International Journal of Machine Learning and Cybernetics (2022) 13:3113–3129
https://doi.org/10.1007/s13042-022-01583-x

ORIGINAL ARTICLE

Fault detection and diagnosis for industrial processes based on clustering and autoencoders: a case of gas turbines

Jose M. Barrera[1,2] · **Alejandro Reina**[1,2] · **Alejandro Mate**[1,2] · **Juan C. Trujillo**[1,2]

Received: 13 October 2021 / Accepted: 13 May 2022 / Published online: 2 June 2022

Abstract
Industrial machinery maintenance constitutes an important part of the manufacturing company's budget. Fault Detection and Diagnosis (henceforth referenced as FDD) plays a key role on maintenance, since it allows for shorter maintenance times and, in the long run, to train predictive maintenance algorithms. The impact of proper maintenance is reflected on an especially costly type of industrial machine: gas turbines. These devices are complex, large pieces of machinery that cause considerable service disruption when downtime occurs. In an effort to shorten these service disruptions and establish the basis for the development of predictive maintenance, we present in this paper an approach to FDD of industrial machinery, such as gas turbines. Our approach exploits the data generated by industrial machinery to train a machine-learning based architecture, combining several algorithms with autoencoders and sliding windows. Our proposed solution helps to achieve early malfunctioning detection and has been tested using real data from real working environments. In order to build our solution, first, we analyze the behavior of the gas turbine from a mathematical point of view. Then, we develop an architecture that is capable of detecting when the gas turbine presents an abnormal behavior. The great advantage of our proposal is that (i) does not require existing disruption data, which can be difficult to obtain, (ii) is not limited to processes with specific time windows, and (iii) provides crucial information in real time to the monitoring staff, generating valuable data for further predictive maintenance. It is worth highlighting that although we exemplify our approach using gas turbines, our approach can be tailored to other FDD problems in complex industrial processes with variable duration that could benefit from the aforementioned advantages.

Keywords Autoencoders · Machine learning · Industrial processes · Fault detection and diagnosis · Internet of things · Data analytics

1 Introduction

The rise of Industry 4.0 has led to a major revolution in industry [1]. Triggered by the rapid advancement of digital technologies and reductions in associated technology costs, Industry 4.0 promises to deliver more flexibility, higher productivity, and lower costs to the manufacturing industry. As a result, companies are focusing their research efforts and budgets to deal with the different barriers between their current status and the envisioned future industry [1].

✉ Jose M. Barrera
jmba12@gcloud.ua.es

[1] Lucentia Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, Alicante 03690, Spain

[2] Lucentia Lab, Av. Pintor Pérez Gil 16, Alicante 03540, Spain

Current approaches for industrial maintenance range from post-failure maintenance, to preventive and predictive maintenance. The first one constitutes the costlier approach, where devices are often broken and/or cause service interruptions until repaired. The latter one is the cheapest, avoiding machinery damage and long service interruptions. However, it is also the most difficult approach to implement, since it requires extensive knowledge about the behavior of the machine at hand.

One of the Industry 4.0 research lines that has attracted more interest is predictive maintenance. This is the maintenance process where faults and breakdowns are predicted before they occur, allowing for timely maintenance just when it is needed. Thus, industries avoid unscheduled shutdowns, machinery damage and optimize maintenance costs. Compared to traditional maintenance, predictive maintenance is performed just when it is required, instead

of regularly scheduled, thus shortening downtimes and costs associated with unnecessary substitutions. Furthermore, it constitutes a substantial improvement over post-failure maintenance, where unscheduled shutdowns cause major disruptions of service and even machinery damage. The first step to predictive maintenance is to know what is happening in the industrial machinery continuously, precisely, and accurately. By using this information, fault detection and diagnosis can be carried out [2].

Unfortunately, there is a general lack of empirical research within the field of maintenance. In [3] authors describe how they found that, out of thousands of papers related to maintenance, only less than 90 displayed empirical real-world evidence. Therefore, empirical approaches that deal with fault detection and maintenance applied to real industry cases with real data are rather scarce.

In this paper, we propose a novel machine learning architecture solution that combines clustering and autoencoders to aid in early failure detection, the first step towards predictive maintenance. After a careful revision of the current state of the art, we checked different combinations of the most appropriate sets of machine learning algorithms and techniques to achieve the best solution with outstanding results empirically tested. One of the great advantages of our approach is that we researched not only one machine learning technique, but the combination of several techniques to achieve the highest performance rate. Our main contributions are:

- The proposed solution is especially useful in scenarios where the different manufacturing stages cannot be clearly identified using time windows or when the stages can vary from one run to another. This makes our solution especially useful in scenarios where the different manufacturing stages cannot be clearly identified using time windows or when the stages can vary from one run to another.
- The proposed solution can be easily tailored to industrial processes other than our application to gas turbines.
- The proposed solution can detect faults that not only present abnormal values above or below safety margins, but also those that affect the normal relationship between magnitudes without exceeding the operation limits, allowing for timely corrective actions.
- Due to the architecture is using an automatic semi-supervised learning, it does not require labelling. Consequently, it avoids human bias.
- Finally, it uses does not require faulty data: applicable to new machine models and processes. Moreover, it avoids the creation of faulty data, which can be very costly in an industrial process.

Therefore, we believe that this paper will be a useful reference for researchers and practitioners that face up with the problem of implementing machine learning solutions for preventive and predictive maintenance in manufacturing. Finally, it should be pointed out that the data used throughout the whole solution has been obtained from a real gas turbine running in a real environment. As a result of that, the authors cope with issues specified in [3] using real data.

The remainder of the paper is structured as follows: section 2 presents the related work in the area of predictive maintenance and AI applications. Section 3 covers the aspects related to clustering, machine learning, and AI that must be considered before presenting our solution. Section 4 presents our proposed solution for fault detection and predictive maintenance on industrial processes. Section 5 discusses the results and limitations of the approach. Finally, Sect. 6, describes the conclusions and sketches future works.

2 Related work

In this section, we review the state-of-the-art literature in the research field and highlight the main advantages that our proposal provides in comparison with existing works.

Artificial Intelligence (AI) has been widely used in industry in different fields in the last years, such as for transforming standard manufacturing into intelligent manufacturing [4] or in smart manufacturing. It has also been used for denoising data [5, 6] or capturing latent factors from a multidimensional space of a distribution [7]. In conjunction with other techniques like data analysis and Big Data techniques, AI has contributed to improve data visualization, dashboard handling, and optimal control on decision-making [8].

As mentioned in the previous section, FDD is a crucial part of industrial processes and therefore, different approaches have been presented in the literature. For instance, in [9] authors present a survey based on Nature-inspired optimization algorithms (NIOAs) and this is used as a base for algorithm's optimization in the first stage of the proposed architecture (clustering stage). On the other hand, in [10] a method based on convolutional neural networks and discrete wavelets is presented. However, this approach is focused on capturing health conditions for specific turbine problems. Furthermore, actual studies present problems dealing with industrial processes with stages that can present variations over time and are not fitted to specific time frames. As such, to tackle this problem, we present a more generic solution that focuses on the health of the industrial process, rather than the turbine, and tackles the problem of non-time-fitted stages of the industrial process.

Consequently, an optimization of the industrial process implies increasing the life expectancy of the turbine as collateral damage is avoided.

Moreover, other techniques that have been used on fault detection in industrial processes are based on SVM (Support Vector Machines) [11]. Nevertheless, this approach uses supervised learning, requiring the labelling of all data and the existence of error tuples. A tuple contains a set of values for each sensor reading in a specific moment. A correct or normal tuple belongs to a normal or correctly working industrial process. Consequently, an erroneous or faulty tuple contains values for each sensor that belongs to an abnormal or incorrect behaviour of the industrial process. Therefore, the labelling of the erroneous tuples can be very difficult to obtain if no data is available (since the obtaining implies provoking an intentional malfunction on the industrial process that could be even destructive for the industrial machinery, and consequently, very expensive). In the presented architecture, we can use non-supervised learning based on normal functioning, making the data capture process easy and less costly.

Focusing on anomaly detection and predictive maintenance, literature shows a close relation with Internet of Things (IoT) techniques [12, 13]. More specifically, autoencoders have been used on industry for anomaly detection on railways [14]. However, in this approach the authors obtain the different stages of the problem by dividing the timestamp of opening and closing the doors in 5 equal bins (using time as the divisor element). Therefore, even though with relevant results, this solution is too tailored to the railway problem and makes it difficult to be directly applied on other cases such as our scenario. The different stages of the operating mode of the turbine can vary its duration, making it difficult to apply any sort of temporary division. In addition, in [?], the authors propose an auto-encoder-based dynamic threshold to reduce false alarm rate in anomaly detection of steam turbines. However, this approach

Autoencoders (more information about autoencoders can be found in chapter 4.2) and semi supervised learning are widely used in faulty detection. In [15–17] authors propose FDD on autoencoders on a Tennessee Eastman Process. Their approach is based on supervised learning (the data from the process is labeled as a "correct" or "faulty") where the input data always corresponds to normal operations [18]. Moreover, in [19] the authors propose a novel deep transfer learning approach based on sparse auto-encoder for fault diagnosis. In [20], a semi-supervised robust projective and discriminative dictionary learning method for industrial process monitoring is presented. [21] introduces the idea of zero-shot learning into the industry field, tackling the zero-sample fault diagnosis task by proposing the fault description task based on the attribute transfer method. Furthermore, in [22], the authors propose a distributed sensor-fault detection and diagnosis system based on machine learning block with an autoencoder. However, their approach is based on the injection of five specific errors (drift bias, precision degradation, spike and stuck faults), whilst our approach is more generic and can detect every deviation regardless the cause. In [23], the authors propose a predictive railways maintenance strategy based on a hybrid neural architecture of long-short term memory autoencoder, providing the results in a test dataset. Nevertheless, our approach shows the results in the test dataset, and in a synthetic one. Finally, in [24] a transfer dictionary learning method for cross-domain multi-mode process monitoring and fault isolation is described.

