.966282	0.945713	1	0.893356	0.94103	0.93743	0.376251	0.308165
ga:bounces	0.891179	0.919976	0.893356	1	0.823135	0.835093	0.435004	0.297327
ga:sessionDuration	0.943996	0.913431	0.94103	0.823135	1	0.943955	0.345791	0.310613
ga:pageviews	0.939359	0.932049	0.93743	0.835093	0.943955	1	0.360779	0.277381
ga:transactions	0.371046	0.392735	0.376251	0.435004	0.345791	0.360779	1	0.814583
ga:transactionRevenue	0.311749	0.30427	0.308165	0.297327	0.310613	0.277381	0.814583	1

Table 4. Normalized cross-correlation coefficients of daily seasonal decomposition residuals.



	ga:users	ga:newUsers	ga:sessions	ga:bounces	ga:sessionDuration	ga:pageviews	ga:transactions	ga:transactionRevenue
ga:users	0	0	0	0	0	0	-27	-33
ga:newUsers	0	0	0	0	0	0	-25	-32
ga:sessions	0	0	0	0	0	0	-27	-33
ga:bounces	0	0	0	0	0	0	-25	-33
ga:sessionDuration	0	0	0	0	0	0	0	0
ga:pageviews	0	0	0	0	0	0	0	0
ga:transactions	-27	-25	-27	-25	0	0	0	0
ga:transactionRevenue	-33	-32	-33	-33	0	0	0	0

Table 5. Lag values corresponding to the highest found cross-correlation coefficient.

			w/o retrain			w/ retrain		
			recall	precision	F1-score	recall	precision	F1-score
	S-H ESD		0,474	1,000	0,643	N/A	N/A	N/A
AE (12,12)	UV rec	mean	0,654	0,761	0,703	0,654	0,761	0,703
		median	0,603	0,870	0,712	0,628	0,803	0,705
		min	0,603	0,810	0,691	0,603	0,770	0,676
		max	0,590	0,821	0,687	0,590	0,807	0,681
		last	0,551	0,741	0,632	0,551	0,729	0,628
	UV CDF	energy	0,590	0,902	0,713	0,577	0,918	0,709
	MV maha	mean	0,600	0,818	0,692	0,613	0,780	0,687
		median	0,640	0,828	0,722	0,627	0,797	0,701
		min	0,493	0,804	0,612	0,533	0,784	0,635
		max	0,587	0,772	0,667	0,573	0,694	0,628
		last	0,520	0,696	0,595	0,493	0,725	0,587

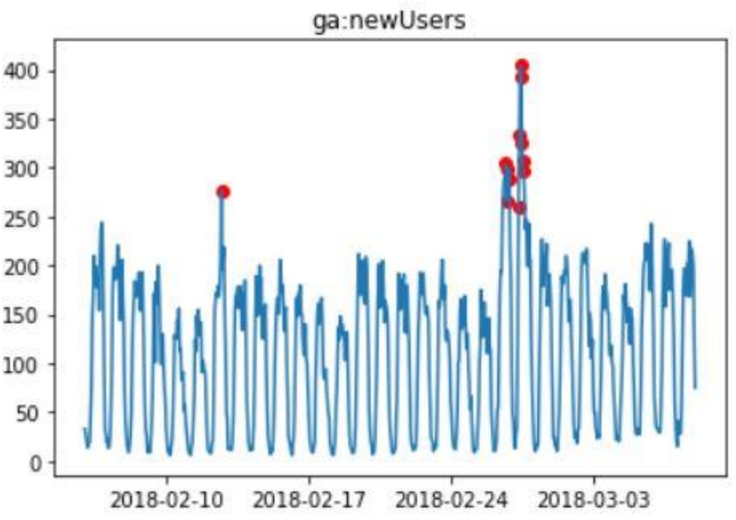
Table 6. Results comparison of retraining on the test set vs not retraining on the test set.
UV rec means univariate recreation (median), UV CDF stands for the univariate Cumulative Distribution Function and MV maha stands for the multivariate Mahalanobis distance.

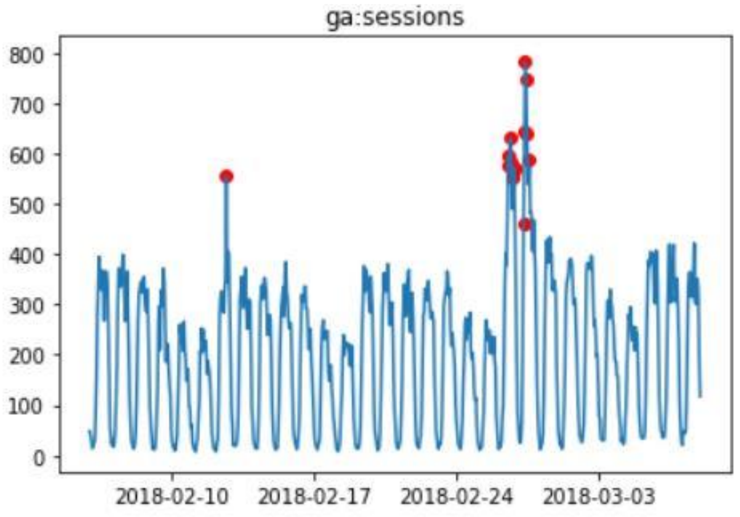
			Hyper-parameters
	S-H ESD		alpha = 1.5, max_anoms = 0.01, longterm=False
AE (12,12)	UV rec	median	Upper bound = 0.996, Lower Bound = 0.00367
	UV CDF	energy	Upper bound = 0.993
	MV maha	median	Upper Bound = 0,948

