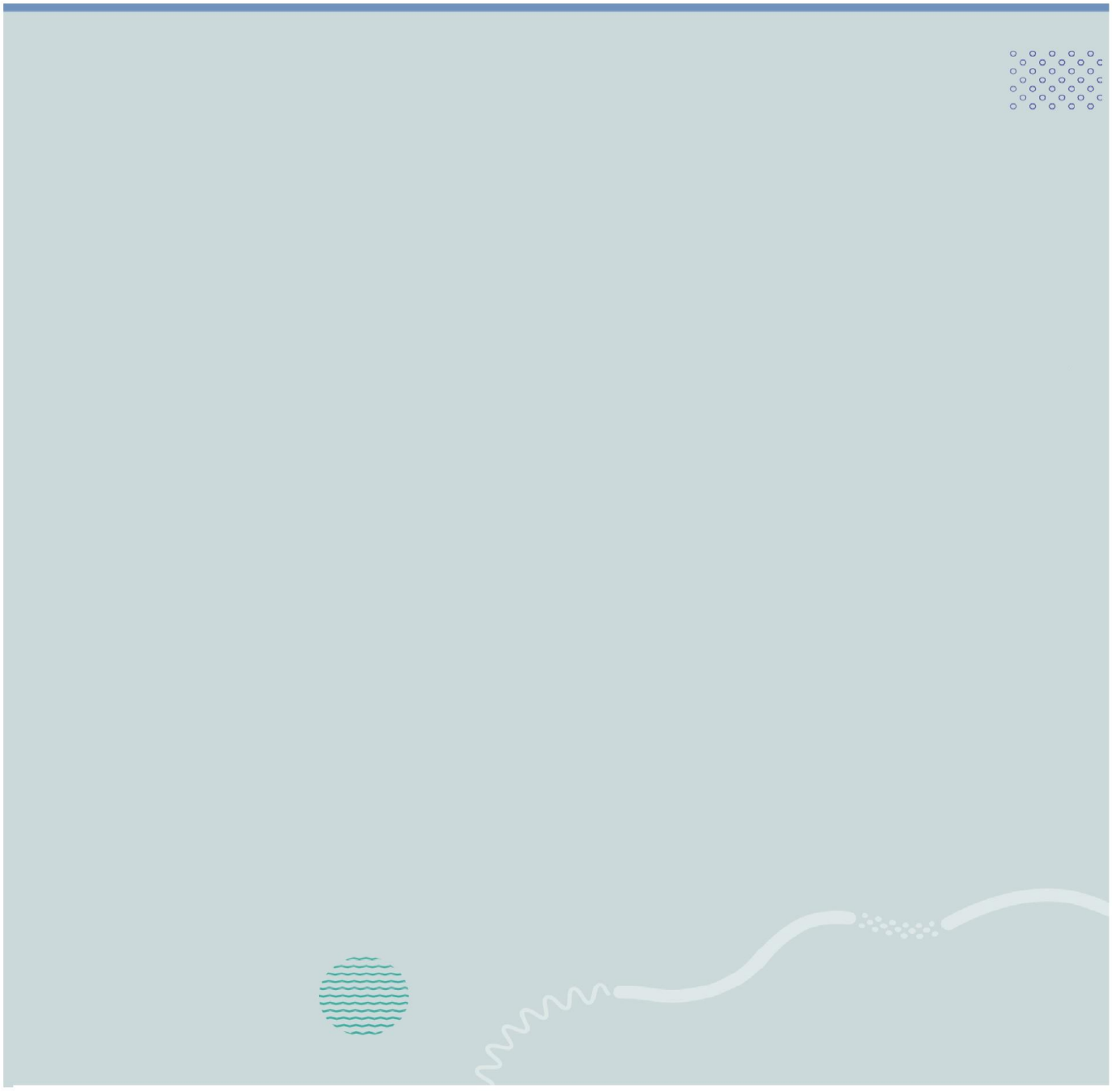
University of South-Eastern Norway
Faculty of Technology and Maritime Sciences
-
Master Thesis in System Engineering With Embedded Systems
Kongsberg Department of Engineering
November 26, 2020

Chrisander BRØNSTAD

Data-driven detection and identification of undesirable events in subsea oil wells



Abstract

Condition-based monitoring (CBM) systems have gained huge popularity in recent years with technological leaps that have arisen. Sensor-technology, communication systems, and computational capability have introduced innovative systems to monitor, analyze, and identify failures in industrial plants, production lines, machinery, and equipment. The gas and oil industry lose billions of dollars yearly related to abnormal events. Thus, abnormal event management (AEM) has become their number one priority, which aims to timely detect and diagnose abnormal events so that preventive actions can be taken.

Similar to AEM, this research deals with the detection and classification of faulty events in offshore oil wells by creating a CBM system. The events used in this work are a part of the 3W database developed by Petrobras, considered the world's third-largest oil producer. Seven events categorized as faulty events are considered, as well as instances considered as normal operation. This work conducts three experiments. The first experiment is related to a new feature extraction strategy, while the last two experiments are related to two different classification scenarios. The proposed systems achieve an overall accuracy of 90%, indicating that the system is not only able to detect faulty events but also successfully anticipating incoming failures.

Acknowledgements

First and foremost, I would like to convey my sincere and deepest appreciation to my supervisor Professor Antonio L. L. Ramos, for his guidance and inspiration during my undergraduate and graduate education. I am thankful for suggesting working on this topic and for the international experience I was able to acquire through his collaboration projects with prestigious higher education institutions abroad. This was a unique opportunity that enriched my learning experience at USN.

I would also like to express my deepest appreciation to Professor Sergio Lima Netto, for his hospitality and availability during my visits to the Federal University of Rio de Janeiro (UFRJ). I am deeply thankful to Professor Sergio for suggesting this specific task for my case study and facilitating access to the data to carry out this research, and for sharing his insight and experience throughout the process. I am deeply grateful for his unconditional guidance and support throughout this process.

I am also grateful to my colleague Eivind Haldorsen, who has also been a great friend and a source of encouragement throughout the master's degree. Many thanks to the students at the SMT Laboratory at UFRJ that always let me feel welcomed and showing great hospitality during my visits to UFRJ. A special thanks to Rafael Padilla, Lucas Cinelli, and Matheus Marins, for taking the time to share their experiences, specially in the initial stages where it was most needed.

Lastly, to my family, thank you very much for the patience and support you have shown me. Your encouragement was worth more than I can express on paper.

Chrisander Brønstad
Kongsberg, Norway, November 26, 2020

Contents

List of Figures

List of Tables

List of Abbreviations

AEM	**A**bnormal **E**vent **M**anagement.
AI	**A**rtificial **I**ntelligence.
CBM	**C**ondition-**B**ased **M**onitoring .
CV	**C**ross **V**alidation.
DT	**D**ecision **T**ree.
KNN	**K**-**N**earest **N**eighbors.
ML	**M**achine **L**earning.
NN	**N**eural **N**etwork.
PCA	**P**rincipal **C**omponent **A**nalysis.
PCK	**P**roduction **C**hoke **V**alve.
PDG	**P**ermanent **D**ownhole **G**auge.
RF	**R**andom **F**orests.
SVM	**S**upport **V**ector **M**achine.
TPT	**T**emperature and **P**ressure **T**ransducer.

Chapter 1

Introduction

The interest in condition-based monitoring of industrial machines has grown in recent years. It is one of the most innovative approaches to cope with machinery failures and is a solution employed by corporations and governments alike. This technique has been around for a quite some time, especially in terms of military usage for aircraft, and can be traced back to the second world war. In that period, Britain's Royal Air Force was plagued by equipment issues and in some squadrons, less than half of the aircraft were operational. The man who was in charge of the maintenance at the time was C.H Waddington, and through his observations, he came to the conclusion that the rate of failure or unscheduled repairs, were more frequent after scheduled maintenance. This phenomenon has become known as the 'Waddington Effect' and led to the very first development in condition-based monitoring. Their solution was to adjust the maintenance process to correspond with the physical condition of the equipment based on reported issues, and the frequency of its use [8, 27].

The advancements in sensor technology, communication systems for data acquisition, storage, and computational capability have caused an era of massive automatic data collection. These advancements in technology and large gatherings of data have caused a paradigm shift, which provides opportunities to develop innovative solutions and systems. Smart technology has become a term for such systems and Germany has successfully launched a project known as "Industry 4.0" to revitalize the industry based on the aforementioned. Today, condition-based monitoring (CBM) applies state-of-art technology and is often related to "smart technology" solutions such as Cyber-physical systems (CPS) and Internet of Things (IoT) systems [15, 16]. These systems aim to improve the efficiency, reliability, and productivity of industrial applications. The solutions are often with data-driven analysis, where acoustic and vibration signals, current, and temperature are examples of features that are monitored to identify the condition of bearings, motors, and other machinery [1].

1.1 Condition-based monitoring

Condition-based monitoring has become a method adopted to monitor and identify the condition of processes, machinery, or component under investigation. It is often used with two different strategies. One of the strategies is directly related to maintenance and aims at perceiving and anticipating the remaining useful lifetime of a component, machine, or process so that planned maintenance can be scheduled. The other strategy aims at anticipating and detecting incoming failures, so that preventive actions can be taken during its continuous operation to preserve a stable production by avoiding unexpected downtime. However, the goal of any intelligent CBM system

is to make decisions without human interaction. An example of enabling technologies of this kind includes sensors with built-in intelligence (SMART sensors), which are capable of extracting rich, high-grade information in combination with algorithms base on machine learning (ML), that can analyze trends within raw-sensory data [10]. CBM systems consist of three main steps, namely data acquisition, data processing, and maintenance decision-making. The first step collects and stores data relevant to the system. The second step starts by preprocessing and analyzing the data collected from the previous step for better understanding and interpretation of the data. Here, techniques such as time-domain and frequency-domain analysis are common. The last and final step of the system is decision-making support, where the goal is to provide prognostics to predict fault or failures before they occur [12].

1.2 Oil and gas industry

The oil and gas industry has some of the most demanding requirements for operational safety, productivity, and efficiency. This is because undesirable abnormal events can cause production losses for days and even weeks. It is estimated that the oil and gas industries lose 20 billion dollars every year. Thus, they have rated abnormal event management (AEM) as their number one problem that needs to be solved. As CBM systems, AEM address fault detection and diagnosis. Through different means, it aims to timely detect, diagnose, and correct abnormal conditions of faults in a process [33].

1.2.1 Current State-of-the-art

Applying algorithms for detection and classification in the oil and gas industry have become important tools during the early (e.g., drilling and construction) and late stages (e.g., production phase and operating the oil well with all its subsystems) of an oil well. This can be as early as the well-testing interpretation, which was investigated by Ahmadi et al. 2017. That work presents an approach to determine underlying reservoir models from noisy pressure data. The authors investigate the use of random forests (RF), Support Vector Machine (SVM), linear regression (LR), and probabilistic neural networks (PNN) as classifiers for well-testing model classification, from pressure transient test data with geometric features. That work demonstrates prominent results, where the RF classifier achieved an accuracy as high as 94.9%. Other examples of detection techniques and algorithms in the early stages of the oil well include the work of Tang et al. 2019. In that work, the authors present a method that uses real-time drilling data to automatically detect flow influx during drilling. The authors investigate the use of statistical features for this purpose, such as, quantifying the increase and decrease in local fluctuations. This approach showed reliable performance and was able to predict undesirable flow influx trends 10 minutes before reported detection, on average.

