Machine learning for anomaly detection and process phase classification to improve safety and maintenance activities

Elena Quatrini*, Francesco Costantino, Giulio Di Gravio, Riccardo Patriarca

Department of Mechanical and Aerospace Engineering, University of Rome "La Sapienza", Via Eudossiana, 18, 00184 Rome, Italy

(elena.quatrini@uniroma1.it, francesco.costantino@uniroma1.it, giulio.digravio@uniroma1.it, riccardo.patriarca@uniroma1.it)

**Corresponding author: Via Eudossiana, 18, 00184 Rome, Italy +39 0644585260; elena.quatrini@uniroma1.it*

Abstract

Anomaly detection is a crucial aspect for both safety and efficiency of modern process industries. This paper proposes a two-steps methodology for anomaly detection in industrial processes, adopting machine learning classification algorithms. Starting from a real-time collection of process data, the first step identifies the ongoing process phase, the second step classifies the input data as "Expected", "Warning", or "Critical". The proposed methodology is extremely relevant where machines carry out several operations without the evidence of production phases. In this context, the difficulty of attributing the real-time measurements to a specific production phase affects the success of the condition monitoring. The paper proposes the comparison of the anomaly detection step with and without the process phase identification step, validating its absolute necessity. The methodology applies the decision forests algorithm, as a well-known anomaly detector from industrial data, and decision jungle algorithm, never tested before in industrial applications. A real case study in the pharmaceutical industry validates the proposed anomaly detection methodology, using a 10 months database of 16 process parameters from a granulation process.

Keywords: Machine Learning; Condition-Based Maintenance; Condition Monitoring; Random Forests; Decision Jungles.

1. Introduction

Condition monitoring and anomaly detection are important topics in industrial productions. Condition monitoring is the process of monitoring the trend of one or more meaningful parameters that determine a machine performance, in order to identify changes that can indicate a feasible fault; anomaly detection refers to the problem of finding patterns in data that do not conform to expected [1]. The prevention of accidents is one of the most important goals of condition monitoring and anomaly detection; in fact, anomaly detection has a key role in ensuring reliability and safety in industrial processes, creating an alert system of dangerous situations to stop or control [2]. An uncorrected trend of a parameter can be strictly correlated with the wear of a component that can involve, without correct maintenance interventions, the loss of containment of dangerous material or the out of control and unexpected growth of temperature with the risk of explosion. As a further example, condition monitoring and anomaly detection of the correct air inflow can detect a dirty air channel, with the insurgency of overheating and the risk of conflagration. Research about anomaly detection is important for several other reasons: forecast a future stop or a future

downtime minimize overall maintenance costs [3], reduce the costs associated to the loss of production caused by machine downtime [4], and increase product quality in order to give more competitiveness in the global marketplace to a company [5]. In this paper, the anomaly detection problem is stated as a classification problem; the analysis of data identifies to what classes of criticality the monitored industrial process belongs to: “Expected”, “Warning”, “Critical”. For the suggested methodology the authors have decided not to consider the ordinality and the correlation of these classes, losing some information. This consideration will be an interesting point of reflection for future researches and future improvements in the methodology. The difficulty to link data from the field to the specific process phase is a common criticality of anomaly detection in a real industrial process. This problem applies to anomaly detection in multiphase processes, because of the inability to understand if the values of the parameters are expected because they can vary depending on the process phase. This assumption refers to the analysis of inhomogeneous time series. Inhomogeneous time series refers to some characteristics of the production process: the length of both process phases and production cycles can vary during the time; the order of the production phases is always the same, but a production cycle may not be complete due to tests or failures. The monitored parameters are always the same as well as their sampling interval. So, every time the process phase is not known in the system that analyses the parameters data, condition monitoring, and anomaly detection probably fails. In actual manufacturing systems, process data unlinked to the industrial process phase is a common situation, thus companies need methods to detect anomalies with a prior automatic recognition of the process phase. With this aim, the paper presents a two steps methodology to solve this multiclass classification situation. Both classifications have more than two output classes: the first classification has as many classes as the process phases, the second classification has three criticality classes. The authors consider as anomalies all the situations that do not represent the expected functioning of the production process, and the criticality classes take into account the distance from compliance requirements and company target levels.

The remainder of the paper is organized as follows. Section 2 presents the related literature. Section from 3 to 5 present the different algorithms tested is the case study. Section 6 presents a general methodology to approach situations like this and in Section 7 is presented a case study. The discussion of the results, the conclusions, and future research are summarized in Section 8.

2. Literature Review

According to [1] anomaly detection has some important fields of application and it is widely discussed in different areas of research, e.g. Cyber-Intrusion Detection, Fraud Detection, Medical Anomaly Detection, Industrial Damage Detection, Image Processing, Textual Anomaly Detection, and Sensor Networks. The main techniques of anomaly detection are Classification Based, Clustering Based, Nearest Neighbor Based, Statistical, Information Theoretic, and Spectral. This paper fits in the context of anomaly detection, specifically considering Industrial Damage Detection, using Classification Based Anomaly Detection. It is not possible to apply directly what is proposed in different areas of application: an anomaly takes on meaning in the specific context of the event, requiring a specific field contextualization for its recognition. Consequently, a relevant problem in the anomaly detection process is the lack of contextual information for identifying the cause of a problem and the domain expertise is necessary to solve it [6]. In fact, the contextualization of an anomaly increases the interpretability of the found outliers [7], because what is anomalous in a context could be not anomalous in another one [8]. It is necessary to

consider that the production process analyzed in this paper is a process that operates "under time-varying conditions", composed by seven different sub-processes, i.e. seven different process phases. In this kind of process, a relevant problem is that normal changes in the operational condition can be interpreted as faults, producing a large number of false alarms [9]. Traditional stationary signal processing techniques cannot be applied effectively in a time-varying conditions context [10]. Due to the time-varying conditions both for the operating conditions and the healthy or faulty state, can be necessary to properly generate data related to the operating context of the analyzed system [11].

The role of the domain experts in the anomaly detection context is crucial even for the definition of the boundaries between normal and anomalous behavior as well as for the economic analysis of the anomalies impact [12] and the extraction of useful details from the raw signals [13].

The interest in anomaly detection is substantial and ever-changing in recent years. Several applications provide insights for data security, e.g. the recent papers on botnet traffic [14], data stream of network infections [15], physical intrusions in building [16]. Safety applications of anomaly detection are not common: some researches forecast specific hazards, e.g. earthquake [17], or apply anomaly detection methods on accident reports to identify risk factors and types of anomaly conditions [18].

For condition monitoring, in last years the interest has been in various [19–22], e.g. vertical form seal and fill machine [23], rotating machines [9], electrical motors [24], continuous stirred tank reactor system and an industrial centrifugal pump [25], bearings [26], gearbox [27], crankshaft forging [20], repetitive machining operations [28], serial multistage manufacturing systems [29], electrical discharge machining [30].

In industrial production, anomaly detection is widely applied for condition monitoring and to get relevant alerts for predictive or on-condition maintenance. Studies on specific types of machinery identify anomalies from the monitoring of industrial parameters, e.g. vibration [31,32] or temperature, pressure, flow, energy consumption [33]. Some recent interesting applications of anomaly detection are in the automotive sector, on CNC machines [34], and in the press-hardening process of automotive components [35]. The energy sector is deeply studied because a lot of data is collected from resources distributed over large areas, such as in wind turbines [36,37] and photovoltaic plants [38], or in production plants [39,40]. Some other recent industrial applications of anomaly detection can be found in the chemical sector [41–44], in the food industry [45], ultraprecision machining processes [46] or for automated quality control in manufacturing processes [47]. Recently, an anomaly detection method has been successfully applied in detecting industrial control system cyber-attacks [48], linking security to the monitoring of industrial process data.

All these applications of anomaly detection consider just one process phase, without the need to interpret the data collected depending on the process phase in which it was generated. However, this scenario is very common in production systems, typically in batch processes, where often the time intervals of process phase are not fixed. Due to this lack of experience in the interpretation of the multiphase processes is difficult to properly identify anomalies from real-time data, ensuring a low number of false alerts.

Research about anomaly detection is achieving excellent results with the Random Forest Algorithm (RFA), which is the most popular Decision Forest prediction model [49]. RFA achieved good results for fault detection in industrial chemical processes [50,51]. RFA applications are available for real-time predictive maintenance systems in data centers [52], energy transmission systems [42], aircraft components [53], processing plants [54,55], supply systems [56]. The high-

level performance of RFA is presented even in the most common application of anomaly detection, which is the intrusion detection and the prevention of data breach [57].
This paper applies a two steps methodology using classification algorithms that properly identifies the process phase from a set of gathered data, and then performs an anomaly detection through a machine learning model that considers the specific process phase. Based on the literature, the anomaly detection and production process monitoring methodologies consider the process phase known or focus on a specific process phase. These approaches are not suitable for production processes in which the same machine carries out several production phases, sometimes all of them; the condition monitoring receive data without the specific contextualization of the measurements nature (label of the belonging process phase). The proposed methodology innovates suggesting a sequential approach, in which the second step, i.e. the anomaly detection, relies on the information gained with the first step, i.e. the process phase labeling. The paper proposes a comparison between the anomaly detection implemented with and without the proposed methodology. This comparison underlines the superiority and consequently the need to apply this methodology. The algorithms of the second step have been chosen from literature considerations, and these are not limiting factors of the methodology. Firstly, the aim is to evaluate the performance of the well known and tested algorithm in the field of anomaly detection, that is the RFA, with the proposed methodology and in the specific context of the application, frequently used for classification problems.
Since the RFA models sometimes bring to a high memory footprint, a valid alternative is Decision Jungle Algorithm (DJA). This latter is an evolution of RFA [58], that revisits the idea of ensembles of rooted decision Directed Acyclic Graphs (DAGs), and requires dramatically less memory than RFA. The application of DJA is a recent field of research, focused on medical classification problems: good results are available for the disease prediction [59], for the fetal health status prediction [60], for the classification of patients and suggestion of suitable diet [61]. So, the authors want to verify the applicability and the performance of the DJA too, that has never been applied in industrial anomaly detection; in fact, the memory footprint can be considered a limited resource and can be important to have a valid alternative for the algorithm.
As far as we know, moreover, applications, and comparisons of RFA and DJA in industrial anomaly detection are not currently available in the literature. All the considerations obtained by the application of these two algorithms must be considered as an additional output of the paper, but not its focus. Future researches will apply the same methodology with different algorithms, verifying the obtained results.
The paper provides a methodology to achieve anomaly detection with real-time data from a multiphase industrial process, using firstly the RFA and secondly DJA.

3. Decision tree

In machine learning, a decision tree is a predictive model where each internal node is a decision node and it represents a predictive variable (feature), an arc to a child node represents a possible value range for that variable, an external node (leaf) represents the predicted value for the target variable, a classification or decision.

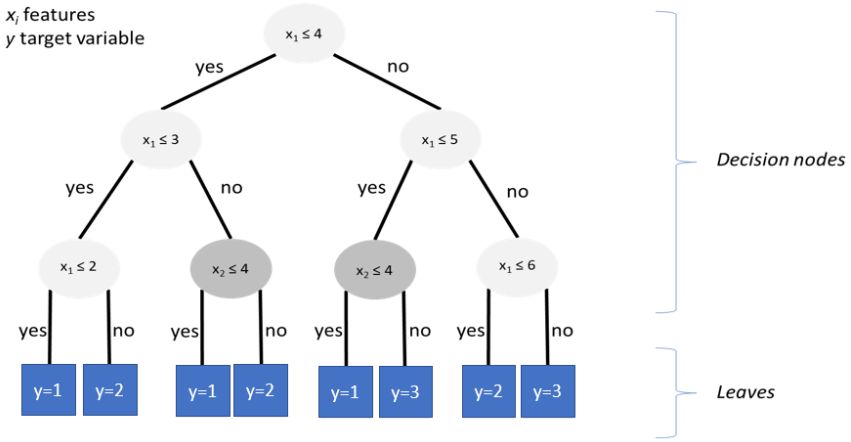

Figure 1. *Example of a decision tree*

The decision tree uses a series of if-then statements to identify boundaries and define patterns in the data. Every internal node split the data into two branches based on a split value, that moves the decision to a further node.

Figure 1 shows a decision tree, where x_1 and x_2 are the variables, 4 is one of the split values in the internal nodes (e.g. in the dark grey nodes), the target value to predict is in the leaves *y=1* or *y=2* or *y=3*.

The order of decision node in a tree corresponds to the best predictor capacity. A split point is the decision tree's version of a boundary. There are several methods to choose the best split [62].