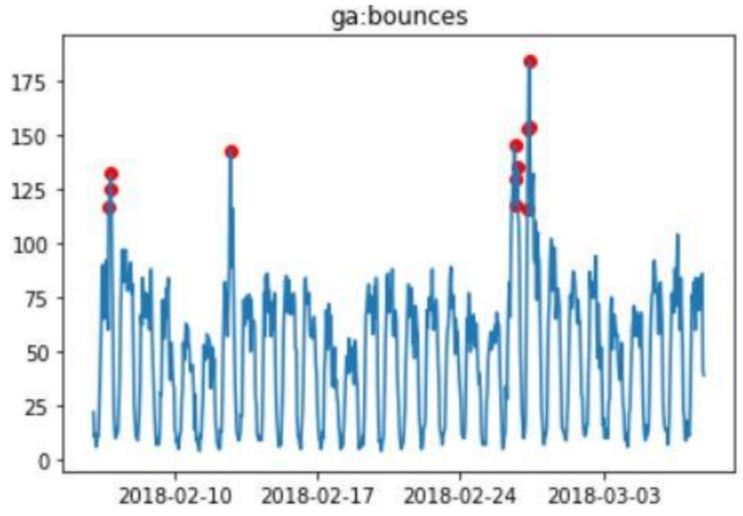
Table 7. Optimal hyper parameter settings.
UV rec means univariate recreation (median), UV CDF stands for the univariate Cumulative Distribution Function and MV maha stands for the multivariate Mahalanobis distance.

	Sequential/intersection			Parallel/union		
	recall	precision	F1-score	recall	precision	F1-score
MV-ED	0,560	0,977	0,712	0,667	0,794	0,725
MV-UV	**0,587**	0,957	**0,727**	0,653	0,790	0,715
MW-TW	0,533	**1,000**	0,696	0,707	0,646	0,675
ED-UV	0,560	0,933	0,700	0,627	**0,887**	**0,734**
TW-ED	0,493	**1,000**	0,661	0,693	0,693	0,693
TW-UV	0,507	**1,000**	0,673	0,693	0,684	0,689
ALL	0,480	**1,000**	0,649	**0,733**	0,625	0,675

Table 8. Anomaly detection ensemble results of all combinations.
MV = multivariate, ED = energy distance, UV = univariate, TW = Twitter package


ga:newUsers
0
50
100
150
200
250
300
350
400
2018-02-10
2018-02-17
2018-02-24
2018-03-03


ga:sessions
0
100
200
300
400
500
600
700
800
2018-02-10
2018-02-17
2018-02-24
2018-03-03


ga:bounces
0
25
50
75
100
125
150
175
2018-02-10
2018-02-17
2018-02-24
2018-03-03

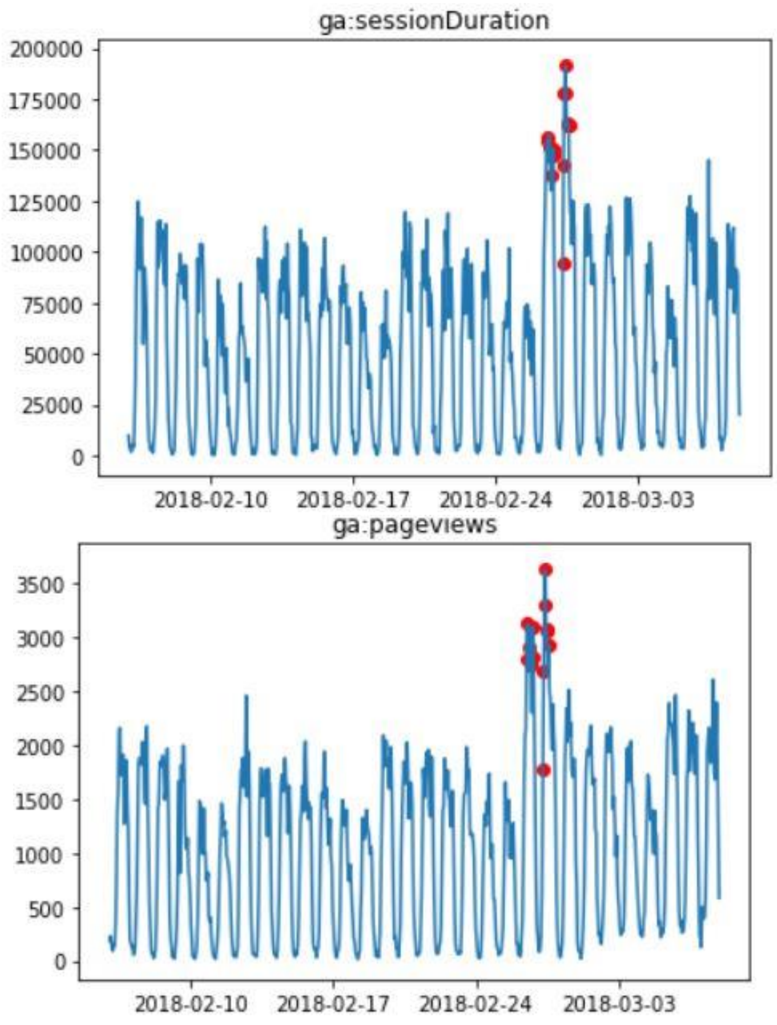


Figure 14. Example anomaly detection on the real dataset.