Fault Detection and Classification in Oil Wells and Production/Service Lines Using Random Forest

Matheus A. Marins[a,*], Bettina D. Barros[a,b], Ismael H. Santos[c], Daniel C. Barrionuevo[c], Ricardo E. V. Vargas[d], Thiago de M. Prego[e], Amaro A. de Lima[e], Marcello L. R. de Campos[a], Eduardo A. B. da Silva[a], Sergio L. Netto[a]

[a] *Electrical Engineering Program, Federal University of Rio de Janeiro, POBox 68504, Rio de Janeiro, RJ, 21947-970, Brazil*
[b] *Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway*
[c] *Petróleo Brasileiro S.A., Cidade Universitária, Rio de Janeiro, RJ, 20211-160, Brazil*
[d] *Petróleo Brasileiro S.A., Barro Vermelho, Vitória, ES, 29057-570, Brazil*
[e] *CEFET-NI, Nova Iguaçu, RJ, 26041-271, Brazil*

Abstract

This papers deals with the automatic detection and classification of faulty events during the practical operation of oil and gas wells and lines. The events considered here are part of the publicly available 3W database developed by Petrobras, the Brazilian oil holding. Seven fault classes are considered, with distinct dynamics and patterns, as well as several instances of normal operation. A random forest classifier is employed with different statistical measures to identify each fault type. Three experiments are devised in order to evaluate the system performance in distinct classification scenarios. An accuracy rate of 94% indicates a successful performance for the proposed system in detecting real events. Also, the system's time of detection was on average 12% of the transient period that precedes the fault steady-state.

Keywords: fault detection and classification; oil well monitoring; abnormal event management; random forest classifier; machine learning.

1. Introduction

Recent advances in data acquisition, transmission, and storage have led many companies to develop large databases associated with their entire production and management chains. Such databases can integrate several kinds of data acquired from all sorts of sources, such as equipment sensors, economic series, human resources, and so on [1, 2]. This trend becomes a two-edged knife when the amount of data surpasses its processing capability, making it difficult to extract useful information from it. Recent developments in the

*Corresponding author.

fields of machine learning, computational intelligence, and data mining, however, have devised efficient algorithms for processing large and diverse databases, capable of providing meaningful insights from underlying patterns [3, 4, 5].

Condition-based monitoring (CBM) is a strategy that verifies the true condition of a system or equipment during its continuous operation. Thus, in contrast to planned and preventive maintenance, CBM can efficiently anticipate production chain problems. CBM has the following advantages [6]: it does not interrupt production to evaluate equipment behavior; it promotes a safer environment for production chain workers; it minimizes costs related to activity planning. According to [6], a CBM system may be structured into three main phases: data acquisition, data processing, and maintenance decision making. The first phase is usually the CBM bottleneck, since it requires an existing infrastructure for the acquisition of reliable data. In the second phase, the data is received and processed in order to better suit the next stage [7]. The third and final CBM stage is the phase in which most researchers spend their time pursuing innovative solutions [8, 9].

Machine learning is the most explored approach for decision-making. For instance, in this context, Yam et al. [10] used recurrent neural networks to develop an intelligent predictive decision support system for CBM, where Xavier and Seixas [11] applied a similar algorithm to analyze a chemical process. Widodo and Yang [12] and Helmy et al. [13] also used machine learning approaches based on support vector machines to model the decision-making process.

This work proposes a CBM system for oil and gas (O&G) wells and production/service lines that acts as a support tool to decision-making systems. The proposed system attempts to detect and identify an anomalous behavior as early as possible, so the operator, given the fault inherent criticality, can intervene accordingly to reduce the associated production losses. The work has been developed using a database developed by Petrobras comprising about 2000 events. They describe the normal operation (Class 0 in this work) and eight different types of fault (Classes 1 to 8) in well operation [14]. Our CBM system is an ensemble of machine learning techniques, from the pre-processing phase until the decision-making. The proposed system starts by extracting nine different statistical features from each available tag, and then performs a principal component analysis [3] to reduce the problem dimension as well as the associated computational complexity. This is followed by a random forest classifier, that is used to detect and classify the faulty conditions, aided by a Bayesian approach [15] employed to tune the classifier hyperparameters.

Three different classification scenarios are devised: fault detector, single-fault detector/classifier, and multi-fault classifier. Results in each experiment reach up to 90% of system accuracy, indicating its ability to detect and classify most of the fault types properly. In order to introduce its technical contributions, this paper is organized as follows: Section 2 describes the general problem of detecting faulty behaviors in O&G wells and production/service lines; Section 3 introduces the proposed CBM system and Section 4 details the database employed in this work, the so-called 3W database; Sections 5 and 6 present the experimental methodology and results, respectively, detailing and analyzing the performance of the proposed system in

the problem at hand. Finally, Section 7 closes the paper emphasizing its main contributions.

2. Problem Description

The production of O&G from underground reservoirs involves chemical and mechanical processes that affect well drilling and operation. Many of these processes may eventually cause a problem with the well, resulting in a decrease in production or in equipment failure. The majority of serious problems can be avoided or postponed by preventive maintenance techniques, or recognized at an early stage by means of regular analysis of production rates, fluids, and the mechanical condition of the well. Such practices can prevent expensive workovers that may be necessary to restore well production and can also prevent total well losses.

2.1. O&G General Fault Assessment

Since 2010, Petrobras has been developing a large database aimed at describing all of its O&G losses within its operational unit in Rio de Janeiro (OU-Rio). The so-called loss integrated management (LIM) platform congregates several complementary databases describing a production loss by a series of up to 85 different features such as: initial/final loss date, platform, affected equipment, equipment operator, related sensor tags, original cause, secondary cause, estimated loss, required initial/secondary actions, subsequent activities, and so on [16, 17]. Figure 1, for instance, shows the relative cumulative volume loss and number of failures between 2014 and 2017 of Petrobras OU-Rio which comprises 298 production/injection wells. From this figure, one can observe the significant amount of loss (23.8%) caused by reservoir- and well-related issues. A breakdown of these losses is depicted in Figure 2, indicating the main origins of the faults associated with such losses.

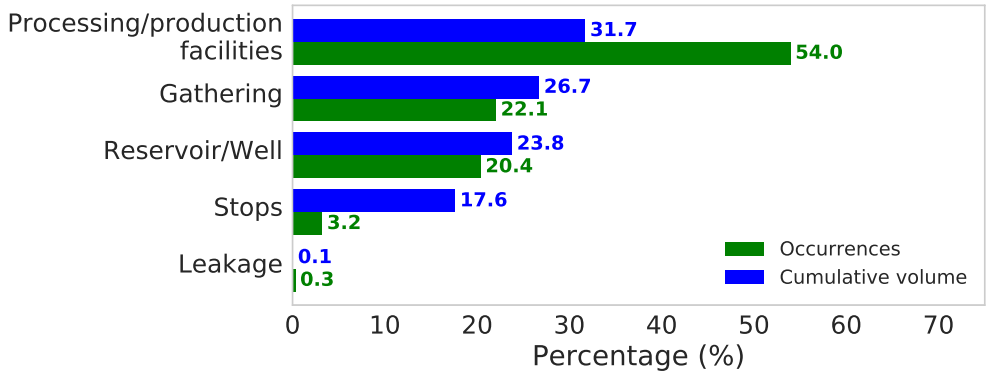

Figure 1: Breakdown of cumulative oil-volume loss (blue bars) and corresponding number of failures (green bars) between 2014 and 2017 in Petrobras OU-Rio for different production systems.