4. Random Forest Algorithm

The RFA is an ensemble learning method for classification. The first idea of Random Forest was introduced independently and almost simultaneously by two research teams [63,64]. It was later revised and popularized from 2001 [65], with a subsequent unified model of RFA [66].

The analytical formulation of the algorithm presented in this section follows the research by [66]. The algorithm works by building multiple decision trees and then voting on the most popular output class.

In classification problem [66], there is an association between a data point $v = \{x_1, \dots, x_d\} \in F$, where F is the feature space with a dimension d, with a discrete class $c \in C, C = \{c_k\}$, $k = 1,\dots,K$, K= output classes. Thus, the goal of the training is to understand how to label every input data v in the dataset with c. In this situation, the target is to determine the posterior distribution $p(c|v)$, that is the probability of the onset of c when the input is v. Forests can easily handle classifications with more than two classes, they produce probabilistic outputs, they generalize in a very good way new data and they reduce tests for data point; the outputs are always discrete and categorical. $\theta_j \in \Gamma_j$ denote the parameters of the test function h at the j-*th* split node, with $j = 1, \dots, J$, J = number of decision nodes; the parametrization of the test function model is $\theta_j = (\varphi, \psi, \tau)$, where ψ defines the geometric primitive used to separate the data, the parameter vector τ captures thresholds for the inequalities used in the binary test and the filter function φ selects some features of choice out of the entire vector v.

Γ is the entire space of the parameters of the test function. Optimizing over Γ is not feasible, because the size of Γ can be extremely large. Instead, when training at the *j-th* node it is available only a small random subset $\Gamma_j \subset \Gamma$ of parameter values.
Training optimizes the parameters of the test function at each split node *j, j= 1, ..., J, J* = number of split nodes, via (1):

$$\theta_j^* = \underset{\theta_j \in \Gamma_j}{argmax}\, I_j \qquad (1)$$

The data point *v* arriving at the node is sent to its left or right child node according to the result of the test function; this test is a function with binary outputs $h(v, \theta_j): F\ x\ \Gamma \rightarrow \{0, 1\}$, where 0 and 1 can be interpreted as "false" and "true" respectively.
The objective function I_j takes the form of a classical information gain defined for discrete distributions (2):

$$I_j = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} H(S_j^i) \qquad (2)$$

With *i* indexes the two child nodes and *j* indexes the parent node, with *i* = L, R (Left and Right), *j= 1, ..., J*.
The symbols S_j, S_j^L e S_j^R denote the sets of training points before (S_j) and after the split, on the left (S_j^L) and on the right (S_j^R).
The entropy of a training set *S* is (3):

$$H(S) = -\sum_{c \in C} p(c)\ log\, p(c) \qquad (3)$$

Training every classification tree by maximizing the information gain involves that the entropy of the class distributions associated with nodes decreases; it means that the prediction confidence increases, from the roots to the leaves. *p(c)* is estimated as the normalized empirical histogram of labels corresponding to the training points in *S*, that is the training set.
The unified model of RFA [66] recommends the use of the entire dataset, but it does not exclude the use of bagging; this reduces the training time.
So, the randomness model is structured in two phases:

1. Bagging, which is also called Bootstrap AGGregatING, is an ensemble learning, it means a methodology that allows you to combine multiple models (typically decision trees for classification, both binary and multiclass tasks, and regression) and to aggregate prediction results in order to obtain final estimates. The most common aggregation techniques include max voting for classification models and averaging or weighted averaging for regression models. Intuitively, the idea behind the ensemble algorithms is to emulate common "human" decision-making processes, combining the decisions that result from multiple models to improve overall performance. Often, bagging produces a combined model that outperforms the model that is built using a single instance of the original data. This statement is true especially for unstable inducers such as decision trees since bagging can eliminate their instability [67]. Breiman [65] refers to bagging decision trees as a particular instance of the random forest technique. With bagging the construction of the models takes

place in a non-sequential but parallel manner and each model is independent of the others. Different models are trained on different subsets, each of which is extracted by sampling with replacement from the original training set (bootstrap samples/bags); this sampling ensures that the probability of extraction of each example is equal to that of the others. A base model is then created on each subset and the final estimates are derived from the combination of the prediction results of the individual models. The excellent quality of bagging is that it reduces the variance of the model, but not involving the increase of the bias. This means that the sensitivity of each tree to the noise of its dataset is "compensated" when it comes to considering the trees in their complexity; this happens because the individual trees are not related.

2. Random Decision Forests add a further layer of randomness to bagging. At each node, the optimization is done not concerning the entire parameter space Γ because for large dimensional problems the size of Γ can be extremely large considering that the feature/attribute dimension of each data point can be large. So, during the training at the *j-th* node, we only make available a small random subset $\Gamma_j \subset \Gamma$ of parameter values. So, in training objective function, randomness is injected via randomized node optimization, with $\rho = |\Gamma_j|$ indicating the amount of randomness. It means, for example, that with $\rho = 1$ each split node takes only a single randomly chosen set of values for θ_j.

The output of a classification forest is probabilistic, which is an entire class distribution for the posterior distribution. Practically, in every node there is the execution of a test, that directs in another specific node; tests stop when there is not a new leaf.
Every leaf produces $p_t(c|v)$, that is the posterior distribution, and the forest output is (4):

$$p(c|v) = \frac{1}{T}\sum_t^T p_t(c|v) \quad (4)$$

With $t = 1, \ldots T$, T= number of forest trees

5. Decision Jungle Algorithm

Decision Jungles [58] have been developed to counteract a limit of trees and forests: with a lot of data, nodes in decision trees grow with depth. This growth does not occur in DJA, so this algorithm requires a dramatically less memory footprint than trees and forests*;* it is important because memory is a limited resource.
The analytical formulation of the algorithm presented in this section follows the research by [58]. DJA revises the idea of ensembles of rooted decision directed acyclic graphs (DAGs); decision trees only allow one path to every node, but a DAG allows multiple paths from the root to each leaf (Figure 2).
Using trees, the risk of overfitting can be significant: overfitting occurs when a statistical model lies down on observed data, because of the excessive number of parameters compared to observations. DJA can avoid overfitting, too.
A Rooted Binary Decision DAG is a way to combine a binary classifier to a multiclass classifier; in DAGs there is a One Root Node, with in-degree 0, Multiple Split Nodes, with in-degree ≥1 and out-degree = 2 and there are Multiple Leaf Nodes, with in-degree ≥1 and out-degree = 0.

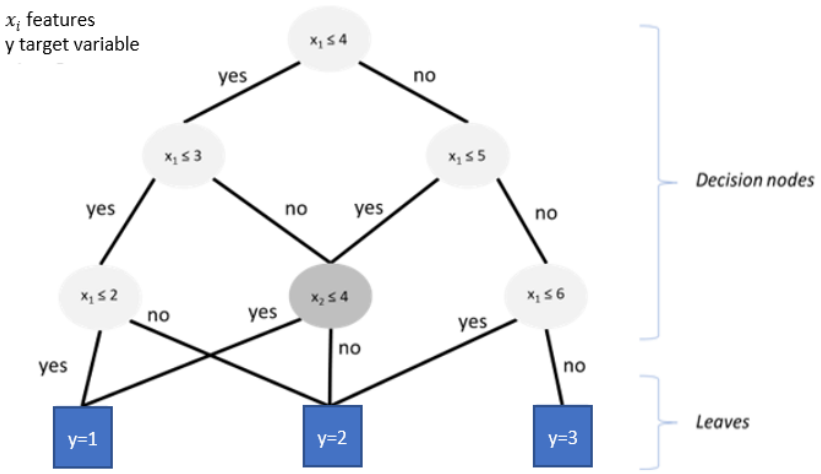

Figure 2. *Example of DAG*

In DJA every DAG is built one level at a time; at each level, the algorithm jointly learns the features and branching structure of the nodes. This is done by minimizing the objective function defined over the predictions made by the child nodes, that are the left and the right nodes reached after the split at *j-th* node, with $j = 1, \ldots, J$, J = number of split nodes.
θ_j denotes the parameters of the split feature function f for parents node $j \in N_p$ and S_j denotes the set of labeled training instances (x,v) that reach node j. With θ_j and S_j it is possible to compute the set of instances from node j to left and right branches (5):

$$\begin{aligned} S_j^L(\theta_j) &= \{(x, y) \in S_j \mid f(\theta_j, x) \leq 0\} \qquad \text{LEFT} \\ S_j^R(\theta_j) &= S_j \backslash S_j^L(\theta_j) \qquad\qquad\quad \text{RIGHT} \end{aligned} \tag{5}$$

l_j denotes the current assignment of the left outwards edge from parent node j to a child node, with $l_j \in N_c$ and $j \in N_p$ and similarly $r_j \in N_c$ for the right outward edge.
N_c and N_p denote respectively a set of child nodes and a set of parent nodes.
After that, the set of instances that reach any child node $z \in N_c, N_c = l_j, r_j$ for the *j-th* split node, is (6):

$$S_z(\{\theta_j\}, \{l_j\}, \{r_j\}) = \left[\bigcup_{j \in N_p \, s.t. \, l_j = z} S_j^L(\theta_j) \right] U \left[\bigcup_{j \in N_p \, s.t. \, r_j = z} S_j^R(\theta_j) \right] \tag{6}$$

The objective function E associated with the current level of the DAG is a function of $\{S_z\}_{z \in N_c}$.
The problem of learning the parameters of the decision DAG can be formulated as a minimization of the objective over θ_j, l_j and r_j. The task of learning the current level of DAG is written as (7):

$$\min_{\theta_{j \in N_p}, \, l_j \in N_c, r_j \in N_c} E(\{\theta_j\}, \{l_j\}, \{r_j\}) \tag{7}$$

The information gain objective requires the minimization of the total weighted entropy of instances, defined as (8):

$$E(\{\theta_j\},\{l_j\},\{r_j\}) = \sum_{z \in N_c} |S_z| H(S_z) \quad (8)$$

S_z has been defined before in (6) and H(*S*) is the Shannon entropy of the class labels *y* in the training instances *(x,y)* $\in$ *S*. The minimization problem (1) is hard to solve exactly; there are two local search-based algorithms for its solution, but this paper presents only the most effective [58], LSearch. The LSearch Method starts from a feasible assignment of the parameters and then alternates between two coordinate descent phases. In the first phase (*split-optimization*), it sequentially goes over every parent node *k* in turn and it tries to find the split function parameters θ_k that minimize the objective function, keeping the values of $\{l_j\},\{r_j\}$ and the split parameters of all other nodes fixed (9):

$$\text{for } k \in N_p$$
$$\theta_k \leftarrow argmin_{\theta'_k} E(\theta'_k \, U \, \{\theta_j\}_{j \in N_p \setminus \{k\}}, \{l_j\}, \{r_j\}) \quad (9)$$

The minimization over θ'_k is done by random sampling similarly to Decision Forest.
In the second phase (*branch-optimization*), the algorithm redirects the branches emanating from each parent node to different child nodes, to yield a lower objective (10), (11):

$$\text{for } k \in N_p$$
$$l_k \leftarrow argmin_{l'_k \in N_c} E(\{\theta_j\}, l'_k \, U \, \{l_j\}_{j \in N_p \setminus \{k\}}, \{r_j\}) \quad (10)$$

$$\text{for } k \in N_p$$
$$r_k \leftarrow argmin_{r'_k \in N_c} E(\{\theta_j\}, \{l_j\} \, r'_k \, U \, \{r_j\}_{j \in N_p \setminus \{k\}}) \quad (11)$$

The algorithm stops when there are no changes and it is guaranteed to converge.
Moreover, Decision Jungles have stronger generalization performance than decision trees, as can be seen in Figure 3:

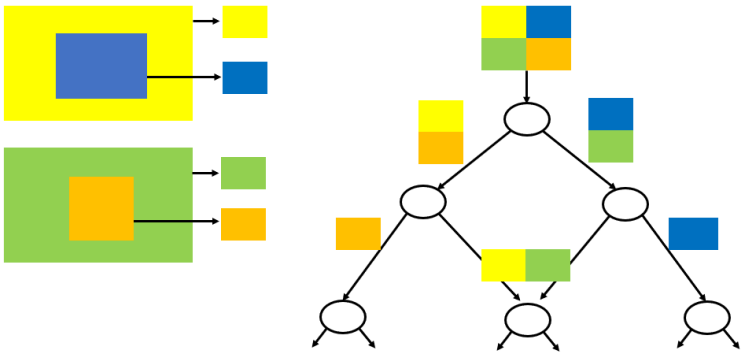

Figure 3. *Generalization of DAG for "background" classification*

Figure 3 shows how a DAG works for classifying image patches as belonging to background, square, or rectangle classes. A stronger generalization means, using the Figure 3 for the explanation, that with DAGs differently colored patches of background (yellow and green), for example, are merged into the same node, because of similar class statistics. So, if the algorithm detects that two nodes are associated with similar class distributions, it merges them, and the model gets a single node with training examples for both the previous nodes. This helps capture the degree of variability intrinsic to the training data and reduce the classifier complexity.