Examples of detection algorithms in later stages include the work of Liu et al. 2011, which examined an approach for semi-supervised classification to detect failures in artificial lift systems. Artificial lift systems are techniques to enhance oil production by increasing the pressure within the reservoir, which directly lifts fluids to the surface. The authors present a framework that applies feature engineering with clustering and semi-supervised learning techniques to enable learning of failures/normal patterns from noisy and poorly labeled multivariate time series. The authors explored three

different classifiers: Decision trees (DT), SVM, and Bayes Net. Among those three, SVM achieved the highest overall accuracy of 98.5%, while Bayes Net achieved 96.4% and DT achieved 97% accuracy. In addition, Liu, Li, and Xu 2019 presents an integrated model for the detection and location of leakages in pipelines. The authors investigate two modules, one that can detect larger leakages and another for micro-leakages. Patri et al. 2015 present an approach to predict valve failures in gas compressors from oil fields, with the use of sensor data from multiple sensors. The authors' approach consisted of the use of feature extraction and selection, combined with DT. Xie et al. 2019 presents data-driven models such as principal component analysis (PCA) and partial least squares regression (PLSR) combined with statistical models to identify influencing factors and predicting failure rates of equipment based on data from six Norwegian oil and gas facilities.

1.3 Thesis statement and contributions

The objective of this work is to research and develop a condition-based monitoring (CBM) system that can identify and detect undesirable abnormal events before and when they occur. A full methodology to develop and implement the aforementioned will be addressed. This includes preprocessing the data in an early stage, calculating statistical features and reducing dimensionality with PCA and training a machine learning algorithm. This thesis comprises the following sub-tasks:

1. Analyse current solutions and applications of CBM systems related to this work.
2. Provide a literature review on machine learning concepts, from the early stages of extracting statistical features, transforming data to principal components, and training the algorithm.
3. Implement and train a CBM system with the acquired 3W dataset to identify and detect real and simulated undesirable abnormal events in oil wells.
4. Test the CBM system and assess the performance and capability of the solution.
5. Provide a final evaluation and assessment for future applications.

1.3.1 Main contributions

The main goal of this thesis was to implement and train a machine learning-based CBM system, which was able to classify normal and faulty events in the 3W dataset. In order to illustrate and experiment with the CBM system, some of the end results is summarized below.

Implementing the CBM methodology

In this work, a proper CBM system has been implemented based on the common methodology used in the industry. This consists of: cleaning raw-sensor data, where factors such as nan values and unlabeled observations are treated; Extracting relevant and rich features from the cleaned sensor data; reducing the dimensionality of the data, such that it can be processed through a machine learning algorithm without causing

memory leakage; and, training and applying an ML algorithm to detect and classify faults.

Exploring individual window sizes for feature extraction

An approach to apply individual window sizes for each fault type was introduced in this work. This was done by applying the grid search for hyperparameter optimization, where the window size was analyzed for all faults. The grid search showed promising results and designated distinct behaviors for each fault type. Therefore, these best results for each classifier were applied for further use when different classification scenarios were investigated.

Investigation of different classification scenarios

The implemented CBM system was introduced with a new classification scenario, which is referred to as "fault versus not fault". This classification scenario aims to implement and train one binary classifier for each fault against all other faults and normal events, in the 3W dataset. Another classification scenario that has been implemented is known as, "fault versus normal". This method also applies binary classifiers but aims at identifying and detecting faulty events against normal events. Furthermore, these classification methods were tested and assessed in a CBM manner, where factors such as reliability and efficiency are important factors.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces the basic concepts of machine learning to provide the technical knowledge required to understand this work. Chapter 3 presents background knowledge about sub-sea oil wells and their sensors. Moreover, the chapter discusses the 3W dataset and the eight different types of faults characterized as undesirable abnormal events in oil wells. Moreover, the challenges related to the 3W dataset is reviewed. Chapter 4 elaborates on the strategy used in the implementation of the CBM system. This relates to how the data is cleaned, which statistical features are used and how they are extracted, and how and why PCA is used. Furthermore, important factors such as algorithm choice, the training routine, and how to evaluate the fit models are reviewed. Chapter 5 provides the test results related to model performance, and an in-depth analysis is conducted. Chapter 6 is the last chapter, which presents conclusions and summarizes the achieved results of the proposed system. Furthermore, future works are discussed with ideas on how to learn more about the problem and perhaps improve the proposed system in this work.

Chapter 2

Machine Learning

This chapter covers the general aspects of machine learning needed to understand the results presented in this work. This has been done by providing an overview of machine learning, where topics such as supervised learning, unsupervised learning, regression, and classification are discussed. Subsequently, the importance of features is reviewed and how to extract those features with statistical tools. Furthermore, why and how to apply dimensionality reduction are reviewed. Next, the focus relies on presenting the methods applied in machine learning for classification, such as decision trees and random forests, where the construction of the classifiers and how they work are explained with a few real-world examples.

2.1 Overview

The goal of machine learning (ML) is to learn and adapt from experience by detecting meaningful patterns in huge amounts of data and then use these learned patterns to analyze future data. The learning of an ML algorithm is usually divided into two main types, supervised and unsupervised learning. The supervised learning methodology is done by mapping inputs x to outputs y, with a target function $f : x \rightarrow y$, given a labeled set of input-output pairs $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$, where $\mathcal{D}$ denotes the training set and N is the number of training samples. On the contrary, unsupervised learning does not apply labeled data. Only the inputs are given, $\mathcal{D} = (x_i)_{i=1}^{N}$, where the goal is to find patterns that might be of interest, which is why it is sometimes referred to as knowledge discovery [23].

The input x of an ML algorithm applying supervised learning can be everything from an image to a sentence or time-series, which is converted to a d-dimensional vector of numbers. For instance, a full high-definition image will be converted to a $1920 \times 1080 \times 3$ dimensional vector ($Width \times Height \times RGB$), where the red-green-blue (RGB) channels of the image represent the features. Furthermore, the output y_i is either categorical or real-valued, depending on the problem. If the problem is to classify animals, such as either a cat or a dog, then the problem is known as classification, and the output form is a finite set of *categorical* values, $y = [C_1, C_2, .., C_n]$, while problems that predict real-values are known as *regression*. Applications such as predicting the steering angle of a self-driving car or predicting stock-market prices are few examples of what a regression problem may be.

2.1.1 Neural networks

Feed-forward neural networks known as multilayer perceptron are a popular architecture for pattern recognition in machine learning. They consist of several neurons, also known as units arranged in a series of hidden layers and have their origin from attempting to represent information processing in biological systems [4].

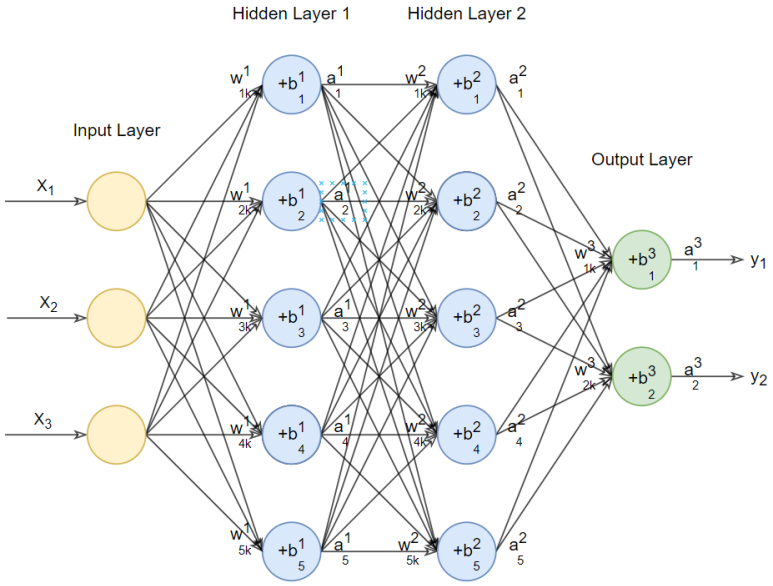


FIGURE 2.1: Illustration of a neural network.

Figure 2.1 illustrates a feed-forward neural network with an input layer, two hidden layers, and an output layer. In this architecture, each hidden layer's neuron is fully connected to all previous and next layers' neurons. The input information (i.e., X_1, X_2, X_3) is fed to the input layer and propagates forward through the i^{th} neuron in the j^{th} hidden layer until it has reached the end of the output layer (i.e., y_1, y_2). In a classifier problem, for instance, the output layer represents the class scores in the form of an output vector. The relation between the next layer's input to the previous layer is given by

$$a_i^j = \sigma((w_{ik}^j * a_k^{j-1}) + b_i^j), \tag{2.1}$$

where σ denotes an activation function, w_{ik}^j represents the weight from the k^{th} neuron in the $i-1^{th}$ layer to the i^{th} neuron in the j^{th} layer. b_i^j is the bias in the j^{th} layer from the i^{th} neuron. Moreover, the activation output a_i^j is the output value of the so-called activation function, commonly defined in Equation (2.2), where z_i^j is used to represent $\sum_k (w_{ik}^j a_k^{j-1} + b_i^j)$, where

$$a_i^j = \sigma(z_i^j) = \frac{1}{1 + e^{-h_i^j}}. \tag{2.2}$$

The purpose of the activation functions in neural networks is to restrict the outputs from achieving huge values. Therefore, non-linear functions are applied to limit the values such that they are mapped in a range between -1 and 1.

2.2 Feature extraction

When dealing with huge datasets, there are often attributes which do not provide any valuable information to solve a complex task. Raw measured data often has noise, and the relationship among the objects are complex. In such circumstances, the data is prepossessed to solve this challenge, which is often referred to as feature extraction. This is done by applying statistical and signal processing tools to find the strongest dependencies among these objects with the intent of extracting meaningful information to the ML algorithm.

2.2.1 Statistical features

Extracting features can be very powerful, and it is used to simplify a complex task. This was shown in Goldberg et al. [9]. For instance, a single statistical feature was applied to monitored signals to detect site disruptions on eBay. The approach excelled compared to the previous rule-based system and was able to detect anomalies that the previous system was incapable of. In the work of Raghavenda et al. [26], a dual moving median was used as a statistical feature in fault prediction for artificial lift systems in oil fields. In that approach, the global median was used to represent long-term performance in terms of months, a mid-term median for recent performance such as over a past week and one short-term median for current performance representing the most recent number of days. A similar approach was used in the work of Tang et al. [30], which investigated a new method to detect flow influx during real-time drilling. One of the statistical features they introduced was the divergence of moving average (DMA). This statistical feature was applied to quantify the increase and decrease in local fluctuations. The maximum, minimum, median, and mean values are a few simple statistical tools that can be applied to build statistical features of raw measured data. Table 2.1 shows a summary of commonly used statistical features in ML, with their respective equations.

TABLE 2.1: Summary of Statistical Features Applied in ML.