s

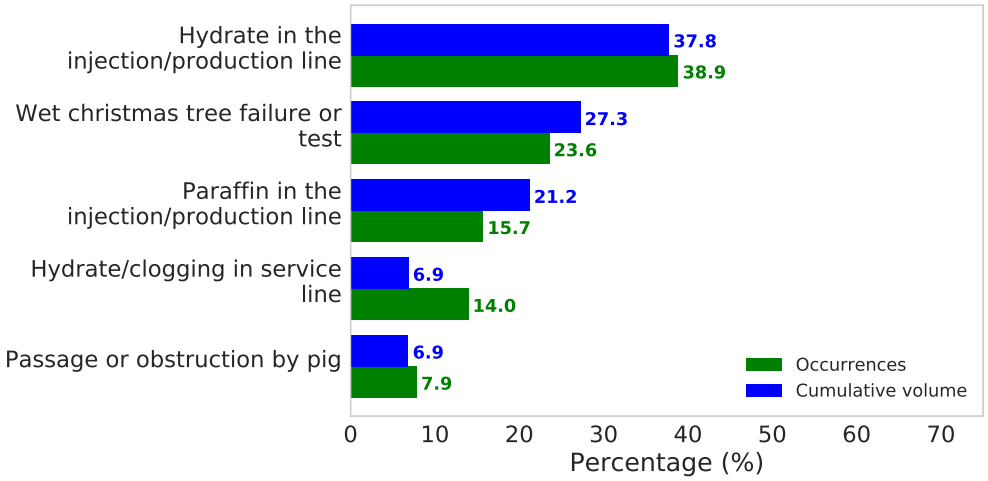

Figure 2: Breakdown of cumulative oil-volume loss (blue bars) and corresponding number of failures (green bars) between 2014 and 2017 in Petrobras OU-Rio for different fault causes.

2.2. Well/Line-Related Fault Description

In mid-2017, Petrobras conceived a project, entitled monitoring of specialized alarms, to develop a new automated system for detecting and classifying eight types of undesirable events in offshore naturally flowing wells (see the 3W database described in Section 4). The selected types of events are [14]:

Class 1 – Abrupt Increase of Basic Sediment and Water (BSW): The BSW is the ratio between water and sediment flow rate and the liquid flow rate, both measured under normal temperature and pressure, providing an insight on the amount of water in the produced oil. During the life cycle of a well, it is expected that the BSW increases due to water production from either the natural reservoir aquifer or artificial injection. An abrupt increase of BSW, however, is often associated with multiple problems related to flow assurance, lower oil production, oil lifting, incrustation, and so on. Early identification of this type of event helps to avoid these undesired conditions.

Class 2 – Spurious Closure of the Downhole Safety Valve (DHSV): The DHSV is installed in the production tubing of wells to ensure their closure in emergency scenarios or physical disconnection. Sometimes the closure function fails in a spurious manner without any indication on the surface. Production losses can be avoided if this spurious closure is detected as precociously as possible, allowing corrective operational procedures at an early stage.

Class 3 – Severe Slugging: Stratified gas-liquid flow, which occurs in scenarios of low liquid and gas flow rate, and a declined production line tend to cause liquid accumulation at the bottom of the riser, which blocks the gas flow until sufficient upstream pressure causes a surge of liquid and gas. After this sudden surge, some of the liquid in the riser returns to the base blocking the flow once again thus restarting the cycle. This transient cyclic phenomenon is called severe slugging and, depending on its periodicity and intensity, may become critical as it leads to stress or even damage to well equipment. Early detection of

this type of event allows preemptive actions to reverse the situation before it becomes critical.

Class 4 – Flow Instability: This type of undesirable event is also characterized by spurious surges of liquid and gas, as is the case of severe slugging, with the difference that it less intense and does not involve the complete cycle of liquid blockage followed by a gas surge. If not dealt with accordingly, flow instability can evolve to severe slugging.

Class 5 – Rapid Productivity Loss: Productivity of a naturally flowing well (as opposed to fluid injected wells) depends on several conditions. When the system energy is less than the minimum necessary to overcome energy loss, for instance, the oil flow slows or even stops. Therefore, early detection of this undesired condition reduces production losses.

Class 6 – Quick Restriction in the Production Choke (PCK): The PCK installed in the production unit is responsible for controlling the well from the surface. Manual operation of this type of valve can lead to unwanted quick restrictions, affecting the oil production directly.

Class 7 – Scaling in PCK: Another PCK-related undesirable event is the scaling due to inorganic deposits along time, which can severely reduce oil production. Early identification of this type of event is also desired, as special actions can be taken, such as injection of scale inhibitors, to avoid additional production losses.

Class 8 – Hydrate in Production Line: As shown in Figure 2, the presence of hydrate in wells and in production/injection lines is one of the biggest problems in the O&G industry, including Petrobras. Hydrates are crystalline compounds, resembling ice in appearance, formed by the reaction of natural gas to water. Early detection of this undesirable event means avoiding production losses for long periods, as unblocking production lines is costly and sometimes a long process.

During the well operation, it is possible that these faults interact with each other. For example, two or more faults can coincide, for instance, a Spurious Closure of DHSV (Class 2) suddenly occurs during a Scaling in the PCK (Class 7), which is a very slow event. In this case, the scaling should be detected before the closure occurs; otherwise, it would be impossible to detect it until the production restarts.

It is also possible that a fault triggers another fault from a different class. For instance, Flow Instability may precede Severe Slugging because these two faults often occur in scenarios of low liquid and gas flowrates and wavy flowlines. When the well starts to oscillate, depending on the line geometry and the flow characteristics, this instability can evolve to a more dangerous scenario of severe slugging.

Also, Severe Slugging (Class 3) and Flow Instability (Class 4) can be caused by faults related to partial flow area blockage, such as Rapid Production Loss (Class 5), Scaling and Quick Restriction in PCK (Class 6), and Hydrate in Production Line (Class 8). All these events may cause a reduction of the production flowrate without completely stopping the well production, leading to slugging and instability.

In this paper, a CBM system is proposed for automatically detecting and classifying the above event types using machine learning techniques. The data-driven approach learns underlying patterns during faulty

well operations and identifies such anomalous behaviors in subsequent system operations, as detailed in the following section.

3. System Overview

The proposed system is represented in Figure 3, which shows the data flow starting from its raw version all the way down to the system classification output, passing through the feature extraction, data transformation, and classification modeling stages. The first two blocks extract information in a compact form, whereas the last block associates the classification labels with the related characteristics according to the task at hand. Details of each system stage are provided in the three ensuing subsections.

Every system block was implemented using Python, which has become one of the standard programming languages for machine learning algorithms, due to its readability, growing community, and available libraries. In particular, in this work, we use two of these major libraries: Pandas [18], for data analysis, and Scikit-Learn [19], for handling the basic algorithms.