6. Methodology

For the implementation of the anomaly detection system for a multiphase process using supervised classification algorithms, the logical scheme in Figure 4 is proposed.

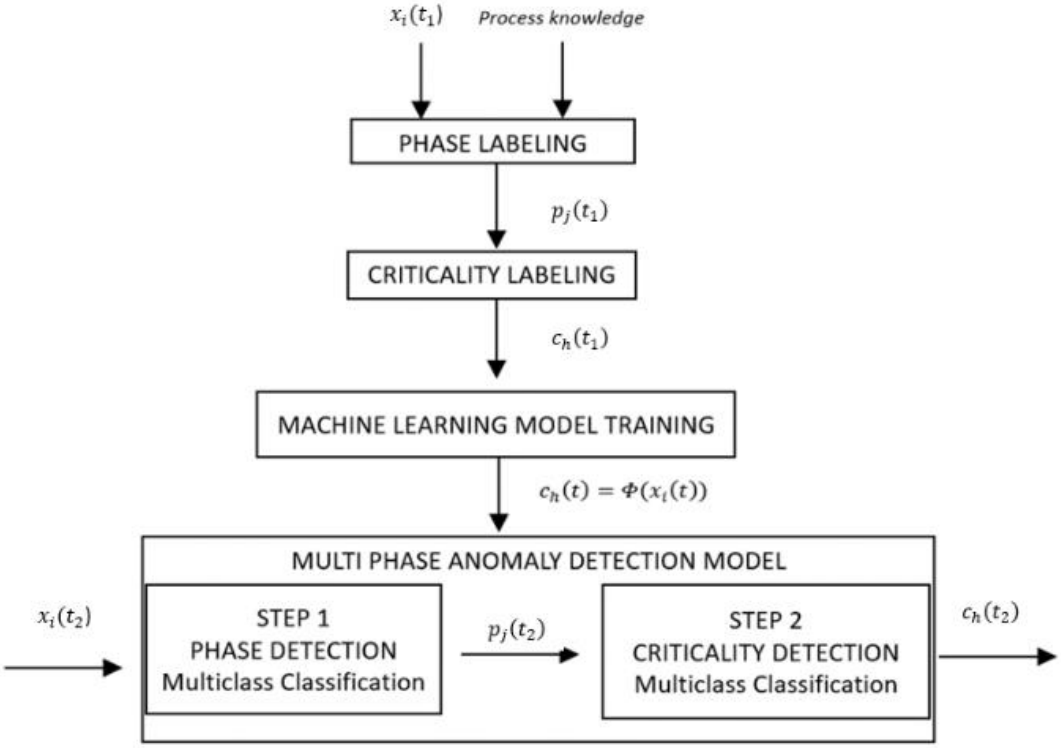

Figure 4. *Methodology logical scheme*

The presented model is a supervised learning model, that works with labeled data samples as inputs to find relations among process parameters (data) and process condition, e.g. anomaly (label). The model uses those relations to classify future unlabeled data. Available data are time-series, with the measurements of some parameters involved in the production process. Thus, the first steps of the methodology consist of build a labeled dataset. This requires a "phase labeling" step and a "criticality labeling" step.
Please, refer to the following notation in the later explanations:

$$t = \begin{cases} t_1 & 0 \le t \le t_{TRAIN} \\ t_2 & t > t_{TRAIN} \end{cases}$$

- $n =$ Measured process parameters
- $x_i(t) =$ Process parameter *i,* measured at time t, with $i \in [1, n]$
- $m =$ Number of process phases
- $p_j(t) =$ Process phase *j* at time t, with $j \in [1, m]$
- $w =$ Number of criticality levels
- $c_h(t) =$ Criticality level at time t, with $h \in [1, w]$
- $x_{kj}(t) =$ Measured significant process parameter to label the process phase *j,* with $k \le n$
- $\Phi\big(x_i(t)\big) =$ Predictive function of $x_i(t)$ for *y*
- $\hat{\imath}$ = Set of parameters among the n process parameters, with $\hat{\imath} = \{h\}, \ h = 1 \dots, I$

In these situations, the comprehension of the production process, or process knowledge, is necessary; this requires interviewing who manages and who carries out the process, considered the experts of the process. The experts of the process have three main important roles:

1. Expected data values and trends depend on what is going on in the process, thus the model requires a logical decomposition of the production cycle. Thus, the experts of the process explain to the data-scientist the process phases, adding meanings to the collected data, values, and trends.
2. They explain the phase signals, that are the combinations of selected parameters and their expected values and trends. The experts of the process guide the data-scientist to select the parameters among all those monitored that should bring information about the anomaly detection, for every phase of the process. The expected values and trends of the selected parameters consider historical values, legal requirements, company standards, good manufacturing practices.
3. They help with the definition of the rules of labeling; the labeling will be explained below.

The experts of the process involved in the development of the methodology are:

- MAINTENANCE MANAGER: he explains and clarifies the maintenance procedures applied and the needs of the plant, as well as the objectives to be pursued with the methodology. He shows to the authors the historical analyses and reports about the accidents and the maintenance actions in the line. He also describes the process requirements levels (e.g. from GMP, safety, and other compliances) and the maintenance requirements levels (e.g. from service continuity, historical costs of interventions).
- PRODUCTION LINE MANAGER: he explains and clarifies the real functioning of the production process. He enables the comprehension of the production process, entering in the details of the specific production phases, and the already known correlation between

process phases and input parameters. He illustrates the consequences of the criticality in each phase on the entire production process, highlighting the most troubling of them. The production line manager is fundamental to fully understand the difficulties in the identification of the process phases.

- IT AREA MANAGER: the role of this manager has been after a full understanding of the production process. He clarifies the condition monitoring and the data recording process. In particular, he explains the measurement frequency, the data aggregation process, the motivation of some missing values, the unavailability of further input parameters. He is the person who physically extracted from the ERP the data for the analysis.

The first three meetings were set individually with the three managers, in the order of maintenance manager, production line manager, and IT area manager. The following two meetings have involved the three experts together, for coordinating the development of the methodology.

At the beginning of the experiment, it is necessary to identify the process parameters, x_i, and to measure them in predefined time-lapse for structure a time-series composed by several $x_i(t_1)$, if this time-series is not available. The time-series will be the initial dataset for the following steps.
The first practical step is to label the dataset with the process phases of the production process. For doing that, it is necessary to define which are the parameters that determine the onset of a phase. This is a qualitative step, because it does not measure or analyze the performance of the phases, but it only determines their sequence.
The process phases can be identified in one or more of the following analysis.

1. $p_j(t_1)$ MEASURED →a historical dataset with the measurements associated with the phases is available. Some ERP (Enterprise Resource Planning) get this information, generally by manual signals from the process worker. With manual signals, the line operator registers firsthand in the ERP system at the beginning of the process phase. This kind of registration has some considerable problems: it is not always suitable, when the process phase is not clearly visible, e.g. where human intervention is not needed, and it is not reliable because of the human factors. A line operator could enter the production phase wrong or not timely. Both these human errors strongly compromise the effectiveness of the following anomaly detection process.
2. $p_j(t_1)$ is obtained for $x_{i\ MIN}(t_1) < x_i(t_1) < x_{i\ MAX}(t_1)$ → the process is in a specific phase when one or more parameters are in specific ranges. It is possible to link the phase to parameter limits, considering legal requirements, company standards, good manufacturing practices. For example, considering a spraying phase, there will be a specific target of the spray percentage that identifies this phase.
3. $p_j(t_1)$ is obtained for $x_i(t_1) = f(t_1)$ → the specific phase is identified by a trend of one or more parameters. For example, the reduction of the temperature can be the symptom of the load in the machine of some materials with a lower temperature than the inside materials.
4. $p_j(t_1)$ is obtained for $x_i(t_1) = g(x_i(t_1 \pm t'))$ → the phase is correlated with the onset of the previous phase, the following phase, or both. The insurgency of a phase can be determined by what is happened previously or what will happen afterward. If the process cycle includes multiple instances of the same phase, but the expected values and trends

depend on the specific instance, it is important to automatically identify the number of its instances. Moreover, sometimes the easiest way to identify the phase is because it always follows another one. For example, a drying phase follows a spraying phase, that moistens the product and therefore makes necessary the drying phase.

Substantially, when $p_j(t_1)$ is not measured yet, it is possible to identify some "if-then-else" rules which determine the link between the measurements and a specific phase.
This if-then-else approach is possible on historical data sets, while for a real-time identification it is generally not a suitable method, mainly for the following reasons:

- In situations in which the identification of the phase depends on a trend of some parameters or it is a function of other phases it is impossible to implement a real-time analysis with if-then-else rules. It becomes possible with a machine learning model;
- With if-then-else rules, it is possible to create a labeled dataset, but when there is the attendance of some anomalies who break the rules it is impossible to determine the phases in real-time. The machine learning model can predict the phase even in these situations;
- When $p_j(t_1)$ is measured, the machine learning model can understand the links between the observed phases and the parameters.

With machine learning, it is possible to enlarge the tolerance to anomalies analyzing the relative behavior of the other parameters. At the end of this step, the result obtained is a labeled dataset with the phase labeling inserts in it.
The second step is the criticality labeling. The analysis of the effective performance of every phase defines the criticality with one of the three following quantitative steps:

1. EVENT DRIVEN → the criticality labeling is available: there is, in fact, a historical dataset of the performances of every phase;
2. REQUIREMENTS DRIVEN → compliance requirements are present as machine target and upper / lower limits. Therefore, the criticality of the phase depends on the position of the value considering the process compliance requirements;
3. STATISTIC DRIVEN → the statistic analysis of the process labels the phase criticality. For every phase, one or more parameters are relevant for the phase performance, with specific thresholds to be defined. The thresholds can be defined in two principal ways: with control charts or with percentiles. The difference between these modalities is mainly their dynamicity. The control charts define a static range, it means that the thresholds of every criticality level are not computed by the system but are predefined based on some statistical analysis. So, for changing the thresholds it is necessary to repeat the statistical analysis for their definition. The percentiles, instead, are dynamic because the thresholds are computed by the system, based on the real values of the parameters measured over time; the percentiles are easily editable, and the system can quickly compute again the new thresholds. A correct and useful criticality labeling, according to us, can be:

    - c_1 = WARNING → the phase trend is slightly different than expected;
    - c_2 = CRITICAL → the phase trend is completely different than expected;
    - c_3 = EXPECTED → the phase trend is coherent with expectations.

The criticality labeling is substantially a function of the input data $x_i(t)$ (12):

$$c_h(t) = \Phi(x_i(t)) \quad (12)$$

The importance of machine learning for real-time anomaly detection system can be summarized in different ways depending on the typology of criticality labeling:

- When the criticality labeling is event-driven, the model is necessary in order to understand the correlation between the event and a specific trend of one or more parameters;
- When the criticality labeling is requirement driven or statistical driven, the model is necessary to understand the correlation between the trend of two or more parameters.

In both the situations, the link between the model and real-time data is present: the machine learning model guarantees the possibility of real-time implementation; likewise, the phase labeling, when the evaluation of the performance requires more than one parameter to be measured, the model during the training provides the correct prediction in real-time. At the end of this step, the output is a labeled dataset with criticality labeling inserts in it. Now it is possible to start the machine learning model training; the model develops the ability to independently and in real-time predict the process phase and its criticality level, through the study of the labeled dataset.
The trained machine learning model is now available for the multiphase anomaly detection system; in fact, the trained model enables to understand the relations between parameters, process phases, and criticality levels. The inputs are the real-time measurements of the machine parameters.
After all these steps the result is a machine learning anomaly detection system that determines the process phase and its criticality level, independently and in real-time, in few seconds.

7. Case study

The case study is structured on a dataset from a pharmaceutical company. It performs a fluid bed granulation. Granulation is a very important process for pharmaceutical companies because it produces tablets and capsule dosage forms, increasing the uniformity of drug distribution in the product and its physical properties. Granulation methods can be divided into two major types: wet methods which utilize some form of liquid to bind the primary particles, and dry methods which do not utilize any liquid [68] The case study belongs to the wet method process. It is an excellent case study because granulation involves important problems with operators and plant safety:

- The high temperatures, the possible introduction of nitrogen and the non-properly management of vacuum can generate explosions or the degradation of the product;
- There is the possibility of pollution, due to the condensation of the evaporated liquid, with high-level risk for the operators;
- The granule breakage entails the emission of powder, with high-level risk for the operators.