No.	Name	Equation	
1	Maximum	$x_j \geq x_i$, for all $x \in X$	(2.3)
2	Minimum	$x_j \leq x_i$, for all $x \in X$	(2.4)
3	Median	$\begin{cases} \frac{X_{\frac{N}{2}} + X_{\frac{N+1}{2}}}{2}, & \text{when N is even.} \\ X_{\frac{N+1}{2}}, & \text{when N is odd.} \end{cases}$	(2.5)
4	Mean	$\frac{1}{n}\sum_{i=1}^{n} x_i$	(2.6)
5	Variance	$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$	(2.7)
6	Standard deviation	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$	(2.8)
7	Skewness	$\frac{1}{N}\sum_{i=1}^{n}\frac{(x_i - \overline{x})^3}{(\sqrt{(x_i - \overline{x})^2})^{\frac{3}{2}}}$	(2.9)
8	Kurtosis	$\frac{1}{N}\sum_{i=1}^{n}\frac{(x_i - \overline{x})^4}{(\sqrt{(x_i - \overline{x})^2})^4}$	(2.10)

2.3 Dimensionality reduction

Reducing the dimensionality to a lower-dimensional subspace is common in ML when dealing with high dimensional data. In the case of images, often it is redundant to use all three colour-channels. Hence, the image can be converted from RGB full colour to grayscale. This means that the three colour-channels are converted to a single channel, which only represents brightness information without any apparent colour for each individual pixel in the image. An important note is that re-scaling the size of an image, which is common in ML is not the same as dimensionality reduction. Even though it may improve the algorithm by removing inessential information. Reducing the dimensionality of the data has several benefits, such as, the computational time of an ML algorithm execution, due to aspects as memory allocation and calculations. In addition, if done correctly, it often results in enhanced model performance, in the manner of predictive accuracy. A common statistical tool to do this is Principal Component

Analysis (PCA), which filters out redundant features and focuses on the essential features [23].

2.3.1 Principal component analysis

The goal of the PCA is to reduce the dimensionality of a dataset consisting of a large number of connected variables, at the same time keeping the variation present in the dataset. Principal Components (PCs) are variables that are transformed and achieving this objective. The PCs are uncorrelated and ordered in a specific hierarchy, such that the first few PCs retain the highest variation present in all of the original dataset [13]. Simplified, the PCA is done in four steps:

1. The first step is normalization, based on observed mean and standard deviation (i.e., mean of zero and standard deviation of 1), also known as standardization.

$$z_i = \frac{x_i - \overline{x}}{S}, \tag{2.11}$$

   where $\overline{x}$ and S denote the mean and standard deviation, respectively.

2. Second step is to calculate the covariance matrix:

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x}) \times (y_i - \overline{y}), \tag{2.12}$$

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{pmatrix}. \tag{2.13}$$

3. Third step is to find the eigenvectors and eigenvalues of the covariance matrix:

$$Cv = \lambda v, \tag{2.14}$$

   where v is the eigenvector of the covariance matrix C with eigenvalue λ.

4. The fourth step is to hierarchically sort the eigenvectors by eigenvalues, where the eigenvectors define the direction of the new axis and the eigenvalues with the highest value define the eigenvector with the highest variance in terms of energy. Thus, the eigenvectors with the smallest eigenvalues will be discarded for the new subspace, which has lower dimensionality.

2.4 Decision tree

In machine learning, decision trees have been widely used in classification and regression problems in different disciplines. The goal of a decision tree is to reach a final conclusion of a given problem statement (e.g., classify a person's gender) by eliminating assumptions with true or false questions. The structure of the model are similar to a tree, where the flow of the input data starts from the root node and propagates through the internal nodes down to the terminal node, also known as the leaves, based on true or false assumptions in each node.

To get a better understanding of how decision trees work, [Louppe, 2014] has defined a decision tree with the following five concepts [19]:

1. A tree is a graph $G = (V, E)$ in which any two vertices (or nodes) are connected by exactly one path.
2. A rooted tree is a tree in which one of the nodes has been designated as the root. In our case, we additionally assume that a rooted tree is a directed graph, where all edges are directed away from the root.
3. If there exists an edge from t_1 to t_2 (i.e., if $(t_1, t_2) \in E$) then node t_1 is said to be the parent of node t_2 while node t_2 is said to be a child of node t_1.
4. In a rooted tree, a node is said to be internal if it has one or more children and terminal if it has no children. Terminal nodes are also known as leaves.
5. A binary tree is a rooted tree where all internal nodes have exactly two children.

With the given concepts above, a decision tree can be defined as a model with a target function $h : X \to Y$, represented with a rooted tree $X_{t_0} = X$ and $X_{t_2} \subseteq X_{t_1} \subseteq X$ for all $(t_1, t_2) \in E$. Furthermore, the internal nodes t of the rooted tree are labeled with a split s_t taken from a set of binary questions (i.e., true or false), $\mathcal{Q}$. The terminal node are labeled with the best guess value $\hat{y} \in Y$. When an input x propagates through a decision tree model, it ends up in a terminal node, also called a leaf, where the label of this leaf is the final output prediction value $h(x)$.

Learning a decision tree according to some dataset $\mathcal{D}$ is basically about deciding the tree structure. This is done by following Occam's Razor principle. The principle concludes that with a given occurrence of any problem, the solution to the problem is most likely the solution with the fewest assumptions. If this is applied to the decision tree. The best-constructed decision tree will most likely be the *shallowest tree* h^*, which minimizes the error E_{in}. The *impurity* measure $i(t)$ evaluates the goodness of a given node t, which is determined by the purity of the node and predictions. The purer the node and the better predictions $\hat{y}_t(x)$ have, the smaller $i(t)$ [19]. The impurity decrease of a binary split $s \in \mathcal{Q}$ is given by

$$\Delta i(s, t) = i(t) - \frac{N_{t_L}}{N_t} i(t_L) - \frac{N_{t_R}}{N_t} i(t_R), \tag{2.15}$$

where N_{t_L} and N_{t_R} denotes the left node and right node, respectively, and N_t is the size of the subset $\mathcal{D}_t$. The two most commonly used impurity functions for classification trees are the Shannon Entropy and the Gini index. The Shannon Entropy quantifies the uncertainty of Y within node t, and is given by

$$i_H(t) = -\sum_{k=1}^{J} p(c_k|t) log_2[p(c_k|t)], c \in Y. \tag{2.16}$$

The Gini index defined as

$$i_G(t) = \sum_{k=1}^{J} p(c_k|t)[1 - p(c_k|t)], c \in Y, \tag{2.17}$$

measures how often a randomly chosen object x would be misclassified if it was randomly labeled by a class $c \in Y$ according to the distribution $p(y|t)$.

To know if a split of a node is good two main factors that have to be evaluated. Those two factors are the entropy and the information gain, where entropy is a measurement of randomness. So, a good binary split $s \in \mathcal{Q}$ of a dataset $\mathcal{D}$ produces two sets of data which provide fewer assumptions required to identify a class, thus, reducing the randomness (entropy) and increasing the information gain. Thus, the best split $s \in \mathcal{Q}$ is the split that achieves the maximum information gain and lowest entropy. Constructing and training decision trees are usually done with randomized methods which will be reviewed in next section.

2.5 Random forests

The principle of randomized ensemble methods for decision trees are known as random forests. Decision trees are prone to the generalization error which is the phenomenon of poor performance on unseen data. Therefore, a randomized ensemble methods is used to implement random perturbations into the learning procedure. By utilizing this method, the prediction variance is reduced while the respective bias is not increased greatly. This is done such that several individual models can be produced from a single learning set $\mathcal{L}$ and then combining the prediction of those models to form the prediction of the complete ensemble [19]. Consider a set of $\mathcal{M} = [m_1, m_2, .., M]$ randomized models which have learned on the same data $\mathcal{L}$, but each model built on a set of random seed of features θ_m. Ensemble methods operate by combining the predictions of these models into a new *ensemble* model, such that the generalization error of the ensemble is reduced compared to any of the individual randomized models. The predictions in classification are usually aggregated by considering the models in the ensemble and then resort to the class with the majority of the votes to form a final prediction [19], and is given by

$$\hat{y} = \arg\max_{c \in Y} \frac{1}{M} \sum_{m=1}^{M} p_m(c|x). \tag{2.18}$$

Random forest methods are typically divided into two main categories, based on how the random perturbations are implemented into the learning procedure. The two methods are not mutually exclusive and can be combined. The two methods are as follows:

- **Bagging:** This method consists of randomizing the data by bootstrap aggregation, such that each tree grows from a random selection of examples in the learning set $\mathcal{L}$. It is done by creating a form of replicate datasets $\mathcal{L}_{\mathcal{B}}$, where each replicated dataset is drawn at random and consists of N_B samples but with replacements of $\mathcal{L}$. Each sample N_B may or may not appear repeated times in any particular $\mathcal{L}_{\mathcal{B}}$ [6]. In order to build the forest, we train a model $\mathcal{M}_{\mathcal{B}}$ using $\mathcal{L}_{\mathcal{B}}$ and repeat the process to produce an ensemble of $\mathcal{M}$ models.

- **Randomized node optimization:** This method consists of using a random selection of features to split each node, where each node is split from a random selection among the K best splits. The K best splits are the subset of features of

interest. Each node can be split by one or more features. Usually a split of more than 1 feature is applied. To dive into greater detail, consider a complete set of parameters $\mathcal{T}$ where each parameter θ_i can be used to make a decision or split at a given node j. When training the j^{th} node, only a small subset $\mathcal{T}_j \subset \mathcal{T}$ of such parameters is available [7]. Therefore, undertraining a tree is achieved by optimizing each split node j, and is defined as

$$\theta_j^* = \underset{\theta_j \in \mathcal{T}_j}{\arg\max.}\ I_j, \tag{2.19}$$

where I represents the information gain based on the entropy. The degree of randomness is controlled by the ratio of two parameters, p and d, where if $p = 1$, the maximum randomness and uncorrelated trees are achieved. On the other hand, all the trees are identical and there is no randomness if $p = d$.

Random forests have been used in a wide set of disciplines, such as in the oil field. Anderson [3] introduced the Petroleum and Analytics Learning Machine (PALM), which is a machine-learning-based analysis system that uses random forests as one of its classifiers to to maximize the performance of oil and gas wells and pipeline systems. This is accomplished by predicting the production volumes of oil, natural gas, and water as each well ages.

2.6 Summary

This chapter have discussed several techniques applied in the field of machine learning, from the early stages of pre-processing the data to feeding an ML algorithm with data for classification. Some relevant features to this work have been presented and discussed along with the impact they can have to different applications. In addition, we have presented why dimensionality reduction is beneficial and how to apply it with PCA. Furthermore, the theory of constructing, optimizing and training machine learning classifiers that will be applied in this work has been covered. In the next chapter, the 3W dataset is introduced, which most of this work is based upon. The different events characterized as faults will be reviewed, and relevant information about offshore oil wells. Furthermore, a discussion is provided to understand the challenges with the dataset.

Chapter 3

Data Analysis

This chapter aims to provide an understanding of the 3W dataset that is used in this work. This is done by presenting relevant information about offshore oil wells and sensors that are used to detect undesirable events during oil well production, followed by a discussion about the dataset and the events that may occur in oil wells.

3.1 Offshore oil wells

An offshore oil well is a unit installed on the bottom of the seabed to produce oil from large oil and gas reservoirs. The term "oil well" is used for a larger system consisting of several subsystems, such as, a production tubing, which is the main path for the well fluid; a wellhead to ensure structural safety during drilling and production; and a "Christmas tree" installed on the top of the wellhead, which gives access to the production tubing and controls the production with several valves and sensors that can be accessed from the surface. The communication link between the surface and the oil well on the seabed is referred to as an "umbilical", which is an electro-hydraulic unit used for transmission of electrical signals and hydraulic power. Furthermore, it is connected between the Christmas tree and the surface control system (i.e., a nearby production platform) [14]. Figure 3.1 illustrates an example of a typical offshore oil well set-up, described above.