It is worth highlighting that this approach is highly interesting because it allows us to correctly classify a binary classification when only correct data is available. If supervised learning techniques are used instead, then it must be taken into account that the input data must be labeled and malfunctioning data must be available. If malfunctioning data is not provided, it should be created and quite often faulty data cannot be generated since it implies forcing a malfunctioning cycle on very costly processes, which can simply become unaffordable. Even more, every different stage can have different faulty data and consequently, every stage must be able to discriminate its own correct and faulty tuples. For instance, in [25] the authors propose a model for FDD in hydraulic machinery integrating LSTM autoencoder detectors and diagnostic classifiers, with promising results. However, it uses a specific dataset for failure classification. The failures that are specified in that dataset are not related with the specific machinery. This can imply labeling faulty data as correct or vice versa, taking into account that some specific factors can not be extrapolated to new distribution (as for instance, ambient factors). Finally, in [26] the authors propose a FDD model based on convolutional neural networks. Neverthless, the create specific failures through a software, and consequently a classification approach is used. In our case, no software is available and no domain knowledge is provided. Thus, our model must extract its insights from the normal operation data distribution. The results between state-of-art and our approach are explained in Results, discussions and limitations.

As it has been shown, there is an extensive literature tackling malfunctioning machinery, where autoencoders are prime candidates for dealing with situations where malfunction data is not available. However, existing approaches have not dealt with a combination of stage-based processes where each stage has its own characteristics coupled with undefined duration and where malfunction data is also unavailable, such as in the case of gas turbines.

Compared to the state of the art, we present a comprehensive and novel architecture, based on different machine learning techniques, developed from the scratch, using data from a real industrial process that tackles these challenges. One of the advantages of our approach is that we also provide a guide to find out the different industrial stages (from

a mathematical point of view instead of from a functional point of view, that could imply the introduction of a human bias on the results) and a system based on sliding windows for "faulty error" communication. Additionally, we must highlight that the model is based on semi-supervised learning, allowing us to train the model with only "correct data". This avoids the need for "faulty data", since forcing a malfunction on a gas turbine is prohibitive from a monetary point of view.

In order to illustrate our approach, in the next section we introduce a running case study on gas turbines and describe its underneath problems, requiring a more complex approach than a single machine learning solution.

3 Running case study: gas turbines

Industrial processes have increased its complexity dramatically in the last decades. The advances in automatization and techniques related with sensorization have opened a vast field of capabilities related with the control of the industrial process.

In this case study, we present the case of a gas turbine used for electric energy co-generation. A gas turbine is the prime mover for the generator. Its input is thermal energy from burning gas and the output is mechanical power that drives the generator. Thus, this device converts the mechanical energy generated by the gas combustion into electrical energy that is supplied as a product at the terminals of the generator.

The industrial process that governs the turbine is very complex. The turbine mode used in the case study incorporates more than 100 sensors, which provide information about every part of the device. The system measures 31 physical values with backup sensors and multiple measuring points distributed along the turbine and the generator, as we can see in Table 1. It must be highlighted that temperature is checked in 16 different places in the exhaustor. In addition, small variations in the relationship between the different magnitudes can lead to a deviation from the optimal operation and consequently, a loss of production.

The sensor network of the turbine is constituted by an IoT architecture. In that architecture, every sensor emits its value and the timestamp associated to it. However, the sensor emission is not synchronized with other sensors. To solve this, the ML model gets as an input the last value received on each tuple every second. That implies that a tuple of data is processed in real-time every second.

Moreover, the progressive automation of industrial processes implies a decrease in human intervention on it. This fact is aggravated by the retirement of personnel with knowledge of the domain and with the inclusion of new personnel who lack that knowledge associated with the operation of the industrial process.

Table 1 Measured variables of the turbine-generator

Measured variables	
Active Load	Exhaust Tempeture 3
Variable Guide Vane	Exhaust Tempeture 4
Compressor Inlet Pressure	Exhaust Tempeture 5
Turbine Exhaust Pressure	Exhaust Tempeture 6
Turbine Inlet Temperature (T52)	Exhaust Tempeture 7
T7 Exhaust Temp Average	Exhaust Tempeture 8
Ambient Air Humidity	Exhaust Tempeture 9
Ambient Air Temperature	Exhaust Tempeture 10
Press Comp Outlet	Exhaust Tempeture 11
Turbine Exhaust Diff Pressure	Exhaust Tempeture 12
Air Intake Diff Pressure	Exhaust Tempeture 13
Compressor Inlet Temperature	Exhaust Tempeture 14
Compressor Outlet Temperature	Exhaust Tempeture 15
Rotor Speed	Exhaust Tempeture 16
Exhaust Tempeture 1	Gas Fuel Temperature
Exhaust Tempeture 2	

As a result, the information provided by the prevention system must be:

1. As fast as possible: If a deviation of the normal process is detected, it must be communicated as soon as possible so that preventive or corrective actions are taken immediately, avoiding potential damage to the machinery and service disruptions.
2. Provided with a suitable abstraction layer: If the information of the FDD system is provided with an ideal abstraction layer, users will be likely to act properly, without providing excessively low level data that may not be understood.
3. Their different running stages during execution time are not always the same ones, and therefore, this makes highly difficult to apply directly a classification technique: Due to the length of every phase in the industrial process can vary, the use of supervised learning and the labelling of the tuples can be tricky. Consequently, an automatic process for labelling must be created to tackle this issue.

Therefore, our goal is to characterize a normal operating state and communicate to the user when a continuous anomaly working process is happening.

However, in order to achieve these goals, the process of the gas turbines presents a few problems that must be properly tackled:

1. Number of stages from expert knowledge can imply a human bias: Since the authors do not have domain knowledge about the possible number of stages that the turbine could have, a brief summary was requested to the experts about how many stages we should be able to detect/consider, and which should the main characteristics that could help to discriminate those stages. Therefore, the industrial process is formed by 4 stages, related with its main dimension (Active Load). However, this approach is based on a physical point of view: experts only focus on the main output. Consequently, only one dimension has been taken in account. That implies that number of stages has not been obtained taking in account the whole multidimensional space (a mathematical point of view), and therefore, a human bias can have been introduced.
2. Correct and faulty data are different for every stage: Every stage differs on its working process to other stages. Consequently, correct data in one stage can be faulty data in another one and vice versa. Therefore, our architecture must be able to discriminate faulty and correct data in every stage.
3. The industrial process can have punctual anomalies in a normal working cycle: Punctual anomalies are part of a normal working process and they should not be communicated to the turbine's user. Thus, our architecture must focus on a continuous anomaly working process detection. Consequently, punctual anomalies must be dampened and only a continuous anomaly working process (in other words, a continuous set of punctual anomalies) must be communicated.

4 Proposed architecture

The proposed architecture is made up of 3 components and the overall architecture of the system is represented on Fig. 1.

The first stage is responsible for automatically determining the number and characteristics of stages that correspond to the industrial process (and consequently it solves problem 1), the second one tests the correctness of the tuple received (therefore it solves problem 2, and the third element muffles false positives (thus, it solves problem 3). On each subsection, we have thoroughly analyzed each of the components and how each parameter has been obtained. To summarize all this information, at the end of each subsection it is provided a table with the main parameters for that stage of the proposed architecture.

4.1 Dealing with problem 1: detecting stages

In order to deal with problem 1, a correct detection of the number of clusters must be performed. The information provided by the experts revealed that the turbine had 4 states based on the most important parameter of the turbine: the supply of electrical power to the electrical grid (Active Load). According to this information, the different stages are:

- stage 1. Idle: The turbine is at rest. It does not provide electrical power to the grid.

