BACHELOR'S THESIS

Degree in Mathematics

Title: Daphne: A tool for Anomaly Detection

Author: Carlos García Ling

Advisor: Daniel Selva Valero

Department: Mechanical and Aerospace Engineering

Academic year: 2017/18

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

Daphne: A tool for Anomaly Detection

Carlos García Ling

June 2018

Supervised by Daniel Selva Valero

Department of Mechanical and Aerospace Engineering
Cornell University, Ithaca, New York

Acknowledgements

I would like to thank my advisor, Daniel Selva, for helping me develop this project. I also thank Antoni Virós, for introducing me to the Daphne project and welcoming me to the Ithaca.

I would like to thank all the sources of funding that have allowed me to do this project in the USA: the TFG scholarship from CFIS and Funcació Cellex, the MOBINT scholarship form Generalitat de Catalunya, and Cornell University.

Abstract

This thesis presents a versatile tool for anomaly detection, which is presented as a new functionality for Daphne's cognitive assistant. This tool allows the user to analyze any kind of time series data, univariate or multivariate, and search for non conforming patterns, i.e. anomalies. Several anomaly detection algorithms have been integrated to analyze either a concrete variable, or the multivariate series as a whole. Multiple algorithms allow the user to search for different kind of anomalies in the data, accounting on data properties (seasonality, correlation), or on problem requirements (isolated anomalies, clustered anomalies)

Complementary features to anomaly detection were also implemented, as an attempt to help the user choose a specific algorithm or understand its output. These features include data properties extraction (Seasonality, Correlation), checking agreement between different algorithms and diagnosing the novel anomalies using a database provided by the user.

To enhance user experience an interactive interface has been implemented. The main components of the interface are a plot and different functionalities. The plot displays the problem's variables and the outputs from the anomaly detection algorithms, and also allows the user to select and obtain information about the data. The functionalities allow the user to: run and configure anomaly detection methods; upload data; execute the complementary features.

The current implementation of the tool can be found at the following link:
`https://www.selva-research.com/anomalies/`

Keywords

Anomaly Detection, Anomaly diagnose, Cognitive Assistant, Unsupervised Anomaly Detection, Time Series Analysis, Multivariate Analysis

Contents

1. Introduction

In machine learning, anomaly detection refers to the identification of patterns of data that do not adjust to the expected behaviour [1]. This patterns are referred to as anomalies or outliers. Another definition for anomaly is: an observation which deviates enough from the others to suspect that it was generated by a different mechanism [2]. According to this, the anomalies can be an indicator that the system analyzed is not behaving in the expected way. As a result, detecting these patterns has multiple real life applications, by spotting crashes or malfunctions early, and preventing their consequences.

One of these applications is fraud detection [3], this includes any kind of undesirable behaviours such as bank fraud or system intrusion. As an example, financial institutions monitor credit card transactions. Applying anomaly detection algorithms, they can spot changes on the normal transactions pattern [4], and this can warn them that the credit card might have been stolen, or that an illicit use of the card is taking place. Another broadly used branch of fraud detection is computer intrusion [5]. Companies survey data, such as account logins or information requests, to guarantee that confidential information is not being compromised.

In many systems considering the behaviour of different variables can help detect and predict faults or overloads. As an example, in mechanical systems, detecting anomalies in bearings can help to predict their remaining useful life [6]. In satellite control systems, addressing control wheels [7] and sensor [8] malfunctions, can prevent catastrophic failures. In satellite power subsystems, anomalies must be monitored to guarantee overall system health [9], and each particular component's [10]. In distribution systems, like natural gas networks [11], detecting anomalies in consumption might warn for a failure like a leak on the distribution system.

In epidemiology, anomaly detection techniques are used to control different factors related with infectious deceases [12]. Anomalies in this factors might indicate events like the start of an outbreak. Early detection and diagnose of these incidents are fundamental for a fast action against them, preventing important consequences in global health.

Anomaly detection techniques can be even applied to computer vision data. In particular, video surveillance data has been fed to these algorithms in order to detect infractions in crowded scenes [13] like sneaking into the subway [14].

This thesis presents a tool for anomaly detection based on Daphne cognitive assistant [15], motivated by the diversity of applications of anomaly detection and the lack of a free versatile and open source tool for this problem. Daphne's anomaly detection tool presents an interface to analyze any kind of time series data, by performing anomaly detection, extracting data properties and understanding and diagnosing the detected anomalies.

The structure of this thesis starts with a related work section 2, where other anomaly detection tools are presented, as well as the current approaches on anomaly detection methods. Sections 3 and 4 explain in detail the Anomaly detection algorithms used and the functionalities implemented respectively. To show the potential of the tool some practical examples are developed in 5. Section 6 presents future work in order to improve the current tool. To end Section 7 presents conclusions and remarks on the work done.

2. Related work

This section presents other already developed tools for anomaly detection (Section 2.1), and the current literature on anomaly detection methods (Section 2.2)

2.1 Anomaly Detection Tools

Detecting anomalies and monitoring temporal data is a topic of interest in many areas. Due to the numerous applications, many companies are interested on integrating this technology. As a result, tools for anomaly detection have been developed to satisfy the market demand.

New tools like *Anodot* [16], allow businesses to monitor variables on marketing and sales environments in order to detect anomalous events, such as drops on sales. By crossing information from diverse sources, such as of social networks or important events, and using big data algorithms, these algorithms provide possible explanations of the anomalies. All these allows companies to react and correct these behaviours.

Other tools like *CA technologies Threat Analytics* [17], are more focused on security concerns. This tool provides an interface to monitor the accesses to the company's network, in order to avoid compromising important information. Using anomaly detection algorithms it warns the user of suspicious activities. Once suspicious activity is spotted, it tracks and displays the context and content in order to recognize the anomalous behaviour.

Both [16] and [17], provide very specific solutions targeted to particular problems. In contrast, *HTM-Studio* [18], another Anomaly Detection tool provides a more generic approach. Using Hierarchical Temporal Memory (HTM) algorithms, patterns in any temporal data provided are modeled and, applying specific anomaly detection methods to HTM [19, 20], anomalous data is located. This tool is quite versatile, as it allows any temporal series to be analyzed. Although the main drawback is that it only allows to analyze univariate time series, and it cannot be used to extract any other information form the data other than anomalies.

In contrast with the existing tools Daphne's anomaly detection tool allows the user to analyze any dataset, and detect anomalies both in univariate series and multivariate series, allowing to use multiple anomaly detection methods to adapt itself to the requirements of each particular problem. It also allows the user to extract intrinsic information about the data such as correlations or seasonality, and to diagnose the anomalies detected when a database of anomalies is provided.

The main outcome of the presented tool is the versatility at analyzing any dataset, and multiple kind of problems. Keeping up with Daphne [15], this tool is also presented as open source and free to access, in contrast with most of the available tools. Although, to present it as a truly relevant tool many of the points presented in future work (Section 6) have to be implemented first.

2.2 Anomaly Detection Algorithms

Anomaly Detection is a broad field that has been studied in the context of a large domain of application. An extensive overview is provided in the literature [1, 2, 21, 22], presenting approaches to different kinds of data, and to different specific problems.

There are three possible approaches to the anomaly detection problem, in terms of whether labeled data is provided: Supervised (labeled normal and anomalous data), Semisupervised (only normal instances) and Unsupervised (generic data). Although there is literature regarding Supervised [23] and Semisupervised [24] algorithms, mainly unsupervised algorithms were consider to be applied. This is mainly because of the majority of the data provided corresponds to unlabeled datasets, and because the supervised methods might fail to detect novel anomalies. Also some of the unsupervised methods can be easily modified to perform in a semisupervised fashion by providing a particular training dataset.

As Daphne's tool is oriented to temporal data, time series approaches are interesting to consider [25]. This techniques usually consider the temporal context, instead all the data instances. The main advantage of these methods is that they adapt to changing patterns in data over time. As a result, they can detect more subtle anomalies, that might be hidden to a more general algorithm. When analyzing time series two type of anomalies can be considered: punctual anomalies, or anomalous sub sequences. The developed tool is mainly focused on punctual anomalies, for what different approaches can be considered.

A common approach consists on using predictive models in order to estimate the series value at each moment. After that, the real values are compared with the estimated ones. If there is not agreement between the predicted and the real values, the point is considered an anomaly. This includes methods such as ARIMA models [26, 27, 28] HTM [20, 19], single-layer lineal network(AR model)[29], nearest cluster [29], or multilayer perceptron [29].

Other methods only check if the data corresponds to the patterns of the data in a determined temporal context. This can be done by using measures like entropy [30], or by checking if each point corresponds to the distribution from its context's data [31]. As time series can be interpreted as sequences, frameworks using Hidden Markov Models (HMM) [32] can be also applied. A common methodology using HMM consists on considering that the data might be generated by two hidden states: an standard hidden state, or an anomalous state.

Another approach, that has not yet been yet considered in Daphne's tool, is to search for anomalous subsequences on the time series. To do so, similarity measures are computed between subsequences of the time series, these serve as a start point for the anomaly detection algorithm. Examples of these measures include Dynamic Time Warping (DTW) [33], or simply, euclidean distance [34].

Time series methods often lack of a general approach, this made take into consideration to also implement algorithms that take into account all the data on the time series, and not just the point's context. There is a wide variety of these kind of algorithms such as: density based methods [35], isolation methods [36], clustering based methods [37, 38] or support vector machines based methods[39]. All these strip away the temporal dimension from the data, and only consider the values of the variables to spot anomalies in an absolute context.

3. Methodologies

Recalling this diversity of the needs of the data, different methods have been implemented to try to include various types of data as well as different problems requirements. Four methods have been implemented, two univariate (Sections 3.1 and 3.2) and two multivariate (Sections 3.3 and 3.4). Both univariate methods are widely known and commonly applied. The multivariate methods are more recent, the first one was published this year, while the second one is already quite known and applied. The algorithms mentioned in this section were implemented using *python* programming language, aided mainly by *numpy* and *pandas* libraries.

3.1 Statistical Methods

Statistical methods are based in one main assumption, the fact that the data sampled follow a concrete statistical distribution. The statistical distribution may be chosen attending to the data properties, or using a priory knowledge regarding the data.

In particular Gaussian model-based methods are widely used, both can be applied to univariate data [2] and to multivariate data [40]. These methods have been used in fields such as product quality control [41], data centers (i.e. web servers, online cloud servers...) [42], sensor data [31], or systems fault prognosis [6] and diagnosis [43].

Using the normal distribution to detect anomalies is quite simple and fast, as a result the Gaussian model based methods can be applied to online anomaly detection algorithms. Although, the performance can be quite poor, but it can be considered as a start point to compare with other developing anomaly detection methods.

3.1.1 Proposed Method

In order to have a benchmark anomaly detection method the Gaussian model based method was implemented. The implementation was done in a very similar way that the one presented in [31].

The anomaly detection algorithm analyzes each data point x_i individually, and classifies it in "anomalous" or "non anomalous". To perform this classification a context C_i for each the data point is considered. This context is defined to contain the w previous data points $C_i = \{x_{i-w}, x_{i-(w-1)}, ..., x_{i-2}, x_{i-1}\}$. The points of C_i are fitted to a normal distribution $N(\mu_i, \sigma_i^2)$ whose are estimated using the following expressions:

$$\hat{\mu}_i = \frac{\sum_{x \in C_i} x}{w} \tag{1}$$

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{x \in C_i} (x - \hat{\mu}_i)^2}{w - 1}} \tag{2}$$

Once the parameters have been estimated, the data point is classified by comparing it to the context data. If does not fit the data, it will be considered an anomaly. To determine if it fits or not, it will be required to belong with high probability to the contexts C_i distribution, $N(\mu_i, \sigma_i)$. To distinguish between anomalous and non anomalous instances, an interval I_i is fixed for each point. The analyzed data point x_i will be considered non anomalous if it lays on the interval $I_i = [\hat{\mu} - t, \hat{\mu} + t]$, and anomalous if it lays outside it. The term t is a threshold fixed attending to $\hat{\sigma}_i$, taking values like $t = 3\hat{\sigma}$, $4\hat{\sigma}$.