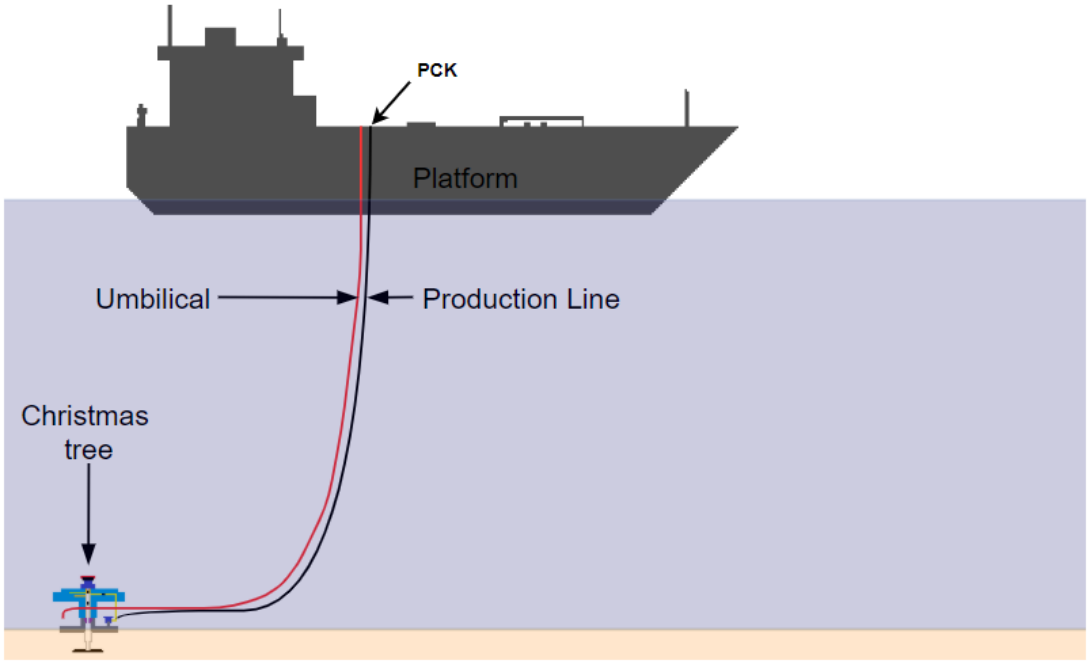


FIGURE 3.1: Simplified schematic of a typical offshore naturally flowing well based on [32].

3.2 3W dataset

The 3W dataset is a public dataset released by Petrobras [32], considered as the world's third-biggest oil company. The dataset consists of real, simulated, and hand-drawn data of oil wells during operation. Moreover, the data shows instances of the oil well during normal operation and more importantly when undesired events in the oil well occur. This is shown through sensor readings extracted from five monitored variables:

1. Pressure at the Permanent Downhole Gauge (PDG);
2. Pressure at the Temperature and Pressure Transducer (TPT);
3. Temperature at the TPT;
4. Pressure upstream of the Production Choke Valve (PCK);
5. Temperature downstream of the PCK;

Temperature and Pressure Transducer (TPT) and *Permanent Downhole Gauge* (PDG) are devices that consist of pressure and temperature sensors. PDG remains fixed in a certain position of the production tubing while the TPT is located at the Christmas tree. The last device is the control valve that is responsible for well control at the surface, this control valve is located at the beginning of the production unit and is called the *Production Choke Valve* (PCK). Furthermore, The *Downhole Safety Valve* (DHSV) is another device that is monitored, to the extent of the closure mechanism. The PCK and DHSV and their impact are explained further on. Figures 3.1 and 3.2 show the location of the mentioned devices.

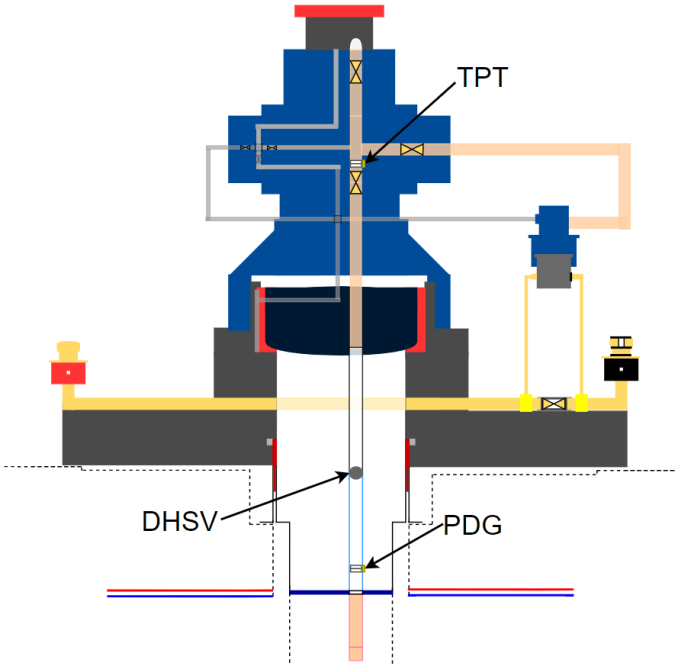


FIGURE 3.2: Simplified schematic of a typical subsea Christmas tree based on [32].

The 3W dataset considers eight types of undesirable events in oil wells. Water, sediment, natural gas, and their ratio and flow rate are important factors to these undesirable events. As mentioned earlier, there are real, simulated, and hand-drawn undesirable events in the dataset, where all real instances have been extracted from the plant information system of Petrobras. The simulated instances have been computer-simulated with OLGA. OLGA is a highly used dynamic flow simulator used in the oil industry. Furthermore, the hand-drawn instances have been made by experts within the field. Simulated and hand-drawn instances were created in manner to reduce the imbalance of the dataset. Due to the reasoning that 58% of the real instances (597 of all 1025 real instances) consist of normal operation, where no undesirable event occurs. Moreover, the distribution among the real instances of undesirable events is uneven, where 80% of all instances (344 instances of all 428 real instances of undesirable events) belong to only one type of undesirable event. Every undesirable event in the dataset is a continuous sequence of observations with three states: *normal*, *faulty transient* and *faulty steady state*. In the normal state, there is no evidence of abnormal behavior. In the faulty transient state, the dynamics caused by undesirable events are ongoing. When these dynamics cease, the faulty steady-state cease. These states were created in order to allow early detection of a given failure event. The units used in this dataset include Pascal [Pa], standard cubic meters per second [sm^3/s], and degrees in Celsius [$^{\circ}C$].

3.2.1 Fault description

The subsection provides the general description of the eight fault types contemplated in the 3W dataset. Vargas et al. [32] has defined the eight fault types as the following:

1. Abrupt increase of basic sediment & water

Basic Sediment and Water (BSW) is defined as the ratio between the water and sediment flow rate and the liquid flow rate, both measured under normal temperature and pressure (NTP). During the life cycle of a well, its BSW is expected to increase due to increased water production from either the natural reservoir aquifer or artificial injection to avoid declining production. However, a sudden increase of BSW can lead to several problems related to flow assurance, lower oil production, oil lifting, incrustation, industrial plant processing, and the recovery factor. Automatic identification of this type of undesirable event may permit actions such as administering production or artificial injection to avoid this sort of problem.

2. Spurious closure of DHSV

The Down-hole Safety Valve (DHSV) is placed in the production tubing, where its purpose is to ensure the closing of the oil well. It provides safety by shutting off the well in situations in which the production unit and well are physically disconnected or in the event of an emergency or catastrophic failure of surface equipment. However, the closing mechanism will eventually fail in a spurious manner (e.g., the pressure drop in the hydraulic actuator). This kind of failure is problematic because there are often no indications of the failure on the surface, which causes production losses and additional cost. Actions can be taken if the spurious closure of this valve is detected in a timely manner, such that the production losses can be reduced.

3. Severe slugging

This type of undesirable event occurs frequently at irregular intervals, on mature oil fields. Severe slugging takes place when "slugs" of liquid separate bubbles of gas through the pipeline. This causes pressure and flow rate oscillations everywhere in the pipe and can cause a substantial decrease in oil production [22]. In the 3W dataset, it is considered a critical type of instability and can result in stress or even damage to equipment in the well and/or the industrial plant. Actions can be taken to prevent damage or production loss if detected in advance.

4. Flow instability

During flow instability, there is a periodical change of pressure but with acceptable amplitudes. Flow instability is not necessarily equal to slugging, what separates those two anomalies is the lack of periodicity. Though flow instability can result in slugging. As instability can progress to severe slugging, its prognosis avoids all the negative aspects associated with this more severe anomaly.

5. Rapid productivity loss

There are several factors that can change the productivity of a naturally flowing well, the factors consist of the diameter of the production line, percentage between water and basic sediment, static pressure of the reservoir, and the viscosity of the produced fluid. When any of these factors are changed to the extent that the system's energy is not sufficient enough to overcome the losses, the flow of the well will slow down or

even stop, which causes productivity loss. To prohibit this, the operating point of the well can be changed if the fault is predicted in advance.

6. Quick restriction in PCK

Production choke (PCK) is a control valve located at the beginning of the production unit. It is responsible for well control and can restrict, control ,and regulate the flow. The choke can be controlled from the surface and when operated manually problems may occur, such as unwanted restrictions. This is referred to as "quick restriction in PCK" and occurs when there is an amplitude above a specified reference (e.g., 5%) and in a short time (e.g., less than 10s). Identifying this event automatically is desirable because unwanted restrictions can be reversed.

7. Scaling in PCK

Inorganic deposits will occur during production. Therefore, it is important to monitor the production choke since it significantly reduces oil and gas production. If detected, Losses of oil and gas production can be avoided. Thus, detecting it in an early stage is favorable, so actions can be taken.

8. Hydrate in production line

This undesirable event occurs when water and natural gas form a crystalline compound, which happens under extreme pressure and temperature conditions. This crystalline compound resembles ice and when it is formed in production lines it can stop production for days and weeks. This is one of the biggest problems in the oil industry. Thus, avoiding this is desirable.

3.3 Data review and challenges

There are several factors that cause the 3W dataset to be challenging. As noted earlier, the dataset is very imbalanced even though measures have been taken to reduce the imbalanced ratio among the fault types. Another concern is that the dataset itself is limited, in the manner that there is only a total of 1984 instances of the different events, which are few instances in general. This can be observed in Table 3.1, which shows the amount and distribution of real, simulated, and hand-drawn instances of each event in the dataset.

TABLE 3.1: Quantitative relation of the instances in the 3W dataset.

Type of Event	Real Instances	Simulated Instances	Hand-Drawn Instances	Total Instances
0. Normal	597	0	0	597
1. Abrupt Increase of BSW	5	114	10	129
2. Spurious Closure of DHSV	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 Restriction in PCK	6	215	0	221
7. Scaling in PCK	4	0	10	14
8. Hydrate in Production Line	3	81	0	84
Total	1025	939	20	1984

Despite its great technical value, the 3W dataset includes a great deal of missing and frozen variables and unlabeled observations. In this case, a 'variable' refers to the monitored operational settings and sensor readings. Furthermore, an 'instance' refers to a recorded event of one of the eight fault types in the 3W dataset, while an 'observation' is a sample from an instance, showing the true label, timestamp, operational settings, and sensor readings. These definitions are used in the following subsections, which review challenges related to the 3W dataset.

3.3.1 Unlabeled observations

An observation is considered unlabeled when there is no label of the fault type for a given sample of an instance. 5,130 (0.01% of all 50,913,215 observations of all 15,872 variables of all 1,984 instances) observations are considered unlabeled in the 3W dataset. Figures 3.3 and 3.4 show an instance that has unlabeled observations of the fault type 'Abrupt increase of BS&W'. The figures show the behavioral pattern of the sensor readings of the pressure at the TPT, temperature at the TPT, and pressure at PDG, respectively. Furthermore, different periods are highlighted with individual colors. The green period illustrates that there are is abnormal behavior (i.e., normal operation), the orange period illustrates the faulty transient state, and the red period illustrates the faulty steady-state. The black period indicates unlabeled observations. In this case, there were seven unlabeled observations. These observations create complication because it is not clear whether the observation is in the faulty steady state or in the faulty transient state.