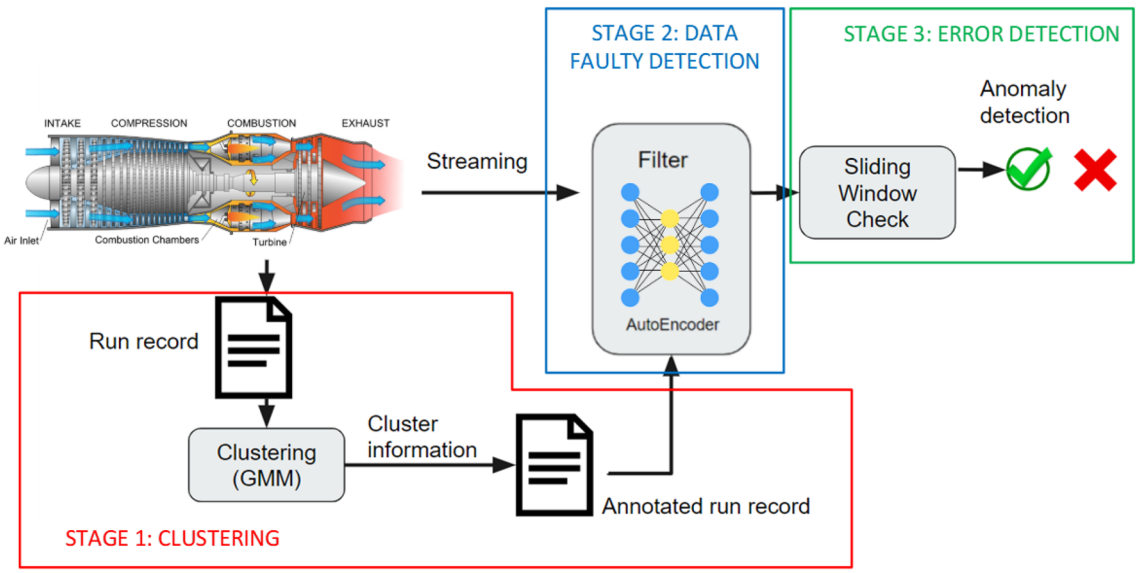

Fig. 1 Proposed three-stage architecture

- stage 2. Loading: The turbine starts its loading process, trying to reach the nominal operating values. During this stage, the active power supply grows continuously.
- stage 3 Loaded. The turbine has finished loading and is operating with its nominal values. During this stage, the turbine provides the maximum active power values into the grid.
- stage 4 Stopping. The turbine has received the stop command and begins to perform the stopping process. During this time, it reduces the electrical energy supplied until the turbine returns to stage 1, at which time it will be ready to restart the cycle.

In a graph representing active load / time, the expected four stages should have (approximately) the form of the Fig. 2.

In order to know the suitable option for detecting the stages that conforms the operating process of the turbine, two different approaches have been compared: a traditional approach focused on classification and knowledge provide by domain experts; and a novel approach focused on clustering with a strictly data driven approach. Due to space restrictions, only the summary of the results of the traditional approach is show: the algorithm works properly for stable phases (phase 1 Idle and phase 3 Full load with a 100% of accuracy, but unfortunately it shows worse results for phase 2 Increase of load and phase 4 Decrease of load, where there is still room for improvement with an accuracy of 60% for phase 2 and an accuracy of 50% for phase 4. Moreover, this approach presents some problems: the labeling of each of the tuples is required for correct training. If such labelling is not provided by the data, then a labeling operation based on expert knowledge is necessary. This is not always easy to achieve with sufficient precision. Furthermore, if the labeling operation is manual, it can have considerable costs depending on the number of tuples to be labelled.

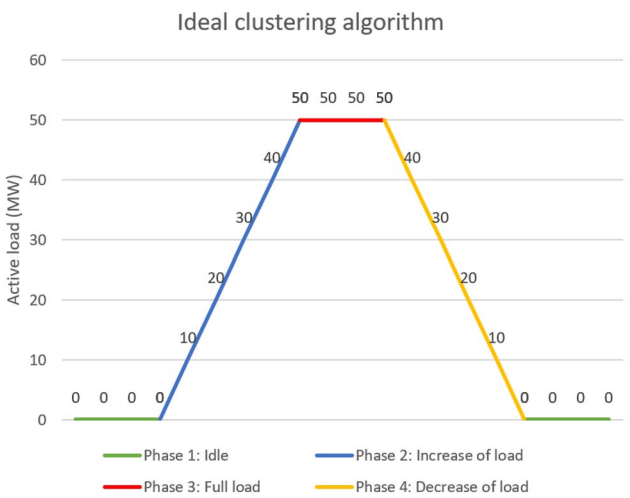

Fig. 2 Ideal working process from an approach based on classification

Consequently, the authors focus on a novel clustering-based approach. By means of the clustering, we can locate in a multidimensional space the different tuples of data that are in the vicinity, and consequently those tuples are labelled with the same label, thus greatly facilitating the labelling tasks.

Clustering is a technique of unsupervised machine learning and it consists on the process of grouping similar entities together [27]. It makes uses of statistical methods, distance based or density-based methods. The goal of this unsupervised machine learning technique is to find similarities in the data point and group similar data points together. Grouping similar elements together helps with the discrimination of the different groups. In addition, it helps to profile the attributes of each group. Due to its not necessary the labelling of the groups as in other techniques as classification (clustering obtain its own number of groups and its own labels), the automation of the process is easier and domain knowledge is not required. Moreover, the human bias is completely avoided, due to the algorithms for clustering are strictly mathematical and only a data driven approach can be performed.

Finally, it should be pointed that whereas in classification it is very easy to establish the degree of accuracy of the presented model (we have a predicted label and real label, so we can compare which model is better than another one with different metrics) [28], in clustering the boundary is a little bit fuzzier, due to the model does not have a "real label" to predict. In order to know which of the clustering models is the best one, we will use the "silhouette coefficient" [29]. This coefficient will provide us a guide about which model is best one, focusing on:

- The density of every of the clusters found (cohesion)
- The distance between the different clusters (separation)

In order to find different alternatives for the clustering operation on the system and find out the suitable solution, a few clustering algorithms have been tested on our model. The metric that would be used in order to know the suitability will be the silhouette coefficient.

The clustering algorithms used have been:

- Kmeans [30]
- DBSCAN (Density-based spatial clustering of applications with noise) [31]
- HAC (Hierarchical Agglomerative Clustering) [32]
- GMM (Gaussian Mixture Model) [33]

Every considered algorithm has different parameters that must be tailored in order to find the suitable configuration

of the algorithm on stages detection. To simplify this, a PSO (Particle Swarm Optimization) algorithm has been used [34] (an optimization tool). The idea of PSO is to emulate the social behavior of birds and fishes by initializing a set of candidate solutions to search for an optimal one. Particles are scattered around the search-space, and they move around it to find the position of the optimal solution. Each particle represents a candidate solution, and their movements are affected in a two-fold manner: (1) their cognitive desire to search individually, (2) and the collective action of the group or its neighbors [35]. Thanks to it, we can obtain an optimal or sub-optimal value of hyperparameters for each clustering algorithm, and therefore we can focus on results. The results of clustering approach can be seen on Fig. 3

After the execution of different clustering algorithms with PSO, looking for the best Silhouette coefficient, we obtain these conclusions:

- Most of the algorithms (GMM, KMeans, HAC) partition the space in almost in same way, varying some punctual tuples in most of the algorithms. Those 3 clustering algorithms can find 6 different stages in the industrial process (DBSCAN has created too much clusters).
- HAC allows an initial partition of the dimensional space, but due to the algorithm's own nature, the clustering of additional tuples is not allowed once the space has been partitioned. Moreover, KMeans has same number of clusters, but some of them has a very reduced number of points. Thus, all these 2 options are discarded and the clustering algorithm chosen is GMM.

From this space-partition we can extract more insights:

- From a mathematical point of view, the turbine has 6 stages. These are similar to those thought by a human from a mechanical point of view, but it must be pointed that during the start of the turbine, an additional stage is detected (stage of red color) surely due to the preparation of the different elements of the turbine before starting to supply active load.
- The last stage (color in black) is actually a different stage from the first one. This is logical since from a purely mathematical point of view, the sensors continue to send completely different values to those corresponding to the idle stage. For instance, take temperature sensors in the idle stage as an example, these sensors emit a value close to 0, but at the moment that the turbine has stopped supplying power to the network, even if the active load is 0, the temperature values of the turbine are high, since the turbine is in the cooling process. After a properly time, the turbine temperature values will return to 0 and therefore this new stage would be overlapped with the initial start-up stage.

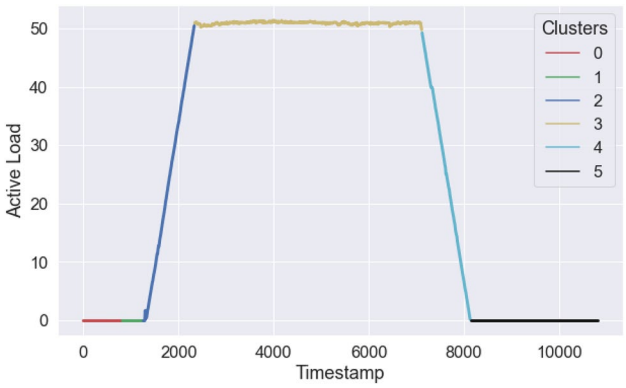

Fig. 3 Clustering results on stages detection. Timestamp shows the amount of seconds from the beginning of the industrial process

These conclusions help with the idea of avoiding human bias and focus on a mathematical point of view stage partitioning. Finally, the algorithm that we choose for partitioning the space is GMM, and it makes this partition of the data provided.

To sum up, in Table 2 we can see the information related to the components and the detection of the number of phases of the proposed architecture.

4.2 Dealing with problem 2: fault data detection

After solving the labeling process, we still need to solve which parts of this multidimensional space correspond to the normal operations and which parts are anomalies that occur when the gas turbine malfunctions. For this purpose, we then consider the use of autoencoders and semi-supervised learning. In order to do that, we will use an approach based on autoencoders. An autoencoder is a basic artificial neural network that consists of:

- An input layer, with N inputs.
- A hidden layer with Y neurons, where $Y < N$.
- An output layer, with N outputs.

The graphic structure of an autoencoder can be seen on Fig. 4.