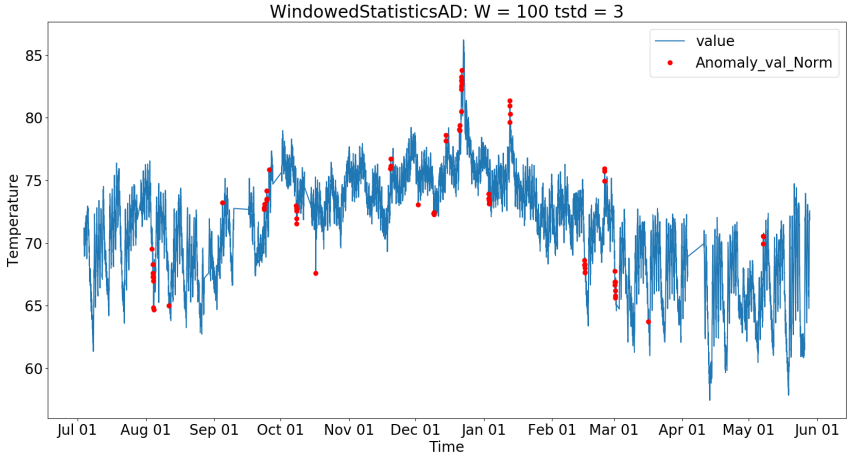
Algorithm 1 WindowedStatisticsAD(X,w,tstd)

Input: X:input data, w:windowing parameter, tstd:threshold parameter
Output: Detected Anomalies
  n = length(X)
  Anomalies = [] {initialize anomalies detected}
  for i=0 **to** i=n **do**
    PreviousData = $X[i - w, i - 1]$
    mean = Mean(*PreviousData*)
    std = Standard Deviation(*PreviousData*)
    if not $X[i]$ **in** $[mean - tdtd \cdot std, mean + tdtd \cdot std]$ **then**
      Add $X[i]$ **to** *Anomalies*
    end if
  end for
  return *Anomalies*

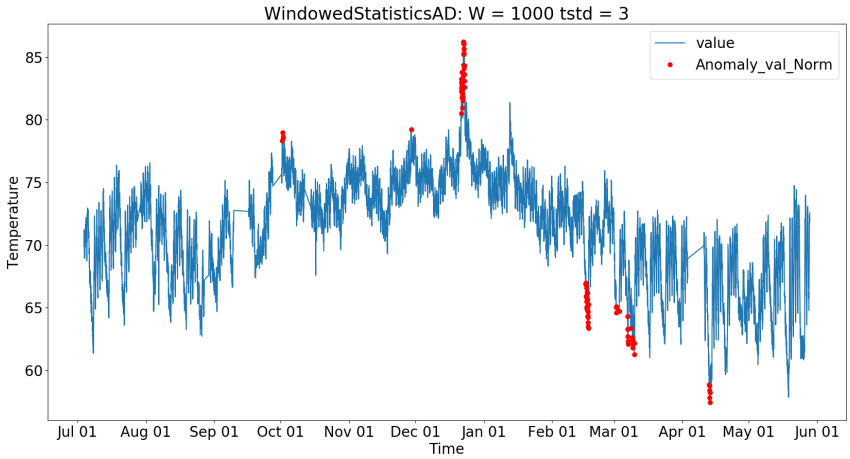
The final pseudo code of the method is presented in algorithm 1. In practice the algorithm is slightly modified, to avoid problems with the first w data points, this instances are not considered. In this algorithm two parameters must be fixed:

- Windowing parameter: w. Fixing an adequate windowing parameter is not trivial, it depends on which kind of anomalies want to be spotted. A higher windowing parameter will only detect anomalies with a more absolute nature, while anomalies detected with a smaller windowing parameters have a more local nature (Figure 1).

- Threshold: t. The threshold parameter will be fixed as a function of $\hat{\sigma}$, by defining a new parameter t_{std}, being $t = t_{std}\hat{\sigma}$. This parameter will only account on the degree of the anomalies detected. A higher t_{std} will account for extreme anomalies while a lower value of the parameter will also detect milder anomalies.



(a) WindowedStatisticsAD Method applied with setting $w = 100$



(b) WindowedStatisticsAD Method applied with setting $w = 1000$

Figure 1: By setting the adequate w parameter, the algorithm can be configured to detect anomalies in a more local context with a smaller window, 1a, or in a more general scope setting w to a higher value, 1b. The data analyzed was obtained form the Numenta Anomaly Benchmark Database [44], it corresponds to the ambient temperature in an office that experimented a failure on the control system.

3.2 ARIMA Model Based

The Auto Regressive Integrated Moving Average(ARIMA) models [45] are commonly find in prediction problems. Examples might include forecasting electricity prices [46], or predicting the run off water resources [47]. It also can be used combined with different algorithms such artificial neural networks [48] to increase the accuracy of the model.

These ARIMA models predict the a value of the time series at every time within a confidence interval. The approach in anomaly detection consists on computing this interval and, for each data point, checking whether it belongs or not to it, classifying them as anomalous if they do not.

3.2.1 ARIMA Models

ARIMA models are used to describe a time series by using the values and errors from the previous terms. Given a time series $\{x_0, x_1, x_2...\}$ is explained by an ARIMA model as follows:

$$(1 - \sum_{i=1}^{p} \alpha_i L^i)(1 - L)^d x_n = (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_n \tag{3}$$

The terms from Equation (3) will be explained below:

- ϵ_n is the prediction error, correspondent to the difference between the estimated term from the ARIMA model and the real term: $\epsilon_i = x_i - \hat{x_i}$. Isolating the x_n term from equation (3) it can be expressed in with the previous terms and error of the series, that would represent the estimated $\hat{x_n}$ (Equation (5)), and the current error (Equation (4)). If the model is fitted correctly the error distribution must follow a normal centered in zero ($\epsilon_n \approx N(0, \sigma)$)

$$x_n = -\sum_{k=1}^{d}(-L)^k \binom{d}{k} x_n + (\sum_{i=1}^{p} \alpha_i L^i)(1 - L)^d x_n + (\sum_{i=1}^{q} \theta_i L^i \epsilon_n) + \epsilon_n \tag{4}$$

$$\hat{x_n} = -\sum_{k=1}^{d}(-L)^k \binom{d}{k} x_n + (\sum_{i=1}^{p} \alpha_i L^i)(1 - L)^d x_n + (\sum_{i=1}^{q} \theta_i L^i \epsilon_n) \tag{5}$$

- **L** is the lag operator, it represents the previous term of the series when applied to an element, for example: $Lx_n = x_{n-1}$, or $L\epsilon_n = \epsilon_{n-1}$
- **p**, **d**, and **q** are respectively the auto regressive, integration, and moving average parameters. These are the ones that can be previously defined for an ARIMA model
  - **p** parameter indicates the number of previous terms of the series considered to fit the data, and can be estimated taking into account the time series partial auto correlation function.
  - **d** parameter stands for the number of times the series is differentiated, as L is the lag function $(1 - L)$ is the differentiating operator ($(1 - L)x_n = x_n - x_{n-1}$). The d parameter is found by differentiating the times series until it can be considered stationary.
  - **q** parameter indicates the number of previous error terms considered to predict the next data point, and it can be estimated taking into account the time series auto correlation function.
- α_i are the auto regressive coefficients, which explain how the series will behave considering the previous values.

- θ_i are the moving average coefficients, and they model the influence of the previous errors calculating the series value on the next term.

Both α_i and θ_i coefficients are found iteratively by maximizing the likelihood that these parameters explain the series. The parameters are specific for each combination of p, d, q, parameters. In order to choose the adequate parameters first d is estimated by differentiating the series until this is stationary. To check the stationarity of the series the Augmented Dickey-Fuller test is performed [49]. Once d parameter is fixed, a grid search is performed to select p and q. The parameters are chosen attending to the AIC metric, which represents a trade-off between the complexity of the ARIMA model, and the goodness of fit.

3.2.2 Seasonality and exogenous variables

To introduce further complexity the model can be accommodated to include the seasonal trend and also the influence of other variables. The resulting model expression is obtained by adding the seasonality terms to the model as follows:

$$(1 - \sum_{i=1}^{p} \alpha_i L^i)(1 - \sum_{i=1}^{P} A_i L^{i \cdot s})(1 - L)^d (1 - L^s)^D x_n = (1 + \sum_{i=1}^{q} \theta_i L^i)(1 + \sum_{i=1}^{Q} T_i L^{i \cdot s})\epsilon_n \tag{6}$$

In the previous equation (6), the new terms added correspond to the seasonality modeling. The s variable indicates the seasonality lag, that represents the period of the seasonality. The resulting model is denominated Seasonal Autoregressive Integrated Moving Average (SARIMA). The parameters P, D and Q, are the equivalent of their correspondent lowercase parameters, they indicate analogous properties but for the seasonal properties. These are estimated similarly to p, d and q. The parameters A_i and T_i are fitted in an iterative process which maximizes the likelihood of the parameters. The fitting of the SARIMA model parameters is done simultaneously (parameters α, θ, A, T). In practice the statsmodels implementation of this method for python was used

Also exogenous variables can similarly be added to the model (SARIMAX). The relations implied in the model are linear, in consequence the variables added must be correlated to the principal time series data analyzed.

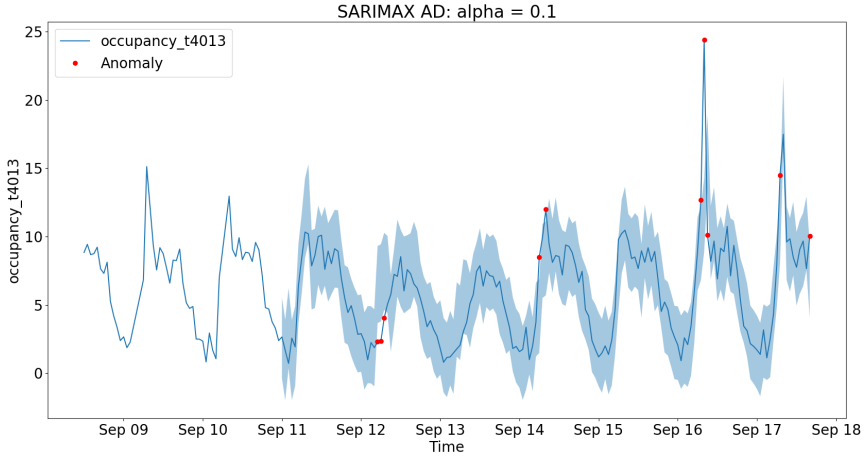
3.2.3 Application to Anomaly Detection

The main use of this model is to be used as a forecasting tool. Although it can be also used as an anomaly detection tool In Algorithm 2 the process to implement anomaly detection is explained in detail. It consists on two main parts which are: building the model, and predicting a confidence interval for each point. The classification of the points on anomalous or non anomalous is quite simple, the anomalous data will lay outside the confidence interval while the non anomalous are the ones located in it.

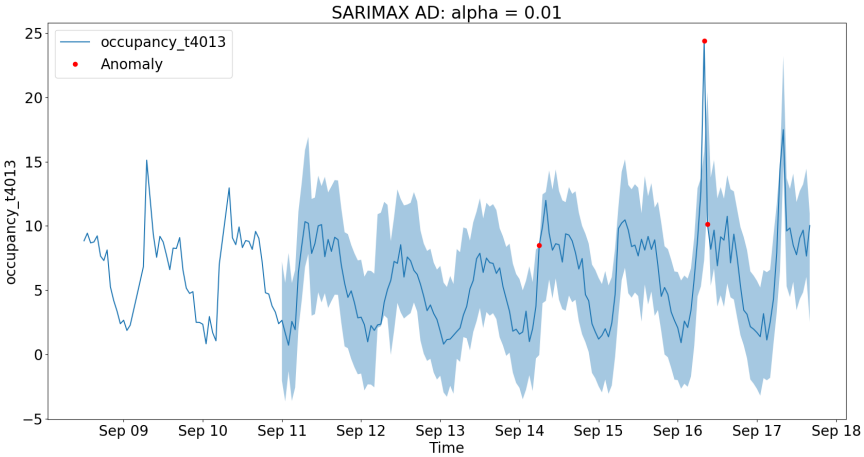
Algorithm 2 SARIMAXAD(X,α,S,ex)

Input: X:input data, α: sensibility parameter, S: seasonality parameter, ex: exogenous variables
Output: Detected Anomalies

```
Anomalies = [] {initialize anomalies detected}
[p, d, q, P, D, Q] = Find Model Parameters(X, S, ex)
SARIMAXModel = SARIMAX(X, S, ex, [p, d, q, P, D, Q])
Fit Model(SARIMAXModel)
for all (t, y) in X do
    CI_y = Confidence interval(SARIMAXModel,t, α)
    if not y in CI_y then
        Add (t, y) to Anomalies
    end if
end for
return Anomalies
```



(a) SARIMAX Method applied to traffic data setting $\alpha = 0.10$



(b) SARIMAX Method applied to traffic data setting $\alpha = 0.01$

Figure 2: The shaded part of the plots represent the confidence intervals. A higher α, 2a, rises the sensitivity by narrowing the confidence interval and detecting more anomalies as a result. On the other hand, lower α values, 2b, mean in a wider confidence interval and only detecting very anomalous data

The anomaly detection algorithm uses α in order determine the amplitude of the confidence interval, the meaning of this parameter is the uncertainty of the prediction, this means than when for example an $\alpha = 0.01$ parameter is chosen, there is a 1% uncertainty of the variable not being in the on the output confidence interval. Translated into anomaly detection terms, a higher value of the parameter means that the method will be more sensitive to anomalies, detecting many but also generating false alarms, while a lower parameter will only detect extreme anomalies, usually overlooking less relevant anomalies (Figure 2).

