Chapter

Multivariate Real Time Series Data Using Six Unsupervised Machine Learning Algorithms

Ilan Figueirêdo, Lílian Lefol Nani Guarieiro and Erick Giovani Sperandio Nascimento

Abstract

The development of artificial intelligence (AI) algorithms for classification purpose of undesirable events has gained notoriety in the industrial world. Nevertheless, for AI algorithm training is necessary to have labeled data to identify the normal and anomalous operating conditions of the system. However, labeled data is scarce or nonexistent, as it requires a herculean effort to the specialists of labeling them. Thus, this chapter provides a comparison performance of six unsupervised Machine Learning (ML) algorithms to pattern recognition in multivariate time series data. The algorithms can identify patterns to assist in semiautomatic way the data annotating process for, subsequentially, leverage the training of AI supervised models. To verify the performance of the unsupervised ML algorithms to detect interest/anomaly pattern in real time series data, six algorithms were applied in following two identical cases (i) meteorological data from a hurricane season and (ii) monitoring data from dynamic machinery for predictive maintenance purposes. The performance evaluation was investigated with seven threshold indicators: accuracy, precision, recall, specificity, F1-Score, AUC-ROC and AUC-PRC. The results suggest that algorithms with multivariate approach can be successfully applied in the detection of anomalies in multivariate time series data.

Keywords: unsupervised learning, pattern recognition, multivariate time series, machine learning, anomaly detection

1. Introduction

Today, the industry is changing by what experts call the “Fourth Industrial Revolution”, also called Industry 4.0. This change is strongly associated with the integration between physical and digital systems through, for example, installing sensors. The integration of these environments allows the collection of a large amount of acquired data in different fields such as: industrial processes, meteorological monitoring stations, stock exchanges etc. This amount of both collected and stored data enables faster and more directed information exchange [1].

In many fields, it is essential for the process to identify unusual patterns that can be generated by unpredictable or unwanted behavior. These behaviors may be due to some problem that may be occurring in the related process, for example, in an industrial environment companies can use machine monitoring data to identify malfunction operating due its abnormal behaviors. This fact, when it is not detected in time, can generate false data and lead experts to misinterpret the operating condition of the machine. Another example would be a credit card operator who can monitor each user's transaction to look for unusual behavior that could point to fraudulent transactions. These unwanted and abnormal behaviors are often called interest patterns and can be extracted from data due a variety of reasons, all presenting a certain level of relevance to the analyst. It is important that this analysis takes into account any changes in the behavior of the parameter to identify opportunities to improve, prevent or correct any situation [2].

The detection of interest/anomaly patterns is usually carried out by specialists which comprises the dynamics of the system under analysis. However, it is often not feasible to analyze and label them due to the large volumes of data generated. Thus, there is a limitation regarding the ability of specialists to process a large amount of data, requiring many hours of work that, in general, are involved in other activities and do not have the time necessary for this relevant activity. Thereby, there is a great need to automate the process of identifying hidden interest patterns in time series data [3].

Unsupervised machine learning (ML) has been research hotspot in intelligence artificial (IA) field to extract useful features from unlabeled raw data. Instead of selecting features by a human operator, the unsupervised learning is quite intelligent and independent of specific knowledge of processing techniques and field expertise in a data-driven way. Thus, there is no escape from the requirement of labeled data to train classifiers at the phase of diagnosis problems, but it is hard to label a mass of collected data before the determination of interest patterns [4].

In an industrial environment, data collection is often carried out through multiple sensors due the possibility of a more robust representation of the phenomena involved, as example, the monitoring of industrial assets is carried out through both acquisition of vibration and temperature data of the machine. Another example is the monitoring of meteorological conditions, which generally collect data on wind speed, air humidity and ambient temperature. However, multivariate data presents a greater challenge for the application of Machine Learning (ML) algorithms, as they must be able to recognize patterns and predict behaviors in a greater amount of data and attributes to be correlated [5].

Anomaly detection algorithms seek for patterns in data that do not conform to an expected behavior. Anomaly detection is essential in industrial applications to optimize economic performance and minimize safety risks. The advent of system health monitoring methods was realized to preserve system functionality within harsh operational environments. Hence, to develop an anomaly detection model based on multi-sensor signals, three major challenges must be faced [6]:

i. online multi-sensor signals are often available in the form of complex, multivariate time series, as different sensors measure various aspects of data over fairly long periods of time, and since there is a large amount of heterogeneous data, it can become impractical to human specialists to label anomalies or unknown events within the data;

ii. especially in industrial applications, it is likely to have extremely imbalanced datasets, since far more data is obtained during normal operations rather than abnormal;

iii. the dataset may contain uncertainties and spurious data, especially when it involves manually recorded data.

One way to mitigate these problems is to perform some type of anomaly detection technique, which are usually computationally intensive algorithms, and then flag unusual patterns for further inspection by human specialists [3]. Other way of dealing with it are based on supervised machine learning anomaly detection algorithms, which require a training dataset that contains a set of instances of anomalies, and a set of instances of non-anomalous (or normal) data, at least. From the training data, the algorithm learns a model that distinguishes between the normal and the anomalous patterns. Such supervised learning algorithms typically require tens or hundreds of thousands of labeled samples to obtain good quality performance. Nevertheless, as stated before, the scarce availability of labeled data poses a challenge to the usage and application of supervised learning techniques for anomaly detection in multivariate time series data.

Hence, unsupervised learning algorithms step up as a viable and feasible alternative to tackle this challenging problem. Since they are designed to deal with unlabeled data, they are able to learn and identify interesting patterns from the data's own internal structure, meaning that they can be used to point out anomalous patterns when the labels are unknown. Thus, unsupervised machine learning models are essential to solve the addressed challenges [7].

Several works proposed the development and application of unsupervised ML algorithms over the past years to detect anomalous patterns in time series datasets, which are based on major approaches that are summarized in **Table 1**. A detailed description and evaluation of each of these approaches is beyond the scope of this chapter.

Most of the works based on unsupervised ML algorithms employs clustering techniques, which are either distance-based [10] or density-based [11]. On the one hand, the advantages of clustering methods are that they are simple, robust, and easy to program. However, the problem is the need to define the parameters related to the data observations beforehand such as defining a similarity function or the number of clusters that should exist in the data, and that becomes the responsibility of the designer to determine how these parameters should be used, even if the data has a random structure [9]. The work [12] proposed the K-means to automate diagnosis of defective rolling bearing. To overcome the sensitivity of choosing the initial clusters number, the initial centers were selected using features extracted from simulated signals. However, K-means depends mainly on distance calculation between all data points and the centers, therefore, the cost of the computational time will be higher for big data.

To reduce the time cost of K-means, [13] proposed a Fast K-means algorithm based on two stage. The first stage is a fast distance calculation using only a small fraction of the data to originate the best possible location of the centers. The second stage is a slow distance calculation in which the initial centers are taken from the first stage. Besides that, the K-means is optimized through grid search method that is efficient when the number of parameters is small.