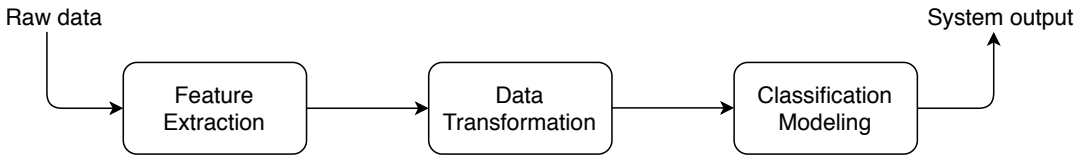

Figure 3: Block diagram of the proposed CBM system for O&G wells and production/injection lines.

3.1. Feature Extraction

Feature extraction is commonly concerned with highlighting important information to help the classification task. In the proposed system we process the raw input data from the available tags, all collected at a rate of one sample per second, and extract n_f features from each N-sample data window for each tag. Given an N-sample tag window $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, the $n_f = 9$ extracted features in the proposed system are the following statistical measurements:

- **Mean value:**

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i; \tag{1}$$

- **Standard deviation:**

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2}, \tag{2}$$

  which provides information on how the data is spread around its mean;

- **Skewness:** The third standardized moment of a random variable [20],

$$s_k = \frac{E[(X-\mu)^3]}{(E[(X-\mu)^2])^{\frac{3}{2}}}, \tag{3}$$

quantifies the asymmetry of the given data distribution: $s_k = 0$ stands for a symmetric distribution, where the mean is equal to the median;

- **Kurtosis:** The fourth standardized moment of a random variable [21],

$$k = \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} - 3, \tag{4}$$

measures the tails of the distribution, which are related to the presence of outliers. Positive and negative kurtosis indicate heavy-tailed and light-tailed distributions, respectively.

- 5**-number summary:** The last five features are given by the minimum, maximum, median, and first and third quartiles values of the data window [22]. The minimum and maximum values provide the range of the data distribution within a given window, the median is the center of the distribution, and the first and third quartiles show how the data is spread within its extremes.

In summary, the feature-extraction block receives an $\mathbb{R}^{M\times n}$ data matrix, with M time samples of n input tags, and transforms it into an $\mathbb{R}^{m\times n_f}$ matrix, where $m = \lfloor \frac{M-N}{s} \rfloor$ is the total number of N-sample data windows, s denotes the shift in samples between two consecutive windows, and $n_f = 9$ is the number of features extracted from each data window.

3.2. Data Transformation

The concept of data transformation is to represent data features in a more appropriate space. In our context, its purpose is twofold. First, it avoids range divergence among the extracted features. Second, it mitigates the "curse of dimensionality" problem, which means that although high dimensionality is expected to provide more information from the data, it also affects negatively its discriminative capability [23]. The first goal is achieved by performing z-score normalization on the data, and the second one by employing principal component analysis (PCA) on the result of the z-score normalization.

The z-score or standardized normalization [24] tends to equalize the features ranges and is performed as follows:

$$Z^i = \frac{X_f^i - E[X_f^i]}{\sigma_f}, \tag{5}$$

where X_f^i is a random variable representing one of our data features, σ_f is its standard deviation, and Z^i is the normalized version of X_f^i.

PCA [3] is a technique for reducing the data dimensionality while keeping most of its energy, that is in general equivalent to keep most of its representative information. Assuming a centered data matrix $\mathbf{X}_f = [\mathbf{x}_f^1, \ldots, \mathbf{x}_f^m]^T \in \mathbb{R}^{m\times n_f}$, with a sample covariance matrix $\mathbf{S} = (m-1)^{-1}\mathbf{X}_f\mathbf{X}_f^T$, the PCA formulation is given by

$$\mathbf{VS} = \mathbf{\Lambda V}, \tag{6}$$

where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_m)$ is the diagonal matrix of eigenvalues λ_i of $\mathbf{S}$ and $\mathbf{V} = [\mathbf{v}^1, \ldots, \mathbf{v}^m]^T$ is the orthonormal matrix of eigenvectors $\mathbf{v}^i$, also known as principal components, of $\mathbf{S}$. The PCA transformation $\mathbf{X}_f^m = \mathbf{V}\mathbf{X}_f$ performs a data projection into a new set of coordinates that are sorted in decreasing order of variance directions (i.e., $\mathbf{v}^1$ and $\mathbf{v}^m$ represent the maximum-variance and minimum-variance directions, respectively).

The dimensionality reduction consists in selecting only a small number $p < m$ of principal components $\mathbf{V}^p = [\mathbf{v}^1, \ldots, \mathbf{v}^p]^T$, which retain a large fraction (99%, for instance) of the original data information/energy, thus reducing the feature matrix dimensions from $\mathbb{R}^{m \times n_f}$ to $\mathbb{R}^{p \times n_f}$.

3.3. Classification Modeling

In the past few years, deep neural networks (DNN) have been applied in several O&G-related problems, including failure classification during drilling operations [25]. However, when compared to other ML algorithms, DNNs have several disadvantages such as the requirement of a large and adequately labeled dataset, the vast choices for the hyperparameter tuning (number of layers, number of neurons in each layer, type of nonlinear activation function, and so on), the complexity of its training procedure (due to a large number of internal coefficients), and its sensitivity to outliers or missing input data.

In this paper, we adopted the random forest [26] algorithm, which is a supervised machine-learning approach suitable for both binary and multiclass problems such as the ones considered here. The random forest technique has already been applied to many distinct problems, such as gene selection [27], remote sensing [28], prediction of proteins [29], and so on. The main characteristics of this algorithm are:

- it is robust to noise and outliers;
- it is faster than bagging and boosting methods [3, 5];
- it can provide useful error information (strength, correlation, and importance of the variable);
- it is simple to deploy (as it has a small number of hyperparameters to be tuned);
- it is easy to parallelize;
- it can handle missing data.

The random forest algorithm is an ensemble of decision trees, known as weak learners, for having low computational cost and low discriminative capabilities. Training a random forest classifier is equivalent to training several independent decision trees. When training each of these trees, distinct subsets of the input data and the extracted features are randomly drawn, so that each tree learns from a different partition of the data, as depicted in Figure 4. Properly combined, these trees can generate strong classifiers using the idea of wisdom of the masses [3, 4, 5]. After the learning procedure a new input data sample is labeled as the class that the majority of the decision tree classifiers voted for.

The vote distribution of all trees within a random forest can be interpreted as a probabilistic distribution for the system output. For example, if we use 100 trees to analyze a given input sample, which 80 trees estimate as Class 0 and the rest as Class 1, we can say that the classifier has 80% and 20% of certainty that this sample belongs to Classes 0 or 1, respectively. This property can be used to establish a threshold τ_i for each output class, with a given sample only belonging to that class if the certainty is above the corresponding threshold. In the context of decision-making systems, these classification thresholds allow one to balance the number of false positives and undetected faults according to the system priority: increasing the threshold values lowers the number of fault misclassifications at the cost of a reduction in the fault detection rates.