In general, pharmaceutical plants every day should consider the risk of emission of toxic material.

Practically, the model gets the industrial process and the product parameters, to identify anomalies and criticalities in production. The dataset consists of 16 parameters, referred to the process or the product; these parameters are:

- x_1 = Date;
- x_2 = Spray Percentage;
- x_3 = Air IN Flow;
- x_4 = Spray Flow;
- x_5 = Air Pressure Spray;
- x_6 = Microclimate Pressure;
- x_7 = Cleaning Pressure;
- x_8 = Air IN Temperature;
- x_9 = Washing Air Temperature;
- x_{10} = Air OUT Temperature;
- x_{11} = Product Temperature;
- x_{12} = Cooling Temperature;
- x_{13} = Absolute Air IN Humidity;
- x_{14} = Relative Air IN Humidity;
- x_{15} = Product Humidity;
- x_{16} = Mill Speed.

Every data point $v = \{x_1, \dots, x_{16}\}$ is a set of 16 measures at the same time. The time rate is 1 minute, it means that the sensors take a data point every minute, for a time frame of 10 months with 3 shifts production.

As explained before, the model considers two classifications: the first classification identifies the process phases, the second classification evaluates the phase performance and it shows anomalies. The analyzed process consists of 7 phases:

- p_1 = Initializing;
- p_2 = Conditioning;
- p_3 =1st Spray;
- p_4 = Heating;
- p_5 = 2nd Spray;
- p_6 = Drying;
- p_7 = Unloading.

Beyond these phases, there is a new phase that indicates whether the measurement is not in a production cycle, named "not producing", as p_8, as stated. Experts of the process and its real-time analysis on field clarify that Air IN Flow >1 and Microclimate Pressure >1 characterize the production cycle. Figure 5 shows the typical trend of a production cycle, highlighting the most significant parameters. This phase considers a single criticality level, specific only for the situation of not producing, i.e. "not in production". From Figure 5 to Figure 8 the presented data are normalized.

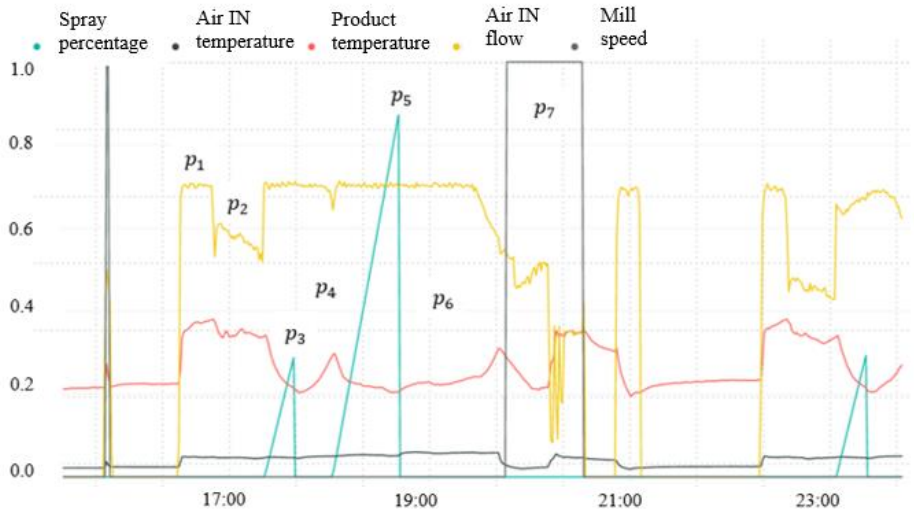

Figure 5. *Run chart of 5 parameters in the granulation production cycle*

As explained in the methodology, the process parameters correlations to process phases are available from interviews with the process experts, as follows:

- Conditioning is characterized by the microclimate pressure between a minimum value and a maximum value.
- For 1st Spray and 2nd Spray, there is a limit spray duration, which is 25 s. After that, there was the initialization of a count that grows if spray percentage(t)>1 and spray percentage($t+1$) ≥ 0. When the count, bigger than 0 for both phases, is minor than spray limit duration the process phase is 1st Spray, else it is 2nd Spray.
- Unloading is characterized by mill speed with a constant value of 500 rpm.
- For the other phases, there are not specific and univocal characteristics that determine the phase. These phases have been interlocked with the previous phase and the following phase. It means, for example, that the heating is inserted between 1st Spray and 2nd Spray.

Thus, the process phases are not always trivially separable. Time series are inhomogeneous and the experts of the process refer that the analysis of data does not permit the identification of every process phase. This situation makes it necessary to use the first classification to understand the intrinsic and hidden correlations between the input parameters and the phases of the process.

After that, the criticality labeling can start; the first part of this step is to associate every phase to one or more parameters that influence process phase performance. In the case study, once again analyzing the process and talking with process experts, the links between phases and parameters are the following:

- Initializing → Air IN Flow
- Conditioning → Air IN Flow and Air IN Temperature
- Heating → Air IN Flow, Microclimate Pressure and Absolute Air IN Humidity
- Drying → Air IN Flow and Absolute Air IN Humidity

- Unloading → Air IN Flow
- 1^{st}Spray and 2^{nd}Spray → Absolute Air IN Humidity

The following figures show some examples of correct and incorrect trends of relevant parameters for some specific phases. The chosen phases are the phases more relevant for plant safety and the quality of the product. Figure 6 and Figure 7 show the correct trend and the incorrect trend of Absolute Air IN Humidity in 1^{st} Spray and 2^{nd} Spray: in fact, the incorrect functioning of these phases involves throwing the whole product without the possibility of recovery. Moreover, these phases are correlated with the peristaltic pump and the air handling unit; the incorrect functioning of these phases involves dangerous situations for those nearby. Observing the parameters of these phases makes the detection easier if the valve of the air handling unit is not properly closed or if the rotor of the peristaltic pump is worn, ensuring the safety of the operators and the plant. Figure 8 shows the correct and the incorrect trend of Air IN Flow for the Conditioning phase; this is an important phase because it guarantees that the environment is suitable for the operation of the machine and its functioning is correlated with the air handling unit, that involves, as explained before, important problems for plant safety.

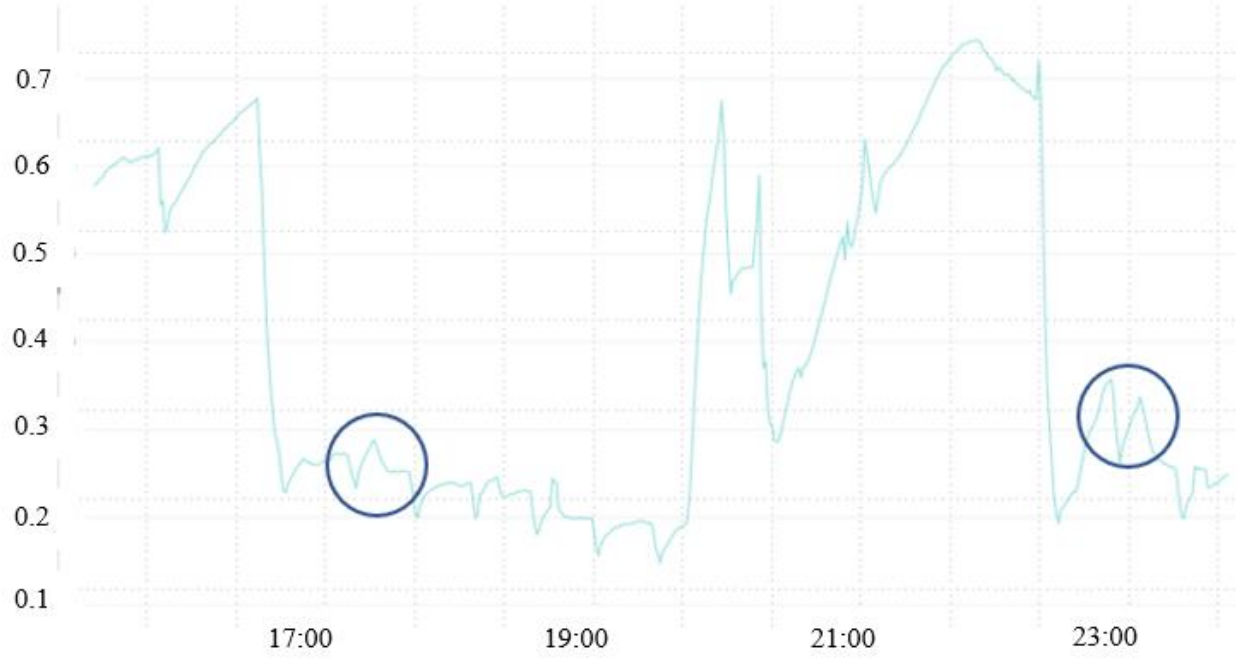

Figure 6. *Incorrect trend (left) and correct trend (right) of Absolute Air IN Humidity (g of water/Kg of air) in 1^{st} Spray*

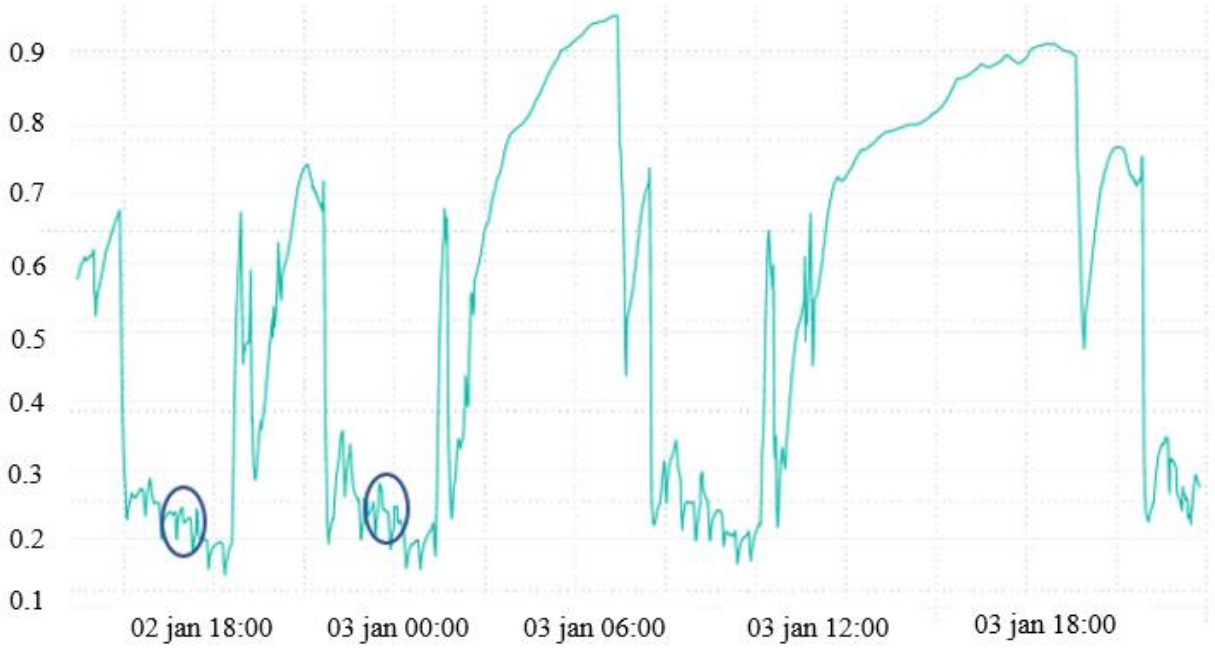

Figure 7. *Incorrect trend (right) and correct trend (left) of Absolute Air IN Humidity (g of water/Kg of air) in 2nd Spray*

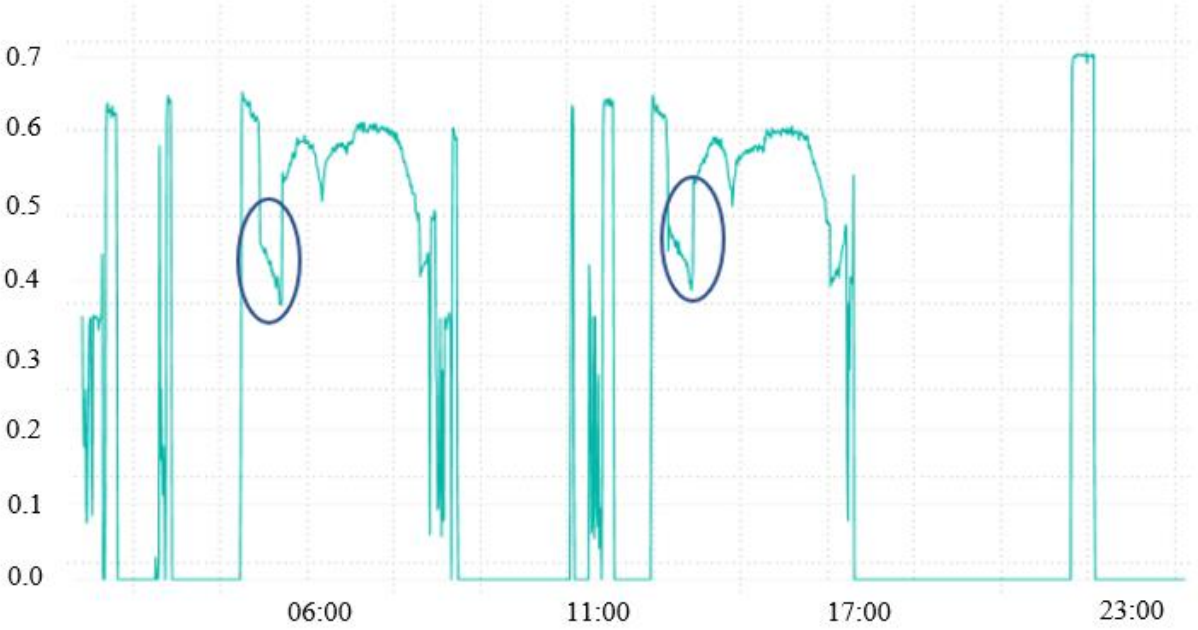

Figure 8. *Incorrect trend (right) and correct trend (left) of Air IN Flow in Conditioning*

An important goal achieved is the correspondence between phases and specific maintenance interventions on some components of the granulator; the anomaly detection system is in real-time connection with the maintenance.