Algorithm	Description
K-means	It is used to divide a group of data points into hard clusters. It assumes a balanced cluster size, the joint distribution of features with equal variance, and independent features with similar cluster density. Determining the optimal K can be difficult, but for small values, it is computationally fast and efficient. In addition, it is important to choose the most appropriated distance or similarity function, since it is one of the key aspects used to determine whose cluster a certain data point belongs to.
Gaussian mixture model	It uses a Gaussian distribution-based parametric model to identify the underlying populations. These can be explained by a normal distribution in the midst of many heterogeneous populations. However, in many practical situations, the data distribution may not have any explicit clusters. As a result, each point can be assigned with different weights or probabilities to soft clusters.
Random forest	It operates by constructing a multitude of decision trees at training time and outputting the class that is either the mode of the classes (classification) or the mean prediction (regression) of the individual trees. Random forest can learn arbitrary relationships between the features and the outcome, even non-monotonic relationships.
k-NN	It assigns data points according to the majority of its nearest neighbors to find anomalous data points by measuring the local deviation. A choice needs to be made on the value of *K*, i.e. the number of neighbors, to avoid overfitting/underfitting issues.
DBSCAN	It recognizes the clusters as dense regions having some coincidence that is diverse from the other sparse region. The algorithm may use a reduced number of points and a distance measure to merge the data points that are similar to each other. Moreover, DBSCAN requires two parameters to operate, which are the epsilon (*eps*) and the minimum points (*minpoints*). *Eps* determines the smallest distance existing between two points in a cluster, while *minpoints* defines the least number of points required to form a dense region [8].
PCA	Principal Component Analysis (PCA) based anomaly detection techniques are able extract the main features of a certain dataset without losing its ability to represent the original data, then using these features to analyze which constitute a normal class and applies distance metrics to identify cases that represent anomalies. This allows to train a model using existing imbalanced data.
Autoencoders	It is a neural network that attempts to reconstruct its input. Similarly to PCA, it can serve as a form of extract main features to produce a compressed representation of its input at the encoder. This representation can be mapped to its original form using a decoder. Anomaly detection uses the reconstruction error to measure how well the decoder is performing.

Adapted [9].

Table 1.
Unsupervised ML approaches found in literature for anomaly detection in time-series.

Aiming to prove the reduction quality of the dataset, [14] demonstrated the superiority of the autoencoder in the feature dimensional reduction comparing with the PCA method. The reduced feature set obtained from the PCA method revealed overlaps of classes and features that are scattered on the large space, while the autoencoder represented the superior ability in the clear and concentrated distribution.

The work [15] presented a similar comparative study to evaluate the performance of the autoencoder to the original feature set, PCA reduction and real-valued negative selection. The dimensional reduction of the autoencoder, once again, have performed a highly improved anomaly detection compared to the others. Moreover, PCA space transformation requires complete knowledge about normal and faulty data classes.

Two methods for feature selection were proposed by [15]. The first one is based on k-NN for clustering using feature similarity influence, and the second one is the pretraining using sparse autoencoders. The classification performance obtained by the k-NN algorithm is comparable to the result obtained from the autoencoder (slightly lower in accuracy). The criterion for choosing the “k” parameter is based on the combinations that frequently appear in the subsets reduced by the technique itself. In other words, the criterion adopted is purely empirical.

Furthermore, even though several unsupervised techniques have been proposed in literature, their performance depends a lot on the data and application they are being used in. This indicates that most of these methods have little systematic advantages over the other when compared across many other datasets.

In this context, this chapter discuss the level of accuracy and reliability of six unsupervised ML algorithms for pattern recognition and anomaly detection with no need of labeled data. Two real cases were applied for performance evaluation of the algorithms abilities to detect the interest patterns in the multivariate time series data. The real cases were: (i) meteorological data from a hurricane season and (ii) monitoring data from dynamic machinery for predictive maintenance purposes.

2. Unsupervised ML algorithms

This section will review the concept and application of six unsupervised ML algorithms for anomaly/pattern detection applied in this research. It is important to inform that among the six algorithms described in this chapter, only the methods in Sections 2.1 and 2.3 has its intrinsic characteristic to perform a multivariate analysis, while the other algorithms are only able to perform a univariate analysis. Therefore, a univariate performance evaluation is computed in each dimension of the data to then calculate an average performance, except algorithms in 2.1 and 2.3 sections.

2.1 C-AMDATS

The Cluster-based Algorithm for Anomaly Detection in Time Series Using Mahalanobis Distance (C-AMDATS) is a clustering ML unsupervised algorithm. The model has only two hyperparameters that user can manipulate: (i) Initial Cluster Size (ICS) and Clustering Factor (CF). First the ICS clusters the observed sequences of time series data A, where each cluster may represent a behavior status. After the initial clustering, a new and better clustering in the dataset is remade according to the data points distribution over timeline. This ability is due to the usage of the Mahalanobis distance in the algorithm. In general, clustering techniques use the Euclidean distance function, which makes the clustering assumes the geometric shape of a circle, then it does not consider the variance of each dimension or feature of the dataset. However, there are situations in which the variance between each dimension (or feature) is different. Conversely, by using the covariance matrix, the Mahalanobis distance can detect the variance of each dimension. Eq. (1) presents the Mahalanobis distance formula.

$$d_m(x,\mu)=\sqrt{(x-\mu)^T S^{-1}(x-\mu)} \tag{1}$$

Where: $d_m(x,\mu)$ is the Mahalanobis distance between a specific point in the time series and its respective centroid; $x = (x_1, x_2, \ldots, x_n)^T$ is a specific variable in the time series data, where n is the number of variables; $\mu = (\mu_1, \mu_2, \ldots, \mu_n)^T$ is a certain cluster centroid; and S is the covariance matrix relative to that cluster.

After the new clustering through the Mahalanobis distance, the algorithm calculates the similarity of each cluster in the time series A to find the respective hidden patterns P. This similarity is calculated using the standard deviation σ_y of the actual values of the A samples, the Y coordinate of each centroid and the CF. If the modulus of the difference between the y coordinate of the centroids of two cluster is less than or equal to the product of CF and σ_y, then these clusters can be merged, meaning that they will represent the same pattern P. This task is carried out until every cluster have been analyzed.

The last step of C-AMDATS is to calculate the probability of the pattern P to be an anomaly R. The Anomaly Score measures the anomaly R for each pattern P (found in the previous step). The score is calculated by the ratio of the size of the entire time series to the sum of the sizes of the clusters present in P. The anomaly score assesses the degree of relevance of P in terms of anomaly detection. Then, all set P is ordered by R in descending order, and the anomalous patterns will be those with the highest anomaly score values. The higher the anomaly score value for a pattern P, the greater probability is of being an anomaly behavior in A [2]. Eq. (2) presents the Anomaly Score formula.

$$Anomaly\ Score_{P_i} = \frac{|T|}{|P_i|} \tag{2}$$

Where $Anomaly\ Score_{P_i}$ is the anomaly score of the pattern P_i, $|P_i|$ is the size of the pattern P_i, and $|T|$ is the size of the time series T.

2.2 Luminol Bitmap

Bitmap is an available unsupervised learning algorithm in Luminol library for anomaly detection or time series correlation. The background of Bitmap algorithm is based on the idea of time series bitmaps. The logic of the algorithm is to make a feature extraction of the raw time series data - by converting them into a Symbolic Aggregate Approximation (SAX) representation - and use it to compute the information about the relative frequency of its features to color a bitmap in a principled way. SAX allows a dimensionality reduction of the raw time series C of arbitrary length n to a string arbitrary length w ($w < n$, typically $w << n$) by a vector $\bar{C}$. It transforms the data into a Piecewise Aggregate Approximations (PAA) representation to symbolize it into a discrete string [16]. Eq. (3) presents the calculation of the *ith* element of $\bar{C}$:

$$\bar{C}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} C_j \tag{3}$$

After transformed a time series dataset into PAA, the algorithm applies a further transformation to obtain a discrete representation with equiprobability [16]. The conversion of the time series into a SAX words is made by a slider window (also called feature window). Bitmap algorithm use two concatenated slider windows together across the sequence, the latter one is called lead window, showing how far to look ahead for anomalous patterns and the former one is called lag window, whose size represents how much memory of the past to remember it.