Table 2 Main elements of the clustering component

Element	Description
Number of phases	6
Clustering detection algorithm	GMM
Hyperparameter optimization	Through PSO
Clustering parameter	Silhouette coefficient

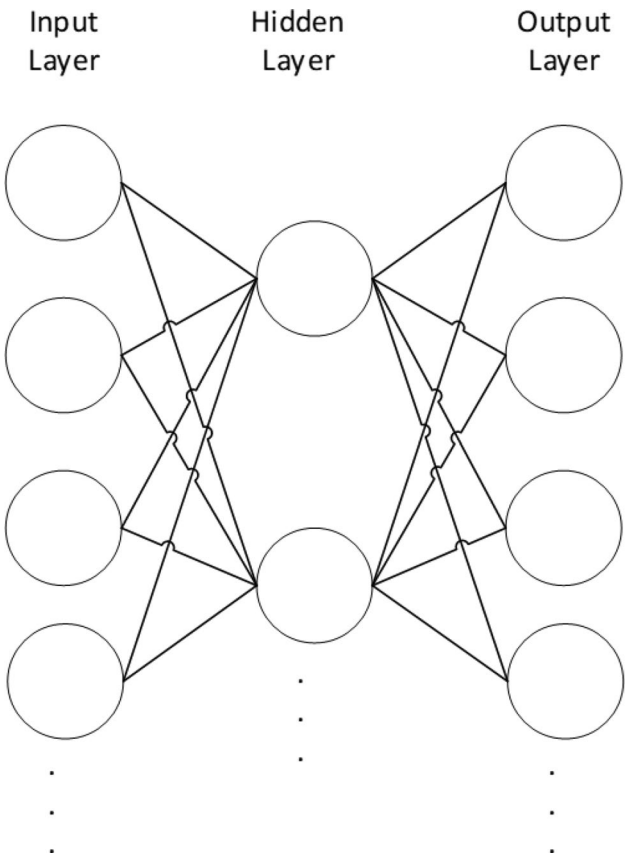

Fig. 4 Structure of an autoencoder

The autoencoder is formed by two stages: a encode stage (from input to hidden layer) and a decode stage (from hidden to output layer). The function of the autoencoder is to compress the input into a latent-space representation, and then to reconstruct the output from this representation, as we can see in Fig. 5. Detail of train/test overfitting point

That means that if the autoencoder has captured properly the latent space representation, when the input element is reconstructed from the latent representation, it will be almost or nearly the input element. If the autoencoder is trained with a set of elements that have a common latent space representation (as for instance, all the data tuples that are in a stage on an industrial process), every tuple after passing through the autoencoder will be reconstructed and the output from the output layer will be very similar to the input tuple.

From a mathematical point of view, we can treat the autoencoder as:

- Encoder: The part that compresses the input into the latent-space representation. It can be represented as $h=f(x)$
- Decoder: The part that reconstructs the input from the latent space representation. It can be represented as $i=g(h)$

To sum up, we can represent the general function of the autoencoder as $i=g(f(x))$, where i should be very similar to x. If we have a dataset D formed by (x_n) elements where $(D = \{x_1, x_2, x_3, \ldots, x_n\})$ in a multidimensional space of n-dimensions $D \in \mathbb{R}^n$, the autoencoder will be able to obtain for each x_n a values of i_n, where $x_n \simeq i_n$.

As we have previously mentioned, the autoencoder will try to reconstruct the input, such as output will be as close to input as possible. Therefore, if we do the operation output-input for every tuple, the result will be close to 0 for a correct tuple (Input–Output $\simeq 0$) and for a faulty one, the absolute value of the operation will be greater than 0 (|Input–Output| $\ggg 0$), due to the reconstruction rules that autoencoders has learnt are only valid for correct tuples (and if they are applied to a faulty tuple, the reconstruction of the tuple will trigger the value of the difference of the rest). In other words, the autoencoder can learn "the correct working process" of the industrial process, if we input "correct tuples" on it. Therefore, a correct tuple will have an output close to 0, and a faulty tuple will have an output much greater than 0, despite the problem that is generating the erroneous tuple.

Moreover, we have based the train on Root Mean Squared Error [36], due to this metric will help us to polarize values of correct and faulty data. It must be pointed out that the use of a single autoencoder is not sufficient since the autoencoder is capable of adequately discerning the ideal tuples that occupy a space. However, it should be pointed out that for each of the turbine stages, there will be a different region of the space with ideal operation. Therefore, the number of stages that conforms the industrial process should be automatically detected by means of a clustering algorithm. Then,

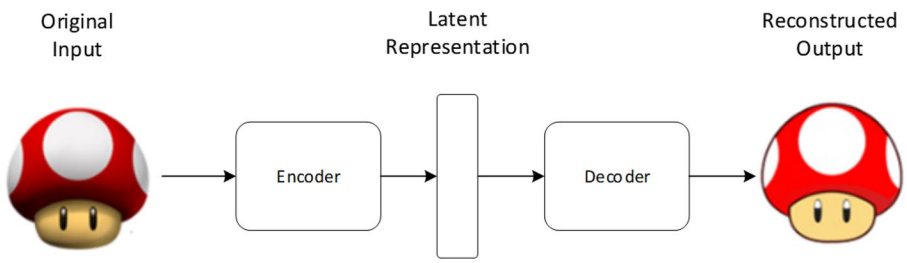

Fig. 5 Working process of an autoencoder

once the number of clusters is known, each cluster will be trained by means of an autoencoder which will be focused on detecting which part of the multidimensional space is the correct one for that specific cluster. Consequently, we will have one autoencoder for each of the stages of the industrial process. In our particular case of Gas Turbines, we will have 6 autoencoders, as previously described in the last section. Furthermore, the number of neurons in the input and the output layer will be determined by the number of dimensions of the architecture (31 neurons). For choosing the number of neurons of the hidden layer, we have selected 2/3 of the number of neurons in the input layer. This proposal has been used in other autoencoders structures in the literature, as for example in [37].

Moreover, if all the correct tuples produce almost the same output in the autoencoder (its reconstruction error is close to 0) and all the faulty tuples produce different results, we could establish a boundary, based on the histograms of the training and the test data on each autoencoder, based on this reconstruction error. This boundary will discriminate correct tuples from faulty tuples, and it will help us to minimize false positives and false negatives, as we can see in Fig. 6.

One problem that can have autoencoders is overfitting [38], i.e, when the ANN "memorizes" the data received and loses its capability of prediction (this can create a false sense of accuracy in the model). Consequently, a split of the data has been done between training models and test models (80–20%), with the aim of obtaining what would be the degree of accuracy of the model trained with completely unknown samples for each stage. Moreover, an excessive number of iterations can contribute to the overfitting problem [39]. To cope with that, a graph relating train/test cost for each iteration has been drawn. Consequently, we can graphically know which is the suitable number of iterations in order to achieve the best performance on unknown tuples. Thus, the training of the autoencoder for each stage has been realized with the optimal number of iterations, as we can see in Fig. 7. This enable to establish a boundary of discrimination between correct and faulty tuples, as it was showed on Fig. 7.

Table 3 Main elements of the classification component

Element	Description
Number of autoencoders	6 (one for each phase)
Number of layers	3 (input, hidden,output)
Number of neurons in input layer	31
Number of neurons in hidden layer	21
Number of neurons in output layer	31
Train-test split	80–20
Optimization parameter	RMSE

To sum up, in Table 3 we can see the information related to the structure and the topology of the autoencoders in this proposal.

4.3 Dealing with problem 3: detecting continuous malfunctioning

Once we have detected if a tuple of data corresponds to "correct" functioning or a "faulty" one, the next step is to discriminate if that tuple corresponds to a punctual isolated faulty data, or if it belongs to a set of faulty tuples that are received in a continuous streaming. In order to do that, we will use a sliding window method.

A window is determined by a predefined number of tuples, which are sliding by default, so naturally, their extent spans the most recent stream elements. Therefore, last received elements are more relevant than the previous ones. This is usually implemented as a FIFO (First In, First Out) queue.

Although there exist a great variety of windows (Count-based, Partitioned, Landmark, Fixed-band, etc) we are focusing on a time-based sliding window [40]. This is probably the most common class of window over data streams and it is suitable one for the problem that must be solved.