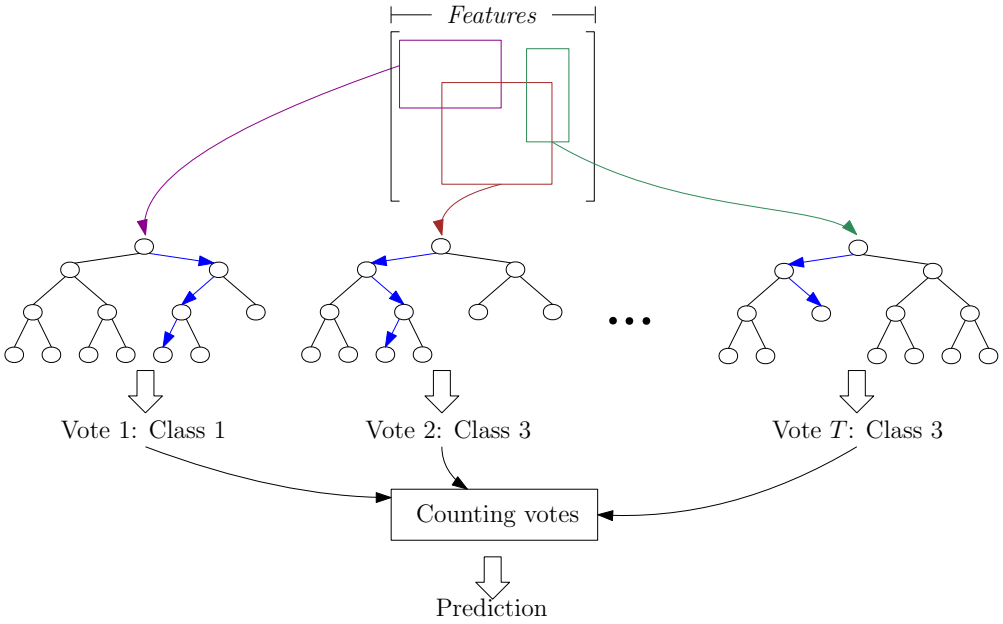

Figure 4: General schematic diagram of the random forest classifier.

4. The 3W Database

The development of the proposed machine-learning CBM system was based on the 3W dataset introduced in [14] (see also Sections 1 and 2). This database is composed by approximately 2000 operational events representing different states of the well, varying from normal operation (Class 0) to the eight distinct faults (Classes 1 to 8) described in Section 2. Each event is a time-series data composed by $n = 8$ tags acquired by 8 different sensors, chosen according to their availability and relevance to the faults at hand, as given in Table 1.

Three event types are included in the 3W database: real, simulated, and sketched[1]. Real events are described by sensor data acquired through the PI System [30] during real well operations. Signals of

[1] In [14], the authors named the sketched events as hand-drawn events.

Name	Description	Unit
P-PDG	Pressure at permanent downhole gauge (PDG)	Pa
P-TPT	Pressure at temperature/pressure transducer (TPT)	Pa
T-TPT	Temperature at temperature/pressure transducer (TPT)	°C
P-MON-CKP	Pressure upstream of production choke (CKP)	Pa
T-JUS-CKP	Temperature downstream of production choke (CKP)	°C
P-JUS-CKGL	Pressure downstream of gas lift choke (CKGL)	Pa
T-JUS-CKGL	Temperature downstream of gas lift choke (CKGL)	°C
QGL	Gas lift flow rate	m^3/s

Table 1: List of tags in 3W database, including tag names, descriptions, and measuring units.

simulated origin were obtained through the OLGA system [31], vastly used by the industry for dynamic multiphase flow simulation. Sketched signals were created through a tool designed by 3W database creators, which uses expert knowledge to sketch the profile of a particular unwanted event. Table 2 contains the quantitative description of 3W dataset per event type.

Class	Description	Real	Simulated	Sketched	Total
0	Normal	597	0	0	597
1	Abrupt BSW Increase	5	114	10	129
2	Spurious DHSV Closure	22	16	0	38
3	Severe Slugging	32	74	0	106
4	Flow Instability	344	0	0	344
5	Rapid Productivity Loss	12	439	0	451
6	Quick PCK Restriction	6	215	0	221
7	Scaling in PCK	4	0	10	14
8	Hydrate in Prod. Line	3	81	0	84
	Total	1025	939	20	1984

Table 2: Quantitative description of 3W database per event type.

When building the 3W database, besides the faulty periods, the authors have indicated for each event instance a period when the samples are not faulty (referred to as normal) and a period of fault transient before the actual steady-state fault consolidation, as depicted in Figure 5. Such annotation procedure allows one to detect a given fault during its initial transient stage. This is the behavior intended in the proposed system, in order to minimize maintenance costs and production losses.

5. Experimental Methodology

In this work, three experiments were devised in order to evaluate and understand the capability of the proposed CBM system:

Experiment 1 – One-class classifier: This scheme consists of a single classifier to discriminate only between normal and faulty operations. Therefore, in this case, all faults are combined into a unique class which is compared against the normal-operation class.

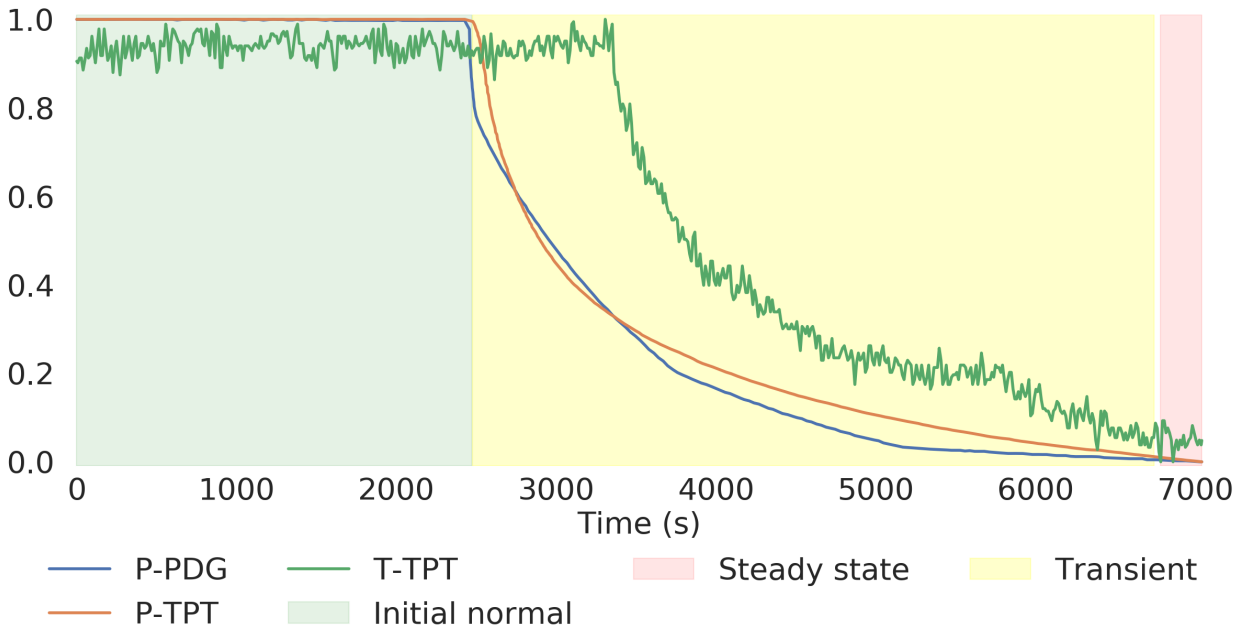

Figure 5: Example of (normalized) tag values of a Class 2 event where the background color indicates the normal (green), transient (yellow), and faulty (red) stages.