In summary, the algorithm approach is to convert both feature windows into SAX representation, then count the frequencies of SAX subwords at the desired level and get the corresponding bitmaps. The distance between the two bitmaps is measured and reported as an anomaly score at each time instance, and the bitmaps are drawn to visualize the similarities and differences between the two windows. The user must choose the length of the feature windows N and the number n of equal sized sections in which to divide N [3].

2.3 SAX-REPEAT

SAX-REPEAT algorithm is an approach that relies on extending the original SAX implementation to handle multivariable data. The algorithm takes as input a set of K multivariable time series X_i of lengths T_i, and dimensionality D, that represent different instances of the raw data to be learned. The user can set the parameters of the final string length N and an alphabet size M.

The algorithm applies SAX to each dimension of the data separately, and then combine the output string by assigning each possible combination of symbols, resulting in D strings to a unique identifier. This leads to a string of length N, but an extended alphabet of length M^D. So, to maintain the requirement of the final string to be an alphabet of symbols M (parameter set by the user), the algorithm clusters the resulting characters into M clusters through K-means method and replace each character with the centroid of its cluster [17].

Although SAX-REPEAT can recognize interesting patterns, the original algorithm does not calculate the probability of the patterns being anomalous. Thereby, this work implemented an anomaly score for each found cluster (pattern). As C-AMDATS algorithm, the score is computed by the ratio between the size of the entire time series and the sum of the sizes of each cluster. Therefore, each cluster is sorted according to the respective anomaly score in descending order, and the anomalous patterns will be those with the highest anomaly score values.

2.4 k-NN

The k-Nearest Neighbors (k-NN) algorithm is one of the most popular method to solve both classification and regression problems. However, in this study, we will use it only for classification problem as unsupervised learning.

The algorithm assumes that similar data points exist in proximity, *i.e*, they are near to each other. The algorithms capture the idea of the similarity (also known as distance, proximity, or closeness), calculating the distance between points on a graph. Distance calculation is usually done by Euclidean distance, but it can be calculated using other distance functions. The Euclidean distance between the points $P = (p_1, p_2,\ldots,p_n)$ e $Q = (q_1, q_2,\ldots,q_n)$, in a n-dimensional Euclidean space, it is defined in Eq. (4):

$$d_e(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4}$$

Where, $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points in Euclidean n-space.

The k-NN algorithm depends on two parameters, a metric used to compute the distance between two points (in this case Euclidean function), and a value k of the number of neighbors to consider. When k is underestimated, the algorithm can overfit, *i.e*. it will classify just based on the closest neighbors instead of learning separating frontier between classes, but if k is overestimated, the algorithm will underfit, in the limit if $k = n$, the algorithm will consider every point belongs to the class that has more samples [18, 19].

2.5 Bootstrap

Bootstrap algorithm uses the computational power to estimate almost any summary statistics, such as the confidence interval, mean, or standard error. The method depends on the notion of a bootstrap sample B, which is a resampling of size n drawn to replace the original dataset $Z = (Z_1, Z_2, \ldots, Z_n)$. The bootstrap sample is represented $Z^* = \left(Z_1^*, Z_2^*, \ldots, Z_n^*\right)$. Each Z_i^* is one of the original Z values randomly selected, the selection probability for each Z value is equipollent, for example: $Z^* = Z_7, Z_2^* = Z_5, Z_3^* = Z_9, Z_4^* = Z_7$, etc. Note that the same original value can appear zero, one or more times, in the example, Z_7 appeared twice, *i.e*, the selection of Z value is not exclusive. The name Bootstrap concern to the use of the original dataset to generate new datasets Z^*. The idea is to generate a larger number of Bootstrap sample B of each size n using a random number device to perform the algorithm training. The number of bootstrap repeats defines the variance of the estimate, *i.e*, higher the number is, better is the variance, but in contrast, the computational cost increases with the increasement of the B number [20, 21].

In this sense, we are interested in calculating a confidence interval using Bootstrap, which is performed by requesting the statistics stored during the training and selecting values in the chosen percentile for the confidence interval. The chosen percentile is denoted as δ (Alpha or Significance Level). Eq. (5) defined the calculation to estimate the distribution of δ^* for each Bootstrap sample.

$$\delta^* = \overline{x}^* - \overline{x} \tag{5}$$

Where: $\overline{x}^*$ is the mean of an empirical bootstrap sample and $\overline{x}$ is the mean of the original data.

Therefore, the confidence interval for a Significance Level of 0.05 is defined by Eq. (6).

$$\text{Confidence interval} = \left[\overline{x} - \delta_{.05}^*, \overline{x} - \delta_{.95}^*\right] \tag{6}$$

Where, $\overline{x}$ is the mean of the original data, $\delta_{.05}^*$ is significance level at the 5th percentile, and $\delta_{.95}^*$ is significance level at the 95th percentile.

So, in order to obtain a very accurate estimate of $\delta_{.05}^*$ and $\delta_{.95}^*$, it is important to generate a large number of bootstrap samples.

2.6 RRCF

Robust Random Cut Forest (RRCF) algorithm is an ensemble technique for detecting outliers. The idea is based on an isolation forest algorithm that uses an ensemble of trees. In graph theory, trees are collections of vertices and edges where any two vertices are only connected by one edge, it is an ordered way of storing numerical data.

In this view, the algorithm takes a set of random data points, cuts them to the same number of points and creates trees. The algorithm starts by constructing a tree of n vertices, then it creates more trees of the same size, which in turn creates the forest. The user can choose the number of trees and the number of data point of each tree has, which is randomly sampled from the dataset. After the construction of the forest, the algorithm injects a new data point p into the trees to follow the cuts and to compute the average depth of the point across a collection of trees. The point is labeled an anomaly if the score overtake a threshold, which corresponds to the average depth across the trees [22].

3. Comparative analysis between the algorithms

This section will discuss the details of the datasets, the algorithm parameters settings, the evaluations performance method, and a comparative analysis between the algorithms.

In order to summarize the advantages and limitations of each algorithm, **Table 2** shows advantage and limitations of the ML algorithms.

ML algorithms	Advantage	Limitations
C-AMDATS	• Multivariate approach • Easy to parameterize • Considers the variances of each dimension	• Take more CPU time due to the need to compute the inverse of the covariance matrix for each cluster • Sensitive to noisy data, missing values and outliers
Luminol Bitmap	• Quick calculation time • Runs well on big data • Not too sensitive to parameter choices	• Univariate approach
SAX-REPEAT	• SAX Multivariate approach	• Hard to parameterize • Slow calculation time due the k-means clustering • Sensitive to missing values and outliers
k-NN	• Quick calculation time • Easy to implement • Variety of distance criteria can be chosen	• Does not work well with large dataset • Does not work well with high dimensions • Sensitive to noisy data, missing values and outliers • Univariate approach
Bootstrap	• Quick calculation time • Does not require large sample size	• Can be computationally expensive depending on the bootstrap sample number • Univariate approach
RRCF	• Different dimensions are treated independently • Designed to run in a streaming data	• Univariate approach • Can be computationally expensive depending on the number of trees • Sensitive to noisy data, missing values

Table 2.
Advantage and limitations of the unsupervised ML algorithms.