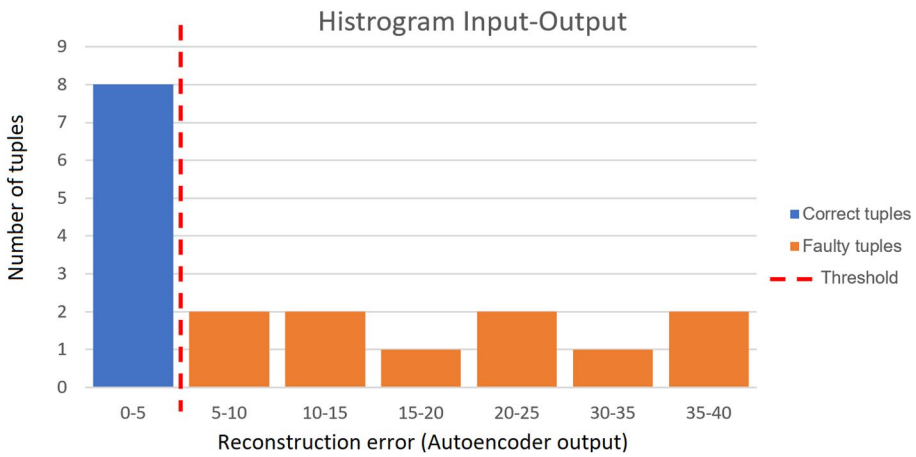

Fig. 6 Example of boundary for acceptance or rejection of a tuple in an autoencoder

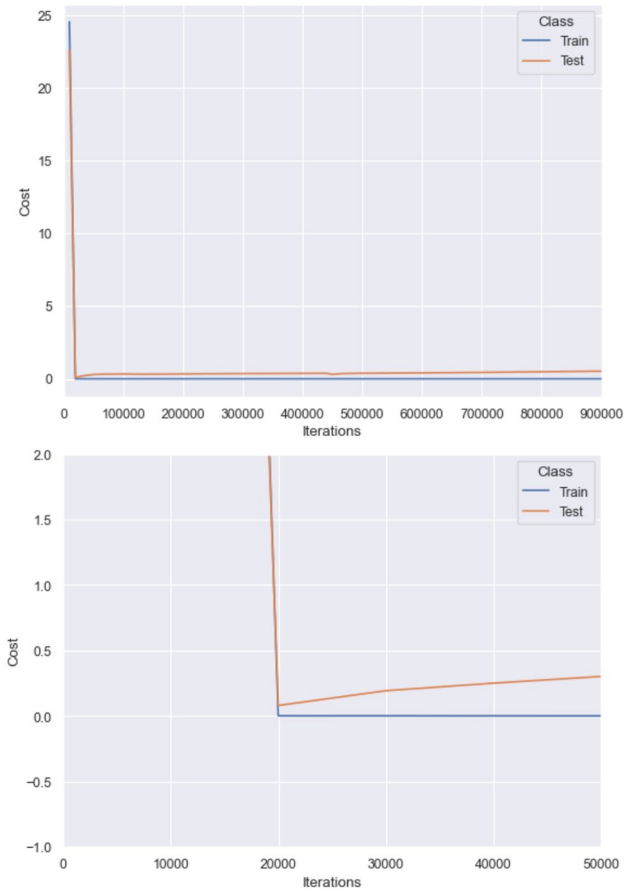

Fig. 7 Avoiding overfitting. Top: Overall train/test cost. Bottom: Detail of overfitting point

The sliding windows methods have been used in the industry for data-driven robots [41] and for FDD [42], blockchain architecture and IoT infrastructure [43, 44] and one of its purposes is to absorb fluctuations to faulty data on a correct streaming.

If we have a faulty tuple inside a streaming of correct data, should it be notified to the dashboard or should it be avoided? If we have a stream of data that is sending 100 data tuples every second, should we communicate to the dashboard 1 faulty tuple when we have 99 correct tuples and then that error is not repeated?

By using the sliding window, you can check whether the deviation from optimal performance is punctual, or if it is really is part of a malfunctioning sequence. The first event will be absorbed by the system, whereas if the error is repeated over time and goes beyond a certain threshold, it will be notified to the turbine staff. In the following image you can see a sliding window with 3 elements. The error is only communicated to the staff when the window is filled with erroneous tuples. If errors with correct tuples coexist in the window, the error is absorbed by the sliding window, as we can see in Fig. 8.

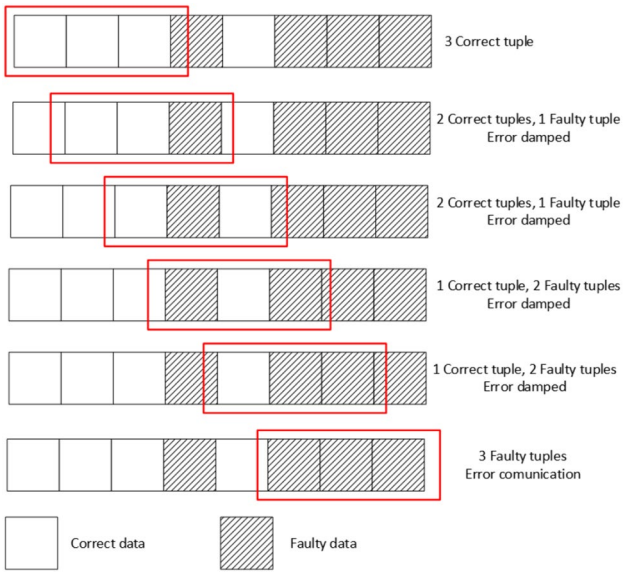

Fig. 8 Absorption of punctual errors with a sliding window

After the detection of faulty data, the system must discriminate if that faulty data belongs to a punctual anomaly (that it must not be communicated to staff) or if that anomaly belongs to an abnormal operation (that must be communicated in order to treat it with the corresponding corrective operations)

It is important to remark that the sliding window has a close relationship with IoT elements and Data Quality [45]. The size of the sliding window is crucial for absorbing fluctuations of the industrial process: the greater the sliding window is, the greater absorption will be. However, as larger that window is, more time will be necessary to communicate that error (due to the sliding window must be fulfilled with more data).

As a result of that, it is paramount to find a correct balance between absorption/communication. Furthermore, the needs of the industrial process and the communication infrastructure must be taken into account when the model is designed: if the tuples of data that are sending the sensors infrastructure are too spaced, maybe the error communication for a big sliding window will take too much time (if we are receiving data every 5 s, and the sliding window is formed by 5 elements, not until 25 s will be communicate the error), and the industrial process cannot allow that.

On the other hand, if the data tuples are received continuously, the model will have a more room for maneuver (if the model is receiving data every 0.1 s, with the same windows as we have commented in last paragraph, the communication error will be communicated after 0.5 s). The specification provided by the customer specifies that the error communication must be provided in less than 10 s.

Table 4 Main elements of the classification component

Element	Description
Type of window	Sliding
Size of the window	5
Tuple reception period	1 s
Time for communicating an anomalous working process	5 s

Finally, in the model presented, we have chosen a sliding window formed by 5 elements, and the error will be communicated if the sliding window is fulfilled with faulty tuples, as it was showed in section 4.2. Due to every tuple is received every 1 s, the communication of the error will be provided after 5 s. Thus, the customer requirement are met.

To sum up, in table 4 we can see the information related to the false alarms damping system of our proposal.

5 Results, discussions and limitations

In this section, we present a discussion of the results obtained by the presented model for the train/test operations on the ANN and for the new synthetic tuples created for testing the abnormal turbine functioning.

The experiments have been carried out using an Intel Pentium i9-7920X CPU 2.90 GHz, with 64 Gb RAM and 12 cores and 2 NVIDIA GeForce RTX 2080 Ti. With this infrastructure has been trained the clustering algorithms with PSO (around 8 h each algorithm) and the 6 autoencoders (one for each stage, around 8 h of training). Although the training of ANN could be improved using GPU's [46], it must be pointed that this has not been tested. Due to the size of data available, the transferring of information to the GPU incurred in a larger overhead than the improvement that it has on temporal cost. As a result of that, all training has been done on CPU.

For the metrics and comparison with other researches in the field, we have used the approach in [47] and [14]. Here the authors base the comparison in two parameters:

- IPR (Impostor Pass Rate): The impostor pass rate is the number of outliers instances that are wrongly classified as belonging to the target class (also known as the false positive rate).
- FAR (False Alarm Rate): The false alarm rate is the number of correct tuples that are wrongly classified as faulty tuples (also known as the false negative rate).

Furthermore, with these two parameters, we can create a Err parameter where Err is a pondering of IPR and FAR (a low Err implies a better result than an Err with high value). Thus, the final metric that would be used in our approach will be:

$$Err = \sqrt{IPR^2 + FAR^2} \tag{1}$$

Moreover, once the histogram for each cluster has been obtained, we have plotted the RMSE of the autoencoder output (as previously explained) for three types of variables: training, test and synthetic. Finally, we have obtained the ROC curve [48] (Receiver operating characteristic). With that information, we have chosen the suitable threshold, according to data and the risk appetite of the organization. For choosing that threshold, we have prioritised the correct classification of test tuples (class "correct") instead of synthetic tuples (class "anomalous"). In figures from 9, 10, 11, 12, 13 and 14, we can see the ROC curves for every cluster, and the confusion matrix for that selected threshold. Anomalous tuples have been considered as 1, whilst correct tuples have been considered as 0, from a binary classification point of view.