The rest of the arguments of the algorithm are the seasonal properties (s) and exogenous variables (ex), adding this instances, increases the specificity of the model to by providing the different patterns and relations between the data, and as a result adapting better to the data analyzed.

3.3 Adaptive Kernel density-based

Density based algorithms is considered as a part of the nearest neighbor based anomaly detection techniques. The approach is quite simple, being the main assumption that normal data is located in higher density regions, while anomalous lays in low density regions [1].

The fist step for this kind of methods is to compute a density for each point of the dataset, to do so a distance must be defined. As a clarification, all the distances calculated in this section refer to the euclidean distances, although, this method is valid for any distance that verifies the definition. The original algorithm developed in this section is presented on [35], it consists on two main steps, which are:

- Calculate the local density for each point
- Calculate a Local Anomaly Score (LAS)

First of all, data must be normalized in order to avoid underestimate or overestimate any variables, due differences in the magnitudes of the values. In this case, the normalization performed on the data consists on standardizing the variables, establishing zero mean and unit variance.

The density is computed using a Parzen window estimate, or Kernel density Estimate (KDE). Given a sample of m samples obtain from the probability density $p(x)$, the estimator is formulated as it follows:

$$\hat{p}(x) = \frac{1}{m}\sum_{i=1}^{m} h^{-n} K(\frac{x - x_i}{h}) \tag{7}$$

The term $K(\cdot)$ represents the kernel function, and h a width parameter that controls the smoothness of the estimator. In this approach, the kernel function considered is the Gaussian Kernel. As a result, the expression for the local density results on:

$$\rho(x_i) = \frac{1}{m-1} \sum_{j \in \{1,2,\dots,m\} \setminus \{i\}} exp\Big\{ - \Big(\frac{d(x_i - x_j)}{r_i}\Big)^2 \Big\} \tag{8}$$

The parameter r_i, is the kernel width, which is not a fixed parameter, instead it is determined for each point. This determines how to weight the terms accounting on their distance to x_i. Wider r_i means that, fixed a distance, the same point will have a greater weight. In anomaly detection it is useful to fix the width narrower in low density regions and wider in high density regions, in order to help highlighting the differences between the normal and the anomalous data. It will be computed as follows for each datapoint:

$$r_i = c[d_{k-max} + d_{k-min} + \epsilon - d_k(x_i)] \tag{9}$$

In the equation above, c represents the overall smoothing parameter, which is recommended to be between 0.5 and 1, in the case of density estimation. The term $d(x_i)$ is the average distance of the k-nearest neighbors. The terms d_{k-max} and d_{k-min} represent respectively the maximum and minimum of the set $\{d(x_i)\}$. Lastly the term ϵ takes a small value, ensures the non zero value of the width parameters, guarantees existence of $\rho(x_i)$.

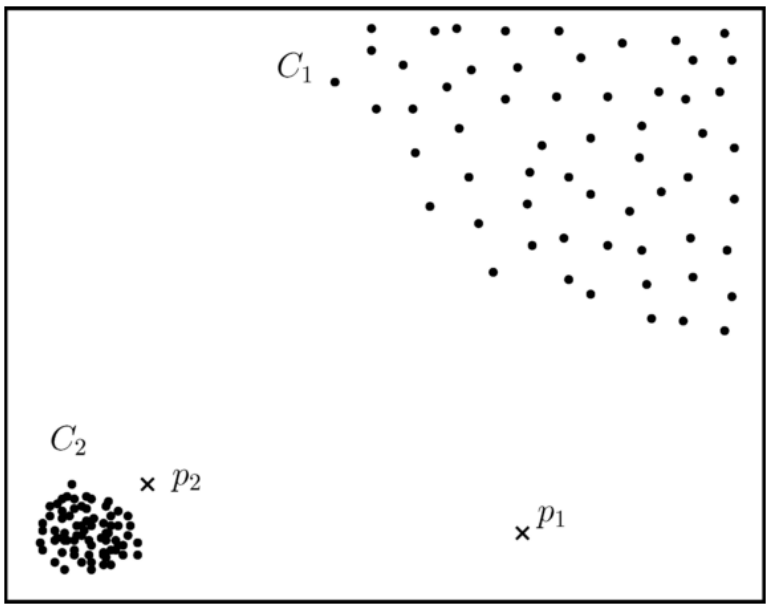


Figure 3: Using an anomaly threshold based only in the density of the points, point p_1 might be always flagged as an anomaly, to classify p_2 as anomalous this will imply to classify normal instances from C_1 as anomalies too
[1]

Once the densities have been computed, the output of the algorithm is computed. The LAS is computed as quotient between the density of the point considered, and the density of the neighboring points.

$$LAS(x_i) = log\left[\frac{\frac{1}{k}\sum_{j \in kNN(x_i)} \rho(x_j)}{\rho(x_i)}\right] \tag{10}$$

The LAS overcome the limitations of a density defined for the globally, by allowing to detect anomalies

in a contextual way, avoiding the miss classification problems showed in Figure 3

To classify a point as an anomaly its density should be significantly lower than its neighbors. The LAS gives a measure to this concept, as it is defined in equation 10, a higher LAS means the densities of the neighboring points are higher than the analyzed point's, and, as a result, the data is characterized as more anomalous. The values of the LAS are not comprised between any predetermined values, as a result, fixing a threshold a priory might not be the most adequate option, although values lower than 0 are not consider as anomalies, as this means that the density of the point is greater than the ones surrounding it.

The pseudocode of the implemented method is found in algorithm 3

Algorithm 3 AdaptiveKDB(X,k,c)

Input: X:input data, k: number of nearest neighbors, c: smoothing parameter
Output: Anomaly Score
  $AnomalyScore$ = [] {initialize anomalies detected}
  D = Compute Distance Matrix(X)
  for all x_i **in** X **do**
    KNN_i = Compute Nearest Neighbors($D(x_i)$)
    $d_k(x_i)$ = Compute Mean(D(KNN_i, x_i))
  end for
  d_{max} = max($d_k(x_i)$)
  d_{min} = min($d_k(x_i)$)
  for all x_i **in** X **do**
    r_i = Compute kernel radius($d_k(x_i)$){Use equation 9}
    ρ_i = Compute density($D(x_i)$, r_i){Use equation 8}
  end for
  for all x_i **in** X **do**
    $AnomalyScore[i]$ = Compute Anomaly Score(ρ_i, ρ){Use equation 10}
  end for
  return $AnomalyScore$

In Algorithm 3 the matrix D represents the distance matrix, in which each element $D_{(i,j)}$ represents the distance between elements i, and j ($D_{(i,j)} = d(x_i, x_j)$). The computation of D matrix is the slowest part of the algorithm, and increases with complexity $O(n^2)$.

To avoid this issue, in problems with big volumes of data, a subset of the data can be used to train the model. If there is a subset of the data that represents the normal behaviour of the data, it can be useful to consider this as the training set. In the implementation done, a random sampling is performed as determined by the user to reduce execution time when working with big data sets.

Although online implementations are not considered yet, the algorithm can be used for online anomaly detection. In this case a training set must be fixed, and also special attention must be payed to equation (9), as the $d(x_i)$ might exceed d_{k-max}, as a result r_i, must be limited to a minimum value ($c[d_{k-min} + \epsilon]$)

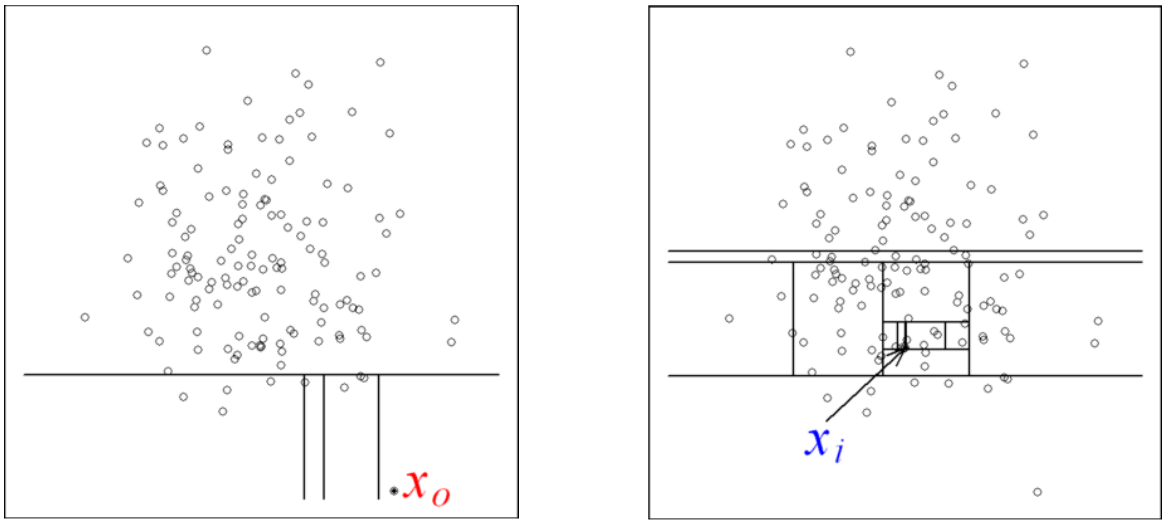
3.4 Isolation Forest

The last Anomaly Detection Algorithms is the Isolation Forest [50]. It is focused on multivariate data sets, but does not take into account the time dimension. The strategy of this algorithm is to randomly divide the data, by recursively partitioning it, using what is denominated as isolation trees. Once many of these trees are obtained, the data points are evaluated through these trees to obtain an Anomaly Score. The Isolation Forest methodology it is based in a property of the data called mass [51], assuming that normal data lays in regions with bigger mass, while anomalous data lay in regions with lower mass. One of the main advantages of using this property, is the ability to efficiently detect clustered anomalies. This algorithm is significantly different to standard anomaly detection algorithms, as instead of profiling the normal data, it directly the isolates anomalies.

To clarify, some elements of the problem are defined. The input of the algorithm consists on a data set $X = \{x_0, x_1 ... x_n\}$, with each of the data points defined by m attributes: $x = (x^{q_1}, x^{q_2}, ... x^{q_m})$, being $Q = \{q_1, q_2, ... q_m\}$ the data attributes . The algorithm processes the data and assigns each point an anomaly score $AS(x_i) \in [0, 1]$, 1 meaning that the data is very anomalous, and 0 meaning the data is not anomalous at all.

This algorithm performs two different steps in order to obtain the anomaly score:

- Create the Isolation Forest (Training stage)
- Evaluate the Data through the Isolation Forest (Testing stage)



(a) (b)

Figure 4: Differences regarding number of partitions in anomalous data 4a regular data 4b [50]

The training stage is responsible of creating the model, by recognizing the data distribution, and in particular, the regions where the data is more "isolated". This translates in the idea that when the data

is randomly recursively partitioned, the number of partitions to arrive to one that contains only a single instance is small(Figure 4a), in contrast, generic data requires a greater number to reach isolation(Figure 4b).

These partitions are denominated Isolation Trees, which can be defined recursively as follows:

Definition 3.1. Isolation Tree Given T a node of an Isolation tree. T can be either an *external node* (*exNode*) with no child or and *internal node* formed by a test and two child nodes (T_l, T_r). A *test* is defined by an attribute $q \in Q$ a split value p, the test divides the elements of T in the nodes: $T_l = \{x \in T | x^q < p\}$ and $T_l = \{x \in T | x_q >= p\}$

Algorithm 4 iTree(X,e,l)

Input: X:input data, e:current tree height, l-height limit
Output: an iTree
  if $e \geq l$ **or** $|X| \leq 1$ **then**
    return *exNode*$\{Size : |X|\}$
  else
    randomly select $q \in Q$
    max: maximum value of attribute q in X
    min: maximum value of attribute q in X
    randomly choose $p \in [min, max]$
    Split X in $X_l = \{x \in X | x^q < p\}$ and $X_r = \{x \in X | x^q \geq p\}$
    return *InNode*(
        Left : *iTree*$(X_l, e+1, l)$,
        Right : *iTree*$(X_r, e+1, l)$,
        SplitAttribute : q,
        SplitValue : p)
  end if

The training phase consists on computing a reasonable number t of iTrees, in order to identify the isolation patterns of the data. By computing a big number of isolation trees and averaging them, the normal behaviour of the data can be profiled. The pseudocode (Algorithm 4), shows how a iTree is computed. To implement this function, two classes (*InNode* and *ExNode*) must be defined as stated in definition 3.1.