The advantages and limitations of each algorithm will be discussed throughout the development of this chapter.

3.1 Characterization of case studies

This chapter brings two sets of real data, which were collected for the purpose of detecting anomalies in multivariate time series. The databases applied in this study will be detailed in the next sections.

3.1.1 Case study 01 - meteocean data in hurricane season

The chosen set is a public meteocean data available online in the National Data Buoy Center of the National Oceanic and Atmospheric Administration's (NOAA). The dataset was collected in the Atlantic Ocean off the Bahamas coast (23,838 N; 68,333 W). The data were structured in hourly frequency and it begins in June 2012 until November 2012 (213 days and 22 hours), comprising 15,315 data points. This period corresponds to the hurricane season in the Ocean Atlantic, which that year was especially active with 19 tropical cyclones (winds above 52 km/h), which 10 cyclones became hurricanes (winds above 64 km/h).

Hurricanes can be detected by several meteorological variables that consequently are directly impacted. In this case study, the following analysis variables were considered: (a) significant wave height (WVHT); (b) sea level pressure (PRES); and (c) wind speed (WSPD). Within the period of the dataset, three hurricanes transited through the Bahamas coast region: (i) Isaac; (ii) Rafael and (iii) Sandy.

Isaac had his first alert issued on August 21 by the National Hurricane Center. Several islands in the Lesser Antilles have been placed under hurricane surveillance or tropical storm warnings. Isaac was tracked between Guadalupe and Dominica on August 22, it passed over Haiti and Cuba with a strong tropical storm force. On August 26, the Isaac approaches Florida Keys and the next day entered the eastern Gulf of Mexico causing several economic impacts in the USA. There was a gradual intensification and Isaac reached its peak intensity as a category 1 hurricane, with sustained 1-minute winds of 80 mph (130 km/h) [23].

Hurricane Rafael produced minor damage in the northeastern Caribbean Sea in mid-October 2012. The first alert was issued to Bermuda on October 14, but was canceled on October 17 when the hurricane passed northeast of the island. On October 16, Rafael reached his peak intensity with maximum sustained winds of 90 mph (150 km/h). Rafael intensified in a category 1 hurricane [24].

Hurricane Sandy was the deadliest and most destructive, as well as the strongest, hurricane of the 2012 Atlantic hurricane season. Inflicting nearly $70 billion USD in damage, Sandy was a Category 3 storm at its peak intensity when it made landfall in Cuba. On October 24, Sandy became a hurricane reaching the coast near Kingston and Jamaica. On October 25, it hit his peak intensity in Cuba. On October 31, Sandy was already off the coast of Maine in the United States of America [25].

The **Figure 1** illustrates the period of hurricanes Isaac, Rafael and Sandy in the multivariate time series data.

In **Figure 1**, it is possible to visualize a behavior similarity between the three hurricanes. During the passage of the hurricanes, the variables WVHT and WSPD presented upward spikes, but on the other hands, PRESS presented downward spikes.

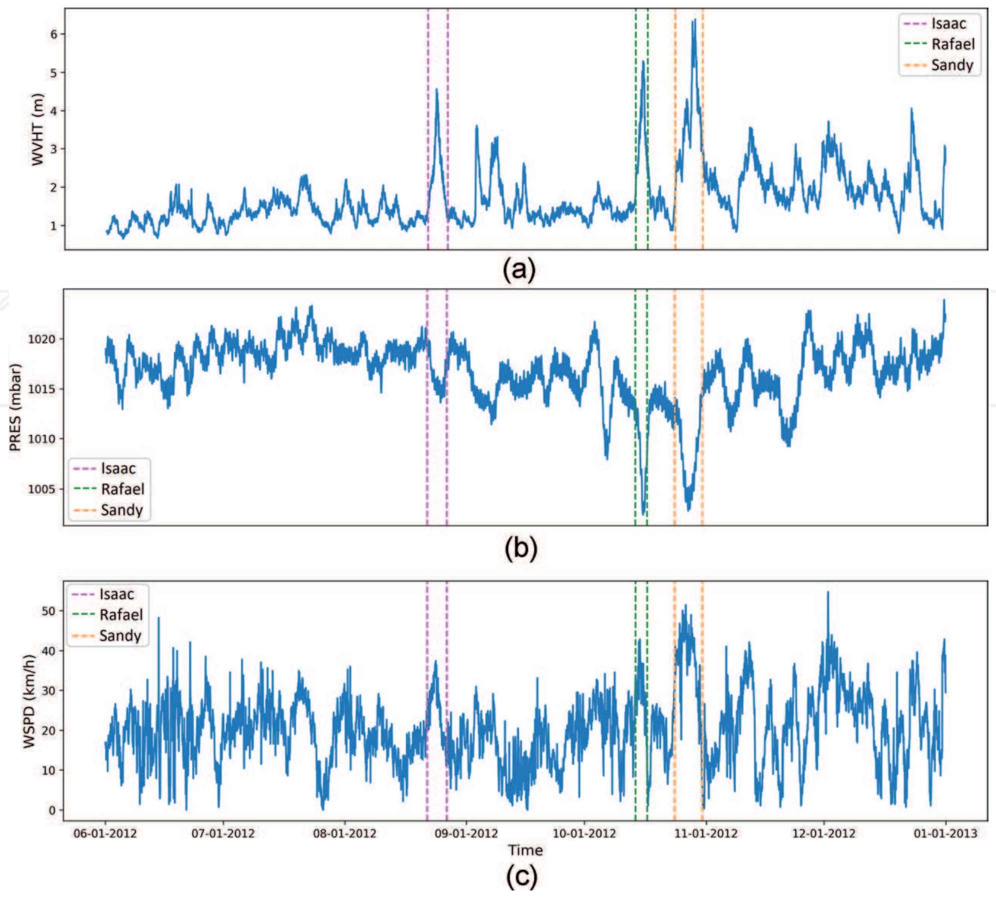

Figure 1.
Visualization of three variables in the same time domain (a) significant wave height, (b) sea level pressure and (c) wind speed. Color boxes represent hurricanes Isaac, Rafael and Sandy. Source: produced by the authors.

It is worth noting that the period of hurricanes in the **Figure 1** represents the time of its trajectory on the coast of Bahamas, and not its life span throughout its trajectory in the Atlantic Ocean.

3.1.2 Case study 02: monitoring data from dynamic machinery

The public dataset provided by the KNIME was acquired from 28 sensors installed in a dynamic machine. The sensors were installed to collect eight mechanical components parts (1st column of **Table 3**). The data starts on January 1st of 2007 and goes until April 20th of 2009 (838 days), comprising 16,660 data points.

The dataset was composed of 28 time series from 28 sensors. The signals were pre-processed with Fast Fourier Transform (FFT). The **Table 3** shows the groups and description of the sensors.

Each sensor group had at least 3 collections with different frequency bands, except the torque variable (M1), which had only one collection.

Signs of rotor malfunction could be traced back to March 6, 2008. The breakdown event happened on July 21, 2008. The break was visible only to some sensors, especially with low frequency bands.

For a cleaner and clearer view, **Figure 2** illustrates the multivariate time series only for sensors that detected the malfunction zone of the dynamic machine. Therefore, of total of 28 sensors, 18 were chosen to illustrate the multivariate time series. The machinery malfunction was detected in all sensor groups, except for the M1.