Granulator components associated with phases are:

- Windmill anemometer → All phases;
- Chiller → 1st Spray and 2nd Spray;
- Peristaltic pump → 1st Spray and 2nd Spray;
- Sifter → Unloading;
- Air handling unit → All phases;
- Product temperature sensor → All phases.

The following step of the methodology consists of criticality labeling. The focus is on the prediction of the criticality level for every phase, it means an indication of how much the behavior of every phase is consistent with the desired trend. There are two groups of phases: for "1st Spray" and "2nd Spray" there are three criticality levels, that are "Expected" if the phase behavior is desirable, "Warning" if the phase behavior is not very consistent with the desired trend and "Critical" if the phase behavior is strongly inconsistent with the desired trend; for the others, there are two criticality levels, with "Expected" if the phase behavior is desirable and "Warning" in all the other situations. This difference comes from a difference of relevance between the phases: the experts of the process consider "1st Spray" and "2nd Spray" the most problematic phases; in fact, anomalies in these two phases lead to the elimination of the whole product batch, without the possibility to recover it. Moreover, the product is in a more dangerous physical-chemical state, and the plant safety deals with important risks; for this reason, the maintenance manager and the production line manager ask for a more detailed analysis for these phases. Moreover, the process history shows the worst performances in these two phases. For all the other phases the recovery possibility is substantially 100%, therefore it is not necessary to consider the level "Critical". Obviously, the identification of criticality should not consider the measurements that are not in a production cycle. The criticality identification should always consider editable thresholds; it is important to structure a model that is reusable, easily modifiable, adaptable, and dynamic. It means that there are not predefined thresholds, but thresholds are computed from the system or defined by the company.
Percentiles help the automatic definition of thresholds, with a method adaptable to every type of dataset or production system; moreover, with percentiles the single company decides the tolerance level for every machine or process phase, based on, for example, the affordable risk level. Thus, the threshold is the *m-th* percentile of the collected values, which is the value that cuts off the first *n* percent of the data values when it is sorted in ascending order.
In the case study, the company was involved in the percentiles definition as well as the thresholds choice, all expressed in *-th* percentile, are in Table 1. The percentiles definition relies on the historical data about maintenance interventions and accidents, as well as the meetings with the pre-mentioned experts. They supplied the process requirements levels (e.g. from GMP, safety, and other compliances) and the maintenance requirements levels (e.g. from service continuity, historical costs of interventions). These requirements can be considered as thresholds, from which percentiles are calculated. Remarkably, thresholds permit the identification of the criticality only

if a historical dataset analysis is available; when the anomaly detector input is a single set of measures without previous data and generic contextualization of the data it is not possible to apply percentile analysis to identify the anomaly. Moreover, the machine learning algorithm helps in identifying further correlations and data patterns not already known by the experts.
Referring to the notation in section 6:

- $c_h(t)$ = "Warning" if $x_{kj}(t) <$ WLL or $x_{kj}(t) >$ WUL, where WLL is the warning lower level, WUL is the warning upper level, e.g. the warning thresholds;
- $c_h(t)$ = "Critical" if $x_{kj}(t) <$ CLL or $x_{kj}(t) >$ CUL, where CLL is the critical lower level, WUL is the critical upper level, e.g. the critical thresholds.

Please note spray phases only consider the "Critical" level.

Phase	***Parameter***	***CLL***	***WLL***	***WUL***	***CUL***
Initializing	Air IN Flow		4	90	
Conditioning	Air IN Flow		5	97	
	Air IN Temperature		5	97	
Heating	Air IN Flow		3	97	
	Microclimate Pressure		1	98	
	Air IN Humidity Absolute		4	90	
1st Spray	Absolute Air IN Humidity	2	5	90	95
Drying	Air IN Flow		2	95	
	Air IN Humidity Absolute		2	95	
2nd Spray	Absolute Air IN Humidity	2	5	90	95
Unloading	Air IN Flow		3	98	

Table 1. *Criticality thresholds*

The knowledge is now available on:

- the process phases definition;
- the relevant parameters for every process phase;
- the criticality thresholds;
- the correspondences between anomalies in process phases and the granulator components.

Other popular data analysis techniques could provide thresholds, i.e. median absolute deviation and the inter-quartile-range-filtering. The first one is defined as the median of the absolute deviations from the data's median; the inter-quartile-range as the range between the first quartile (25^{th} percentile) and third quartile (75^{th} percentile) and the upper bound and the lower bound are computed by adding 1.5 times the inter-quartile-range to the third quartile or subtracting 1.5 times the inter-quartile-range from the first quartile. The multiplier 1.5 can be handled according to the necessity of the analysis. Those techniques rely only on the distribution of values, not considering

how the historical trend is related to the process requirement levels and maintenance requirement levels.
Thresholds can not be used in a real-time approach where the system does not have access to historical data. Moreover, from the labeling step, the machine learning classification could identify a not evident pattern in data, and thus it identifies criticality even if thresholds are not passed. The main aim of the methodology is its applicability in real-time and not analyzing historical data.

It is now possible to create the labeled dataset. Two Python scripts have been developed to support the labeling phases: the first adds to every line of the dataset the label of the process phase, and the second adds the label of the measurement criticality level. The labeled dataset is the input for the predictive model.
The dataset for the identification of the process phases is in Table 2.

Process Phase	***Instances***
Initializing	3434
Conditioning	5366
Heating	5219
1st Spray	3633
Drying	13681
2nd Spray	7969
Unloading	7468
Not producing	85371

Table 2. *Dataset for the identification of process phases*

The dataset for the evaluation of the performance is in Table 3.

Performance Class	***Instances***
Expected	34855
Warning	11107
Critical	808
Not in production	85371

Table 3. *Dataset for the evaluation of the performance*

The number of “Critical” is low because it is present only in two phases, i.e. 1st Spray and 2nd Spray, and it is rare. “Expected” and “Warning” are in every phase.
For both the algorithms, RFA and DJA, the choice of parameters consider previous applications of these algorithms.

Specifically, the authors have considered previous approaches available in “Azure AI Gallery”, that is a gallery in which the community of developers and data scientists can share solutions implemented with Azure Machine Learning Studio, the platform used by authors for structured the predictive solutions. The authors have started with a setup confirmed with previous performance in different contexts.
The impact of these setup parameters and the model sensitivity analysis will be the focus of further researches.

The RFA parameters for this model are summarized in Table 4.

Parameter	***Value***
Number of decision trees	8
Maximum depth of decision trees	32
Number of random splits per node	128
Minimum number of samples per leaf node	1
Allow unknown values for categorical features	Yes

Table 4. *Parameters of RFA*

- **Number of decision trees:** it sets up the maximum number of decision trees; a bigger number of trees increases the coverage of parameters combinations, at the expense of the time training. The case study presents a big dataset, the necessity of real-time implementation, the thresholds of the model can be changed frequently, and training could often take place.
- **Maximum depth of decision trees:** it sets up the maximum depth of every tree or the length of the longest path from the root to the leaves of the tree; increasing the depth of the trees means increasing the precision of the model, increasing, therefore, training time and overfitting risk.
- **Number of random splits per node:** it sets up the number of splits for every node, which is the number of random values to test.
- **Minimum number of samples per leaf node:** it denotes the minimum number of cases needed for the creation of a new leaf of the tree, or how many times an event must occur for this "rule" to have value. Working on anomaly detection, even the single occurrence of an event should be considered.
- **Allow unknown values for categorical features:** it lets the algorithm consider an input unknown value associated with an output value. This option has been set to accept not only values that were present in the training data.

Similarly to Table 4, the DJA parameters for this model are summarized in Table 5:

Parameter	*Value*
Number of decision DAGs	8
Maximum depth of the decision DAGs	32
Maximum width of the decision DAGs	128
Number of optimization steps per decision DAGs layer	2048
Allow unknown values for categorical features	Yes

Table 5. *Parameters of DJA*

- **Number of decision DAGs**: the maximum number of graphs that can be created in the ensemble.
- **Maximum depth of the decision DAGs**: the maximum depth of each graph.
- **Maximum width of the decision DAGs**: the maximum width of each graph. A smaller value of width improves accuracy, keeping memory constant, but must be trained longer.
- **Number of optimization steps per decision DAG layer**: how many iterations over the data to perform, when building each DAG.
- **Allow unknown values for categorical features**: allows the algorithm to consider an unknown input value associated with an output value. This option has been set to accept not only values in the training data.

Both for RFA and DJA, 70% of the data went for the training, and 30% of the data for testing the model. The dataset is large enough for accurate training and for testing the model on an important data set.
The models have reached important results; the general parameters of quality for the algorithms are Precision and Recall. The precision explains which is the correct percentage of positive predictions, and the recall measures the percentage of correctly predicted positive events.
The formulas for these measures focus on the importance of retrieval of the positive cases (Table 6).

Measure	*Formula*	*Evaluation focus*
Precision	$\frac{True\ positives}{True\ positives + False\ Positives}$	Agreement of predictions class-by-class within the real data. E.g. correct predicted "Warning" versus all predicted "Warning"
Recall	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$	Coverage of real data class-by-class by the predictions. E.g. True predicted "Warning" versus all real data "Warning"
F-Score	$2 * \frac{Precision * Recall}{Precision + Recall}$	Harmonic average of precision and recall