In contrast with other anomaly detection algorithms, in which more data means a better fit, isolation trees work better when keeping the training data small, as large datasets reduce the capacity of isolating the data. As a result, in the training phase the iTrees are not defined using the whole data X, instead, a sub sample of the data $X' \subset X || X'| = \psi$, is obtained to fit each iTree. The subsampling size parameter ψ is fixed to a number, fixing $\psi = 256$ generally provides a good performance of the algorithm [50].

Once the iForest is computes, the testing stage is performed. In the testing stage the average path length is computed for each $x \in X$, in order to obtain the Anomaly Score.

Definition 3.2. Given an instance x and an iTree T, the path length of x referred to T ($h(x)$) is the number of edges that must be gone through in order to reach an external node.

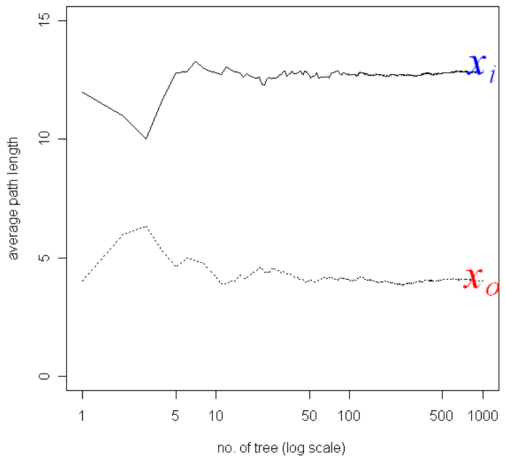


Figure 5: Convergence of the average path length[50]

When increasing the number of generated isolation trees, the average number of partitions converges to a concrete value as shown in Figure 5. The interest of the algorithms lays in identifying elements with shorter paths, as a result, there is no point in calculating in detail the path of elements with the longest paths. When defining the iTree, a limit of height was fixed in order to avoid this computations. This height is defined as the average path of an iTree ($\log_2(\psi)$), and represents the l parameter from Algorithm 4. In order to estimate the longitude remaining when an external node is reached, a term based ($c(s)$) in the size of the external node (s) is added to the path length of the node. This term is computed based in equation 11 that represents the average path length of an unsuccessful search in a Binary Search Tree (BST) [52] of size n. The algorithm to compute the path length is detailed in Algorithm 5

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{11}$$

In equation 11 the term $H(i)$ represents the harmonic number which can be estimated by $H(i) = \ln i + \gamma$, being γ the Euler-Mascheroni constant.

Algorithm 5 pathLength(x,T,e)

Input: x: a data point, T: an iTree , e: current path length (initialized to 0)
Output: path length of x

```
if T is exNode then
    return e + c(T.Size) {c() defined in }
end if
q = T.SplitAttribute
if x^q < T.SplitValue then
    return pathLenght(x, T.left, e + 1)
else
    {x^q ≥ T.SplitValue}
    return pathLenght(x, T.right, e + 1)
end if
```

Once the average path has been computed the anomaly score can be computed. The output score of this method is based on the average path length of a BST with size the sub sampling size: $c(\psi)$. The score is defined by equation 12. In the Anomaly score equation the term $E(h(x))$ indicates the expected value of the path length of element x, which can be estimated by averaging the lengths over the trees computed $E(h(x)) \approx \bar{h(x)}$

$$AS(x) = 2^{-\frac{E(h(x))}{c(\psi)}} = 2^{-\frac{\bar{h(x)}}{c(\psi)}} \tag{12}$$

Considering equation 12 the anomaly score can be easily interpreted. Data points x with values $AS(x) \leq 0.5$ mean that $\bar{h(x)} \geq c(\psi)$, i.e. the mean path length is greater than an the mean unsuccessful BST path, which implies that x is not characterized as anomalous. On the other hand, if x has an anomaly score significantly greater than 0.5 this will indicate an anomalous point.

Once all the stages of the algorithms have been defined, the pseudocode of iTree can be found at Algorithm 6

Algorithm 6 iForest(x,ψ,t)

Input: x: a data point, ψ: subsampling size , t: number of iTrees computed
Output: Anomaly Score

```
l = Set Limit iTree Height (ψ)
T = initialize iTree array
for i=0 to i=t do
    Xsample = Subsample a ψ sized data from x
    T = Append(T, iTree(Xsample, 0, l))
end for
AnomalyScore = initialize Anomaly Score array
for all y in x do
    PathLenghts = initialize Path Lenghts array
    for all iT in T do
        PathLenghts = Append(PathLenghts, pathLenght(y,iT,0))
    end for
    AnomalyScore = Append(AnomalyScore, Compute Anomaly Score (PathLenghts)){Use equation 12}
end for
return AnomalySore
```

4. Daphne integration

The main objective of this project is to integrate these algorithms into one tool which is easy to use, that provides the user an understanding on why the data is classified as anomalous, and aids the diagnose of these anomalies. The integration has been done in *JavaScript* and in *Python*, the logic has been integrated in Python using the Django framework, while all the user interface was done using the *Vue* framework based on *JavaScript*.

4.1 User Interface

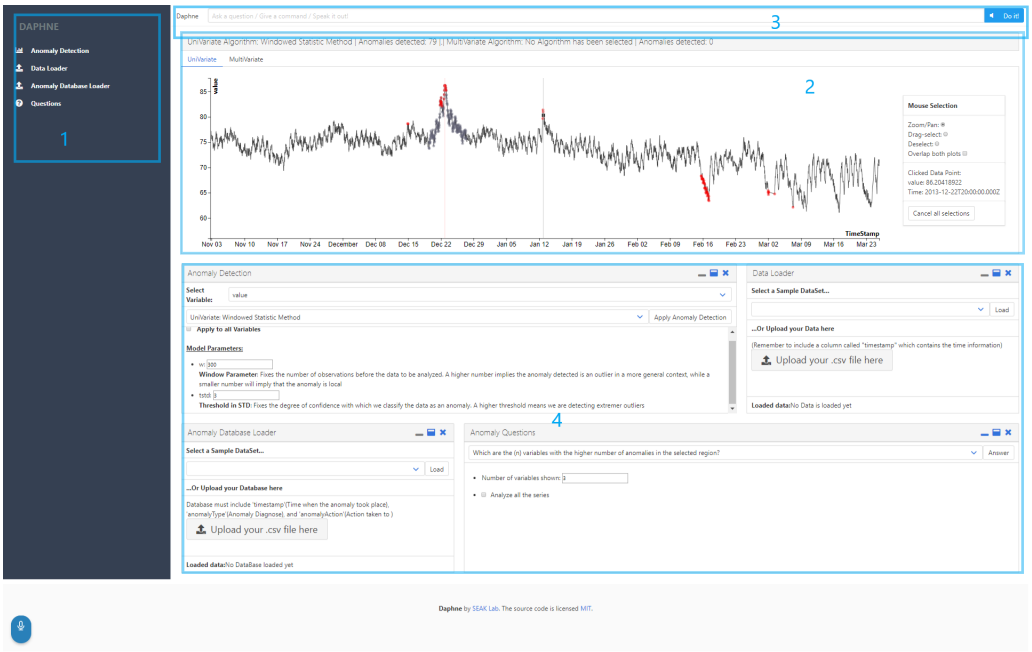


Figure 6: DAPHNE Anomaly Detection Feature Interface. 1 **User Menu**, 2 **Anomaly Plot**, 3 **Question Bar** and 4 **Functionalities**

The user interface uses Daphne's [15] design and disposition as a template. The interface is divided in four parts that can be seen in Figure 6 which are:

- **1.** User Menu: Displays the available features
- **2.** Anomaly Plot: Graphically represents the data and the anomalies detected, it also allows the user to select different regions or to obtain information about a concrete instance of the data.
- **3.** Question Bar: Allow the user to ask questions to Daphne, either by writing it in the bar, or by speaking it to a microphone.

- **4.** Functionalities: Allows interaction with the data, user can use these to upload data, perform anomaly detection algorithms or ask questions too.

4.1.1 User Menu

The user menu (Figure 7) consists on a list with the functionalities available to the user. These will be shown in white color when displayed in the functionalities section, and in grey when hidden. To show a hidden functionality the user must only click on it. To hide one, it must be done by closing it directly at the functionality section.

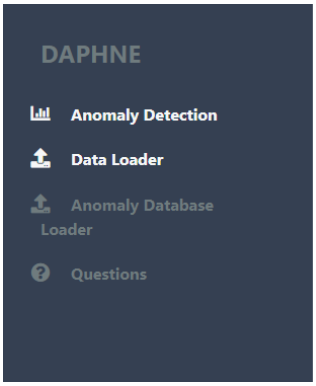


Figure 7: User Menu. Highlighted in white color appear the functionalities displayed.

4.1.2 Anomaly Plot

The Anomaly Plot displays the data selected at the moment by the user. It also allows the user to select data to feed specific operations on the data, and to click and display information of a specific datapoint. The main components of the Anomaly plot are listed below:

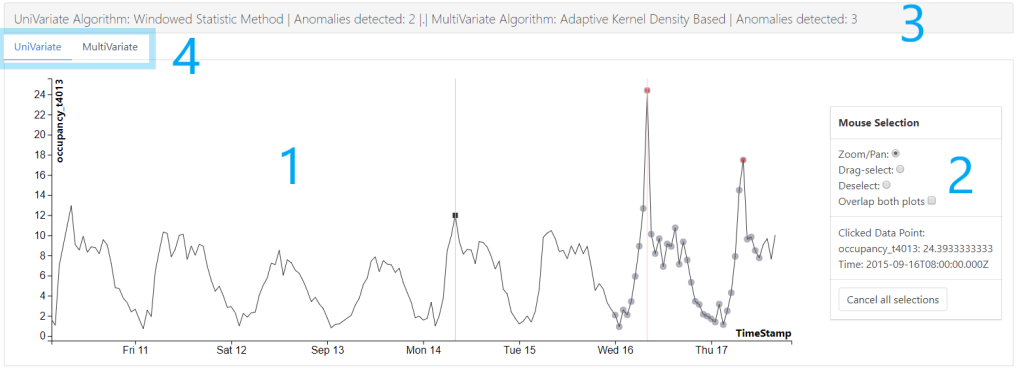


Figure 8: Anomaly Plot Sample. The elements highlighted are: 1 **Series plot**, 2 **options menu**, 3 **information bar** and 4 **selection tabs**

- The **series plot** is the main content of the plot. It contains a series plot in which are represented either the selected variable, the anomaly score of a Multivariate method or both.

  The selected variable will be represented with a black color line, and the axis with the scale will be located on the left side. The Anomaly Score series is plotted in dark blue, and the axis and scale are both on the right side. To alternate between both plots two **selection tabs** were implemented in the top left of the plot, by changing the active tab, the plotted series is alternated. In order to plot both series, a check box must be activated in the options menu.

  In the series plot the anomalies are represented as red dots for univariate anomalies and as purple dots for multivariate anomalies. The selected points are also represented as dots, but these are bigger and grey.

  In order to provide some feedback of the interaction of the user a cursor is implemented as a vertical grey line for the hovered points with a square on the data point. Also when an observation is clicked this is selected an displayed in the options menu.

- The **options menu** is located on the right side of the plot. Its main function is to enable the user to change the mouse mode, to overlap the both plots, and display information. The user can also cancel all the selected and clicked points by clicking on the "*Cancel all Selections*" button.

  There are three modes available, the zoom and pan mode, the select mode and the deselect mode. The select and deselect mode work by simply dragging the mouse over the points that the user wants to select.

- The **information bar** is located on the top part of the plot, and displays relevant information about the detection algorithms and the anomalies detected. Both the univariate and multivariate selected algorithms are displayed on the bar, and also the number of the detected anomalies by each method. This methods are the ones whose anomalies are displayed at the plot.