Sensor Group	Sensor Description
A1	Input shaft vertical
A2	Second shaft horizontal upper bearing
A3	Third shaft horizontal lower bearing
A4	Internal gear 275 degrees
A5	Internal gear 190.5 degree
A6	Input shaft bearing 150
A7	Input shaft bearing 151
M1	Torque kNm

Table 3.
The eight parts of the rotor monitored through groups of sensors.

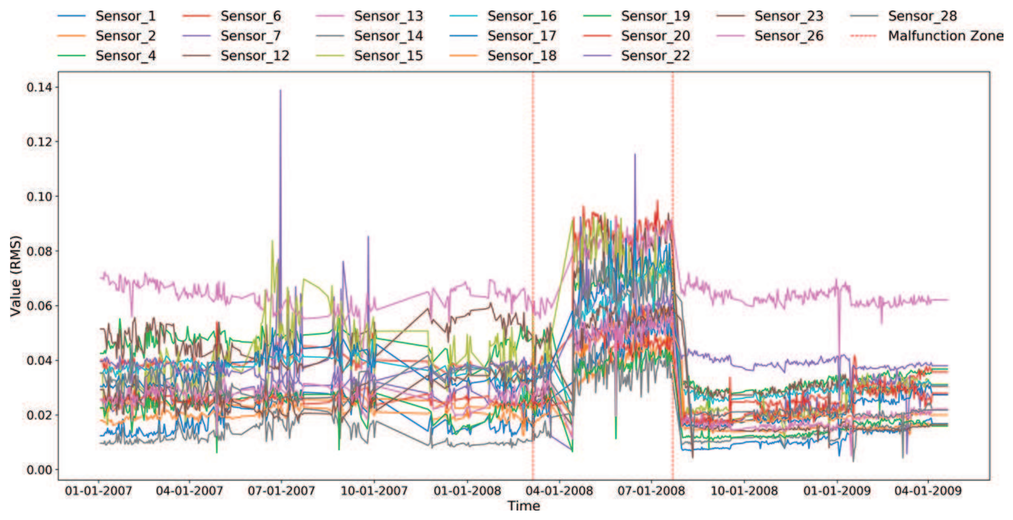

Figure 2.
Multivariate time series of 18 sensors that detected the malfunction zone of the machine.

In **Figure 2**, it is possible to verify a behavior change in the multiple sensors inside the malfunction zone (begging of March until the end of July). The **Figure 2** also illustrates two alarms in the beginning of 2007 triggered by the KNIME system. However, these two alarms can be considered as pre-mature, as the history of the machinery continued to run normally over one year. Another detail is that, afterwards the breakdown and rotor replacement, the signals were recorded much cleaner.

3.2 Parameterization of the ML algorithms

The parameters of unsupervised ML algorithms were settings to achieve the best possible performance to find the patterns of interest. The parameterization requires several attempts of success and error to achieve the best possible result. SAX-REPEAT was the most difficult method of setting the parameters due the high sensitivity of the variables. Whereas the Luminol Bitmap revealed not too sensitive to parameter choices, where the Bitmap Detector Score demonstrated the most determinant parameter for the algorithm. The Bootstrap showed a similar result for iterations above 200 and confidence level above 95%. C-AMDATS, RRCF and k-NN are easy algorithms to set the parameter due the small number they have. As an example of experiment case 2, it was necessary to run SAX-REPEAT with 76 different combinations of parameters to identify the best configuration, k-NN was necessary to run 21 times, C-AMDATS 20 times, RRCF 11 times, Luminol 12 times, and Bootstrap 10 times.

The **Table 4** summarizes the parameter settings of the presented algorithms for the two real cases applied in this chapter.

All ML algorithms in this paper were implemented in Python 3.6 programming language and executed on a high performance computing named AIRIS (Artificial Intelligence RSB Integrates System) at the Supercomputing Center for Industrial Innovation at SENAI CIMATEC. The AIRIS processor model is the Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz and has 376 GB RAM memory.

3.3 Case study experiment 01 - meteocean data in hurricane season

All monitoring variables at the meteocean data were processed in ML algorithms using the settings presented in the **Table 4**. The results were compared to the period of hurricanes life as shows in **Figure 1**. The hurricanes behaviors are more visually clear through the WVHT variable. Thereby, for the better understanding of the reader, we only illustrated the ML results in the WVHT variable, even though was made a multivariate analyzed. **Figure 3** shows the detection of the six algorithms.

In **Figure 3**, the C-AMDATS algorithm detected three distinct behavior patterns in the multivariate time series. Patterns 0 and 1 had the highest anomaly score and are well situated in hurricanes regions, so these patterns where considered as anomaly behavior. However, pattern 1 also appear in November and December, which had no records of hurricane or tropical depression or tropical storm in the Bahamas cost, revealing to be a false positive signal. The Luminol bitmap and RRCF algorithms failed to isolate the patterns of interest. Luminol demonstrated a little sensitivity for detecting anomalies in this experiment, because it was possible

Algorithm	Parameter Setting	
	Case 01	Case 02

C-AMDATS	CF = 3.0
	ICS = 24
		CF = 1.8
		ICS = 30

Luminol Bitmap	Bitmap Detector Score = 1.0 Precision = 40 Lag Window Size = 10 Future Window Size = 10 Chuck Size = 24	Bitmap Detector Score = 0.35 Precision = 40 Lag Window Size = 10 Future Window Size = 10 Chuck Size = 30
SAX-REPEAT	Window size = 1155 PAA size = 3 Alphabet length = 2	Window size = 120 PAA size = 3 Alphabet length = 3

k-NN	k = 5
	Metric = Euclidean Distance
		k = 150
		Metric = Euclidean Distance

Bootstrap	Confidence = 0.95 Number of Iterations = 200	Confidence = 0.95 Number of Iterations = 200
RRCF	Number of Trees = 10 Tree Size = 5105	Number of Trees = 20 Tree Size = 595

Table 4.
Parameter setting of the Unsupervised ML algorithms. CF: Cluster Factor - ICS: Initial Cluster Size - PAA: Piecewise Aggregate Approximations - k: Neighbors number.