Experiment 2 – Multiple binary classifiers: This strategy consists in designing several classifiers, each one specialized in discriminating an individual fault against the normal-operation mode. In this strategy, one can infer which faults are the hardest to be identified, as opposed to the previous scheme which analyzes the faulty conditions altogether in a single class.

Experiment 3 – Single multiclass classifier: This scheme employs a single system to identify all different classes individually. In this case, each class (normal operation or any specific fault) is discriminated against all the remaining classes, thus providing more information to the system operator but posing a much harder problem to the machine-learning algorithm.

As described in Section 4, each annotated event in the 3W dataset has three phases: normal, transient, and steady state. Our main research goal was building a CBM system capable of anticipating a fault as much as possible, in order to give as much time as possible for the operator to intervene and minimize the losses. For this reason, in all experiments we have opted to use only the normal and transient phases of the event to train the random forest classifier. Also, for a more realistic scenario, the sketched instances were ignored and Class 7 was disregarded as it is under-represented in the 3W database. After all these considerations, the 3W database was split into the training and test sets, following a 70%/30% ratio, as indicated in Table 3.

Class	Description	Real	Simulated	Total
0	Normal	468(129)	0	468(129)
1	Abrupt BSW Increase	3(2)	78(36)	81(38)
2	Spurious DHSV Closure	15(7)	11(5)	26(12)
3	Severe Slugging	22(10)	55(19)	77(29)
4	Flow Instability	260(84)	0	260(84)
5	Rapid Productivity Loss	8(4)	340(99)	348(103)
6	Quick PCK Restriction	4(2)	170(45)	174(47)
8	Hydrate in Prod. Line	0(3)	56(25)	56(28)
	Total	780(241)	710(229)	1490(470)

Table 3: Number of instances in both training and test (between parenthesis) sets employed in the development of the proposed CBM system. Notice the absence of any Class 7 and sketched events.

5.1. Training, Validation, and Test

Building a system with a wide variety of hyperparameters demands a robust and reliable training routine. In this work, we performed a cross-validation [32] using $k = 5$ folds. In order to avoid contamination during this procedure, all samples of a given event belong to the same fold. Also, we kept the same class proportions in the training set throughout every cross-validation iteration.

In our experiments, we have considered a hyperparameter to evaluate the amount of balance between normal (including n_0 samples from Class 0 and n_{N} samples from the initial normal phase in the faulty instances) and faulty samples. We refer to this parameter as b, and, given n_0 and n_{N}, we have $b = n_0/n_{\mathrm{N}}$. So, the total number of normal and faulty samples used during training becomes

$$n_{\mathrm{normal}} = n_{\mathrm{faulty}} = (b + 1)n_{\mathrm{N}}. \tag{7}$$

Even though b is not mandatory when training a random forest classifier, incorporating it to our hyperparameter search enables us to increase the performance on the samples from the initial normal phase in the faulty instances. Increasing b makes n_{N} less relevant in comparison to n_{faulty}, feeding the model with less information about those n_{N} samples.

Many hyperparameters have a direct impact on the system architecture, and thus can influence its overall performance. In the proposed system, for instance, we considered, for the PCA stage, the minimum number of components that guarantee a 99% threshold for the accumulated energy. A fixed sliding window step of $s = 50$ samples was also employed.

The strategy applied to assess the best set of the remaining hyperparameters was a combination of the traditional grid search and a Bayesian optimization method. The Bayesian method applied here [15] is a non-convex optimization algorithm that samples the objective function according to a Gaussian process. In each optimization iteration, the algorithm considers all past observations to evaluate which parameter regions are worth exploring, thus narrowing the hyperparameter search space following a probabilistic-based strategy.

For choosing the sliding window size and the balance ratio, we performed grid search using the following sets of values: $M = \{100, 200, 300, \cdots, 900, 1000\}$ and $b = \{1, 2, 3, 4, 5\}$. As for the number of trees N_t and the maximum depth M_d of each tree in the random forest algorithm, we applied 50 iterations of a non-convex optimization algorithm based on Bayesian sampling of the objective function, as described in [15]. The following parameter ranges have been used: $50 \leq N_t \leq 250$ and $5 \leq M_d \leq 70$.

5.2. Metrics Employed for Performance Assessment

In this work we evaluate the classifier models through three different metrics:

- Accuracy (ACC): considering all normal and faulty samples, the ACC can be computed as:

$$\text{ACC} = \frac{TP + TN}{n_{\text{total}}}, \tag{8}$$

  where TP and TN are respectively the number of true positives and true negatives, in samples, and n_{total} is the overall number of samples;

- Faulty-normal accuracy (FNACC): is the accuracy computed when considering only the normal samples preceding a faulty instance;

- Real faulty-normal accuracy (RFNACC): is the accuracy computed when considering only the normal samples preceding a real faulty instance.

While the standard ACC addresses the overall model performance, it does not evaluate its capability of discriminating between the transient phases in the faulty events and the normal samples preceding them. Therefore, in order to assess the system false-alarm rate, the FNACC and RFNACC are more indicated, with the later proving insight only of the performance for real faulty events.

6. Experimental Results and Discussion

In this section, we present the results for all three classification strategies described in Section 5. This provides a comprehensive assessment of the final system performance, under different fault-detection scenarios.

6.1. Experiment 1

After evaluating 2500 different binary models (following 50 runs of the Bayesian optimizer, for ten different values of window size M and five values of balance ratio b), the best classifiers according to the three distinct evaluation metrics are shown in Table 4, along their average validation performance.

For the evaluated configurations, we may state that larger values of b are associated to larger values of ACC and smaller values of RFACCC, as depicted in Figure 6. This happens because a larger b leads to an increased Class 0 sample representation in comparison to the normal-phase samples preceding faults in the

Model	M	b	N_t	M_d	ACC	FNACC	RFNACC
1.1	700	5	179	70	0.986	0.935	0.660
1.2	600	1	86	53	0.979	0.965	0.733
1.3	500	2	118	38	0.983	0.961	0.773

Table 4: Characterization of best classifiers in Experiment 1, each one according to a specific accuracy metric, along their validation performances: window length M, normal-class balance ration b, random forest number of trees N_t, and maximum tree depth M_d.

training datasets. From Figure 6, one may also notice that the window length M and the number of trees N_t do not have a strong impact on the final results, whereas the maximum tree depth M_d may hinder the final performance in this experiment if one chooses $M_d < 15$.

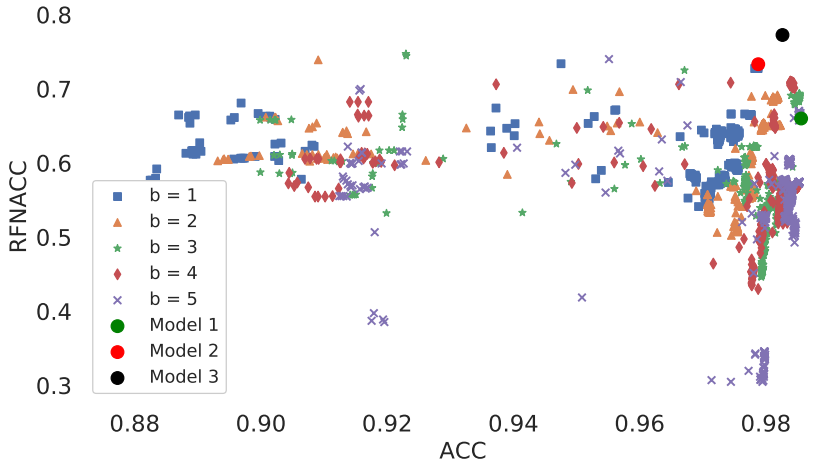

Figure 6: ACC × RFACC relationship for all 2500 classifier configurations evaluated in Experiment 1, indicating that larger balance ratios b correspond to larger ACC and smaller RFACC values.