4.1.3 Question Bar

The question bar is inherited from the original interface from Daphne [15]. This object continues in development, its objective is to be able to introduce directly the questions available for Daphne in this input box, either by writing directly the request on the question bar, or by speaking it to the microphone.

4.1.4 Functionalities

Functionalities are displayed in the windows below the anomaly plot. These can be arranged, widen or narrowed by the user for its convenience, They can also be removed by clicking on the "x" button. The following functionalities have been implemented:

- **Anomaly detection**: Through this functionality the user can execute the different algorithms implemented, and choose the variables to be analyzed.

  There are two list fields, one to select a variable and another to select an anomaly detection method.

The variable selected is the variable that will be considered to any analysis that will have as an objective a single variable. This will also be the one displayed in the anomaly plot.

All the algorithms are presented in the methods list and are preceded by the Multivariate or Univariate word in order to specify which kind of method is chosen. Next to this field there is a button that runs the selected algorithms with the parameters chosen.

The parameters of the chosen method are displayed below with input boxes, where the parameters can be introduced. Also in the particular case of the Univariate methods there is a checkbox available in order to perform the method to all variables at once.

- **Data Uploader** and **Database Uploader**: Both functionalities have the same layout. They have a list where a sample dataset/database can be chosen and an upload button to upload any dataset/database. At the bottom, it is indicated whether any file has been loaded, and if the file is a sample dataset or one uploaded by the user.
- **Anomaly Questions**: It contains all the questions that can be asked to Daphne and that are detailed in sections 4.2, 4.3 and 4.4

  A list displays all the questions the user can ask. Next to the list there is a button to answer the question chosen. Below, once a question has been selected, different parameters are shown to set the question. Once the questions are answered the response is displayed in this space

4.2 Algorithms questions

One of the objectives of the tool is to provide information of the nature of the anomalies, whether they are triggered by particular variables, or how anomalous are compared to other instances. To do this analysis the following questions are presented.

4.2.1 Anomalies Spotter

Although the multivariate anomaly detection methods output an anomaly score, the anomalies are not automatically classified. Instead, a threshold t must be fixed to set the limit from which a data point is considered or not as anomalous.

To classify the anomalies two possibilities have been implemented:

- Directly set the threshold t: The points x_i that have an anomaly score that verify $AS(x_i) > t$ are categorized as anomalies.
- Set the number of detected anomalies to n: The n instances with the highest anomaly score are classified as anomalies.

In both cases the output of the request consist on the anomalies detected directly marked on the multivariate chart, and a written response that consists on the number of anomalies spotted and the threshold set. An example of the output can be seen in Figure 12

This method was also implemented with the possibility to add the selection feature, allowing to classify as anomalies only the data which has been previously selected. This can be particularly useful while monitoring a concrete region of the data where the user wants to extract the most anomalous points.

4.2.2 Agreement Between algorithms

Different algorithms use different properties from the data in order to classify the points as anomalous. The request presented in this section allows the user to compare two algorithms by checking if the anomalies detected with one algorithm corresponds to the anomalies detected with another.

The comparison between algorithms can be done with univariate or multivariate algorithms. When an univariate method is selected for comparison, a concrete variable must be selected too. Checking agreement between the anomalies detected for a concrete variable, and the anomalies detected for the whole system, can be interesting when trying to determine which are the variables triggering the anomalies.

The agreement between two methods A and B computed by checking which are the anomalies that coincide and the anomalies that are related:

Definition 4.1. Given an anomaly detected by method A: $x_A = (t_A, y_A)$, and an anomaly detected by method B: $x_B = (t_B, y_B)$, the anomalies **coincide** if they occur at the same time: $t_A = t_B$

Definition 4.2. Given an anomaly detected by method A: $x_A = (t_A, y_A)$, and an anomaly detected by method B: $x_B = (t_B, y_B)$ and a time margin $tm > 0$, the anomalies **are related** if they occur within the time margin : $|t_A - t_B| < tm$

The objective of the relation stated on definition 4.1 is clear, to asses if both method detect the same anomalies. If this happens it provides a higher confidence in considering the data an anomaly. Related anomalies help the user by identifying anomalous regions of the data, and also when comparing the anomalies from one variable, and a multivariate method, by helping to infer if the variable is triggering the anomaly in the system, or vice versa.

The output of the request consists on an agreement metric, which is expressed in %. The premises taken to do this computation are:

- If an anomaly detected by a method A **coincides** with an anomaly detected by method B, there is a 100% of agreement from method A, with method B, in the mentioned anomaly.
- If an anomaly detected by a method A **is related** with an anomaly detected by method B, there is a 50% of agreement from method A, with method B, in the mentioned anomaly.

Three types of agreement metrics can be computed, a general metric, and one for each direction of relationship. It must be recalled that definition 4.1 implies definition 4.2, as a result the agreement metrics can be computed as:

- Agreement of method A to method B: $AM_{A \to B} = \frac{R_{A \to B} + C_{A \to B}}{2N_A}$

- Agreement of method B to method A: $AM_{B\to A} = \frac{R_{B\to A}+C_{B\to A}}{2N_B}$
- General agreement of methods A and B: $AM_T = \frac{R_{A\to B}+R_{B\to A}+C_{A\to B}+C_{B\to A}}{2(N_A+N_B)}$

In the equations above, the terms $R_{P\to Q}$ represent the number anomalies detected by method P that are related to an anomaly detected by method Q, $C_{P\to Q}$ the number of anomalies detected by method P that coincide with method an anomaly detected by Q, and N_P, the total number of anomalies detected by method P.

The response of the data consists on a written answer which shows the agreement metrics, the number of the anomalies detected by each of the methods compared, and the number of anomalies that are related and that coincide with the other method. Examples of this can be seen in Figures 13 and 17.

4.2.3 Anomaly isolation

The next feature presented aims to help diagnose which are the anomalous variables in the system. The request is quite simple, it counts the number of anomalies in an specific region or in the whole data, and for all the variables and displays the ones with the greater number of anomalies.

This might result particularly useful in datasets with a big number of variables, it can help to show which are the anomalous variables in a region characterized as anomalous by a multivariate method.

4.3 Data Related questions

The questions developed in this sections refer to intrinsic properties of the data, in particular, correlation between variables and seasonality of a particular variable. The objective of the functionalities detailed on this section is allowing the user to get a quick and simple understanding of the structure and relations of the data. This knowledge is useful when deciding the method used to detect anomalies, or when choosing the parameters of the model used.

4.3.1 Correlation

Two definitions of correlations are available in Daphne, Pearson correlation and Spearman correlation. Both correlation types are computed between variables at the moment the data is loaded, and can be accessed directly when requested. The calculation is done quite simply by applying the mathematical formula of both correlations[53]:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{13}$$

$$(r_s)_{X,Y} = \rho_{rg_X, rg_Y} = 1 - \frac{6\sum d_i^2}{n(n^2-1))} \tag{14}$$

In equation (13), $\rho_{X,Y}$ refers to Pearson correlation between variables X and Y, $cov(X, Y)$ to the co-variance between both variables and σ_X and σ_Y to the variables standard deviations. Co-variance is further developed in its definition, in which $E()$ refers to the expectation and μ_X and μ_Y are the means of the variables.

Spearman Correlation, is defined in equation (14) as $(r_s)_{X,Y}$, as Pearson correlation of the rank variables (ρ_{rg_X,rg_Y}). The correlation is further developed and resulting in the right hand side of the equation, where $d_i = rg_{x_i} - rg_{y_i}$, i.e. the difference between the ranks of each observation, and n is the total number of observations.

The main differences between these two types of correlation is the type of relationship that they asses. Pearson correlation indicates whether there is a lineal relationship between variables X and Y. This can be particularly useful when deciding the algorithm applied, for example for highly correlated variables, it might be interesting to use principal components based anomaly detection (Section 6.1), or if two variables are highly correlated, this might mean that it might be interesting to include on in the ARIMA model of the other as an exogenous variable (Section 3.2.2). Spearman Correlation, only reflects if there is a monotonic relation between the variables analyzed, although this relation might be not as useful to choose an algorithm or fit its parameters, it provides information to the user and helps diagnosing the anomalous variables.

The questions related to correlation that the user can ask are two:

- Show the k most correlated variables
- Show the k most correlated variables to variable q

The the difference between both queries is that in the first case all the pairs of variables are taken into consideration, while in the second case only the pairs where variable q is one of the variables are considered. An example of an answer is shown in Figure 9

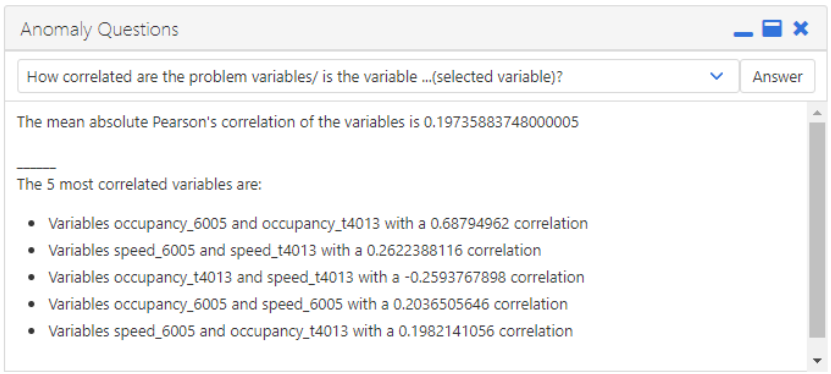


Figure 9: Daphne provides the average of the absolute value of the correlations computed, and shows the most correlated pairs of variables and the sign of the correlation

4.3.2 Seasonality

The notion of seasonality used in this section is based on the developed in subsection 3.2.2, as the main objective of this question is to be able to adequate choose the seasonality parameter in a SARIMA model.

To determine seasonality auto correlation function (ACF) is analyzed [54]. The ACF is defined as follows:

$$ACF(t) = \frac{E[(X - \mu)(L^t - \mu)]}{\sigma^2} \tag{15}$$

The previous equation defines the ACF given a fixed lag t. The terms that appear the equation, μ and σ^2, correspond respectively to the mean and variance of variable X. In theory, a process in which seasonality is present, the seasonality term appears as a single spike on the ACF, in practice, a local extrem appears in the function, this can be seen in Figure 10.

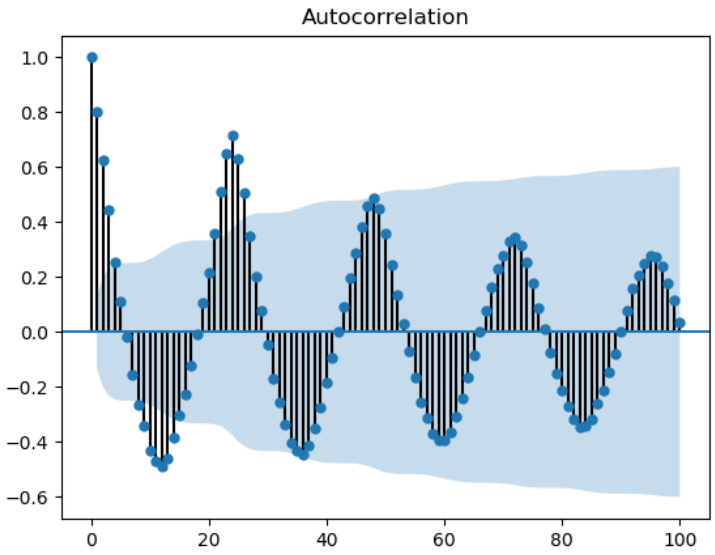


Figure 10: Sample of a ACF plot: The local maximums at lags 12, 24, 36, ... suggest a 12 lag seasonality

The method implemented to asses if there is seasonality in the data, and consists on analyzing the ACF searching local extreme. The peaks of the function are classified according to their relative amplitude, if the amplitude of the peak is higher than 0.25 times the lag analyzed, the lag is showed as possible for being a seasonality corresponding to it and if it is higher than 0.40 times is displayed as a lag with high probability of containing a seasonality. It must be also noted that is common, when there is seasonality, to find the multiples of the values also as likely seasonalities. This lags actually reinforce the idea of the first one having a seasonality.

The seasonality question analyzes the lags until a maximum fixed by the user. The respond consists on a list of seasonalities ordered by likelyhood. An example of response is found in Figure 11

4.4 Anomaly Diagnose

Anomaly detection aside, the main objective of Daphne is to help the user understand the anomalies detected by the implemented algorithms. To do so an anomaly diagnose feature has been implemented. This feature is based on a database that the user must provide. It works by comparing the anomalies detected to the anomalies present on the data base and returns a possible diagnose that helps the user to understand what is happening, and gives a path of actuation in order to fix the anomaly

4.4.1 Anomaly Database

A database must be provided to Daphne in order to infer from it a consistent diagnose. The current algorithm needs to be fed a data set of anomalies in which the appear the value of the variables when the anomaly took place, the date and time when the it took place in a the *timestamp* column, the diagnose of the anomaly in a column denominated *anomalyType* and the action taken on the anomaly in an *anomalyAction* column.