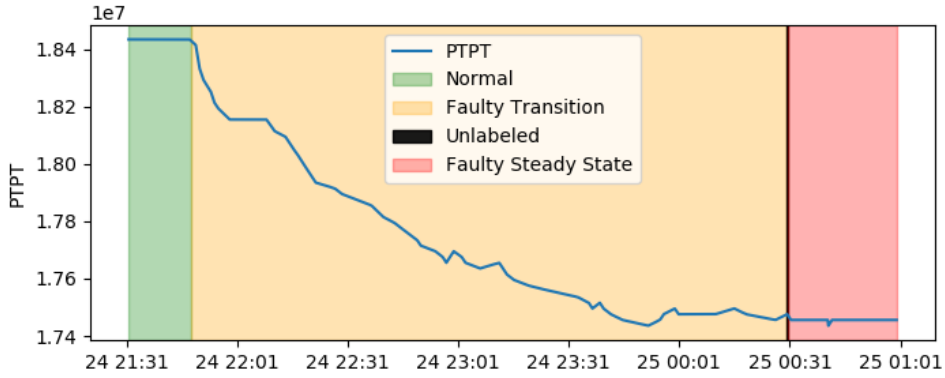


FIGURE 3.3: Real instance of 'Abrupt increase of BS&W' showing pressure at TPT.

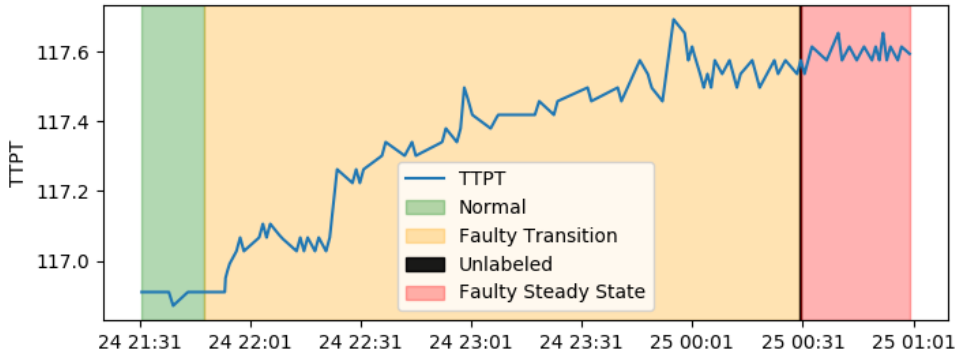


FIGURE 3.4: Real instance of 'Abrupt increase of BS&W' showing temperature at TPT.

3.3.2 Missing and frozen variables

A variable is considered missing when all observations of that particular variable in an instance have a missing value. 4,947 (31.17% of all 15,872 variables of all 1,984 instances) variables are considered missing in the 3W dataset. In the case of frozen variables, they are considered frozen when all observations of that particular variable in an instance have the same constant value. 1,535 (9.67% of all 15,872 variables of all 1,984 instances). variables in the 3W dataset are considered frozen. Figure 3.5 is an example of a variable that is frozen. All the observations of the 'pressure at PDG' have the exact same value. Missing and frozen variables occur due to sensor, system configuration, or network communication issues.

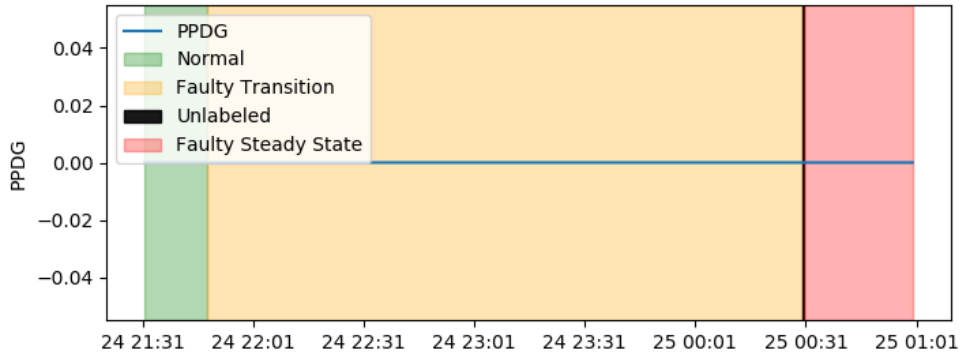


FIGURE 3.5: Real instance of 'Abrupt increase of BS&W' showing pressure at PDG.

3.3.3 Hand-drawn instances

A challenge of this dataset is related to hand-drawn instances, this is because the behavior varies a lot when it is compared to real instances. The hand-drawn instances are too artificial and are quite distinct from the real ones. This is evident when comparing the hand-drawn instance in figure 3.6 against the real instance in figure 3.7, where both instances are of the same fault type. Due to the mentioned difficulties, fault type seven (Scaling in PCK) are completely omitted from all subsequent analyses, since 10 of all 14 instances are hand-drawn.

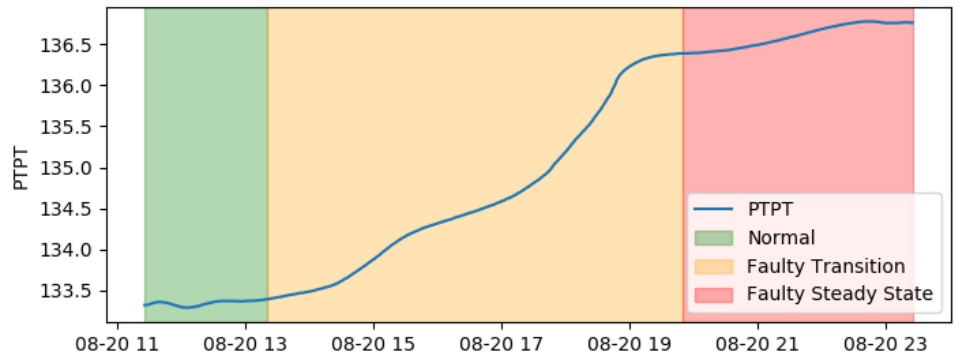


FIGURE 3.6: Hand-drawn instance of 'Scaling in PCK' with sensor variable P-TPT.

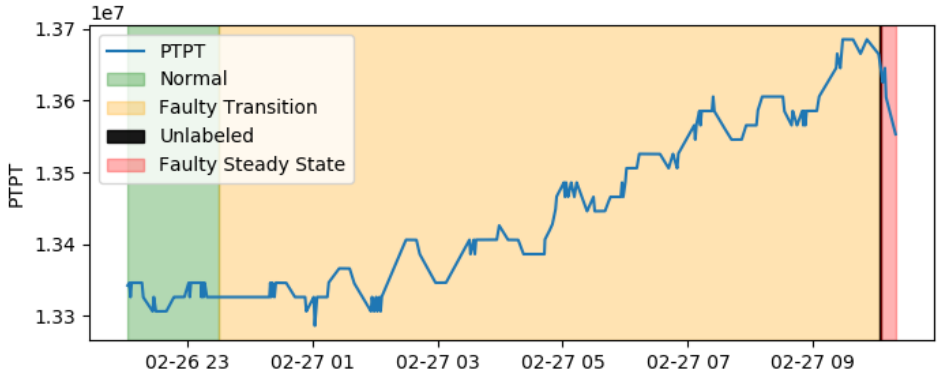


FIGURE 3.7: Real instance of 'Scaling in PCK' with sensor variable P-TPT.

3.4 Summary

In this chapter, the 3W dataset has been detailed. Offshore oil wells and their sensors have been reviewed before discussing the undesirable events that occur in oil wells. Furthermore, the chapter has evaluated the 3W dataset and provides information regarding the challenges associated with the observations and variables in the dataset. The next chapter provides the technical background required to understand the machine learning concepts applied in this work.

Chapter 4

Methodology

This chapter discusses the methodology and covers the proposed system framework as a condition monitoring system. The chapter aims to address both the theoretical and practical aspects to gain a broader understanding of the different matters, provided in this thesis. Firstly, a brief introduction to the framework is presented to get an overview of the complete system. The following sections discuss how the data is prepared in three different stages before it is fed to the chosen classification algorithm. Lastly, it also discusses the training routine of the classification algorithm and how the performance of the fitted model is measured.

4.1 System framework

This subsection provides a brief introduction to the proposed framework used in this work. The goal of the system framework is to serve as a condition monitoring system. In other words, the system must be able to perceive and distinguish undesirable events (anomalies) from normal conditions, based on raw sensor signals.

The framework consists of four important stages, shown in figure 4.1, where the functions of each stage are as follows. Firstly, the raw input data is preprocessed, such that it is applicable. Secondly, this preprocessed data is transformed into statistical features with the objective to present the initial data with deeper and different insights. Thirdly, these statistical features are transformed into principal components to enhance the performance of the anomaly detection algorithm. The fourth and final stage receives the transformed data and model it with a given classification algorithm.

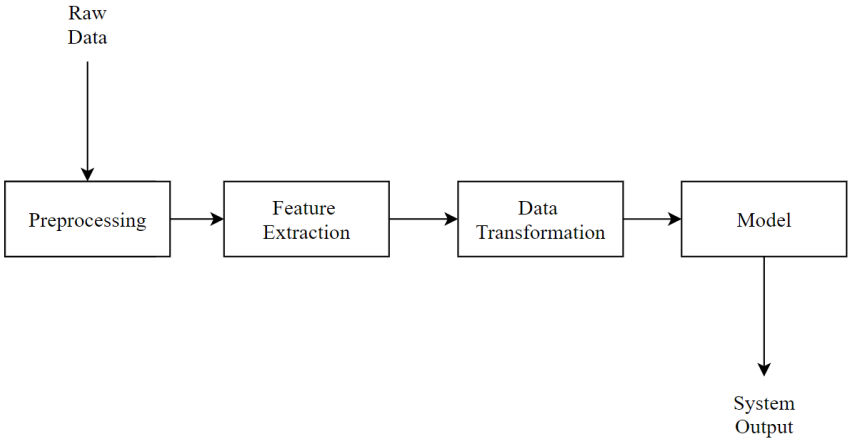


FIGURE 4.1: Block diagram of the framework.

In Section 4.2, the first block of the framework is presented, explaining how the raw data is preprocessed and the necessity of this step. Section 4.3 explains in detail the features that have been considered in this work and how they're extracted, which is the second block of the framework. Section 4.4 explains the third block of the framework, giving the details about the transformation strategy that has been applied. Section 4.5 reviews the last block in the framework, which discusses what the role of a classifier is and which classifier that is used in this work. Section 4.6 explains the factors that are important to consider when training a machine learning algorithm and the training routine used in this work.

4.2 Preprocessing

The first block of the system represents the preprocessing stage. This stage consists of two important steps: firstly, the data is split into a training and test set, such that we can train an algorithm and test it on unseen data; and secondly, the raw data is cleaned, such that it is applicable in the next stage, where the statistical features are extracted. Table 4.1 shows the number of instances for all the events for each dataset, where the data was spread randomly with a 70/30 distribution ratio. All hand-drawn instances were removed since they are completely different from real instances, which can be seen in Section 3.3. Consequently, event type 7 (scaling in PCK) was omitted as it has very few (only four) real/simulated instances.

TABLE 4.1: Quantitative relation between the training and test set.