From a practical perspective, analyzing the model performance in the real instances is far more critical than analyzing its performance in the simulated ones. Moreover, the model must avoid false positives, as each time this happens, the operator loses confidence in using the system. For these reasons, we chose to analyze Model 1.3 test performance. The breakdown of the test result is shown in Table 5, including its performance in classifying the steady-state phase of the faults (which were not used in training).

Event	Class 0	Initial Normal	Transient	Steady-State	Overall
Simulated	–	0.990	0.995	0.958	0.971
Real	0.995	0.673	0.822	0.972	0.971
Overall	0.995	0.938	0.982	0.960	0.971

Table 5: Model 1.3 test accuracy results for each separate sample type, including simulated/real events. The empty entry indicates the absence of data for that particular event type.

Although Model 1.3 was able to reach a very large overall accuracy of ACC = 0.971, Table 5 results show its difficulty in detecting properly the normal samples preceding faults in real events, what may lead

to an undesired large number of false alarms. One of the possible causes for this poor behavior is the system difficulty in modeling the distinct dynamics of all different faults combined into a single class. The next experiment avoids this issue by designing individual classifiers for each separate fault.

6.2. Experiment 2

In this experiment, we select the three configuration models chosen in the previous experiment (that optimize each of the three distinct performance measures), as given in Table 4, and retrain them under the new circumstances. In this case, we use a different classifier to distinguish between the normal and faulty operations for each fault type. Table 6 summarizes all the test results for each model and for each fault type (Classes 1 to 8, except Class 7). As one can notice, in general, Models 1, 2 and 3 yield larger ACC, FNACC, and RFNACC values, as each model tends to prioritize each of these metrics, respectively.

Fault Class	Model	ACC	FNACC	RFNACC
1	1.1_1	0.992	0.932	0.170
1	1.2_1	0.994	0.975	0.704
1	1.3_1	0.994	0.982	0.787
2	1.1_2	0.999	0.994	0.992
2	1.2_2	0.998	0.985	0.970
2	1.3_2	0.998	0.981	0.963
3	1.1_3	1.000	–	–
3	1.2_3	1.000	–	–
3	1.3_3	1.000	–	–
4	1.1_4	0.990	–	–
4	1.2_4	0.989	–	–
4	1.3_4	0.989	–	–
5	1.1_5	0.984	–	0.503
5	1.2_5	0.962	–	0.965
5	1.3_5	0.968	–	0.875
6	1.1_6	0.966	0.976	0.999
6	1.2_6	0.963	0.967	0.923
6	1.3_6	0.962	0.977	1.000
8	1.1_8	0.969	0.994	1.000
8	1.2_8	0.968	0.998	1.000
8	1.3_8	0.969	0.992	1.000

Table 6: Experiment 2 test results for each classifier model and for each fault type (Classes 1 to 8, except Class 7). Empty entries indicate the absence of data for that particular event type. The models are named as 1.X_Y, where X corresponds to the model chosen in Experiment 1 and Y to the Fault Class index.

These results show that the proposed system can properly detect and classify all faults, particularly the ones of Classes 2, 3, 4, 6, and 8, where the accuracy levels reached 0.9 or higher for every metric. Even Classes 1 and 5 yielded good accuracy values but presented lower RFNACC when compared to other classes.

This experiment shows that the developed system can identify properly all individual faults against the normal operation, even when using the same model parameters for all classes. As pointed out in Section 2,

different faults can occur at the same time or even evolve to other fault types. Therefore, in the next experiment, we tackle these scenarios altogether using a multiclass strategy.

6.3. Experiment 3

In this scenario, a new hyperparameter search for the multiclass system was performed considering the same ranges explored in Experiment 1. Table 7 brings the results for the three system models considering the different accuracy metrics as before.

Model	M	b	N_t	M_d	ACC	FNACC	RFNACC
3.1	500	5	102	24	0.973	0.936	0.515
3.2	400	1	86	27	0.962	0.963	0.715
3.3	800	2	238	30	0.970	0.962	0.719

Table 7: Characterization of best classifiers in Experiment 3, each one according to a specific accuracy metric, along their validation performances: window length M, normal-class balance ration b, random forest number of trees N_t, and maximum tree depth M_d.

When comparing Tables 4 and 7, one readily notices the similar performances achieved in each case, despite the different classification problem. This can be explained by the difficulty posed in Experiment 1 for grouping all fault types into a single class, not respecting their distinct natures and dynamics, what can be inferred from the results obtained in Experiment 2.

A breakdown of all classification test results obtained by Model 3.3 is provided in Table 8. These results correspond to an overall system accuracy of 0.94 and excellent classification rates specially for Classes 3, 4, 5, and 6.

We can also notice that classes with less samples have worst performances, as less data hinders the learning process during the system training. This culminates in Class 8, which had no real samples in the training set and presented the lowest test accuracy, indicating that the simulated instances do not emulate properly the real cases. To verify this hypothesis, an experiment was conducted where Model 3.3 was retrained with the number of Class-4 events reduced to only 10% of the available instances. In that scenario, the overall test accuracy dropped to 92.3% and the accuracy within the Class 4 dropped from 95.5% to only 51.4%, as explained above.

Also, an overall confusion matrix is shown in Figure 7, which shows that the model successfully identifies almost all fault cases. The misclassification of type-8 into type-3 faults, as seen in this matrix, can be explained by the fact that severe slugging (Class 3) often results from hydrate in the production line (Class 8), which is an example of flow area blockage, as mentioned in the end of Section 2

These error patterns may be mitigated in a real scenario by providing additional side information to the system operators, thus enabling them to sort out the system output based on their own practical knowledge.

Event	Type	Initial Normal	Transient	Steady-State	Overall
Class 0	Real	–	–	–	0.994
Class 1	Real	0.719	0.454	0.000	0.508
	Simulated	1.000	0.996	0.944	0.978
Class 2	Real	0.982	0.882	0.817	0.888
	Simulated	0.978	0.849	0.035	0.234
Class 3	Real	–	–	0.791	0.791
	Simulated	–	–	0.956	0.956
Class 4	Real	–	–	0.954	0.954
	Simulated	–	–	–	–
Class 5	Real	0.443	0.953	–	0.833
	Simulated	–	0.997	0.940	0.949
Class 6	Real	0.832	0.153	0.146	0.710
	Simulated	0.994	0.983	0.873	0.908
Class 8	Real	0.000	0.000	0.000	0.000
	Simulated	0.998	0.985	1.000	0.989
Overall	Real	0.615	0.511	0.901	0.930
	Simulated	0.998	0.992	0.918	0.944
Overall	–	0.934	0.955	0.916	0.940

Table 8: Model 3.3 test accuracy results for each separate sample type, including simulated/real events. Empty entries indicate the absence of data for that particular event type.

6.4. Event-Based Assessment

Regarding the system operating in a real scenario, its main goal is detecting an event as soon as possible despite misclassifying some time samples. Therefore, the goal of this subsection is to analyze the performance of Model 3.3, which focuses on real events, at the event level.