Table 6. *Measures of the classification*

Since the problem is a multi-class classification problem, with the prediction of one of the three levels of criticality ("Expected", "Warning", Critical"), true positives are the sum of the calculated true positive for every class (e.g. levels of criticality).
The authors have decided to discard the "Overall Accuracy", that is the ratio between the number of correctly predicted items and the total number of items to predict, because of the imbalanced dataset: the number of "Warning" and "Critical" situations is strongly lower than the number of "Expected" situations. In fact, in production systems that "Warning" and "Critical" are exceptional situations, but "Expected" represents the common situation. Moreover, a false negative prediction for "Warning" or "Critical" can involve accidents and not-timely maintenance, and recall is important to quantify these occurrences. Similarly, it is important having few false positives, to guarantee a lower number of non-necessary stop or downtime, and precision is the correct measure for this performance. About this contextualization, precision represents good predictions, focusing on false positives, and recall represents good predictions, focusing on false negatives. Moreover, based on [69], the authors have considered some invariance properties of measures, evaluating the most important for the analyzed case. Firstly, it is important to consider that precision and recall do not depend on True Negative (TN), but only on the correct labeling of positive examples (True Positives-TP) and the incorrect labeling of examples (False Positives-FP and False Negatives-FN). These characteristics accord the anomaly detection in a production process strongly prefers to be unbalanced toward positive prediction, in a precautionary principle approach: a false alarm is better than an unpredicted alarm. Precision and Recall are invariant under the change of TN, where a measure is invariant under a property if its value does not change when the confusion matrix suffers a specific change because the measure does not detect the change in the confusion matrix [69].
For both metrics, there are micro and macro averages. Macro-average means that the metric is computed for every class independently and then there is the average of these values; micro-average instead computes the metric with all data and for every class. In multiclass classification,

micro-average is preferred when the dataset is imbalanced, so the authors have considered micro-average precision and micro-average recall. In conclusion, both precision and recall have a very good performance if there is an uneven class distribution, as in the presented situations. They both focus on the performance of positives classification.
The inputs of the methodology are the 16 inputs presented at the beginning of section 6. Methodology, i.e. from x_1 to x_{16}. Starting from these inputs, two of them are not considered in the methodology, i.e. x_1 (Date) and x_{15}(Product Humidity). x_1 does not add useful information for the evaluation of the process and x_{15} assumes always the same value, becoming invariant for the analysis. So, at the end of this cleaning process, the inputs are 14. The inputs are getting raw, as shown in Figure 5, Figure 6, Figure 7, and Figure 8, but not normalized. Based on these inputs, the outputs are two: the process phase classification, with 8 possible classes, and the criticality level of the measurements, with 4 possible classes, which is the result of the anomaly detection process. The original dataset contains 132141 instances, has been split in 70% for the training data, and 30% for the test data. In conclusion, the training dataset contains a 92498 x 16 table (because it is labeled, it means with the evidence of the process phase and the criticality level) and the test data contains a 39643 x 14 table (because it is unlabelled).
For the cross-validation, the training dataset has been divided into 10 sub-datasets, named also "folds". In every fold, a part of the data is used for the training process and the remaining part, also called *holdout fold* is used to test and validate what learned during the training step.
The authors have carried out the comparison between criticality class classification, i.e. anomaly detection, with and without the proposed methodology, that is with and without the process phase identification step before the anomaly detection step.
The anomaly detection process without the first step of the methodology, i.e. the process phase identification, cannot achieve satisfactory results. The results on the test dataset are shown in Figure 13. The authors have applied a well-known anomaly detector, the RFA. Moreover, authors have implemented the tune of the hyperparameters on the training dataset to find the best settings of the algorithm. A hyperparameter is a parameter whose value is set before the learning process begins. The other parameters are the ones obtained via training. The tune of the hyperparameters builds and tests multiple machine learning models, using different combinations of settings, and compares metrics over these models to get the combination of settings with the best performance. In particular, the authors have considered the F-Score as a metric of this tune of the hyperparameters. Especially, the authors have tested the hyperparameters using the randomized search method, it means randomly selected parameter values over a system-defined range, specified in the following list. Even in this case, the authors have applied the cross-validation to the training dataset, for ensuring the best choice of the hyperparameters. The tune of the hyperparameters on the training dataset for RFA ends with the best F-Score of 97,9%. These results may lead to erroneous considerations. Looking at Figure 13 one can note that without the proposed methodology, the algorithm is not able to properly detect the critical and the warning class on the test dataset. The value of F-Score is affected by the higher number of the two classes correctly predicted by the algorithm. The tune of the hyperparameters tested 500 different settings combinations on the training dataset. The set thresholds for the tune of the hyperparameters of the algorithm are:

- RANDOM FOREST ALGORITHM:

- Number of decision trees: 1-500.
- Maximum depth of the decision trees: 1-500
- Number of random split per node: 1-500
- Minimum number of samples per leaf node: 1-100

The best settings results:

- RANDOM FOREST ALGORITHM:

  - Number of decision trees: 63
  - Maximum depth of the decision trees: 76
  - Number of random split per node: 3
  - Minimum number of samples per leaf node: 4

Considering the obtained results, except for a setting that achieves an F-Score = 0.463, all the tested settings achieve an F-Score between 0.639 and 0.976.
The results of the tune of the hyperparameters on the training dataset are presented in Figure 9, Figure 10, and Figure 11.
Figure 9 represents the different F-Score obtained with 500 different settings of the maximum depth of the decision trees and the number of decision trees.

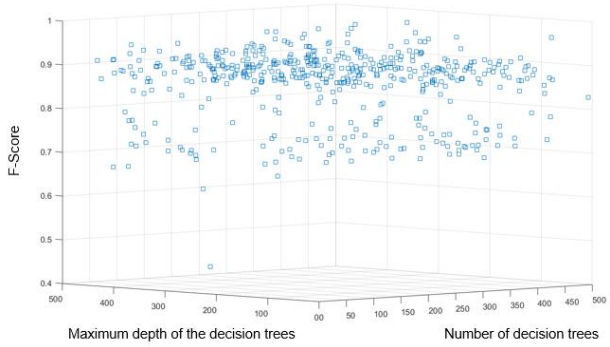

Figure 9. *F-Score without the proposed methodology based on different maximum depths of decision trees and number of decision trees on the training dataset*

As well as Figure 9, Figure 10 represents the different F-Scores obtained with 500 settings of the number of random splits per node and the minimum number of samples per leaf node.

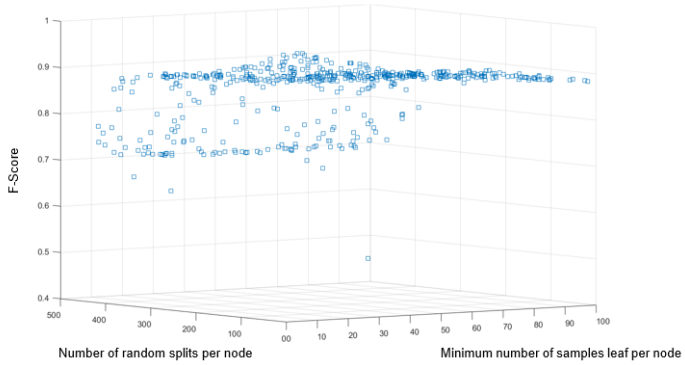

Figure 10. *F-Score without the proposed methodology based on different random splits per node and number of samples per leaf node on the training dataset*

Finally, Figure 11 offers a complete overview of the tune of the hyperparameters process and its results. Every line in the figure represents a tested setting, presenting all of them, with the evidence of the F-Score obtained by each of them. The black line represents the best setting.

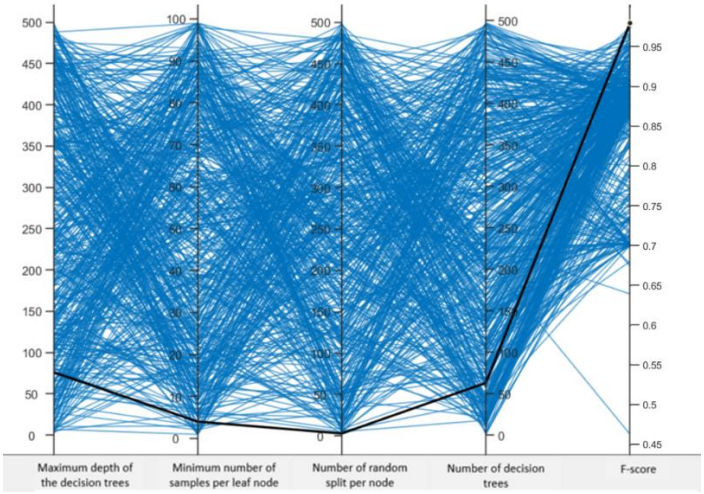

Figure 11. *F-Score without the proposed methodology based on different 4 hyperparameters combinations on the training dataset*

The distribution of the F-Score during the tune of the hyperparameters on the training dataset is presented in Table 7 and graphically in Figure 12.

Represented Value	***Numerical Value***
MAXIMUM	*0.98*
QUARTILE 3	*0.90*
MEDIAN	*0.88*
AVERAGE	*0.86*
QUARTILE 1	*0.85*
MINIMUM	*0.46*

Table 7. *F-Score distribution without the proposed methodology for the tune of the hyperparameters on the training dataset*

Figure 12. *F-Score distribution without the proposed methodology for the tune of the hyperparameters on the training dataset*

In Figure 12 the black line represents the median of the F-Score distribution and the red dot the average.
It is necessary to clarify that in Figure 13, Figure 16 and Figure 17 the expression "not" represent the criticality level "not in production". In Figure 14 and Figure 15 "not" represents the process phase "not producing".
The results of the anomaly detection without the proposed methodology on the test dataset are presented in Figure 13.

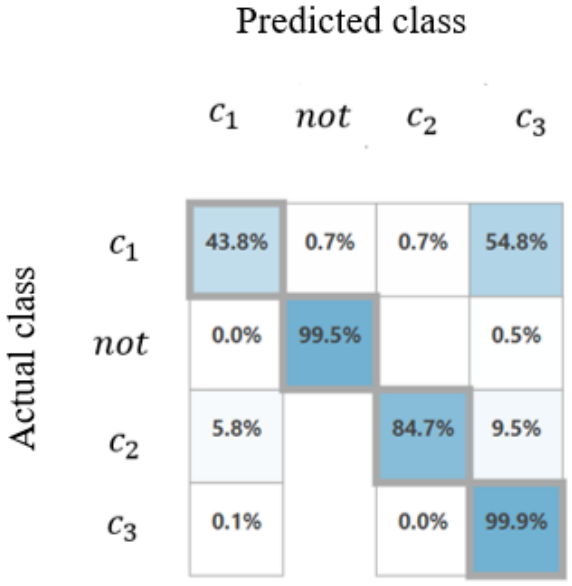

Figure 13.*Percentages of right predictions for every criticality level using RFA on the test dataset, without the proposed methodology*

For the proposed methodology, the resultant metrics of the cross-validation on the training dataset are presented in Table 8, Table 9, Table 10 and Table 11.
The cross-validation has been applied to evaluate the goodness of the model parameters presented in Table 4 and Table 5 and to verify the presence of overfitting.

Algorithm	***Mean***	***Standard deviation***
Precision	0,999373	0,00047
Recall	1	0
F-Score	0,999525	0,0003

Table 8. *Cross-validation results applying RFA on the training dataset for the process phase identification with the proposed methodology*

Algorithm	***Mean***	***Standard deviation***
Precision	0,999232	0,001016
Recall	0,999531	0,000554
F-Score	0,999382	0,00072

Table 9. *Cross-validation results applying DJA on the training dataset for the process phase identification with the proposed methodology*

Algorithm	***Mean***	***Standard deviation***
Precision	0,99612	0,00473443
Recall	0,997055	0,00289489
F-Score	0,996586	0,00375

Table 10. *Cross-validation results applying RFA on the training dataset for the prediction of the criticality level with the proposed methodology*

Algorithm	***Mean***	***Standard deviation***
Precision	0,9886615	0,0100771
Recall	0,9725915	0,03524998
F-Score	0,980273	0,017948

Table 11. *Cross-validation results applying DJA on the training dataset for the prediction of the criticality level with the proposed methodology*

After the cross-validation of the training dataset, the following step is the application of the trained methodology on the test dataset, that has never been seen by the model. As explained before, the test dataset represents 30% of the original dataset.
The results of the application of the proposed methodology in the test datasets are presented in Figure 14 and Figure 15 for the process phases identification and in Figure 16 and Figure 17 for the anomaly detection. As explained at the beginning of the papers, the authors have decided to test even the applicability of DJA with the proposed methodology.
Comparing the confusion matrices in Figure 16, Figure 17 and Figure 13 can be strongly asserted that the proposed methodology achieves better anomaly detection performance in production processes, to monitor more than one phase.

The results for the identification of process phases on the test dataset are, considering the precision, the recall, and the F-Score, in Table 12:

Algorithm	***Precision***	***Recall***	***F-Score***
RFA	99.97%	99.97%	0,9997
DJA	99.96%	99.96%	0,9996

Table 12. *Precision, Recall and F-Score for the identification of process phases on the test dataset, with the proposed methodology*

Figure 14 and Figure 15 show the percentage of right predictions on the test dataset for both algorithms, for every process phase.