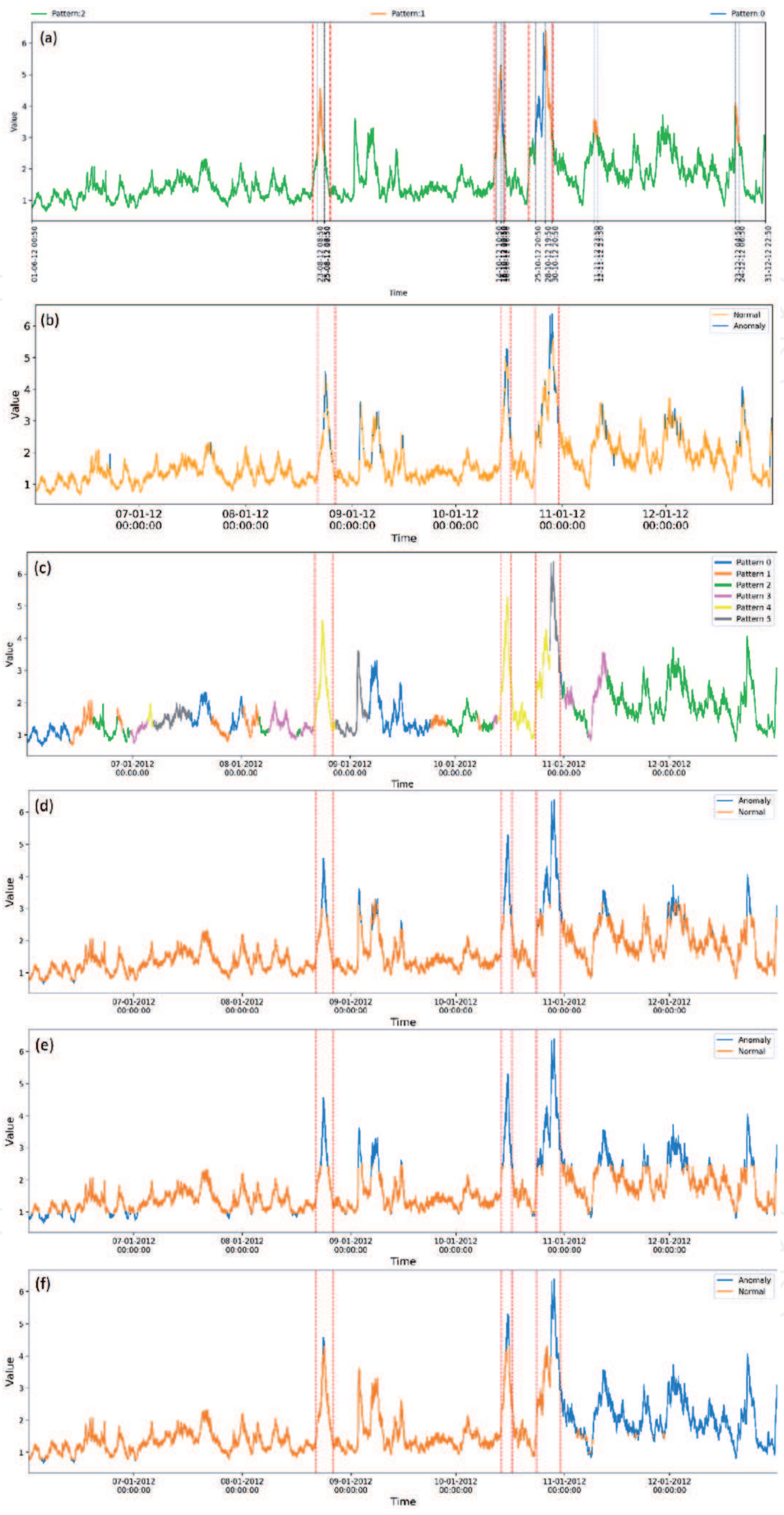

Figure 3.
Results of the unsupervised algorithms of Case 1. (a): C-AMDATS, (b): Luminol Bitmap, (c): SAX-REPEAT, (d): K-NN, (e): Bootstrap, and (f): RRCF.

to verify a few data points detected as anomaly. SAX-REPEAT returned 6 distinct patterns, which patterns 4 and 5 were the top 2 of the anomaly score. These two patterns are precisely in the regions of interest, however, it is possible to verify these patterns also in other regions, also indicating false positive signals. Bootstrap and k-NN had similar results, both algorithms detected spikes caused by hurricanes, but with many false positives, especially Bootstrap.

Therefore, it is possible to ascertain that the algorithms with the best performance in detecting the patterns of interest in case 01 were C-AMDATS and SAX-REPEAT. But a quantitative analysis will still be performed.

3.4 Case study experiment 02 - monitoring data from dynamic machinery

Analogous to the experiment performed in case 1, the experiment case 2 brings the results of the patterns and anomalies detection of unsupervised learning algorithms in the KNIME dataset.

All 28-monitoring data were processed using the settings in **Table 4**. The results were compared to the period of malfunction of the machine as show in **Figure 2**.

The machine malfunction is more visually clear through the sensor 1. Therefore, **Figure 4** only illustrated the results of Sensor 1, although the analysis was performed in a multivariable way.

In **Figure 4**, the C-AMDATS algorithm detected three distinct behavior patterns in the multivariate time series. Patterns 0 had the highest anomaly score and is well situated in the interest region, so this pattern was assumed to be anomalous. Luminol and RRCf again failed to isolate the fault, both algorithms had many false positives and false negatives. SAX-REPEAT detected 15 different patterns, which is not desired as it makes difficult for the specialist to analyze many patterns. Nevertheless, patterns 0 and 4 had the lowest punctuation in the anomaly score ranking, so these patterns were assumed to be normal and the others as anomaly. The k-NN and Bootstrap methods also demonstrated a good performance in isolating the period of interest, with few false positives and false negatives.

Therefore, the algorithms were able to isolate the anomalous region well in case 02, with exception of Luminol Bitmap and RRCF.

3.5 Performance evaluation

The performance evaluation of the algorithms - in their ability to identify the same anomalous patterns - was performed through the calculation of seven metrics: accuracy (ACC), precision (PR), recall (REC), specificity (SP), F1-score (F1), area under the curve (AUC) of receiver operating characteristics (AUC-ROC), and AUC of precision and recall curve (AUC-PRC).

However, the performance evaluation would not be properly fair, as the Luminol, k-NN, Bootstrap and RCCF algorithms made the analysis univariably (different from C-AMDATS and SAX-REPEAT). Thereby, in an attempt to obtain a more appropriate analysis, the threshold metrics was calculated for all proposed variables and then extracted an average evaluation, except C-AMDATS and SAX-REPEAT.

All the evaluation metrics are calculated by comparing the real data points (classified by experts) with the predicted data points (predicted by ML algorithms). So, the ACC reveals the correct prediction in a general approach, but it may hide the error rate of the model, that is why it is prudent to measure the performance jointly with other metrics. PR indicates the true positive value compared to the false negative. REC reveals out the true positive value with the false positive. Both metrics (PR and REC) reveal the model's ability to predict positive values, but with different perspectives. SP demonstrated the capacity of the model to predict the true negative over false positives perspective. The F1 is a harmonic average between REC and PR.