We consider that an event is correctly detected and classified if the model obtains an accuracy rate higher than a given threshold on the samples of that particular event. In this paper, we have opted for a conservative accuracy rate of 0.9 at the sample level to consider an event properly classified. In Table 9, we present the classification results for each fault type, discriminating the number of correctly classified events, for each event phase. Even though the classifier was not trained with steady-state samples, we used Model 3.3 to predict it, considering the label of the same fault transient label. The column "All" represents the scenario where the model detected a fault during its transient or during the subsequent steady-state period, and correctly detect the initial normal phase. These event-level results show a quite satisfactory performance for the proposed system, with low false-alarm rates and high true-positive identification/classification rates.

In Figure 8, we have an example of a sample-level classification (red crosses), where the '0' and '1' outputs correspond to normal and faulty states, respectively. Two new time-intervals (in seconds) are also defined in this figure: the time of detection, t_1, between the beginning of the transient phase and the fault detection provided by the system (as confirmed by 120 consecutive correct detections); and the time to take action, t_2, between the system fault detection and the fault consolidation (beginning of steady-state phase). According to the definitions of t_1 and t_2, the value of $(t_1 + t_2)$ is constant and represents the duration of the

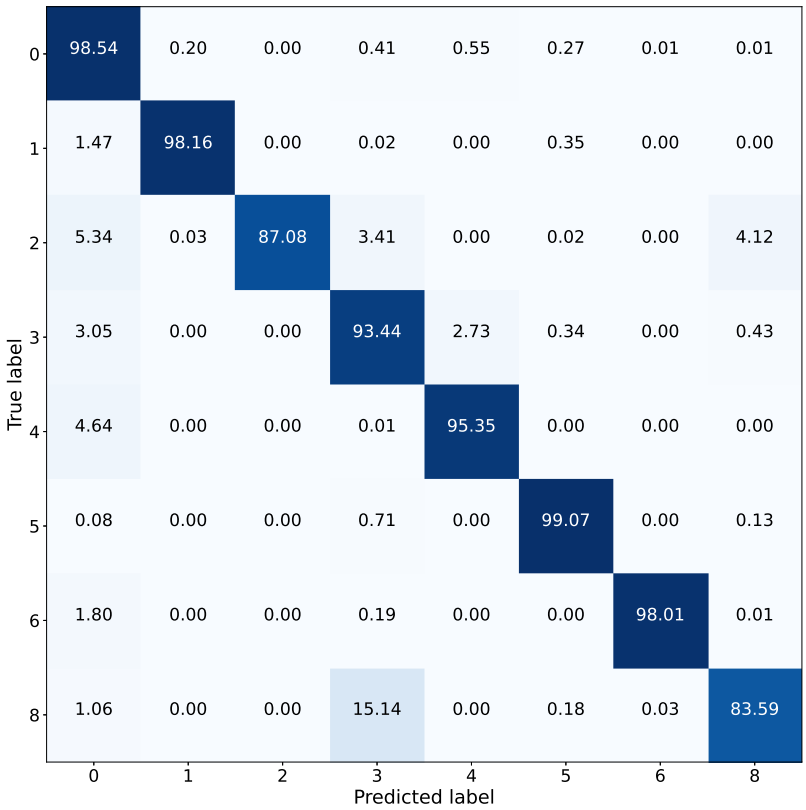

Figure 7: Confusion matrix showing final test results (in percentages) for Experiment 3.

Class	Initial Normal	Transient	Steady-state	All
0	127(129)	-	-	-
1	37 (38)	36 (38)	30 (38)	36 (38)
2	10 (12)	8 (12)	1 (9)	7 (12)
3	-	-	23 (29)	-
4	-	-	74 (84)	-
5	1 (4)	99 (103)	85 (101)	0 (4)
6	45 (47)	44 (47)	36 (47)	43 (47)
8	25 (27)	24 (27)	17 (20)	24 (27)

Table 9: Event-level successful detection/classification results (and total number of events between parenthesis) for each event phase and fault class. Column "All" includes the events where the model detected the fault during its transient phase or during the steady-state period, and correctly identified the initial normal phase preceding the fault. Blank entries indicate lack of particular data in 3W dataset. The classifier used was the one of the model 3.3.

transient phase. So, as t_1 decreases (which indicates less time to detect faults), t_2 decreases.

The average values of t_1 and t_2 for each fault class in the test set are shown in Table 10, as well as the average percentage values of these parameters with respect to the total transient-phase duration. Classes 0, 3, and 4 are omitted here as their transient phase is not specified in the 3W database. These results show that the system can not only act as a fault classifier, but it can also anticipate the failure during its early stage. In the worst scenario, the system detects the failure within 12% of its transient duration, giving an

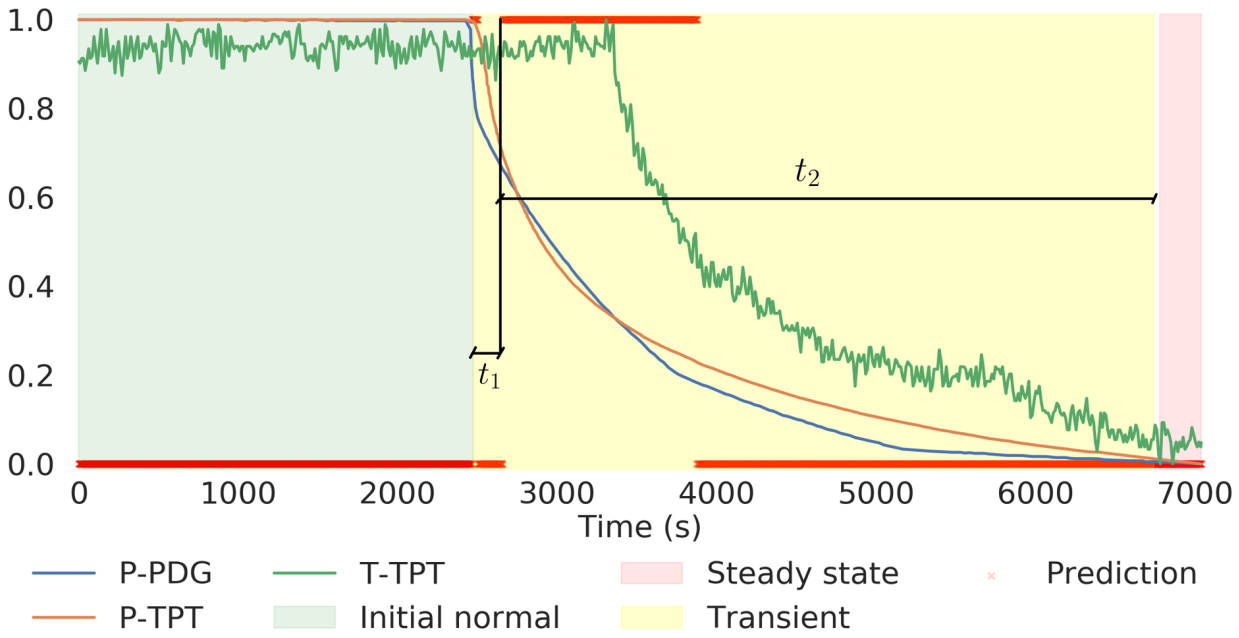

Figure 8: Example of a real instance of Class 2 alongside the system classification (red crosses) for each window sample: '0' and '1' outputs correspond to normal and faulty states, respectively.

additional time for the operators to intervene and prevent major production losses..