To sum up, in Table 5 we can see the final results for every cluster after establishing the threshold to

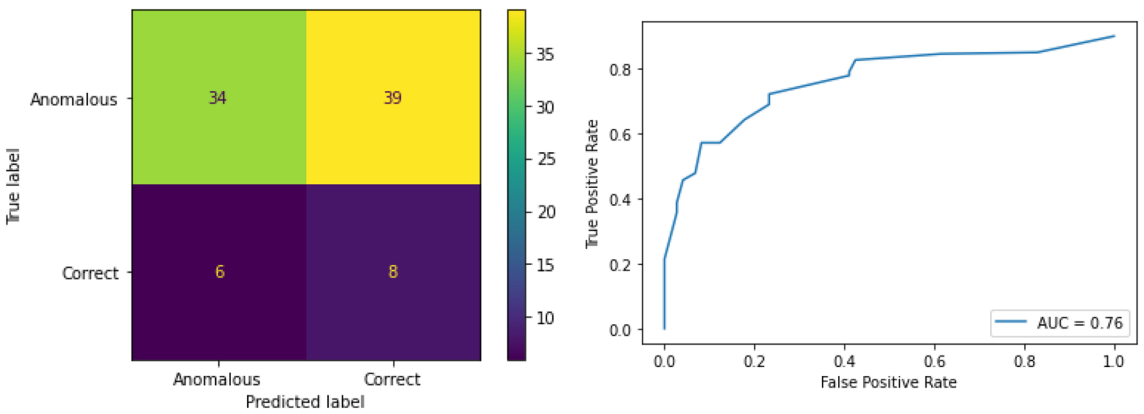

Fig. 9 Confusion matrix (left) and ROC curve (right) for cluster 0

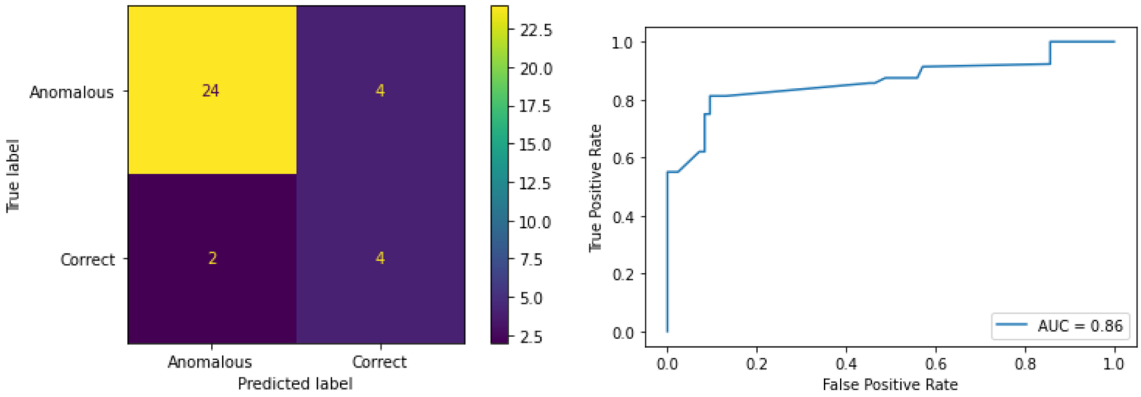

Fig. 10 Confusion matrix (left) and ROC curve (right) for cluster 1

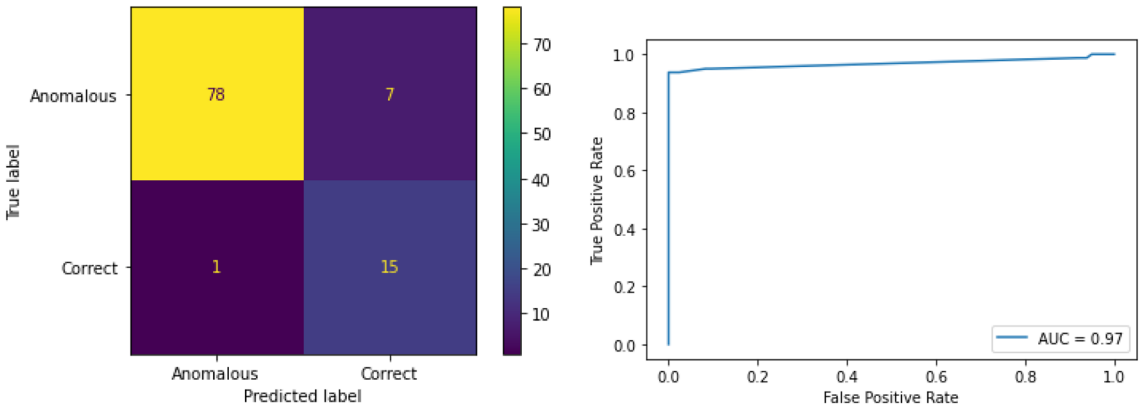

Fig. 11 Confusion matrix (left) and ROC curve (right) for cluster 2

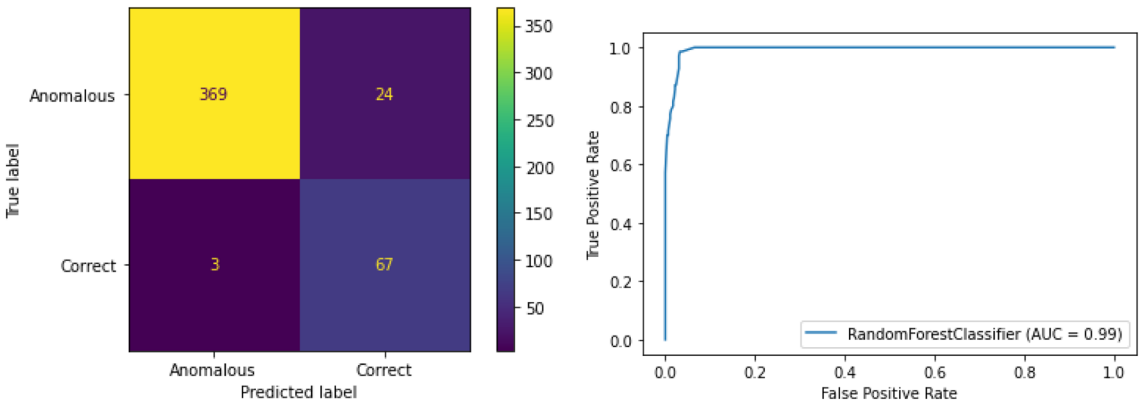

Fig. 12 Confusion matrix (left) and ROC curve (right) for cluster 3

discriminate correct and faulty data as well as the number of samples fore every cluster (for train, test and synthetic partitions). Best results have been highlighted in bold.

As we can see in Table 5, cluster 0 (idle state) is the most difficult to discriminate: this cluster has the highest number of overlapping correct-faulty tuples. The test tuples are prone to be miss-classified with the actual threshold, but only 14 tuples have been assigned to the test group, so this result would not be representative. In addition, the synthetic group has an accuracy of 45.58%. This is due to the most

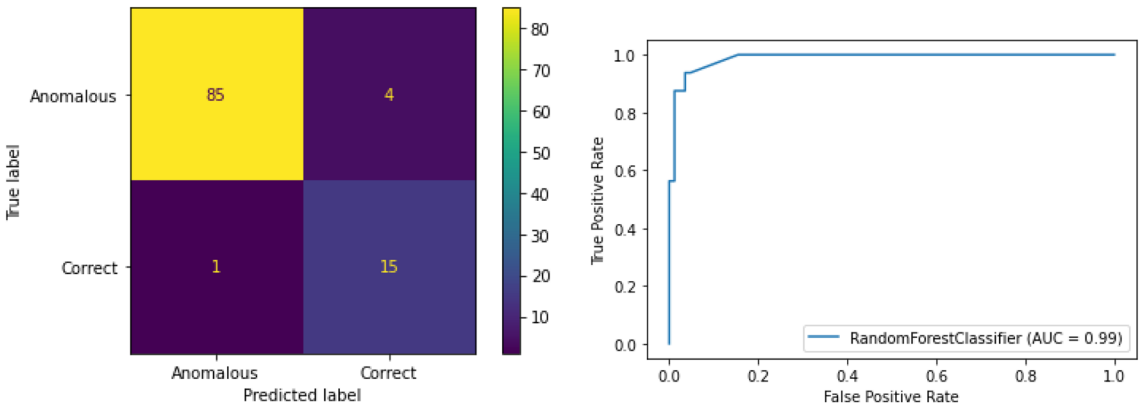

Fig. 13 Confusion matrix (left) and ROC curve (right) for cluster 4

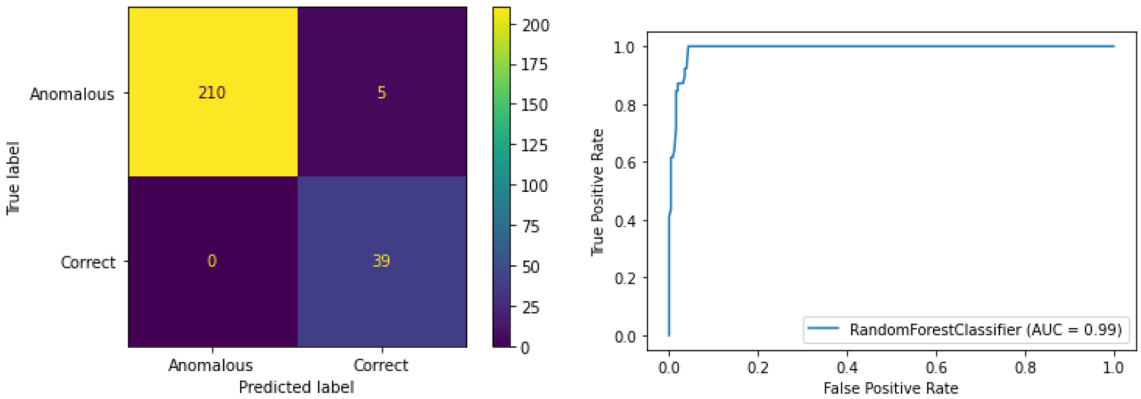

Fig. 14 Confusion matrix (left) and ROC curve (right) for cluster 5

values are really small and maybe the increasing of 200% in one dimension does not provide a faulty tuple. As a result, these tuples may not be faulty, and consequently the 46.58% is not representative.