This database is uploaded through a functionality called *Anomaly Database Loader*, which extracts the information to feed the diagnose algorithm. The anomaly, database, with size l will be denoted as $ADB = \{a_1, a_2, ..., a_l\}$

4.4.2 Diagnose algorithm

In order to classify the anomalies the algorithm simply finds the anomaly that is most similar to the novel anomaly. To do so, the following steps are followed:

- Normalization of the data: both the data from the anomaly database, and the detected anomalies are normalized. The normalization is done attending to the problem data. Both mean μ and standard deviation σ are extracted from the data, after, the data is normalized: $\hat{x} = \frac{x-\mu}{\sigma}$
- Distance calculation: for every novel anomaly na the distance to all the anomalies of the database is computed $d(na, a_i)$
- Nearest neighbor selection: the anomaly a_j with minimum distance $d(na, a_j) = \min_{a_i \in ADB} d(na, a_i)$, is selected and taken as reference for the novel anomaly.
- Diagnose: once a reference anomaly is selected, the novel anomaly is classified attending to the type from a_j, and the action suggested is also the one that corresponds to the reference anomaly.

The implementation also considers the case when the variables of the data provided and the anomaly database don't coincide, in this case the algorithm is executed taking into account only the common

variables, but shows a warning to the user. Once the algorithm has obtained the most similar anomaly on the database Daphne responds as shown in Figure 18

5. Practical examples

Two datasets were used to test Daphne's anomaly detection tool. The fist was extracted from the Numenta Anomaly Benchmark [44], and the other cosists on real data from the battery system of a cubesat [55]

5.1 Traffic data

This sample dataset contains real traffic data from the Twin Cities Metro area in Minnesota. The original dataset contains 7 variables, but, due to incomplete data, only the variables that represent occupancy and speed have been chosen to the analysis. Also data have been re sampled to one hour frequency due to inconsistent time steps in the original data. As a result the data analyzed consists on a time series of 5 variables, 2 occupancy variables and 3 speed variables, with a sampling frequency of one hour.

The fist step to the analysis consist on run the anomaly detection methods. Both univariate methods were executed to all variables with the same parameters. In the case of the Windowed Statistics the window parameter was fixed to $w = 100$. This number has been chosen intuitively in order to provide information from the previous 4 days. The SARIMA based algorithm was executed taking into account a 24 hour seasonality, this can be induced from an a priori knowledge of the data (the traffic patterns show a daily seasonality), or asking daphne about it. When this question is asked to any of the variables on the dataset, a similar response to the one displayed in Figure 11 is obtained, always with high probability in lag 24, as expected.

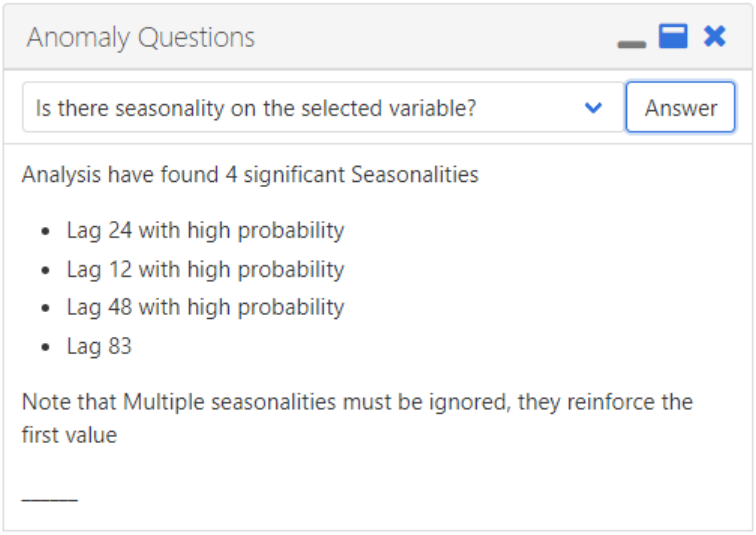


Figure 11: Answer about seasonality of variable *occupancy_t4013*. The seasonalities are displayed from most importance to less. Having 48 as a probable value reinfoces the idea that there is a seasonality in lag 24. A high probability in lag 12 also makes sense, the acf plot that corresponds to this variable is the one in Figure 10, that shows that 12 is a value to consider.

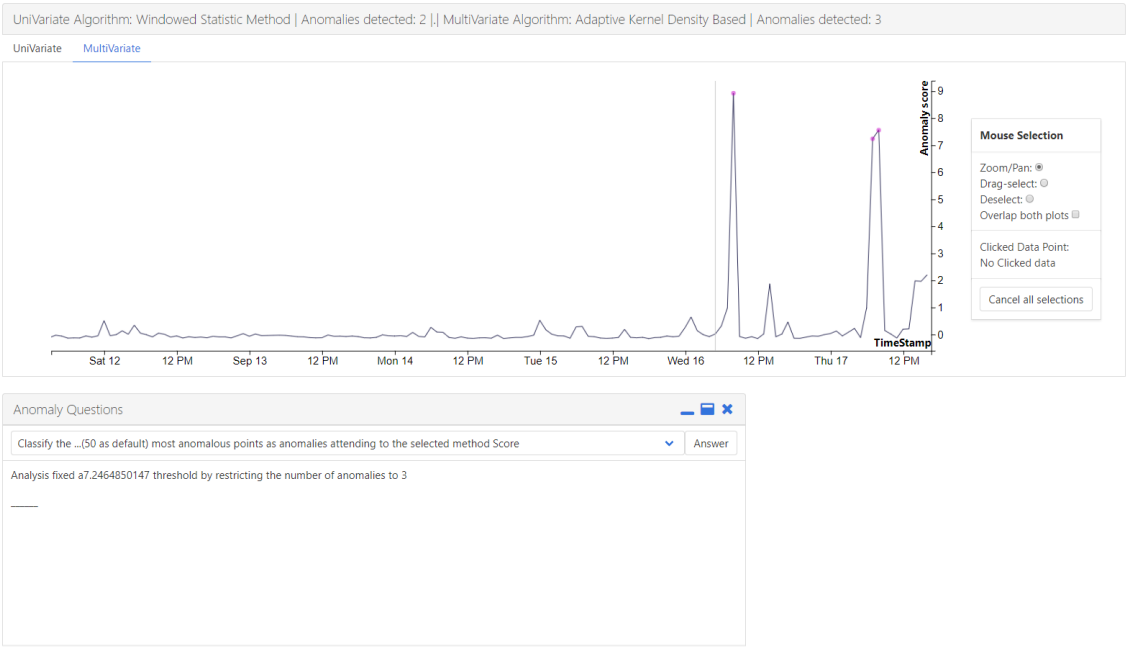


Figure 12: Adaptive Kernel Density Based method's Anomaly Score. When Daphne is asked to classify the tree most anomalous points, a threshold is returned, and the points classified as anomalies are marked in the anomaly plot with a purple dot

Multivariate methods were executed by keeping the default parameters, except in the case of the isolation forest, in which the sampling parameters was modified to consider enough training points. When analyzing the anomaly score of the Adaptive Kernel Density Based method, three clear anomalies are spotted (Figure 12). Because of this, Daphne was asked to classify the three most anomalous points using the score of the multivariate methods for both algorithms.

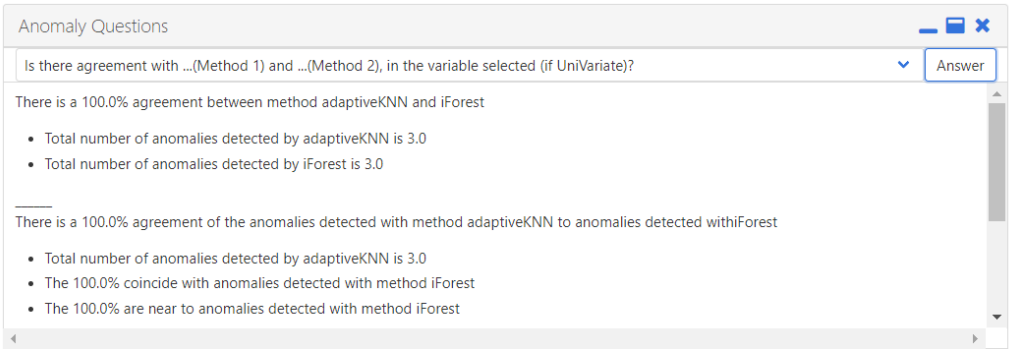


Figure 13: When both methods are asked to classify the 3 most anomalous points there is a correspondence of 100%, as the anomalies detected are the same. All the anomalies are marked as coincident and as related according to definitions 4.1 and 4.2, respectively

Also Daphne was asked whether there is agreement between both multivariate methods. The response returned (Figure 13) shows there is complete agreement between the methods, providing higher certainty about these points being anomalies.

Once there is certainty enough that the marked data points are in fact anomalies, Daphne can be used to spot the variables that trigger, or are related to these anomalies. In particular, when the points in the surrounding the anomalies are selected, the user can ask which are the variables with the higher number of spotted anomalies on the selection (Figure 14). Once these variables are known the plot of the variable and the anomaly score can be overlapped to understand the behaviour of the system.

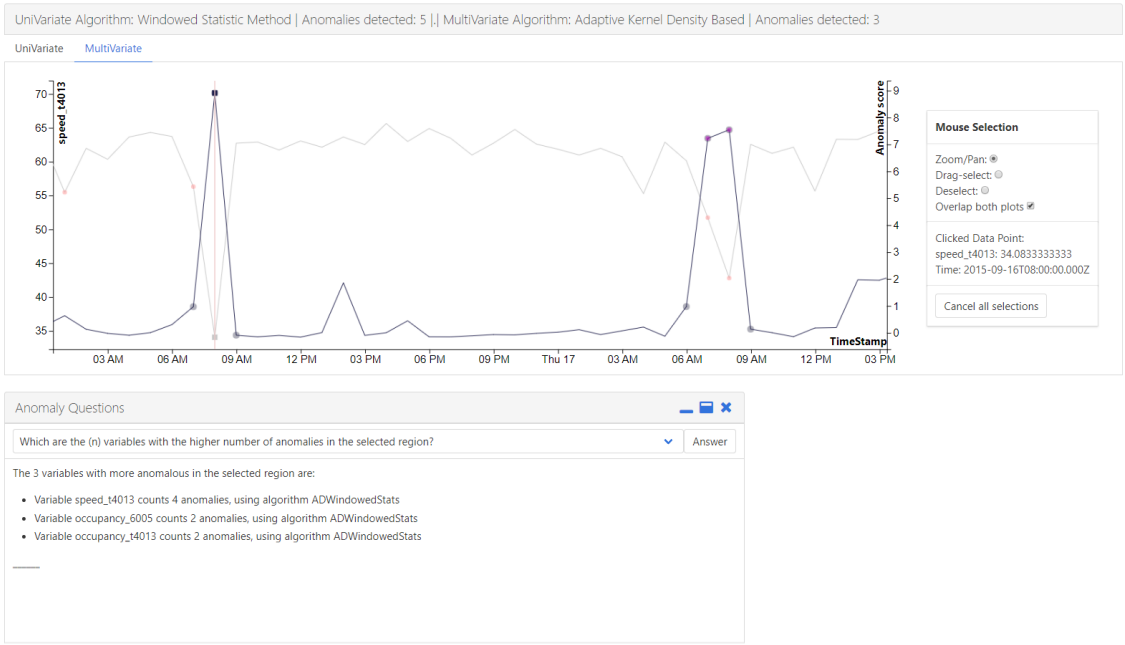
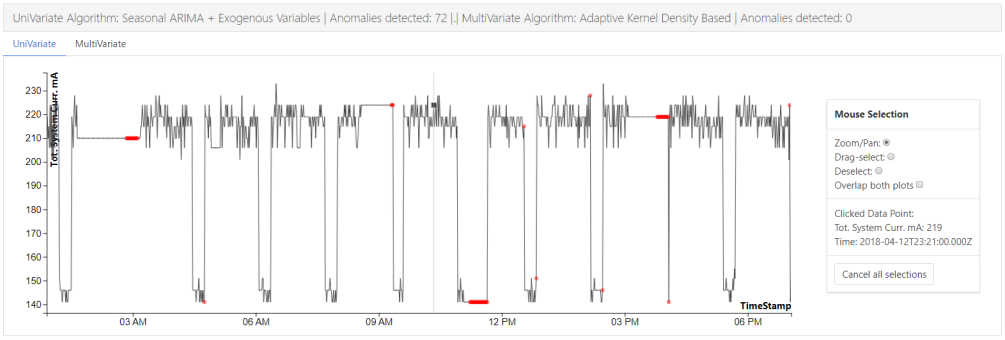