Class	t_1 [s]	t_2 [s]
1	293 (2.76%)	39804 (97.24%)
2	267 (8.77%)	4063 (91.23%)
5	6 (0.19%)	4966 (99.81%)
6	27 (3.69%)	6895 (96.31%)
8	2865 (11.09%)	15742 (88.91%)

Table 10: Average delay (t_1) and anticipation (t_2) intervals, in seconds, for the proposed multiclass Model 3.3 system. The number designated in parenthesis is the percentage of the corresponding time interval with respect to the total transient-phase duration. Classes 0, 3, and 4 are omitted here as their transient phase is not specified in the 3W database.

7. Conclusion

This paper described a full methodology for detecting and classifying faulty events during the practical operation of O&G production wells and lines. Seven fault types were considered along with the normal operation state. The developed system uses a classifier based on the random forest algorithm and a Bayesian non-convex optimization strategy to tune the system hyperparameters.

Three experiments were devised to evaluate system capability and robustness in different fault detection/classification scenarios: Experiments 1 and 2 consider the binary normal $\times$ faulty conditions, where the faults are treated altogether and individually, respectively; Experiment 3 addresses the multiclass scenario, where the system performs simultaneous fault detection and classification, which is best for practical usage.

In the multiclass configuration, for instance, overall accuracy results above 94% indicate successful performance of the proposed system in detecting and classifying all faults types, thus reducing risk and production losses in a real operation scenario. Alongside the high accuracy, the system also achieved a short detection delay, identifying the fault before completing 88% (in average) of its transient period, thus providing additional time to the operator to mitigate associated damages.

Acknowledgments

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, and Petrobras. The authors would also like to thank the Petrobras team from the *Unidade de Negócios-Espírito Santo* for making the 3W dataset publicly available.

References

[1] T. Segaran and J. Hammerbacher, *Beautiful Data: The Stories Behind Elegant Data Solutions*, O'Reilly Media, 2009.
[2] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *Science*, vol. 332, pp. 60–65, 2011.
[3] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, New York, 2006.
[4] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 4th ed., 2009.
[5] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data* AML Book, 2012.
[6] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, Oct. 2006.
[7] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 4th ed., San Francisco, 2016.
[8] W. Liu, B. Tang, J. Han et al., "The structure healthy condition monitoring and fault diagnosis methods in wind turbines: A review," *Renewable and Sustainable Energy Reviews*, vol. 44, pp. 466–472, Apr. 2015.
[9] A. Grall, C. Bérenguer, and L. Dieulle, "A condition-based maintenance policy for stochastically deteriorating systems," *Reliability Engineering and System Safety*, vol. 76, no. 2, pp. 167–180, May 2002.
[10] R. C. M Yam, P. Tse, L. Li et al. "Intelligent predictive decision support system for condition-based maintenance," *International Journal of Advanced Manufacturing Technology*, vol. 17, no. 5, pp. 383–391, Feb. 2001.
[11] G. M. Xavier and J. M. Seixas, "Fault detection and diagnosis in a chemical process using long short-term memory recurrent neural network," *Proc. International Joint Conference on Neural Networks*, pp. 1–8, Rio de Janeiro, Brazil, 2018.
[12] A. Widodo and B.-S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 21, no. 6, pp. 2560–2574, Aug. 2007.
[13] T. Helmy, A. Fatai, and K. Faisal, "Hybrid computational models for the characterization of oil and gas reservoirs," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5353–5363, July 2010.
[14] R. E. V. Vargas, C. J. Munaro, P. M. Ciarelli, A. G. Medeiros, B. G. Amaral, D. C. Barrionuevo, J. C. D. Araújo, J. L. Ribeiro, L. P. Magalhães, "A realistic and public dataset with rare undesirable real events in oil wells," *Journal of Petroleum Science and Engineering*, vol. 180, pp. 62–77, Oct. 2019.
[15] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms", *Proc. International Conference on Neural Information Processing Systems*, pp. 2951–2959, Nevada, 2012.
[16] I. H. Santos, A. R. M. Gaban, A. A. Lima, T. de M. Prego, B. D. Barros, M. A. Marins, E. A. B. da Silva, M. L. R. Campos, and S. L. Netto, "On the automatic identification of critical production systems from a production loss database," *Proc. Rio Oil & Gas Expo and Conference*, pp. 1–7, Rio de Janeiro, 2018.
[17] I. H. Santos, H. F. Lisboa, T. de S. Feital, M. M. Câmara, R. M. Soares, M. A. Marins, B. D. Barros, T. de M. Prego, A. A. Lima, and S. L. Netto, "Hydrate failure detection in production and injection lines using model and data-driven approaches," *Proc. Rio Oil & Gas Expo and Conference*, pp. 1–13, Rio de Janeiro, 2018.
[18] W. McKinney, "Data Structures for Statistical Computing in Python," *Proceedings of the 9th Python in Science Conference*, pp. 56–61, Texas, Jun. 2010.
[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research* vol. 12, pp. 2825–2830, Oct. 2011.
[20] S. Kokoska and D. Zwillinger, *CRC Standard Probability and Statistics Tables and Formulae*, CRC Press, Boca Raton, 1999.
[21] L. T. Decarlo, "On the meaning and use of kurtosis," *Psychological Methods*, vol. 2, no. 3, pp. 292–307, 1997.
[22] D. C. Hoaglin and F. Mosteller, *Understanding Robust and Exploratory Data Analysis*, Wiley-Interscience, New York, 2000.

[23] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," *Proc. International Conference on Database Theory*, pp. 420–434, Berlin, Jan. 2001.
[24] C. Feng, H. Wang, N. Lu, T. Chen, H. He, Y. Li and X. M. Tu, "Log-transformation and its implications for data analysis," *Shanghai Archives of Psychiatry*, vol. 26, no. 2, pp. 105–109, 2014.
[25] A. Ambrus, N. Saadallah, S. Alyaev, and F. Iversen, "Automatic detection of anomalous drilling operations using machine learning methods and drilling process simulations," *Oil Gas European Magazine*, vol. 45, pp. 15–16, 2019.
[26] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[27] R. Díaz-Uriarte and S. Alvarez de Andrés, "Gene selection and classification of microarray data using random forest," *BMC Bioinformatics*, vol. 7, no. 1, pp. 3–16, Jan. 2006.
[28] M. Belgiu and L. Drăgut, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, no. 1, pp. 24–31, Apr. 2016.
[29] J. Jia, Z. Liu, X. Xiao, B. Liu and K. Chou, "pSuc-Lys: Predict lysine succinylation sites in proteins with PseAAC and ensemble random forest approach," *Journal of Theoretical Biology*, vol. 394, no. 1, pp. 223–230, Apr. 2016.
[30] OSI Soft, *PI System*, [Online]. Available at: https://www.osisoft.com/pi-system/. [Access in 08/13/2019].
[31] Schlumberger, *OLGA Dynamic Multiphase Flow Simulator*, [Online]. Available at: https://www.software.slb.com/products/olga. [Accessed in 08/13/2020].
[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, New York, 2001.