In cluster 2 (preparation for increasing active load), the number of tuples is not representative, due to that stage is conformed only for 28 tuples (and consequently the test is conformed only for 6 tuples). Despite of that, it seems that differences between correct and faulty tuples are more pronounced that in cluster 0.

In cluster 3 (increasing active load), the results are better than in previous stages: in this stage we have more data, and consequently the autoencoders can discriminate better faulty data and correct tuples, with an accuracy on this stage greater than 94%. In addition, it must be remarked that this stage conforms the nominal performance of the industrial process and therefore, the turbine will be working on this stage most of the time. In this stage only 4 tuples of incorrect data are miss-classified (when the turbine is changing from stage 2 to stage 3, and when it is changing from stage 3 to stage 4)

In cluster 4 (decreasing active load) we can see the best accuracy of the whole system (96,42%), with a solid separation between correct-faulty data.

Furthermore, in cluster 5 we can see a solid separation in between correct-faulty data, with an accuracy greater than 97%.

Thus, the proposal classifies with high accuracy the test and synthetic datasets on representative clusters (2,3,4, and 5). It must be highlighted that cluster 3 belongs to the nominal working process, and this cluster is the most important one.

Finally, in Table 6 we can see the relation between the FAR and the IPR for every cluster.

As we can see, the Err parameter has low values for clusters 2,3,4,5. However, as we have explained before, the values for clusters 0,1, 2 are not representative due to lack of data.

If we compare these results with other in literature, in [14] the authors have an Err value of 0.085 for open stage and 0.129 for close stage. In our model, we have an error slightly better for clusters 3,4,5 and slightly worse (than open stage)

Table 5 Data results and accuracy on clusters

Cluster	Tuple type	Number of tuples	Established Threshold	Estimated wrong classified tuples	Accuracy %
0	Train	73	0.001	0	100.00%
	Test	14		6	57.14%
	Synthetic	73		39	46.58%
1	Train	28	0.012	0	100.00%
	Test	6		2	66.67%
	Synthetic	28		4	85.71%
2	Train	85	0.008	10	88.24%
	Test	16		1	93.75%
	Synthetic	85		7	91.76%
3	Train	393	0.055	10	97.46%
	Test	70		3	**95.71%**
	Synthetic	393		24	**93.89%**
4	Train	89		0	100.00%
	Test	16		1	**93.75%**
	Synthetic	89		4	**95.51%**
5	Train	215	0.005	2	99.07%
	Test	39		0	**100.00%**
	Synthetic	215		5	**97.67%**

Table 6 FAR, IPR and Err for every cluster

Cluster	FAR	IPR	Err
0	0.4286	0.5342	0.6849
1	0.3333	0.1429	0.3627
2	0.0625	0.0824	0.1034
3	0.0429	0.0611	0.0746
4	0.0625	0.0449	0.077
5	0	0.0233	0.0233

for cluster 2. Moreover, in [15] the authors have an accuracy of 88,2% (Err = 0.118) and 96,8% (Err = 0.032). However, their approach is based in classification (supervised learning) and the authors have labeled data to apply that. In our approach we offer a self-trainable system where labeling is not necessary (although we cannot offer the error discrimination type that the authors can offer in that paper)

In [16] the authors have an accuracy of 86,5% whereas in our approach our accuracy for clusters 3,4,5 is greater than 90%. However, it should be pointed that the authors do a discrimination of error types. In [19] the authors have an accuracy of 99.82%. Nevertheless, it must be highlighted that they are not facing the different stages and the authors tackle the problem as a unique distribution function. In [20] the authors show results in failure detection between 85.07% to 94.93%. Our approach overperforms almost every test whilst they are tackling the problem of zero-shot learning. In addition, in [24] the authors present different FAR values, according to different datasets. Those values are between 1% and 10%. Our approach has better results in some clusters (2, 3, 4 and 5) and worse results in other clusters (0, 1). Moreover, in [22] the authors show an accuracy between 67.9% and 100% in a 6-phases classification problem. In our proposal, we achieve an accuracy between 88.24% and 100% for representative clusters (2,3,4, and 5). Furthermore, we provide the errors in miss-classification for test tuples and synthetic tuples. Finally, in [23] authors show an average accuracy of 95.89% for a test dataset, whilst in our proposal we have a weighted-accuracy of 96.45% for the test dataset, and a weighted-accuracy of 94.88% for the synthetic dataset.

It must be pointed that the Err parameter that we have established must vary and be tailored depending on the industrial process. In this approach, IPR and FAR have the same weight, but in some cases one of them could be more important than the other one: for instance, if we think in an industrial process that provide us the correctness of a crucial and cheap piece of a plane, a high value of FAR would imply to discard a correct piece (just a little economic loss); however, a high value of IPR would imply to accept a crucial piece that is not correct, and to put this piece on a plane with customers. In that case, obviously IPR should have more weight that FAR.

The results presented have their limitations. First of all, malfunctioning data for the testing stage has been synthesized. This is due to the fact that no real faulty data was available, and it was important to evaluate the response and tolerance of the system. In this sense, we must highlight that this does not invalidate or alter at all the model obtained, as the training is done using only real data from normal operations. Furthermore, the test stage involves both real (normal operations) and synthetic (malfunctioning) tuples.

In order to create that faulty data, we have used the real data tuples on the dataset to create additional faulty tuples, adding noise and increasing the value randomly of one of the dimensions between 25–100% its nominal value for that stage. With that, we are trying to check if the autoencoder can detect tuples that deviated from learned distribution.

Aside from the considerations on synthetic malfunctioning tuples, we consider that the results for clusters 0 and 1 cannot be deemed conclusive since little data is available. Since the model has been trained with just one working cycle, it is expected that additional working cycles would increase its precision and accuracy. Thus, the more correct data available, the easier it will be to establish a more precise threshold to separate between correct and faulty tuples.

6 Conclusions

In this paper we have tackled the initial step for predictive maintenance by carrying out Fault Detection and Diagnosis (FDD), identifying malfunctioning in complex stage-based industrial processes. To this aim, we have proposed an architecture that combines clustering with autoencoders for automatically identifying the existing stages in an industrial process. Compared to the state of the art, our approach presents several advantages:

1. It can deal with processes where a clear-cut time division between stages is not feasible.
2. It is able to identify malfunctioning using only data from normal operations, thus avoiding costly or potentially unaffordable generation of malfunction data.
3. It is able to identify malfunctioning even if magnitude relationships vary from one stage to another.

Our approach has been tested with real data coming from a case study based on gas turbines, obtaining promising results. This serves as a solid first step for predictive maintenance in complex stage-based industrial processes.

Regarding the limitations of our approach, while the approach should be easily applied to other industrial cases, we are still working on testing its performance on other contexts, and identifying potential problems that did not appear when working with the gas turbines. On the other hand, we are also working on moving the management of sequences of tuples from sliding windows to recurrent neural networks (RNN) to test whether they improve the results.

As part of our future work, we plan to continue working on predictive maintenance, using the data generated by identifying malfunctioning to train a predictive algorithm. Furthermore, we also intend to analyze the impact of Data Quality on the capabilities of the system in order to establish quality rules that improve the training of the machine learning algorithms and the overall system.

Acknowledgements This paper has been co-funded by the ECLIPSE-UA project (RTI2018-094283-B-C32) and AETHER-UA (PID2020-112540RB-C43) from the Spanish Ministry of Science, Innovation and Universities, CDTI (DQIoT, Project No. INNO-20171086) and EUREKA (Project No. E!11737); both Jose M. Barrera (I-PI 98/18) and Alejandro Reina (I-PI 13/20) hold an Industrial PhD Grants co-funded by the University of Alicante and the Lucentia Lab Spin-off Company.

Author Contributions Conceptualization, JMB, AR, AM and JCT; Data curation, AR; Formal analysis, JMB and AR; Methodology, JMB, AR, AM and JCT; Project administration, AM; Resources, JCT; Software, JMB, AR, AM and JCT; Validation, JMB and AR; Writing - original draft, JMB and AR; Writing - review & editing, AM and JT.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability The data that support the findings of this study are available from Siemens Energy but restrictions apply to the availability of these data, which were used under licence for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of Siemens Energy

References

1. Kamble SS, Gunasekaran A, Sharma R (2018) Analysis of the driving and dependence power of barriers to adopt industry 4.0 in Indian manufacturing industry. Comput Ind 101:107–119. https://doi.org/10.1016/j.compind.2018.06.004
2. Qin SJ (2012) Survey on data-driven industrial process monitoring and diagnosis. Annu Rev Control 36(2):220–234. https://doi.org/10.1016/j.arcontrol.2012.09.004
3. Fraser K, Hvolby HH, Tseng TLB (2015) Reliability paper Maintenance management models: A study of the published literature to identify empirical evidence a greater practical focus is needed. Int J Qual Reliab Manag 32(6):635–664. https://doi.org/10.1108/IJQRM-11-2013-0185
4. hu Li B, cun Hou B, tao Yu W, bing Lu X, wei Yang C (2017) Applications of artificial intelligence in intelligent manufacturing: a review (jan). https://doi.org/10.1631/FITEE.1601885
5. Yu J, Zheng X, Wang S (2019) Stacked denoising autoencoder-based feature learning for out-of-control source recognition in multivariate manufacturing process. Qual Reliab Eng Int 35(1):204–223. https://doi.org/10.1002/qre.2392
6. Langarica S, Núñez F (2004) Dual Blind Denoising Autoencoders for Industrial Process Data Filtering arXiv:2004.06806
7. Luo X., Shang M, Li S (2017) Efficient Extraction of Non-negative Latent Factors from High-Dimensional and Sparse Matrices in Industrial Applications, Institute of Electrical and Electronics Engineers (IEEE), pp 311–319. https://doi.org/10.1109/icdm.2016.0042
8. Shang C, You F (2019) Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data Era (dec). https://doi.org/10.1016/j.eng.2019.01.019
9. Precup RE, Angelov P, Costa BSJ, Sayed-Mouchaweh M (2015) An overview on fault diagnosis and nature-inspired optimal control of industrial process applications. Comput Ind 74:1–16. https://doi.org/10.1016/j.compind.2015.03.001
10. Chen R, Huang X, Yang L, Xu X, Zhang X, Zhang Y (2019) Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform. Comput Ind 106:48–59. https://doi.org/10.1016/j.compind.2018.11.003
11. Chen KY, Chen LS, Chen MC, Lee CL (2011) Using SVM based method for equipment fault detection in a thermal power plant.