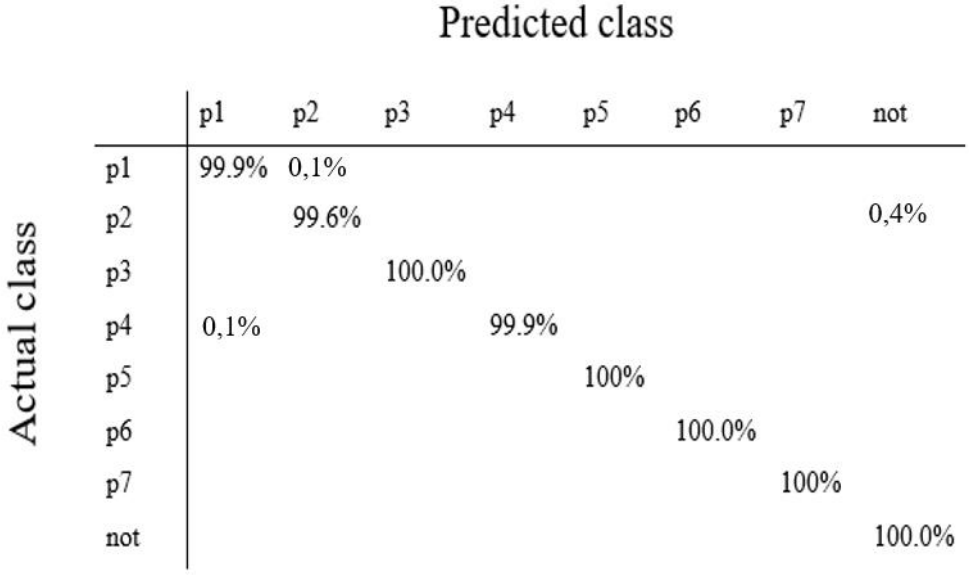

Figure 14. *Percentage of right predictions for every phase using RFA on the test dataset, with the proposed methodology*

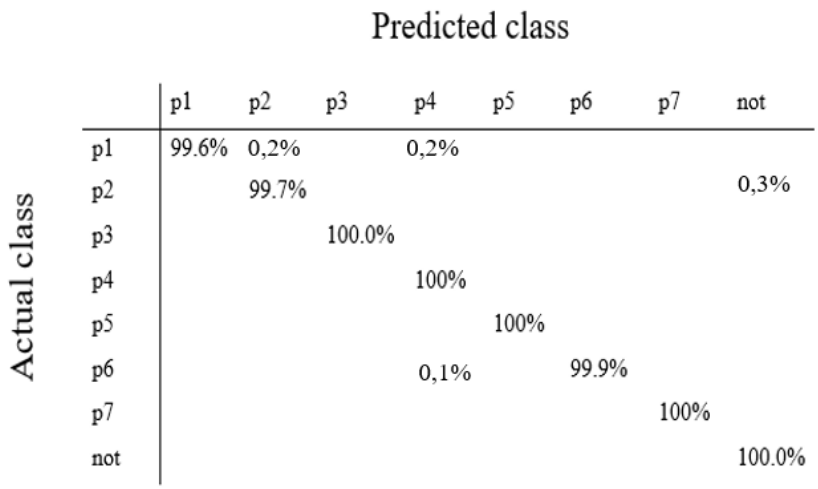

Figure 15. *Percentage of right predictions for every phase using DJA on the test dataset, with the proposed methodology*

The authors can assert that the risk of overfitting is passed. Firstly, the presented results refer to the test data and not the training data. Both cross-validation and different tests using different sub-datasets of test verified the absence of overfitting.

The results for the evaluation of the performance on the test dataset are, considering the precision, the recall, and the F-Score, in Table 13:

Algorithm	***Precision***	***Recall***	***F-Score***
RFA	99.90%	99.90%	0,9990
DJA	99.21%	99.21%	0,9921

Table 13. *Precision, Recall and F-Score for the evaluation of the performance on the test dataset, with the proposed methodology*

Figure 16 and Figure 17 show the percentage of right predictions on the test dataset for both algorithms, for every criticality level.

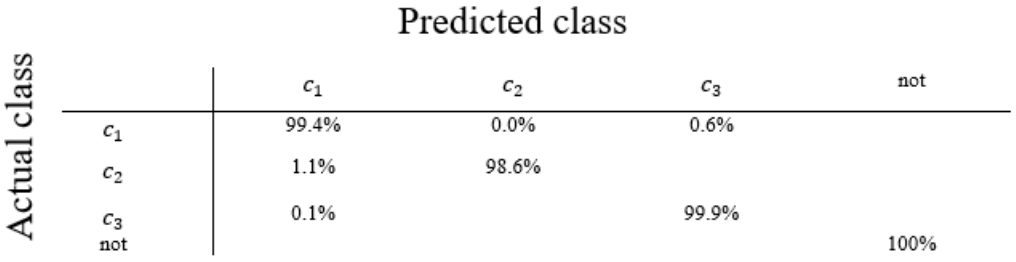

Figure 16. *Percentages of right predictions for every criticality level using RFA on the test dataset, with the proposed methodology*

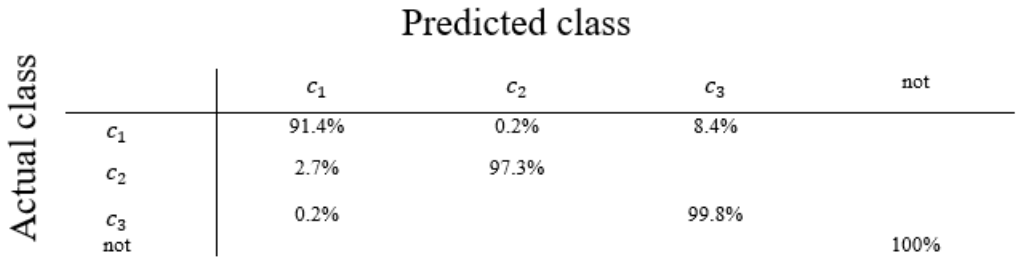

Figure 17. *Percentages of right predictions for every criticality level using DJA on the test dataset, with the proposed methodology*

The authors used Azure Machine Learning Studio to develop the anomaly detection model. Azure Machine Learning Studio is an on-line platform by Microsoft for building, testing and deploying personalized predictive analytics solutions. This platform enables not only the development of predictive models but also their availability as web service, to interact with external digital environments.

8. Discussions and conclusions

A methodology to achieve anomaly detection with real-time data from a multiphase industrial process is presented. The non-detection of the process phase leads to low results in predicting anomalies. A situation is anomalous if contextualized in the specific production phase. The literature considers production processes in which the production phase is well known, or focuses on a specific production phase. The paper proposes a two steps methodology for anomaly detection in this kind of production process. The first step identifies the process phase in running and the second step, based on the information gained in the first step, implements the anomaly detection. The proposed methodology applies in both steps classification algorithms. The anomaly detection step considers some predefined criticality classes. The development of the proposed methodology involved three experts of the production process presented in the case study, i.e. maintenance

manager, production line manager, and IT area manager. Their early contribution enables the concrete definition of the objectives of the methodology as well as the total understanding of the production process and the production settings. A future reproduction of the proposed methodology cannot be implemented without their full involvement. It is possible to assert that the model validity has been proved with a case study with an adequate number and variety of anomalous cases.

The results presented in the case study demonstrate the need to apply the proposed methodology. In fact, with the proposed methodology, the anomaly detection step can guarantee high-performance results with a precision and a recall that exceed 99%. Without the proposed methodology, the anomaly detection step fails to detect warning and critical situations in the production phases.

The first step of the methodology, i.e. the identification of the process phases, is essential in several real production processes, where often there is no evidence of the production phase nor an evident correlation between the monitored process parameters and the phases of production. In these situations, a machine learning model can understand intrinsic and hidden relationships between process parameters and production phases, allowing the identification of the latter in an effective way, as shown in the case study presented.

The tested algorithms provide some additional outputs on their effectiveness and their differences. One can affirm that the anomaly detection model must prefer RFA to DJA for the anomaly detection process. Based on the economical and safety necessities of the company, the 8.4% of error between warning (actual class) and expected (predicted class) and the 2.7% of error between critical (actual class) and warning (predicted class) carries out by the DJA cannot be neglected. Both RFA and DJA have few false anomalies.

The authors presented a typical situation with a single training and test, form a batch dataset; future researches will study the effect of the repetition of training and test while gathering new data. With this ongoing training, the real anomaly detection systems will update the algorithm parameters; this is necessary if the class profiles change over time. Moreover, future researches will focus on the ordinality of the criticality classes as well as their interdependency. Furthermore, it will be interesting to test the behavior of different algorithms with the proposed methodology. Finally, future research will test DJA in other industrial situations, e.g. different situations of anomaly detection and different production processes.

Acknowledgments

The authors thank Isurika Manimendra for her early contribution to the phase identification module.

Declaration of Competing Interest

The authors declare that they have no known competing for financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Chandola V, Banerjee A, Kumar V. Anomaly Detection: A Survey. ACM Comput Surv 2009;41. https://doi.org/10.1145/1541880.1541882.

[2] Harrou F, Sun Y, Khadraoui S. Amalgamation of anomaly-detection indices for enhanced process monitoring. J Loss Prev Process Ind 2016;40:365–77. https://doi.org/10.1016/j.jlp.2016.01.024.

[3] Zaher AS, McArthur SDJ. A multi-agent fault detection system for wind turbine defect recognition and diagnosis. 2007 IEEE Lausanne POWERTECH, Proc 2007:22–7. https://doi.org/10.1109/PCT.2007.4538286.

[4] Aminu KT, McGlinchey D, Cowell A. Acoustic signal processing with robust machine learning algorithm for improved monitoring of particulate solid materials in a gas flowline. Flow Meas Instrum 2019;65:33–44. https://doi.org/10.1016/j.flowmeasinst.2018.11.015.

[5] Goyal D, Vanraj, Pabla BS, Dhami SS. Non-contact sensor placement strategy for condition monitoring of rotating machine-elements. Eng Sci Technol an Int J 2019. https://doi.org/10.1016/j.jestch.2018.12.006.

[6] Chakraborty S, Shah S, Soltani K, Swigart A. Root cause detection among anomalous time series using temporal state alignment. Proc. - 18th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2019, 2019, p. 523–8. https://doi.org/10.1109/ICMLA.2019.00098.

[7] Zhao X, Zhang J, Qin X, Cai J, Ma Y. Parallel mining of contextual outlier using sparse subspace. Expert Syst Appl 2019;126:158–70. https://doi.org/10.1016/j.eswa.2019.02.020.

[8] Wang X, Davidson I. Discovering contexts and contextual outliers using random walks in graphs. Proc. - IEEE Int. Conf. Data Mining, ICDM, 2009, p. 1034–9. https://doi.org/10.1109/ICDM.2009.95.

[9] Li X, Yang Y, Bennett I, Mba D. Condition monitoring of rotating machines under time-varying conditions based on adaptive canonical variate analysis. Mech Syst Signal Process 2019;131:348–63. https://doi.org/10.1016/j.ymssp.2019.05.048.

[10] Uma Maheswari R, Umamaheswari R. Trends in non-stationary signal processing techniques applied to vibration analysis of wind turbine drive train – A contemporary survey. Mech Syst Signal Process 2017;85:296–311. https://doi.org/10.1016/j.ymssp.2016.07.046.

[11] Leturiondo U, Mishra M, Galar D, Salgado O. Synthetic data generation in hybrid modelling of rolling element bearings. Insight Non-Destructive Test Cond Monit 2015;57:395–400. https://doi.org/10.1784/insi.2015.57.7.395.

[12] Hendrickx K, Meert W, Mollet Y, Gyselinck J, Cornelis B, Gryllias K, et al. A general anomaly detection framework for fleet-based condition monitoring of machines. Mech Syst Signal Process 2020;139. https://doi.org/10.1016/j.ymssp.2019.106585.

[13] D'Amato J, Patanian J. Method and system for predicting hydraulic valve degradation on a gas turbine. Proc. Annu. Conf. Progn. Heal. Manag. Soc. PHM, vol. 2016- Octob, 2016, p. 129–36.

[14] Abraham B, Mandya A, Bapat R, Alali F, Brown DE, Veeraraghavan M. A Comparison of Machine Learning Approaches to Detect Botnet Traffic. Proc Int Jt Conf Neural Networks 2018;2018-July:1–8. https://doi.org/10.1109/IJCNN.2018.8489096.

[15] Ariyaluran Habeeb RA, Nasaruddin F, Gani A, Targio Hashem IA, Ahmed E, Imran M. Real-time big data processing for anomaly detection: A Survey. Int J Inf Manage 2018:1–19. https://doi.org/10.1016/j.ijinfomgt.2018.08.006.

[16] Khaleghi A, Moin MS. Improved anomaly detection in surveillance videos based on a deep learning method. 2018 Artif. Intell. Robot. IRANOPEN 2018 8th Conf. Artif. Intell. Robot., IEEE; 2018, p. 73–81. https://doi.org/10.1109/RIOS.2018.8406634.

[17] Jiao ZH, Zhao J, Shan X. Pre-seismic anomalies from optical satellite observations: A review. Nat Hazards Earth Syst Sci 2018;18:1013–36. https://doi.org/10.5194/nhess-18-1013-2018.