Type of event	Train Instances	Test Instances	Total Instances
0. Normal	418	179	597
1. Abrupt Increase of BSW	84	35	119
2. Spurious Closure of DHSV	27	11	38
3. Severe Slugging	74	32	106
4. Flow Instability	241	103	344
5. Rapid Productivity Loss	316	135	451
6. Quick Restriction in PCK	155	66	221
8. Hydrate in Production Line	59	25	84
Total	1374	586	1960

The continuation of the process is to prepare and clean the data. All observations (samples) from all instances that consist of a numeric data type that can be interpreted as not a number (Nan values) are replaced with zeroes. Also, all observations that cannot be classified due to missing labels are kept when extracting features but removed when training the classification algorithm. Table 4.2 shows the number of Nan values

and unlabeled observations that were removed from each event. Keep in mind that the remaining 633 unlabeled observations of all 5,130 unlabeled observations belonged to event type 7, which was removed altogether from our experiments, as mentioned above.

TABLE 4.2: Total amount of Nan values and unlabeled observations for each event in the dataset.

Type of event	NaN Values	Unlabeled Observations
0. Normal	17,537,620	0
1. Abrupt Increase of BSW	26,729,233	1,019
2. Spurious Closure of DHSV	2,071,568	1,026
3. Severe Slugging	13,368,136	0
4. Flow Instability	4,289,933	0
5. Rapid Productivity Loss	38,948,805	1,461
6. Quick Restriction in PCK	23,286,058	622
8. Hydrate in Production Line	6,742,972	369
Total	132,974,325	4,497

4.3 Feature extraction

The second block in the framework extracts statistical features. Prior to this stage, the data has been divided into a training and test set, and furthermore cleaned from Nan values, so that it is possible to compute arithmetic functions on the data. Thus, the input data to this block is cleaned time-series data, which has been sampled second by second from eight different sensors. Many statistical features are often used in machine learning applications and in this work nine popular features have been extracted from the raw time-series sensor data, which can be seen below. In addition to the features below, the arithmetic mean was computed. The mathematical definitions for the features below are provided in Chapter 2 in Table 2.1.

Standard deviation

Standard deviation is a measure of how far the data fluctuates from the mean, where the variance represents the power of these fluctuations. In some cases, when the mean and standard deviation is given, the comparison among them can yield the relationship between the measured signal and the noise, known as signal-to-noise ratio (SNR). In these circumstances, the standard deviation represents the noise, while the mean describes what is being measured [29].

Skewness and kurtosis

Skewness is used to measure the lack of symmetrical shape of data distributions. A perfect symmetrical dataset has a skewness of zero. In this case, the median, mode, and mean all have equal values. On the other hand, the distribution is skewed if the symmetrical shape of the distribution has a long tail in one direction. Skewness with positive values means that the curve is skewed to the right (right-tail), while skewness with negative values suggests skewing to the left (left-tail). Kurtosis measures the peakedness or flatness of a data distribution, where positive values are interpreted as a more peaked distribution, and negative values suggest a flatter data distribution, compared to the normal distribution [5].

5-number summary

The five-number summary represents a set of features, which is the minimum, lower quartile, middle quartile (median), upper quartile, and maximum values of a data distribution. When these features are combined and provided, it gives a proper indication of how spread the data is. This is evident when considering which attributes are achieved when they are combined. In short, the minimum and maximum represent the range of the data, and the median represents the center of the distribution. Moreover, the first quartile represents the center between the minimum and median, while the upper quartile represents the center between the maximum and median [24].

The nine statistical features explained above have been extracted through a sliding-window, where each individual statistical feature has been computed over the same sliding window size. The extraction is completed by transforming a set of $N \times n$ samples (with N samples from n variables) into a vector of nf elements. This is illustrated in figure 4.2, which can be interpreted as a function that maps a real matrix $a \times b$ (initial data X) into a real matrix $c \times bf$ (transformed data X_{tr}), i.e., $f : \mathbb{R}^{a \times b} \longmapsto \mathbb{R}^{c \times bf}$. Considering a sliding window size of length N and a step size s between two consecutive windows yields a function output $c = \frac{a-N}{s} + 1$ instances. Thus, $X_{tr} = f(X)$ yields a real $a[\frac{M-N}{s}] + 1 \times nf$ matrix, considering M instants represented in the input matrix.

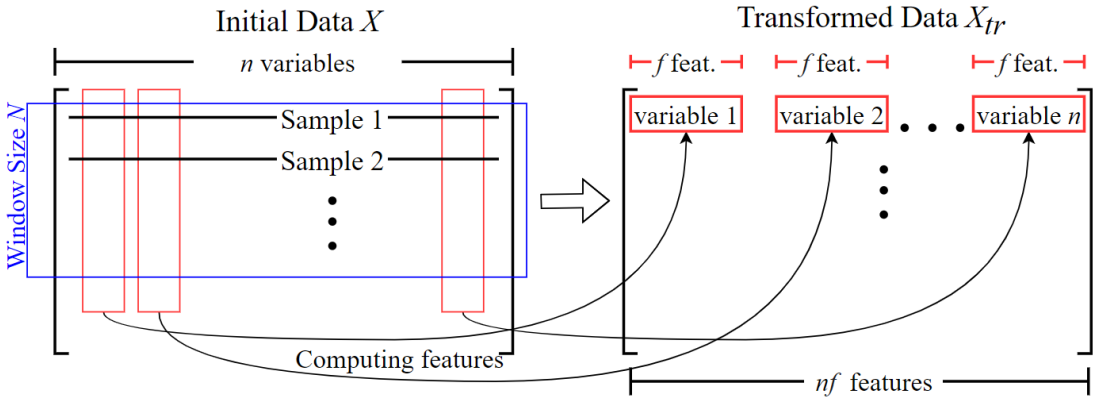


FIGURE 4.2: The raw data is represented as the initial data X (matrix on the left side) which has n variables. The matrix on the right side represents the transformed data X_{tr}, obtained by computing features from the initial data X. This is done over a sliding window of N rows along with all its columns with a step size of s rows between each transformation. For each variable that is transformed inside a window, a feature k is produced. Thus, the transformed data ends up with nk columns. The figure and approach of extracting features is based on Marins [20].

The process of extracting features adds two hyper-parameters to the system, the window size N, and the step size s. The window size N controls the length of the window that is transformed, and it may be changed without great consideration, as opposed to the step size s. Extracting features drastically reduces the amount of data if there is a rise in the step size. Therefore, to avoid reducing the amount of data of an already small dataset, the step size is not experimented on and is kept constant of one row for each feature. Despite the latter, the data must be reduced in some manner: So, that it can fit directly into the memory when training a classification algorithm. The data can be reduced because each instance of each event has several virtually unaltered observations/samples representing the same characteristics, which yields several redundant samples. Thus, subsampling samples from the training set for each instance for each event is possible without removing a great extent of vital information. In this work, each instance is sub-sampled by extracting every 10th sample for events that are associated as an anomaly (event/class 1 to 8) and every 50th sample for each instance belonging to the event associated as normal operation (event/class 0).

4.4 Data transformation

In this section, the data transformation block is explained. Data transformation is done to represent the data in a different manner and to reduce the dimensionality. One can argue that when extracting statistical features, the amount of information is increased. Thus, a more efficient way to represent the data may be beneficial. Consider a dataset with three feature variables, and for each of these original feature variables, some statistical analysis might be beneficial to classify anomalies. If three common statistical analyses, such as the minimum, median and maximum are extracted and used to represent the center and range of the distribution, then the number of input features has increased threefold if it is applied to every original feature variable. As witnessed, when many statistical analyses are used to extract features to gain additional information, the dimensionality increases rapidly. Processing additional features take additional

time and the new features alone are not ensured to be profitable for a classification algorithm. Therefore, a method of reducing the dimensionality of the feature space without the expense of losing sensitive information will allow us to exploit the additional information gained through statistical analysis while reducing the time to train an algorithm. To achieve this Principal Component Analysis (PCA) which was earlier introduced in the Machine learning chapter can be applied.

4.4.1 Principal component analysis

The principal component analysis has been widely used in various machine learning tasks, such as face recognition to image compression. The technique is used to find components that capture the maximal variance in the data. This is because high energy features in terms of variance in the Euclidean space are often important and consist of sensitive information. On the other hand, low energy features often hold lesser important information. Therefore, it is reasonable to assume that if the high energy features are kept while the low energy features are removed, it should be possible to reduce the dimensionality of the data without removing any sensitive information (i.e., the high energy features).

There is an important factor to analyze when removing the low energy features and that is the threshold. The threshold of the PCA is an important hyperparameter, which decides the amount of principal components that are kept for further use. If the transformed data have n eigenvectors and eigenvalues, and only the first ρ eigenvectors (the components with the most energy) are chosen, then the final dataset has been reduced to only ρ dimensions [28]. Figure 4.3 shows the PCA applied to the training set after extracting features. The green, yellow, orange and red lines respectively highlight the number of principal components required to represent 75%, 90%, 95% and 99% of the cumulated variance in the training data.

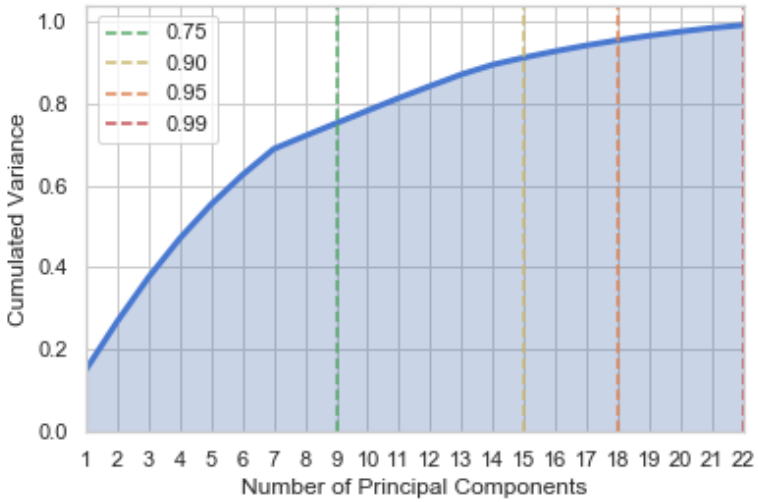


FIGURE 4.3: Principal component analysis performed on the training set of the 3W dataset. In this case, the data had a total of 72 features, which were extracted from the computation of nine statistical features.

4.5 Classification modeling

This section describes the last block in the system framework. Prior to this stage, the raw input data has been cleaned, statistical features have been extracted and these features have been transformed into principal components. The next step is to provide a mapping of the input data to a specific category by analyzing the transformed features. A classification algorithm is required to achieve the latter. That is because a classification algorithm provides a probabilistic output, which yields the probability of the input data belonging to a certain class label.

Before choosing a classification algorithm, one must choose between using a machine learning or a deep learning approach, where both approaches have their pros and cons. The most known trait of the two different approaches is also what separates them, i.e., deep learning algorithms require massive amounts of data to train, while machine learning algorithms can train on smaller datasets. Machine learning algorithms tend to require lesser amounts of data to train due to the limited choices of hyperparameters to be tuned, compared to deep learning algorithms. Deep learning algorithms often consist of large networks that have a vast choice of hyperparameters to be tuned, e.g., number of layers, number of neurons in each layer, and type of activation function. Based on these factors, and the amount of time and the small quantity of data at disposal, machine learning will be applied.

4.5.1 Random forests

There are a few factors that must be considered when making the decision of which machine learning algorithm to be used. The most important factor to be considered is the complexity of the data at hand. The data at hand is highly imbalanced between the distribution ratio among the classes, besides, there is a lot of missing variables in the data, where almost 5000 variables have a missing value. Random forests are a machine learning classifier that can be applied to the mentioned challenges. This is due to its well-known traits:

- It can handle missing data.
- It can handle small datasets.
- It is simple to train and deploy.
- It is easy to parallelize.
- It is robust to noise and outliers.