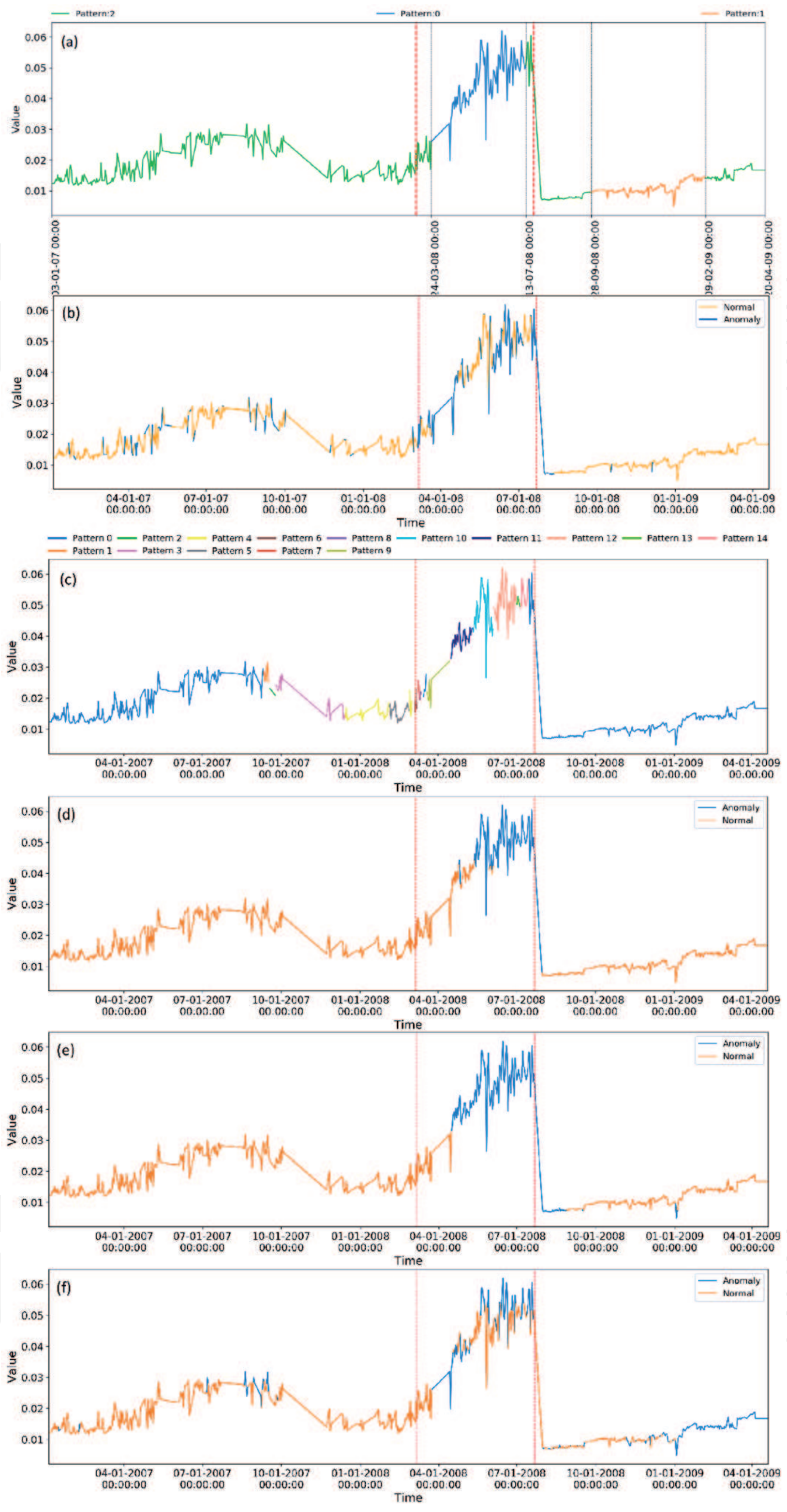

Figure 4.
Results of the unsupervised algorithms of Case 2. (A): C-AMDATS, (B): Luminol Bitmap, (C): SAX-REPEAT, (D): K-NN, (E) Bootstrap, and (F): RRCF.

AUC-ROC is the area under the curve on the true positive (REC) and false positive (1- SP) rates. The AUC-PRC is the area below the curve between PR and REC. AUC-PRC is an important metric for assessing unbalanced datasets, being a great advantage over the others, since in the vast majority of cases, especially real data, have a higher volume of normal than abnormal data.

The seven performance assessment metrics for all proposed variables of case 1 and case 2 experiments are listed in the **Table 5**.

The performance evaluation presented in the **Table 5** revealed that the C-AMDATS was the one that stood out amongst the other algorithms. C-AMDATS was superior

ML Algorithms	Metrics	Case #1	Case #2	Average
C-AMDATS	ACC	96%	96%	96%
	PR	90%	98%	94%
	REC	80%	89%	85%
	SP	80%	89%	85%
	F1	84%	92%	88%
	AUC-ROC	81%	89%	85%
	AUC-PRC	76%	88%	82%
Luminol Bitmap	ACC	86%	72%	79%
	PR	55%	58%	57%
	REC	55%	61%	58%
	SP	55%	61%	58%
	F1	55%	58%	57%
	AUC-ROC	56%	61%	58%
	AUC-PRC	52%	56%	54%
SAX-REPEAT	ACC	86%	89%	88%
	PR	66%	82%	74%
	REC	92%	90%	91%
	SP	92%	90%	91%
	F1	70%	85%	78%
	AUC-ROC	92%	90%	91%
	AUC-PRC	66%	79%	73%
k-NN	ACC	94%	86%	90%
	PR	80%	81%	80%
	REC	67%	68%	67%
	SP	67%	68%	67%
	F1	69%	71%	70%
	AUC-ROC	66%	68%	67%
	AUC-PRC	61%	67%	64%
Bootstrap	ACC	79%	85%	82%
	PR	59%	72%	65%
	REC	73%	75%	74%
	SP	73%	75%	74%
	F1	59%	73%	66%
	AUC-ROC	73%	75%	74%
	AUC-PRC	56%	70%	63%

ML Algorithms	Metrics	Case #1	Case #2	Average
RRCF	ACC	80%	66%	73%
	PR	55%	44%	50%
	REC	55%	45%	50%
	SP	55%	45%	50%
	F1	54%	44%	49%
	AUC-ROC	55%	45%	50%
	AUC-PRC	53%	49%	51%

Table 5.
Performance Evaluation of unsupervised ML algorithms to detect interesting/anomalous patterns in multivariate time series data.

in ACC, PR, AUC-PRC and F1 metrics against SAX-REPEAT, which was the second algorithm that stood out. Nevertheless, it is relevant to note that C-AMDATS was 10% superiority in AUC-PRC of SAX-REPEAT. Then, in decreasing order of algorithm position in the performance evaluation would be: (i) C-AMDATS, (ii) SAX-REPEAT, (iii) k-NN, (iv) Bootstrap, (v) Luminol and (vi) RRCF. Both algorithms that have a multivariate analysis intrinsically were superior. However, more case studies must be carried out to affirm the superiority of the algorithms studied here.

Therefore, the results presented in this study strengthens the idea that unsupervised machine learning algorithms can assist the data annotation and labeling process. This approach can optimize much of the specialists' time and leverage the supervised AI models.

4. Conclusions and future work recommendations

This work demonstrated the effectiveness of a multivariate analysis using six different unsupervised ML algorithms for time series. To verify the performance of the unsupervised ML algorithms to detect interesting/anomalous patterns in real time series data, the six algorithms were applied in two different real cases: (i) meteocean data in hurricane season and (ii) monitoring data from dynamic industrial machinery. The experimental results showed that clustering methods as C-AMDATS have higher capacity to recognize and isolate the anomaly region, revealing the ability to assist experts to label raw data with unsupervised ML algorithms with great performance.

Future works include the extension of this analysis to more real cases aiming to develop a broader analysis, as well as an extensive study and investigation of approaches of semi-supervised learning to train deep learning algorithms to predict and classify unknown data in different dataset.

Acknowledgements

The authors thank the NOAA and KNIME for providing to the scientific community publics datasets, leveraging researches and technology development. The authors also thank the research group CS2i and the Reference Center for Artificial Intelligence at SENAI CIMATEC.