[18] Song B, Suh Y. Narrative texts-based anomaly detection using accident report documents: The case of chemical process safety. J Loss Prev Process Ind 2019;57:47–54. https://doi.org/10.1016/j.jlp.2018.08.010.

[19] Kan C, Yang H, Kumara S. Parallel computing and network analytics for fast Industrial Internet-of-Things (IIoT) machine information processing and condition monitoring. J Manuf Syst 2018;46:282–93. https://doi.org/10.1016/j.jmsy.2018.01.010.

[20] Kim J, Huang Q, Shi J. Latent variable based key process variable identification and process monitoring for forging. J Manuf Syst 2007;26:53–61. https://doi.org/10.1016/j.jmsy.2007.12.001.

[21] Wang J, Liu S, Gao RX, Yan R. Current envelope analysis for defect identification and

diagnosis in induction motors. J Manuf Syst 2012;31:380–7. https://doi.org/10.1016/j.jmsy.2012.06.005.

[22] Yang L, Zhao Y, Peng R, Ma X. Opportunistic maintenance of production systems subject to random wait time and multiple control limits. J Manuf Syst 2018;47:12–34. https://doi.org/10.1016/j.jmsy.2018.02.003.

[23] Langone R, Alzate C, De Ketelaere B, Vlasselaer J, Meert W, Suykens JAK. LS-SVM based spectral clustering and regression for predicting maintenance of industrial machines. Eng Appl Artif Intell 2015;37:268–78. https://doi.org/10.1016/j.engappai.2014.09.008.

[24] Sardá-Espinosa A, Subbiah S, Bartz-Beielstein T. Conditional inference trees for knowledge extraction from motor health condition data. Eng Appl Artif Intell 2017;62:26–37. https://doi.org/10.1016/j.engappai.2017.03.008.

[25] Li X, Yang X, Yang Y, Bennett I, Collop A, Mba D. Canonical variate residuals-based contribution map for slowly evolving faults. J Process Control 2019;76:87–97. https://doi.org/10.1016/j.jprocont.2019.02.006.

[26] Li C, Valente De Oliveira J, Cerrada M, Pacheco F, Cabrera D, Sanchez V, et al. Observer-biased bearing condition monitoring: From fault detection to multi-fault classification. Eng Appl Artif Intell 2016;50:287–301. https://doi.org/10.1016/j.engappai.2016.01.038.

[27] Wang P, Ananya, Yan R, Gao RX. Virtualization and deep recognition for system fault classification. J Manuf Syst 2017;44:310–6. https://doi.org/10.1016/j.jmsy.2017.04.012.

[28] Liu R, Kothuru A, Zhang S. Calibration-based tool condition monitoring for repetitive machining operations. J Manuf Syst 2020;54:285–93. https://doi.org/10.1016/j.jmsy.2020.01.005.

[29] Lu B, Zhou X. Quality and reliability oriented maintenance for multistage manufacturing systems subject to condition monitoring. J Manuf Syst 2019;52:76–85. https://doi.org/10.1016/j.jmsy.2019.04.003.

[30] Shabgard MR, Badamchizadeh MA, Ranjbary G, Amini K. Fuzzy approach to select machining parameters in electrical discharge machining (EDM) and ultrasonic-assisted EDM processes. J Manuf Syst 2013;32:32. https://doi.org/10.1016/j.jmsy.2012.09.002.

[31] Ben Ali J, Fnaiech N, Saidi L, Chebel-Morello B, Fnaiech F. Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals. Appl Acoust 2015;89:16–27.

https://doi.org/10.1016/j.apacoust.2014.08.016.

[32] Cerrada M, Sánchez RV, Li C, Pacheco F, Cabrera D, Valente de Oliveira J, et al. A review on data-driven fault severity assessment in rolling bearings. Mech Syst Signal Process 2018;99:169–96. https://doi.org/10.1016/j.ymssp.2017.06.012.

[33] Schneider T, Helwig N, Schütze A. Industrial condition monitoring with smart sensors using automated feature extraction and selection. Meas Sci Technol 2018;29. https://doi.org/10.1088/1361-6501/aad1d4.

[34] Gandhi K, Schmidt B, Ng AHC. Towards data mining based decision support in manufacturing maintenance. Procedia CIRP 2018;72:261–5. https://doi.org/10.1016/j.procir.2018.03.076.

[35] Lejon E, Kyösti P, Lindström J. Machine learning for detection of anomalies in press-hardening: Selection of efficient methods. Procedia CIRP 2018;72:1079–83. https://doi.org/10.1016/j.procir.2018.03.221.

[36] Qian P, Zhang D, Tian X, Si Y, Li L. A novel wind turbine condition monitoring method based on cloud computing. Renew Energy 2019;135:390–8. https://doi.org/10.1016/j.renene.2018.12.045.

[37] Stetco A, Dinmohammadi F, Zhao X, Robu V, Flynn D, Barnes M, et al. Machine learning methods for wind turbine condition monitoring: A review. Renew Energy 2019;133:620–35. https://doi.org/10.1016/j.renene.2018.10.047.

[38] De Benedetti M, Leonardi F, Messina F, Santoro C, Vasilakos A. Anomaly detection and predictive maintenance for photovoltaic systems. Neurocomputing 2018;310:59–68. https://doi.org/10.1016/j.neucom.2018.05.017.

[39] Hajdarevic A, Dzananovic I, Banjanovic-Mehmedovic L, Mehmedovic F. Anomaly detection in thermal power plant using probabilistic neural network. 2015 38th Int Conv Inf Commun Technol Electron Microelectron MIPRO 2015 - Proc 2015:1118–23. https://doi.org/10.1109/MIPRO.2015.7160443.

[40] Rossetti D, Squartini S, Collura S, Zhang Y. Power plant condition monitoring by means of coal powder granulometry classification. Meas J Int Meas Confed 2018;123:39–47. https://doi.org/10.1016/j.measurement.2018.03.028.

[41] Harrou F, Sun Y, Madakyaru M. Kullback-Leibler distance-based enhanced detection of incipient anomalies. J Loss Prev Process Ind 2016;44:73–87.

https://doi.org/10.1016/j.jlp.2016.08.020.

[42] Li X, Hu Y, Zhu L, Leng X, Ye K, Xiao F. The Research of Anomaly Detection Method for Transformer Oil Temperature Based on Hybrid Model of Non-Supervised Learning and Decision Forests. IOP Conf Ser Earth Environ Sci 2018;192:012020. https://doi.org/10.1088/1755-1315/192/1/012020.

[43] Mansouri M, Nounou M, Nounou H, Karim N. Kernel PCA-based GLRT for nonlinear fault detection of chemical processes. J Loss Prev Process Ind 2016;40:334–47. https://doi.org/10.1016/j.jlp.2016.01.011.

[44] Zhao C, Huang B. A full-condition monitoring method for nonstationary dynamic chemical processes with cointegration and slow feature analysis. AIChE J 2018;64:1662–81. https://doi.org/10.1002/aic.16048.

[45] Cirera J, Quiles M, Carino JA, Zurita D, Ortega JA. Data-driven operation performance evaluation of multi-chiller system using self-organizing maps. Proc IEEE Int Conf Ind Technol 2018;2018-Febru:2099–104. https://doi.org/10.1109/ICIT.2018.8352513.

[46] Kan C, Cheng C, Yang H. Heterogeneous recurrence monitoring of dynamic transients in ultraprecision machining processes. J Manuf Syst 2016;41:178–87. https://doi.org/10.1016/j.jmsy.2016.08.007.

[47] Milo MW, Roan M, Harris B. A new statistical approach to automated quality control in manufacturing processes. J Manuf Syst 2015;36:159–67. https://doi.org/10.1016/j.jmsy.2015.06.001.

[48] Myers D, Suriadi S, Radke K, Foo E. Anomaly detection for industrial control systems using process mining. Comput Secur 2018;78:103–25. https://doi.org/10.1016/j.cose.2018.06.002.

[49] Rokach L. Decision forest: Twenty years of research. Inf Fusion 2016;27:111–25. https://doi.org/10.1016/j.inffus.2015.06.005.

[50] Ragab A, El-Koujok M, Poulin B, Amazouz M, Yacout S. Fault diagnosis in industrial chemical processes using interpretable patterns based on Logical Analysis of Data. Expert Syst Appl 2018;95:368–83. https://doi.org/10.1016/j.eswa.2017.11.045.

[51] Accorsi R, Manzini R, Pascarella P, Patella M, Sassi S. Data Mining and Machine Learning for Condition-based Maintenance. Procedia Manuf 2017;11:1153–61. https://doi.org/10.1016/j.promfg.2017.07.239.

[52] Su CJ, Huang SF. Real-time big data analytics for hard disk drive predictive maintenance. Comput Electr Eng 2018;71:93–101. https://doi.org/10.1016/j.compeleceng.2018.07.025.

[53] Yan W, Zhou JH. Predictive modeling of aircraft systems failure using term frequency-inverse document frequency and random forest. IEEE Int Conf Ind Eng Eng Manag 2018;2017-Decem:828–31. https://doi.org/10.1109/IEEM.2017.8290007.

[54] Auret L, Aldrich C. Unsupervised process fault detection with random forests. Ind Eng Chem Res 2010;49:9184–94. https://doi.org/10.1021/ie901975c.

[55] Amihai I, Gitzel R, Kotriwala AM, Pareschi D, Subbiah S, Sosale G. An industrial case study using vibration data and machine learning to predict asset health. Proceeding - 2018 20th IEEE Int Conf Bus Informatics, CBI 2018 2018;1:178–85. https://doi.org/10.1109/CBI.2018.00028.

[56] Robles-Durazno A, Moradpoor N, McWhinnie J, Russell G. A supervised energy monitoring-based machine learning approach for anomaly detection in a clean water supply system. Proc IEEE Int Conf Cyber Secur Prot Digit Serv (Cyber Secur 2018) 2018:1–8. https://doi.org/10.1109/CyberSecPODS.2018.8560683.

[57] Avdagic I, Hajdarevic K. Survey on machine learning algorithms as cloud service for CIDPS. 2017 25th Telecommun Forum, TELFOR 2017 - Proc 2018;2017-Janua:1–4. https://doi.org/10.1109/TELFOR.2017.8249467.

[58] Shotton J, Sharp T, Kohli P, Nowozin S, Winn JM, Criminisi A. Decision Jungles: Compact and Rich Models for Classification. Adv Neural Inf Process Syst 26 27th Annu Conf Neural Inf Process Syst 2013 Proc a Meet Held December 5-8, 2013, Lake Tahoe, Nevada, United States 2013:234–42.

[59] Gunarathne WHS., Perera KDM, Kahandawaarachchi KADCP. Performance Evaluation on Machine Learning Classification Techniques for Disease (CKD). Ieee 2017:291–6. https://doi.org/10.1109/BIBE.2017.00056.

[60] Akbulut A, Ertugrul E, Topcu V. Fetal health status prediction based on maternal clinical history using machine learning techniques. Comput Methods Programs Biomed 2018;163:87–100. https://doi.org/10.1016/j.cmpb.2018.06.010.

[61] Wickramasinghe MPNM, Perera DM, Kahandawaarachchi KADCP. Dietary prediction for patients with Chronic Kidney Disease (CKD) by considering blood potassium level using machine learning algorithms. 2017 IEEE Life Sci Conf LSC 2017 2018;2018-Janua:300–3.

https://doi.org/10.1109/LSC.2017.8268202.

[62] Chandra B, Kothari R, Paul P. A new node splitting measure for decision tree construction. Pattern Recognit 2010;43:2725–31. https://doi.org/10.1016/J.PATCOG.2010.02.025.

[63] Ho TK. Random Decision Forests. Proc 3rd Int Conf Doc Anal Recognit 1995:278–82. https://doi.org/10.1109/ICDAR.1995.598994.

[64] Amit Y, Geman D. Randomized Inquiries about Shape; an Application to Handwritten Digit Recognition. Tech Rep 401 1994:49.

[65] Breiman L. Random Forest. Mach Learn 2001;45:5–32. https://doi.org/10.1023/A:1010933404324.

[66] Criminisi A, Shotton J, Konukoglu E. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Found Trends® Comput Graph Vis 2011;7:81–227. https://doi.org/10.1561/0600000035.

[67] Breiman L. Bagging predictors. Mach Learn 1996;24:123–40. https://doi.org/10.1007/BF00058655.

[68] Parikh DM. Handbook of Pharmaceutical Granulation Technology. Handb. Pharm. Granulation Technol. Second Ed., 2005, p. 1–625. https://doi.org/10.1201/9780849354953.ch17.

[69] Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inf Process Manag 2009;45:427–37. https://doi.org/10.1016/j.ipm.2009.03.002.