Each tree in the random forest classifier is independent and trained separately. The trees are trained by a random subset of the data and a small random set of drawn features. By applying this approach, each tree can learn from a different partition of the data and each tree will be uncorrelated with other trees. This is illustrated in figure 4.4, where the input features represent the previously extracted statistical features of the dataset. Furthermore, each split in a tree is based on their respective set of randomly drawn features. When each tree in the forest has given a class label to the input data based on their respective independent splits, a majority voting is done. The majority

voting provides a probabilistic distribution for the system output. In a binary classification problem, this probabilistic distribution can be interpreted as if 70 trees out of 100 trees predict a certain class, then it is said to have 70 % certainty of an observation belonging to that given predicted class. Thus, the final classification of the random forest classifier will be the outcome of the probabilistic distribution given by the majority voting.

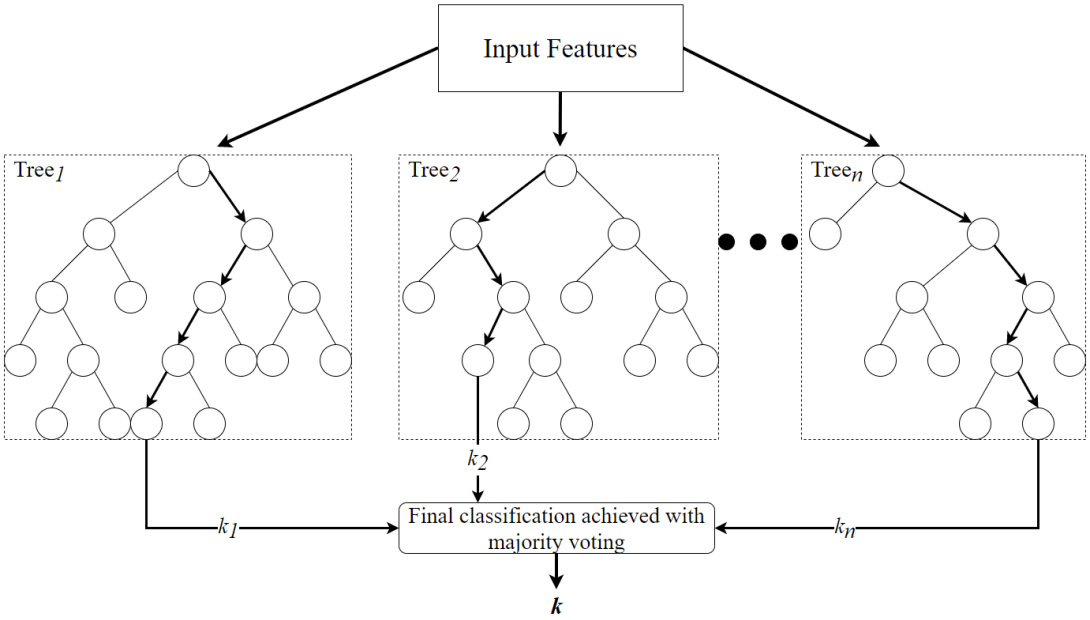


FIGURE 4.4: A simplified schematic of the random forest architecture for a classification problem. The features are fed to the trees, where each tree yields a classification and the majority class will be the final classification k.

There are two hyper-parameters that has to be tuned to achieve the best results with the random forest classifier. Those two hyper-parameters are:

- Max tree depth: This hyper-parameter decides how deep each tree can go in its splits.
- Number of estimators: This hyper-parameter decides how many trees to be used in the random forest.

In this work, Python 3 and the machine learning library scikit-learn has been used to implement and train the random forest classifier. How the classifier has been trained and the importance of the training routine will be discussed in the next section.

4.6 Classifier training

This section covers the training of the model and the strategy that will be used to achieve the best results. How the training routine is performed is an important part in order to achieve a good performing machine learning model. There are certain things that must be taken into mind, such as generalizing well to new data that the model has never seen before. That is the main objective of a machine learning model. The main parametric to measure a model performance is something known as the generalization

error, which is the expected value of the misclassification rate when averaged over future data. In other words, how accurate a machine learning model can predict outcome values for previously unseen data. So, to achieve a model that generalizes well on unseen data, there are two events that we try to avoid when training the model, which is overfitting and underfitting. Overfitting happens when we model every minor variation in the input data. As a result, the model does not just learn the true signal but also the noise which relies in the data. This leads to an inaccurate model, which categorizes based on too many details. Underfitting is the opposite event of overfitting, it happens when we fail to model the major variation in the input data, in other words, when the model does not fit the data well enough [23].

When the training of one or several models are done, we need to know if the model(s) overfit or underfit. In other words, the model must be tested to reveal if it generalizes well because the training accuracy does not always reflect the performance of the model. A designated test set is not available in this thesis; Therefore, the model is tested, and an error rate is estimated by holding out a subset of the training set from the training process and then testing the model on this particular subset. This certain approach of holding a subset and testing is known as the validation set approach which will be discussed in the next subsection.

4.6.1 Validation set and cross-validation

The validation set approach consists of randomly dividing the available data into two parts, a training set, and a validation set or hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the data in the validation set. This results in a validation set error-rate, which yields an estimate of what the test error-rate is. According to James et al., 2014, the validation set approach has two potential drawbacks:

- The test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- The validation set error rate tends to overestimate the test error rate. This is because only a subset of the observations is included in the training set (approximately half the size of the full dataset), which is used to fit the model. As a result, the model learns from fewer observations, which suggests that it will perform worse.

In order to address the drawbacks of the validation set approach, a refined method has been created. The method is known as Leave-one-out Cross-Validation (LOOCV) and they are closely related. Instead of creating two subsets of comparable size, a single observation (x_1, y_1) is used as a validation set and the remaining observations $(x_2, y_2), ..., (x_n, y_n)$ will be used as the training set. This procedure is repeated by selecting (x_2, y_2) as the validation data and training the model on the remaining $n - 1$ observations, i.e., $(x_1, y_1), (x_3, y_3), ..., (x_n, y_n)$. This certain procedure is repeated until all possible combinations are performed and then the error is averaged for all trials, where the number of possible combinations is equal to the number of data points (observations) in the original dataset [11]. The complexity of the implementation and computation is based on the number of data points; Therefore, it would be very time

consuming to use the LOOCV based on the available dataset in this work. Thus, an approach that is based on the validation set approach and the LOOCV will be applied, which will be introduced in the next subsection.

4.6.2 K-fold cross-validation

A widely used cross-validation technique is the k-fold strategy. In this strategy, the training set is split into a k number of subsets, also called a fold. The strategy consists of iterations, and for each iteration, two steps are performed:

1. A model is trained on all folds with the exception of the i-th fold;
2. The trained model is tested on the i-th fold;

Figure 4.5 brings an illustration of this process with a 3-fold cross-validation strategy. During the first iteration, the first fold is used as a testing set for the model, and the remaining two folds are used as training sets. The second iteration uses the first and the last folds as a training set, and the second fold as a testing set. The last iteration uses the last fold as a testing set and the remaining two folds as training sets.

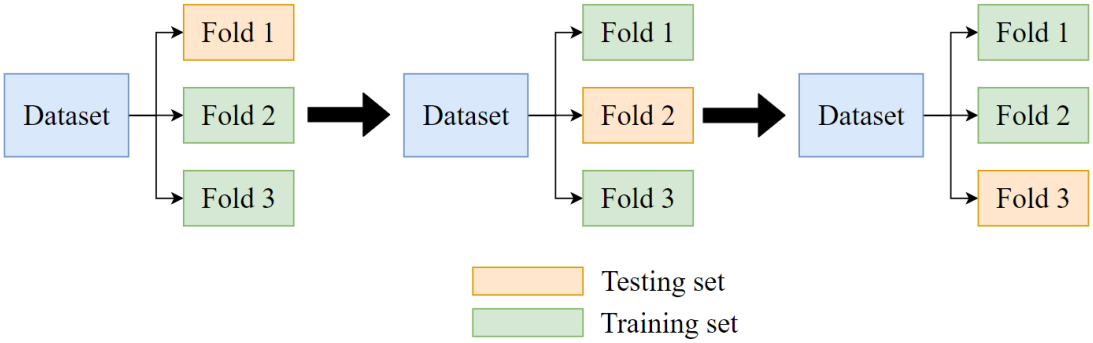


FIGURE 4.5: Illustration of the k-fold strategy. The dataset is split into k subsets, where each iteration uses the orange fold as a test set to evaluate the model and the remaining green folds are used to train the model. This process is repeated until each orange fold of all k folds have been used as a testing set to evaluate the model.

In this work, since the data is time-series there is a risk of contaminating the data. Therefore, every sample of a given event from the same instance belongs to the same fold. This is done by assigning a group to every sample and then splitting each fold based on these groups. This strategy is known as group k-fold cross-validation and figure 4.6 illustrates how the groups are assigned. Moreover, the samples between $t_0 - t_1$, $t_1 - t_2$ and $t_2 - t_3$ are assigned to group 1, group 2 and group 3, respectively. In this case, event 1 denotes any undesirable events of anomalies, while event 0 denotes samples from normal operation (event/class 0) and initial normal (i.e., samples that show normal dynamic behavior before an anomaly event has begun).

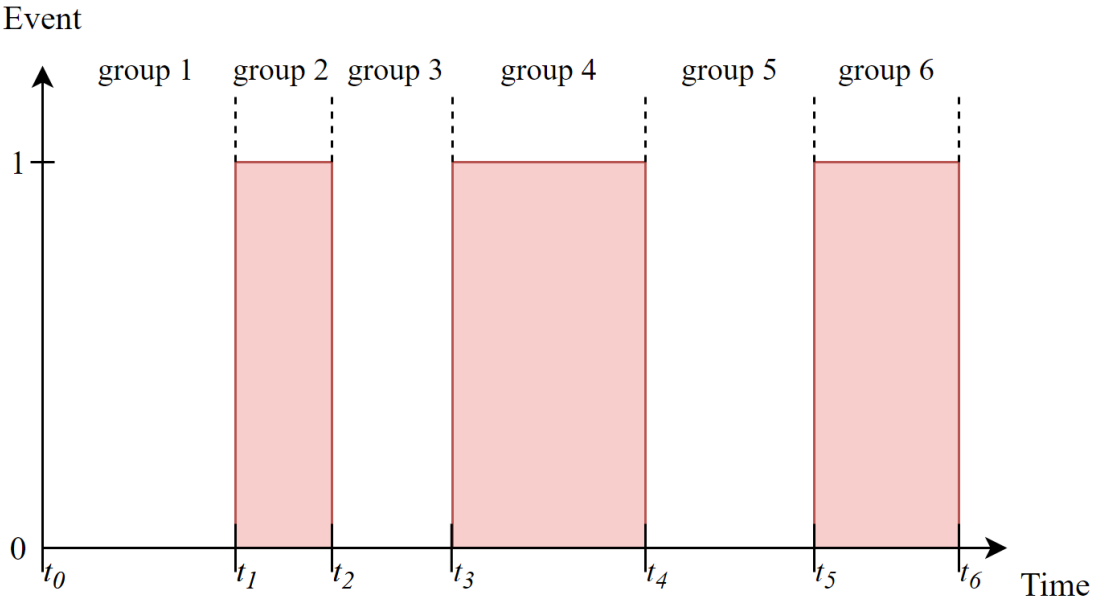


FIGURE 4.6: Illustration of how the samples are categorized into groups based on the condition of the event. These groups are used when applying the Group k-fold strategy during training.