Figure 14: When asked to show the most anomalous variables, Daphne returns the number of anomalies spotted in the selection, as well as the variable, and the anomaly detection method associated to the anomalies. If the *speed_t4013* is displayed, it is clear that the anomalies spotted are driven by this variable

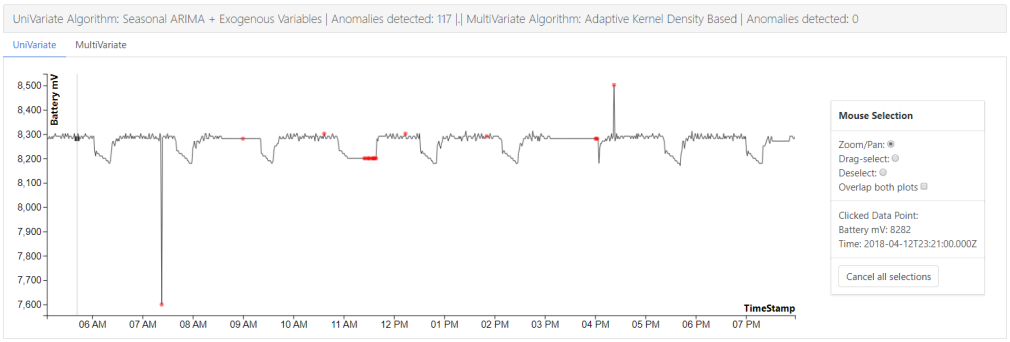
This procedure is an example of how, Daphne can be used to understand the behaviour of the data, spot the anomalies on the data, and understand the behaviour of the system in this anomalies.

5.2 Satellite data

The original housekeeping data from the cubesat contained 14 variables, but, in order to simplyfy the analysis of the data only three of the variables were kept. The variables discharded had to do with the orientation of different parts of the satellite, such as the solar panels or the chasis. The battery level (mV), photo voltaic current (mA) and total System current (mA), were the remaining variables considered.



(a) SARIMA Method applied to total system current



(b) SARIMA Method applied to battery

Figure 15: Both methods detect anomalies o the data when this constant steps don't adjust to the seasonal data pattern. In the case of the battery 15a also the introduced anomalies are detected, due to the seasonal model, the these anomalies induce false alarms

As the datasets available on the internet correspond to normal functioning of the satellite mostly didn't contained anomalous data. Analyzing the data some irregularities might be spotted at first glance, some sections of the data appear suspiciously constant, which might indicate overflow of data. This anomalies are not detected by the Windowed Statistic methodology, but can be detected using a SARIMA based algorithm (Figure 15).

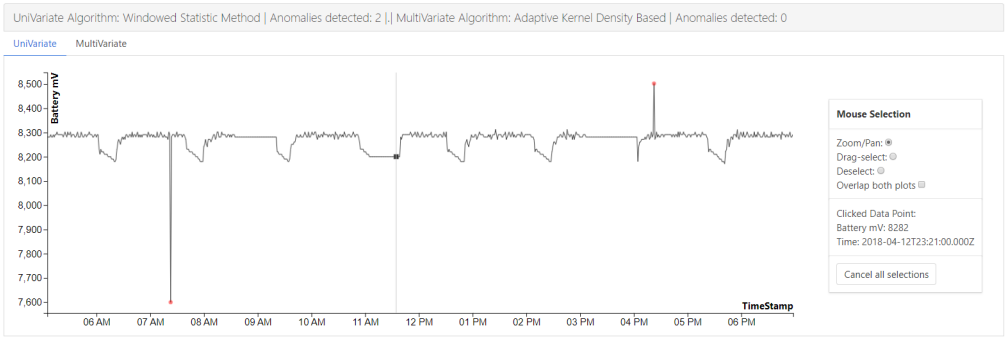


Figure 16: Windowed Statistic method applied to Battery variable. Using this method, only the introduced anomalies are detected, but not the data overflow anomalies.

Although these overflow data anomalies are detected, the values do not correspond to anomalous behaviour of the system. In order to test the diagnose function of Daphne, two anomalies were introduced, by setting anomalous values at the battery variable, one corresponding to a low battery level and another corresponding to an overcharge on the batteries.

The anomalies were successfully detected by the Windowed Statistic (Figure 16), SARIMA based (Figure 15b), and appear as peaks in the anomaly score of the Kernel Adaptive Density Based Algorithm (Figure 17)

Because of the coincidence of a lot of the values of the data due to low resolution, the performance of the iForest algorithm is quite poor. When the 2 most anomalous points according to the iForest are compared with the 2 according to the Adaptive Kernel Density Based, only one anomaly coincides (Figure 13).

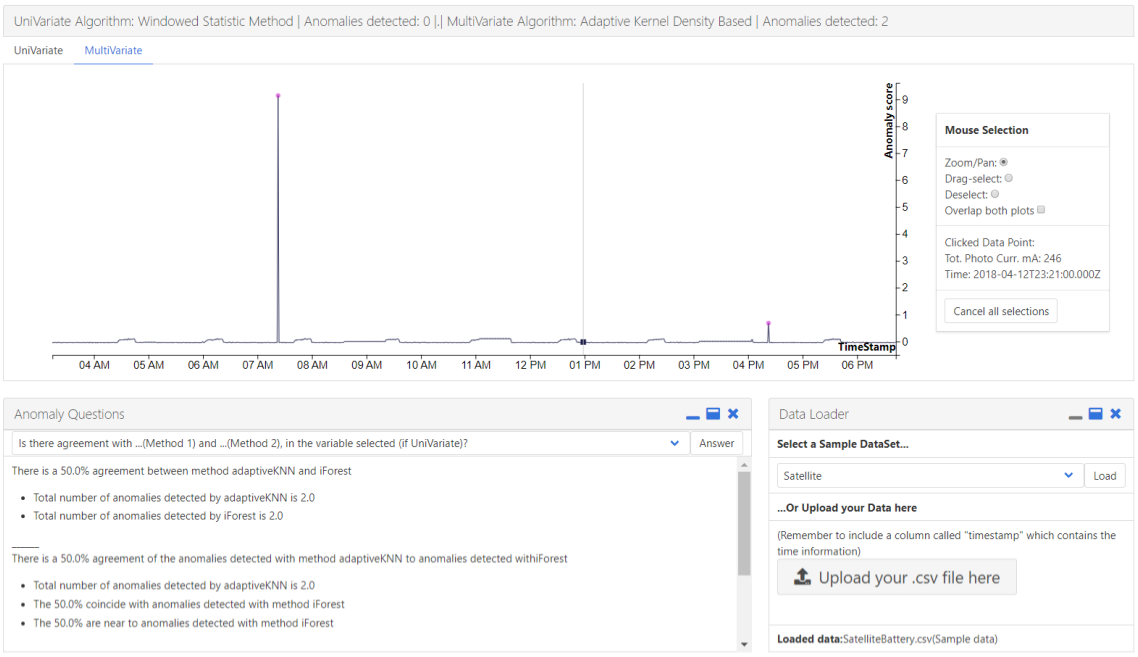


Figure 17: Adaptive Kernel Density Based Algorithm Anomaly scores and Agreement with iForest. Two spikes appear on the anomaly score, corresponding to the anomalies introduced.

To test the anomaly diagnose feature a sample database was uploaded, containing anomalies corresponding to low battery levels and overcharge of the battery. These anomalies are compared with the detected ones to diagnose them. The response of this diagnose is shown in Figure 18.

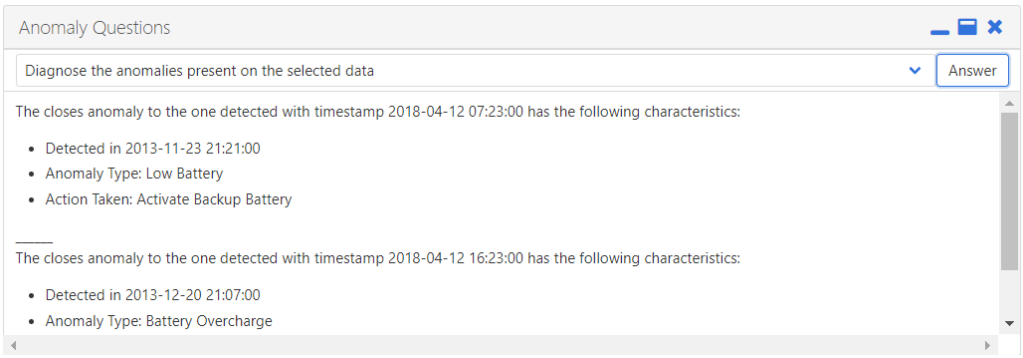


Figure 18: Anomaly Diagnose Response. A Nearest Neighbors algorithm matches the novel anomalies with the ones on the database, and similar anomalies are displayed to the user.

The procedure followed shows how can inconsistencies in seasonality can be found, and how the anomaly diagnose feature results useful into understanding the anomalies, and taking decisions regarding the actuation that must be followed

6. Future Work

The current version of DAPHNE's Anomaly Detection feature is the first step in building a versatile tool to detect irregularities on any kind of data. This section is dedicated to present the possibilities of improvement for the current implementation. This might be done by adding new anomaly detection algorithms specific for other kinds of anomalies, by improving current functionalities such as the Anomaly Diagnose, or by adapting the tool for new needs such as Online Anomaly detection.

6.1 Algorithm additions

Daphne's anomaly detection tool mainly relies on unsupervised algorithms in order to detect anomalies on the input data. Deepening in this kind of methods, other algorithms might be useful to implement, in order to address different problems in a more specific way.

When dealing with systems in which accumulative effects are relevant, is useful to use anomaly metrics as the cumulative sum (CUSUM), that do not focus that much on the deviation of the data from normal instances, but on the sum of these deviations over time. They can be applied to computer traffic data to detect flooding attacks [56], or to monitor power systems [10]. When the system variables are highly correlated, this CUSUM approach, can be performed together with Principal Components Analysis (PCA). This has been successfully applied to early detect the collapse of a hospital's emergency department [57].

Another interesting direction to improve Daphne's anomaly detection tool might be to consider the temporal dimension on multivariate methods. The current implemented approaches in Daphne, only consider the general distribution of the data, but no the temporal scope. To do so, Dynamic Time Warping (DTW) similarity measure [58] might be interesting to introduce with anomaly detection techniques [33]. This measure evaluates the similarity in shape between two sequences, and can be applied to multivariate sequences. DTW has been already applied to detect anomalies in Satellite Telemetry Data [59].

Numenta's Anomaly detection tool [18] is based in HTM networks that mimic the processes that take place in the human brain [60]. The algorithm developed for anomaly detection [20], has been proven to significantly overpass other state of the art algorithms [61]. This algorithm is developed for univariate series, and it can be interesting to add it to the list of implemented methods.

6.2 Anomalies Diagnose

The current approach consists on an empirical model, also called data driven models, these do not depend on specific introduced expert knowledge, but, instead only on the patterns of the data provided [62]. The anomaly diagnose algorithm implemented is quite simplistic and may induce error, as it only relays on the nearest neighbors of the novel anomaly.

In order to improve the reliability of the model more complex classification algorithms might be interesting to implement. Combining Kernel Fisher Discriminant Analysis and the k-nearest neighbors can be used to improve the performance of fault diagnosis [63]. Also, using associated rules, might be an

interesting upgrade, as it has successfully been used in a problem similar to the one presented in Section 5.2 [9].

In contrast to empirical models, mechanistic models are applied to systems for which the behaviour is well known. In this case there is many literature regarding fault diagnose [64], given a process model Kalman filters [7] and Particle filters [65] frameworks can be applied with high reliability. Although the precision and effectiveness of these methods is quite decent, their approach is very domain specific, and, as a result, the integration of these frameworks in Daphne might not be adequeate.

6.3 Online integration

Online detection is an important field of research, with many different algorithms proposed for this task [66]. Many systems rely on these algorithms to early detect critical events, like intrusions [5] or system faults [67]. The main requirement for an online Anomaly Detection algorithms, is to have low computational complexity, in order to be able to keep up with the data stream.