Author details

Ilan Figueirêdo, Lílian Lefol Nani Guarieiro and Erick Giovani Sperandio Nascimento*
SENAI CIMATEC, Salvador, Brazil

*Address all correspondence to: erick.sperandio@fieb.org.br

References

[1] Rodner E, Barz B, Guanche Y, Flach M, Mahecha M, Bodesheim P, et al. Maximally Divergent Intervals for Anomaly Detection. In: ICML Workshop on Anomaly Detection [Internet]. New York, NY, USA; 2016. Available from: http://arxiv.org/abs/1610.06761%0A

[2] Sperandio Nascimento EG, Tavares O, De Souza A. A Cluster-based Algorithm for Anomaly Detection in Time Series Using Mahalanobis Distance. In: ICAI'2015 - International Conference on Artificial Intelligence [Internet]. Las Vegas, Nevada, USA: ICAI'15 - The 17th International Conference on Artificial Intelligence; 2015. p. 622-8. Available from: https://www.researchgate.net/publication/282330724_A_Cluster-based_Algorithm_for_Anomaly_Detection_in_Time_Series_Using_Mahalanobis_Distance

[3] Wei L, Kumar N, Lolla VN, Keogh EJ, Lonardi S, Ratanamahatana C (Ann). Assumption-Free Anomaly Detection in Time Series. In: 17th International Conference on Scientific and Statistical Database Management, SSDBM [Internet]. Berkeley, CA, USA; 2005. p. 237-42. Available from: https://pdfs.semanticscholar.org/909b/8226968d41f76cc14d0eef2d365572e7a37b.pdf?_ga=2.68813208.804195361.1594613685-826585445.1591805583

[4] Liu H, Zhou J, Xu Y, Zheng Y, Peng X, Jiang W. Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks. Neurocomputing [Internet]. 2018 Nov;315:412-24. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0925231218308695

[5] Figueiredo IS, Guarieiro LLN, Santos AAB, Nascimento EGS. Algoritmo de aprendizagem de máquina não supervisionado para detecção de anomalias em séries temporais multivariadas aplicadas a temporadas de furacões. In: V Seminário de Avaliação de Pesquisa Científica e Tecnológica [Internet]. Salvador, Bahia: SENAI CIMATEC; 2020. p. 3. Available from: https://www.researchgate.net/publication/343252454_ALGORITMO_DE_APRENDIZAGEM_DE_MAQUINA_NAO_SUPERVISIONADO_PARA_DETECCAO_DE_ANOMALIAS_EM_SERIES_TEMPORAIS_MULTIVARIADAS_APLICADAS_A_TEMPORADAS_DE_FURACOES

[6] Kim M, Ou E, Loh P-L, Allen T, Agasie R, Liu K. RNN-Based online anomaly detection in nuclear reactors for highly imbalanced datasets with uncertainty. Nucl Eng Des [Internet]. 2020 Aug;364(April):110699. Available from: https://doi.org/10.1016/j.nucengdes.2020.110699

[7] Schwabacher M, Oza N, Matthews B. Unsupervised Anomaly Detection for Liquid-Fueled Rocket Propulsion Health Monitoring. J Aerosp Comput Information, Commun [Internet]. 2009 Jul;6(7):464-82. Available from: https://arc.aiaa.org/doi/10.2514/1.42783

[8] Elavarasan D, Vincent DR, Sharma V, Zomaya AY, Srinivasan K. Forecasting yield by integrating agrarian factors and machine learning models: A survey. Comput Electron Agric [Internet]. 2018;155(October):257-82. Available from: https://doi.org/10.1016/j.compag.2018.10.024

[9] Khan S, Liew CF, Yairi T, McWilliam R. Unsupervised anomaly detection in unmanned aerial vehicles. Applied Soft Computing [Internet]. 2019

Oct;83:105650. Available from: https://doi.org/10.1016/j.asoc.2019.105650

[10] Ward CP, Weston PF, Stewart EJC, Li H, Goodall RM, Roberts C, et al. Condition monitoring opportunities using vehicle-based sensors. In: Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit. 2011.

[11] Yin S, Ding SX, Xie X, Luo H. A review on basic data-driven approaches for industrial process monitoring. IEEE Transactions on Industrial Electronics. 2014.

[12] Yiakopoulos CT, Gryllias KC, Antoniadis IA. Rolling element bearing fault detection in industrial environments based on a K-means clustering approach. Expert Systems with Applications [Internet]. 2011 Mar;38(3):2888-911. Available from: http://dx.doi.org/10.1016/j.eswa.2010.08.083

[13] Wang X-B, Zhang X, Li Z, Wu J. Ensemble extreme learning machines for compound-fault diagnosis of rotating machinery. Knowledge-Based Syst [Internet]. 2019; Available from: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85071672820&doi=10.1016%2Fj.knosys.2019.105012&partnerID=40&md5=a47687f0f999c07d4b3a5a2ed5a8eb2b

[14] Nguyen VH, Cheng JS, Yu Y, Thai VT. An architecture of deep learning network based on ensemble empirical mode decomposition in precise identification of bearing vibration signal. Journal of Mechanical Science and Technology 2019;33(1):41-50.

[15] Abid A, Khan MT, Khan MS. Multidomain Features-Based GA Optimized Artificial Immune System for Bearing Fault Detection. IEEE Trans Syst Man, Cybern Syst [Internet]. 2020 Jan;50(1):348-59. Available from: https://ieeexplore.ieee.org/document/8031077/

[16] Lin J, Keogh E, Lonardi S, Chiu B. A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03 [Internet]. New York, New York, USA: ACM Press; 2003. p. 2-11. Available from: http://portal.acm.org/citation.cfm?doid=882082.882086

[17] Mohammad Y, Nishida T. Robust learning from demonstrations using multidimensional SAX. In: 14th International Conference on Control, Automation and Systems (ICCAS) [Internet]. Gyeonggi: IEEE; 2014. p. 64-71. Available from: http://ieeexplore.ieee.org/document/6987960/

[18] Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans Inf Theory [Internet]. 1967 Jan;13(1):21-7. Available from: http://ieeexplore.ieee.org/document/1053964/

[19] Gou J, Ma H, Ou W, Zeng S, Rao Y, Yang H. A generalized mean distance-based k-nearest neighbor classifier. Expert Syst Appl [Internet]. 2019 Jan;115:356-72. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0957417418305293

[20] Efron B. Bootstrap Methods: Another Look at the Jackknife. In: Breakthroughs in Statistics [Internet]. 1992. p. 569-93. Available from: http://link.springer.com/10.1007/978-1-4612-4380-9_41

[21] Efron B, Rogosa D, Tibshirani R. Resampling Methods of Estimation. In: Wright JD, editor. International Encyclopedia of the Social & Behavioral Sciences [Internet]. Second Edi. Oxford: Elsevier; 2015. p. 492-5. Available from:

http://www.sciencedirect.com/science/article/pii/B9780080970868421653

[22] Guha S, Mishra N, Roy G, Schrijvers O. Robust random cut forest based anomaly detection on streams. In: 33rd International Conference on Machine Learning, ICML 2016 [Internet]. 2016. p. 3987-99. Available from: http://proceedings.mlr.press/v48/guha16.pdf

[23] National Hurricane Center Administration. Tropical Depression Nine Public Advisory One [Internet]. NINTH DEPRESSION OF THE SEASON FORMS EAST OF THE LESSER ANTILLES. Miami; 2012. Available from: https://www.nhc.noaa.gov/archive/2012/al09/al092012.public.001.shtml

[24] Avila L. Hurricane Rafael Tropical Cyclone Report [Internet]. Miami; 2013. Available from: http://www.nhc.noaa.gov/data/tcr/AL172012_Rafael.pdf

[25] Blake ES, Kimberlain TB, Berg RJ, Cangia, Losi JP, Beven II JL. Tropical cyclone report Hurricane Sandy [Internet]. National Weather Service, National Hurricane Center. Miami; 2013. Available from: https://www.nhc.noaa.gov/data/tcr/AL182012_Sandy.pdf