Comput Ind 62(1):42–50. https://doi.org/10.1016/j.compind.2010.05.013
12. Liu Y, Zhou G (2012) Key Technologies and Applications of Internet of Things. In: 2012 Fifth International Conference on Intelligent Computation Technology and Automation, IEEE, pp 197–200. https://doi.org/10.1109/ICICTA.2012.56. http://ieeexplore.ieee.org/document/6150221/
13. Liu Y, Yang Y, Lv X, Wang L (2013) A Self-Learning Sensor Fault Detection Framework for Industry Monitoring IoT. Mathematical Problems in Engineering. https://doi.org/10.1155/2013/712028
14. Ribeiro RP, Pereira P, Gama J (2016) Sequential anomalies: a study in the Railway Industry. Mach Learn 105(1):127–153. https://doi.org/10.1007/s10994-016-5584-6
15. Park P, Di Marco P, Shin H, Bang J Fault detection and diagnosis using combined autoencoder and long short-term memory network, Sensors (Switzerland) 19 (21). https://doi.org/10.3390/s19214612
16. Qiu Y, Dai Y (2019) A Stacked Auto-Encoder Based Fault Diagnosis Model for Chemical Process, in: Computer Aided Chemical Engineering, Vol. 46, Elsevier B.V., pp 1303–1308. https://doi.org/10.1016/B978-0-12-818634-3.50218-6
17. Gülçehre Ç, Bengio Y Knowledge Matters: Importance of Prior Information for Optimization arXiv:1301.4083
18. Klose A, Kruse R (2005) Semi-supervised learning in knowledge discovery. Fuzzy Sets Syst 149(1):209–233. https://doi.org/10.1016/j.fss.2004.07.016
19. Wen L, Gao L, Li X (2019) A new deep transfer learning based on sparse auto-encoder for fault diagnosis. IEEE Trans Syst Man Cybern Syst 49(1):136–144. https://doi.org/10.1109/TSMC.2017.2754287
20. Huang K, Wu Y, Wang C, Xie Y, Yang C, Gui W (2021) A projective and discriminative dictionary learning for high-dimensional process monitoring with industrial applications. IEEE Trans Industr Inf 17(1):558–568. https://doi.org/10.1109/TII.2020.2992728
21. Feng L, Zhao C (2021) Fault description based attribute transfer for zero-sample industrial fault diagnosis. IEEE Trans Industr Inf 17(3):1852–1862. https://doi.org/10.1109/TII.2020.2988208
22. Jan SU, Lee YD, Koo IS (2021) A distributed sensor-fault detection and diagnosis framework using machine learning. Inf Sci 547:777–796. https://doi.org/10.1016/J.INS.2020.08.068
23. Hu L, Dai G (2022) Estimate remaining useful life for predictive railways maintenance based on LSTM autoencoder. Neural Computing and Applications 2022:1–12. https://doi.org/10.1007/S00521-021-06051-1. https://link.springer.com/article/10.1007/s00521-021-06051-1
24. Huang K, Wen H, Zhou C, Yang C, Gui W (2020) Transfer dictionary learning method for cross-domain multimode process monitoring and fault isolation. IEEE Trans Instrum Meas 69(11):8713–8724. https://doi.org/10.1109/TIM.2020.2998875
25. Mallak A, Fathi M Sensor and component fault detection and diagnosis for hydraulic machinery integrating lstm autoencoder detector and diagnostic classifiers, Sensors 21 (2). https://doi.org/10.3390/s21020433. https://www.mdpi.com/1424-8220/21/2/433
26. Ge X, Wang B, Yang X, Pan Y, Liu B, Liu B (2021) Fault detection and diagnosis for reactive distillation based on convolutional neural network. Computers and Chemical Engineering 145:107172. https://doi.org/10.1016/j.compchemeng.2020.107172. https://www.sciencedirect.com/science/article/pii/S0098135420309741
27. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193. https://doi.org/10.1007/s40745-015-0040-1
28. Hossin M, Sulaiman (2015) A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manag Process (IJDKP) 5 (2). https://doi.org/10.5121/ijdkp.2015.5201
29. Rousseeuw PJ (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20 (C) 53–65. https://doi.org/10.1016/0377-0427(87)90125-7
30. Jain AK Data Clustering: 50 Years Beyond K-means, in: Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–4. https://doi.org/10.1007/978-3-540-87479-9_3
31. Sander J, Ester M, Kriegel HP, Xu X (1998) Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. Data Min Knowl Disc 2(2):169–194. https://doi.org/10.1023/A:1009745219419
32. Murtagh F (1983) A Survey of Recent Advances in Hierarchical Clustering Algorithms. Comput J 26(4):354–359. https://doi.org/10.1093/comjnl/26.4.354
33. Chellappa R, Veeraraghavan A, Ramanathan N, Yam C.-Y, Nixon MS, Elgammal A, Boyd JE, Little JJ, Lynnerup N, Larsen PK, Reynolds D (2009) Gaussian Mixture Models, in: Encyclopedia of Biometrics, Springer US, pp. 659–663. https://doi.org/10.1007/978-0-387-73003-5_196
34. Kennedy J, Eberhart R bls Gov, Particle Swarm Optimization, Tech. rep
35. James Miranda LV,PySwarms: a research toolkit for Particle Swarm Optimization in Python Software • Review • Repository • Archive https://doi.org/10.21105/joss.00433. https://doi.org/10.21105/joss.00433
36. Chai T, Draxler RR (2014) Root mean square error (RMSE) or mean absolute error (MAE)?-Arguments against avoiding RMSE in the literature. Geosci. Model Dev 7:1247–1250. https://doi.org/10.5194/gmd-7-1247-2014. www.geosci-model-dev.net/7/1247/2014/
37. Shafi I, Ahmad J, Shah SI, Kashif FM (2006) Impact of varying neurons and hidden layers in neural network architecture for a time frequency application, in. IEEE International Multitopic Conference 2006:188–193. https://doi.org/10.1109/INMIC.2006.358160
38. Jin L, Kuang X, Huang H, Qin Z, Wang Y (2005) Study on the overfitting of the artificial neural network forecasting model. Acta Meteor Sin 19(2):216–225
39. Abbas Q, Haider Bangyal W, Ahmad J (2013) The Impact of Training Iterations on ANN Applications Using BPNN Algorithm, International Journal of Future Computer and Communication 567–569 https://doi.org/10.7763/ijfcc.2013.v2.228
40. Patroumpas K, Sellis T (2006) Window specification over data streams, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4254 LNCS, Springer Verlag, pp. 445–464. https://doi.org/10.1007/11896548_35
41. Mitrevski A, Plöger PG Data-Driven Robot Fault Detection and Diagnosis Using Generative Models: A Modified SFDD Algorithm, Tech. rep. https://github.com/
42. Zhang L, Lin J, Karim R (2017) Sliding Window-Based Fault Detection From High-Dimensional Data Streams. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47(2):289–303. https://doi.org/10.1109/TSMC.2016.2585566
43. Dorri A, Kanhere SS, Jurdak R Blockchain in internet of things: Challenges and Solutions arXiv:1608.05187
44. Koshy P, Babu S, Manoj BS (2020) Sliding Window Blockchain Architecture for Internet of Things. IEEE Internet Things J 7(4):3338–3348. https://doi.org/10.1109/JIOT.2020.2967119
45. Perez-Castillo R, Carretero A, Caballero I, Rodriguez M, Piattini M, Mate A, Kim S, Lee D (2018) DAQUA-MASS: An ISO 8000–61 Based Data Quality Management Methodology for

Sensor Data. Sensors 18(9):3105. https://doi.org/10.3390/s18093105. http://www.mdpi.com/1424-8220/18/9/3105
46. Che S, Boyer M, Meng J, Tarjan D, Sheaffer JW, Skadron K (2008) A performance study of general-purpose applications on graphics processors using CUDA. Journal of Parallel and Distributed Computing 68(10):1370–1380. https://doi.org/10.1016/j.jpdc.2008.05.014
47. Hempstalk K, Frank E, Witten IH LNAI 5211 - One-Class Classification by Combining Density and Class Probability Estimation, Tech. rep
48. Fawcett T (2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874. https://doi.org/10.1016/J.PATREC.2005.10.010

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com