Algorithms 2, 3, and 6 can be easily transferred to perform in an online fashion, under a reasonable data stream. All methods can be trained independently, and, once trained, the computational complexity needed to evaluate incoming data is assumable for moderated streams of data.

Specific algorithms are designed to assimilate Big Data Streams [68]. In the particular case of statistical methods, like algorithm 1, these can also be easily transfer to an online version for larges amounts of data [42], due to the low complexity.

Also, other online algorithms might be also interesting to implement in the future, like HTM based [19] (already mentioned in section 6.1), that are considered to have high performance, or algorithms to address specific requirements like small training samples [69].

7. Conclusions

The anomaly detection problem can be complicated due to the different definitions of anomaly. Depending on the requirements of problem presented, different statistics must be considered, and even when it is clear which are the ones to be monitored, the concept of anomaly can be very subjective. As a result, often, an individual analysis must be done.

By presenting the analyzed cases, it is shown how the tool can be used to monitor variables, detect and diagnose unexpected behaviours. In particular, tools like the anomaly diagnose, anomaly isolation questions or agreement questions, allow the user to understand with a more high level perspective of what is happening. In order to clear part of this subjectivity from the anomalies, the interface helps the user in understanding the data by extracting properties of the data, such as seasonality or correlation between variables.

The current implementation still has a long path to become an useful tool. Just a few of detection algorithms have been implemented, but in order to try to cover a broader scope of this problem variety, other algorithms are already considered to be implemented in the future. Because of the way the tool is programmed, adding new methodologies for anomaly detection can be done easily. And this being an open source project, allows it to become used by anyone, and easily developed by other collaborators.

References

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, 2009.

[2] D. M. Hawkins. *Identification of Outliers*. 1980.

[3] Yufeng Kou, Chang-tien T Lu, S Sirwongwattana, Y P Huang, and Sirirat Sinvongwattana. Survey of fraud detection techniques. *Networking Sensing and Control 2004 IEEE International Conference on*, 2(3):749–754, 2004.

[4] Divya Iyer, Arti Mohanpurkar, Sneha Janardhan, Dhanashree Rathod, and Amruta Sardeshmukh. Credit Card Fraud Detection Using Hidden Markov Model. *Ieee Transactions On Dependable And Secure Computing*, 5(1):1062–1066, 2011.

[5] Wei Wang, Thomas Guyet, René Quiniou, Marie-Odile Cordier, Florent Masseglia, and Xiangliang Zhang. Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks. *Knowledge-Based Systems*, 70:103–117, nov 2014.

[6] Xiaohang Jin, Yi Sun, Zijun Que, Yu Wang, and Tommy W.S. Chow. Anomaly detection and fault prognosis for bearings. *IEEE Transactions on Instrumentation and Measurement*, 65(9):2046–2054, 2016.

[7] Afshin Rahimi, Krishna Dev Kumar, and Hekmat Alighanbari. Fault estimation of satellite reaction wheels using covariance based adaptive unscented Kalman filter. *Acta Astronautica*, 134(February):159–169, 2017.

[8] Xueqin Chen and Ming Liu. A two-stage extended kalman filter method for fault estimation of satellite attitude control systems. *Journal of the Franklin Institute*, 354(2):872–886, 2017.

[9] Dawei Pan, Datong Liu, Jun Zhou, and Guoyong Zhang. Anomaly detection for satellite power subsystem with associated rules based on Kernel Principal Component Analysis, 2015.

[10] Li Yuqing, Yang Tianshe, Cheng Xueliang, Wang Rixin, and Xu Minqiang. An Anomaly Detection Algorithm of Satellite Power System Based on CUSUM Control Chart. *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pages 829–833, 2016.

[11] Hermine N. Akouemo and Richard J. Povinelli. Probabilistic anomaly detection in natural gas time series data. *International Journal of Forecasting*, 32(3):948–956, 2016.

[12] Luís M. A. Bettencourt, Ruy M Ribeiro, Gerardo Chowell, Timothy Lant, and Carlos Castillo-Chavez. Towards Real Time Epidemiology: Data Assimilation, Modeling and Anomaly Detection of Health Surveillance Data Streams. In *Intelligence and Security Informatics: Biosurveillance*, pages 79–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[13] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1975–1981, 2010.

[14] L Kratz and K Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. *IEEE Conference on Computer Vision and Pattern Recognition (2009)*, pages 1446–1453, 2009.

[15] Harris Bang, Antoni Viros, Arnau Prat, and Daniel Selva. Daphne : An Intelligent Assistant for Architecting Earth Observing Satellite Systems. *AIAA SciTech 2018*, pages 1–9, 2018.

[16] Anodot — Automated anomaly detection system and real time analytics. `https://www.anodot.com/`.

[17] CA Threat Analytics for Privileged Access Manager. `https://www.ca.com/us/products/ca-threat-analytics-for-privileged-access-manager.html`.

[18] Hierarchical Temporal Memory (HTM) Studio — Numenta. `https://numenta.com/applications/htm-studio/`.

[19] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[20] Jia Wu, Weiru Zeng, and Fei Yan. Hierarchical Temporal Memory method for time-series-based anomaly detection. *Neurocomputing*, 273:535–546, 2018.

[21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Encyclopedia of Machine Learning and Data Mining. 2016.

[22] Charu C. Aggarwal. *Outlier Analysis*. Springer New York, New York, NY, dec 2013.

[23] Nico Goernitz, Marius Micha Kloft, Konrad Rieck, and Ulf Brefeld. Toward Supervised Anomaly Detection. *Journal Of Artificial Intelligence Research*, 46:235–262, 2014.

[24] Dino Ienco, Ruggero G Pensa, and Rosa Meo. A Semisupervised Approach to the Detection and Characterization of Outliers in Categorical Data. *IEEE Transactions on Neural Networks and Learning Systems*, 28(5):1017–1029, may 2017.

[25] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier Detection for Temporal Data: A Survey. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 26(9):2250–2267, 2014.

[26] Bonnie Zhu and Shankar Sastry. Revisit dynamic ARIMA based anomaly detection. *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*, (1):1263–1268, 2011.

[27] Eduardo H.M. Pena, Marcos V.O. De Assis, and Mario Lemes Proença. Anomaly Detection Using Forecasting Methods ARIMA and HWDS. *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, pages 63–66, 2017.

[28] Qin Yu, Lyu Jibin, and Lirui Jiang. An Improved ARIMA-Based Traffic Anomaly Detection Algorithm for Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2016, 2016.

[29] David J. Hill and Barbara S. Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling and Software*, 25(9):1014–1022, 2010.

[30] Przemysław Bereziński, Bartosz Jasiul, and Marcin Szpyrka. An entropy-based network anomaly detection method. *Entropy*, 17(4):2367–2408, 2015.

[31] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: An application to sensor data. *Knowledge and Information Systems*, 11(2):137–154, 2007.

[32] Jinbo Li, Witold Pedrycz, and Iqbal Jamal. Multivariate time series anomaly detection: A framework of Hidden Markov Models. *Applied Soft Computing*, 60:229–240, 2017.

[33] Seif-Eddine Benkabou, Khalid Benabdeslem, and Bruno Canitia. Unsupervised outlier detection for time series by entropy and dynamic time warping. *Knowledge and Information Systems*, 54(2):463–486, 2017.

[34] Eamonn Keogh, Jessica Lin, and Ada Fu. HOT SAX: Finding the most Unusual Time Series Subsequences: Algorithms and Application. *Icdm*, 2005.

[35] Liangwei Zhang, Jing Lin, and Ramin Karim. Adaptive kernel density-based anomaly detection for nonlinear systems. *Knowledge-Based Systems*, 139:50–63, 2018.

[36] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):1–39, 2012.

[37] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. *Proceedings of the Twenty-eighth Australasian*, 38(January):333–342, 2005.

[38] N. Pandeeswari and Ganesh Kumar. Anomaly Detection System in Cloud Environment Using Fuzzy Clustering Based ANN. *Mobile Networks and Applications*, 21(3):494–505, jun 2016.

[39] Hossein Saeedi Emadi and Sayyed Majid Mazinani. A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks. *Wireless Personal Communications*, 98(2):1–11, 2017.

[40] V Barnett. The Ordering of Multivariate Data. *Journal of the Royal Statistical Society. Series A (General)*, 139(3):318–355, 1976.

[41] Walter Andrew Shewhart. *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[42] Chengwei Wang, Krishnamurthy Viswanathan, Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, and Karsten Schwan. Statistical techniques for online anomaly detection in data centers. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 385–392. IEEE, may 2011.

[43] Matthew Daigle, Michael Foygel, and Vadim Smelyanskiy. Model-based diagnostics for propellant loading systems. *IEEE Aerospace Conference Proceedings*, 2011.

[44] Numenta Anomaly Benchmark. `https://github.com/numenta/NAB/tree/master/data`.

[45] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[46] Javier Contreras, Rosario Espínola, Francisco J. Nogales, and Antonio J. Conejo. ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, 2003.

[47] Wen chuan Wang, Kwok wing Chau, Dong mei Xu, and Xiao Yun Chen. Improving Forecasting Accuracy of Annual Runoff Time Series Using ARIMA Based on EEMD Decomposition. *Water Resources Management*, 29(8):2655–2675, 2015.

[48] C. Narendra Babu and B. Eswara Reddy. A moving-average filter based hybrid ARIMA-ANN model for forecasting time series data. *Applied Soft Computing Journal*, 23:27–38, 2014.

[49] statsmodels.tsa.stattools.adfuller — statsmodels 0.8.0 documentation. `http://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html{#}id2`.

[50] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, dec 2008.

[51] Kai Ming Ting, S C Tan, and F T Liu. Mass: A new ranking measure for anomaly detection. *Gippsland School of Information Technology, Monash University*, (December), 2009.

[52] Preiss Bruno R. Search Trees. In *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*, page 660. Wiley, 1999.

[53] Jerome L. Myers and A. (Arnold) Well. *Research design and statistical analysis*. Lawrence Erlbaum Associates, 2nd edition, 2003.

[54] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. 2016.

[55] Whole Orbit Data. `http://warehouse.funcube.org.uk/wod.html?satelliteId=2`.

[56] Vasilios A. Siris and Fotini Papagalou. Application of anomaly detection algorithms for detecting SYN flooding attacks. *Computer Communications*, 29(9 SPEC. ISS.):1433–1442, 2006.

[57] Fouzi Harrou, Farid Kadri, Sondes Chaabane, Christian Tahon, and Ying Sun. Improved principal component analysis for anomaly detection: Application to an emergency department. *Computers and Industrial Engineering*, 88:63–77, 2015.

[58] Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[59] Datong Liu, Jingyue Pang, Ben Xu, Zan Liu, Jun Zhou, and Guoyong Zhang. Satellite Telemetry Data Anomaly Detection with Hybrid Similarity Measures. *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, (2):591–596, 2017.

[60] Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. Cortical Learning Algorithm and Hierarchical Temporal Memory. *Numenta Whitepaper*, pages 1–68, 2011.

[61] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms - The numenta anomaly benchmark. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, pages 38–44, 2016.

[62] John MacGregor and Ali Cinar. Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. *Computers and Chemical Engineering*, 47:111–120, 2012.

[63] Zhi Bo Zhu and Zhi Huan Song. A novel fault diagnosis system using pattern classification on kernel FDA subspace. *Expert Systems with Applications*, 38(6):6895–6905, 2011.

[64] Zhiwei Gao, C Cecati, and S X Ding. A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis. *IEEE Transactions On Industrial Electronics*, 62(6):3768 – 3774, 2015.

[65] M Orchard and G Vachtsevanos. A Particle Filtering Approach for On-Line Fault Diagnosis and Failure Prognosis. *Measurement And Control*, 31(3-4):1–18, 2007.

[66] Zirije Hasani. Robust Anomaly Detection Algorithms for Real-time Big Data. *2017 6th MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING*, (June):11–15, 2017.

[67] Shen Yin, Xiangping Zhu, and Student Member. Intelligent Particle Filter and Its Application to Fault Detection of Nonlinear System. *IEEE transactions on industrial electronics*, 62(6):3852–3861, 2015.

[68] Laura Rettig, Mourad Khayati, Philippe Cudre-Mauroux, and Michal Piorkowski. Online anomaly detection over Big Data streams. *2015 IEEE International Conference on Big Data (Big Data)*, pages 1113–1122, 2015.

[69] Li Dong, Shulin LIU, and Hongli ZHANG. A method of anomaly detection and fault diagnosis with online adaptive learning under small training samples. *Pattern Recognition*, 64(May 2016):374–385, 2017.