4.7 Summary

In this chapter, the system has been discussed in detail, following the data from the first block of the system all the way to its outputs. This chapter has aimed at giving the reader a thorough understanding of the complexity of the system and its many choices that must be decided. This has been evident from the very first block of the system, where an important factor in preprocessing is to choose a strategy on how to deal with Nan values. Following the second block of the system, which discussed the strategy on how to extract features and the decisions that must be taken regarding new hyperparameters that arise, such as window length and step size. The decisions continue through the data transformation block to the classification modeling block, where factors such as the PCA threshold, number of estimators, and tree depth are under focus. In the remaining sections of the chapter, a strategy is proposed to train the model. There are many opportunities available, and the system is complex. Thus, a list of candidates to conduct experiments are reviewed at the beginning of the next chapter. Furthermore, the chapter reviews the chosen experiments and analyze their respective results to assess system performance.

Chapter 5

Implementation and Results

This chapter covers the implementation and results. The vast number of candidates for experiments is reviewed to show the opportunities that are related to this system and how complex it is. The chosen experiments and contributions of this work are discussed, where factors such as system configuration and implementation are provided. The final sections analyze and elaborate on the experimental results, and factors such as reliability and efficiency of the system are discussed.

5.1 Candidates for experiments

There are a vast number of experiments that can be performed with this system. The system is very complex and an empirical approach to the problem is required to evaluate what this proposed system is capable of. In this work, the following candidates are considered relevant as proper experiments:

1. Evidently, the 3W dataset consists of many classes, and there are several possibilities and methods that can be used to detect and identify these classes. The system does not necessarily need to be a multiclass classification system even though there are several classes, it can also be looked upon as a binary classification problem. Thus, some of the experiments that can be done with the two different techniques are:

   (a) Binary classification:

      i. Using a single binary classifier to perceive and discriminate between normal and faulty operations. This is done by combining every class considered a fault into a single unique class.

      ii. Using multiple binary classifiers, where each classifier is uncorrelated to the other classifiers and is designed and trained to discriminate between a specified fault against the normal operation mode. In this approach, one can analyze which faults are the hardest to classify.

   (b) Multiclass classification:

      i. Using a single multiclass classifier to identify and discriminate between faulty and normal operations. In this strategy, there will be eight classes to identify, where seven of the eight classes consist of the types of events that are considered as faults, and the last class is considered as a normal event. The only event which will not be classified is "Scaling in PCK".

2. Principal component analysis:

   (a) Exploring the impact of the PCA. This can be done by experimenting what difference PCA offer compared to not using any data transformation techniques at all.
   (b) Experiment on the influence of the PCA threshold offer. This experiment consist of using PCA as feature selection with different thresholds τ, such as $\tau \in 0.90, 0.95, 0.99$. Then analyze which parameter value which yields the best model performance.

3. Features:
   (a) Experimenting on different statistical features, e.g., extracting several types such as the mean, the minimum and maximum.
   (b) Exploring feature importance by analyzing the influence of each individual feature and how they impact the system.
   (c) Evaluate the impact of the feature extraction parameters. There are two hyperparameters related to feature extraction, which is the window size N and the step size s. For instance, an experiment could be done by testing different window sizes, such as $N \in 60, 300, 900$.

4. Random Forest:
   (a) Experimenting on the number of trees (estimators) in the forest (e.g., $E \in 20, 100, 200$).
   (b) Experimenting on the maximum depth of the forest (e.g., $d \in 5, 10, 20, \infty$).

5. Hyperparameter optimization: This consists of finding the hyperparameters of the machine learning algorithm that yields the best performance, measured on the validation set. Two popular hyperparameter optimization techniques that can be explored are *grid search* and *Bayesian hyperparameter optimization*.

It is evident that there are many opportunities to explore with the proposed system, and not all of the candidates can be carried out to be experimented on. This is because this master's thesis has a time constraint, and it would not be possible to explore every possible opportunity. The next section presents the candidates that were carried out as experiments and their respective results.

5.2 Experiments

This section presents a comprehensive analysis of system performance. The analysis has been conducted by approaching this as a binary classification problem under two classification scenarios. In this work, these classification scenarios are characterized as *Fault versus Normal Operation* and *Fault versus Not Fault*.

Fault versus normal operation

In this scenario, each binary classifier is designed to discriminate and identify between normal operation and a single specific class, categorized as a fault event. Thus, seven individual uncorrelated classifiers are required (one for each event representing an

anomaly). Furthermore, all samples from normal operation (event/class 0) are combined to a single unique class, with the samples that show normal dynamic behavior preceding the given fault class (initial normal).

Fault versus not fault

In this scenario, each binary classifier discriminates and identifies between a single specific fault against everything that does not belong to that specific fault. This is done by designing seven uncorrelated classifiers, one for each fault, and combining the remaining events into a single unique class.

The experiments above have been conducted to achieve a greater understanding of the system and to acquire more information about each fault class. Besides, this continues and contributes to the work of Marins et al. [21], which have shown that a CBM system can be used to classify the faults, with binary and multiclass classifiers. This work introduces a new classification scenario (fault vs not fault) and a new classification strategy by placing more emphasis on dynamic behavior. This is done by proposing an individual window size for each fault type, based on the assumption that each fault type shows a distinct dynamic behavior in the *initial normal*, *faulty transient*, and *faulty steady* states. Individual window sizes will be the initial experiment in this work and grid search will be applied to find the best hyperparameters.

The samples that belong to the faulty transient and faulty steady states for each fault are combined to one unique class in the two classification scenarios, described above. In this work, the model performance is assessed in detail: firstly, by measuring the overall model performance; secondly, by measuring the capability to discriminate between normal operation and each transitional state; and lastly, by individually measuring the simulated and real instances of the latter two metrics.

5.3 Experiment 1: parameter search

As mentioned earlier, this work proposes individual window sizes for each fault. The proposal is an attempt to provide a strategy that simplifies the identification by enhancing distinct patterns of the dynamic behavior for each fault. The traditional grid search is a method to assess the best hyperparameters, which has been applied to the window size. In short, the method computes a set of chosen hyperparameters for a specified algorithm. The performance of the set of hyperparameters is indicated through the measurement of the cross-validation set.

A grid search computed for the Random Forest classifier and the PCA transformation shows that $E = 200$, $d = 15$, and $\tau = 0.99$ are the best hyperparameters. These hyperparameters denote the number of estimators, maximum depth, and PCA threshold, respectively. In this case, these hyperparameters are the best based on mean accuracy and computational time. The conducted set of hyperparameters in this search were $E \in 100, 200, 300$; $d \in 10, 15, 20$; and $\tau \in 0.95, 0.99$. Additionally, other conducted parameters consist of group k fold cross-validation with $CV = 3$ splits/folds and a window size of $N = 600$. The experiment of the window size was carried out with the best hyperparameters from the previous result. This is shown in the list below, where all parameters are listed for the grid search of the window size.

- Random Forest parameters:
  - Number of trees (estimators): $E = 200$.
  - Maximum depth: $d = 15$.
- Feature extraction parameters:
  - Window size: $N \in 100, 200, 300, ..., 1200$.
  - Step size: $s = 1$.
- Other parameters:
  - PCA threshold: $\tau = 0.99$.
  - Subsample factor: 10 for all faults.

The results to discover the best window size for each fault indicate that individual window sizes improve the performance. Figure 5 illustrates this, which shows the results for the grid search of the window size. The red crosses show the local maximum, which represents the highest mean validation accuracy for each class. The mean validation accuracy is the computed mean of the accuracy over all splits of the cross-validation set. In this search, only real samples of the transitional states from each fault were employed. Only classes 3 and 4 used samples from class 0 (normal operation), due to the absence of samples that represent the 'initial normal' phase. The reason to limit the window size to 1200 is because of the delay it yields, where a window size of 1200 equals a delay time of 20 minutes.

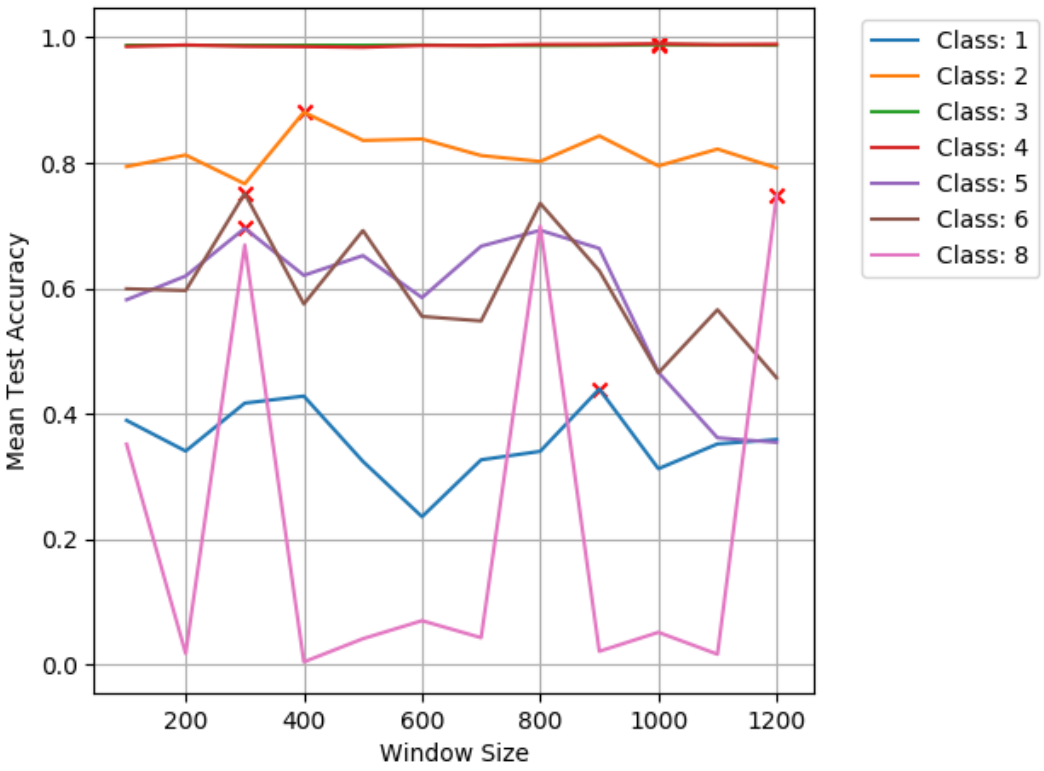


FIGURE 5.1: Grid search results of the window size, showing mean test accuracy for each fault and their respective window sizes.

The individual window size for each classifier besides classes 3 and 4 was chosen based on the best options from the grid search results. Classes 3 and 4 achieved similar results for each window size, and therefore a decision was made to use a smaller window than 1000 to reduce the delay. Therefore, classes 3 and 4 will be experimented on further with a window size of 300, while classes 1, 2, 5, 6, and 8 respectively apply window sizes of 900, 400, 300, 300, and 1200. These window sizes are within the limit of not exceeding the average time-interval of the transient state for their respective class faults, which can be seen in Table 5.1. Moreover, grid search results for each classifier can be found in Appendix A. Keep in mind that not a single instance from the test set have been applied during the grid search nor during the training of the forests.

TABLE 5.1: Average time of the transient state for each fault, based on all real instances in the 3W dataset. Empty entries indicate the absence of data.

Fault	Average Transient Time	Real Instances
Class 1	259.38 min	5
Class 2	66.96 min	22
Class 3	-	32(10)
Class 4	-	344(103)
Class 5	441.88 min	12
Class 6	17.36 min	6
Class 8	440.3 min	3

5.4 Experiment 2: fault versus normal operation

This section reviews the experiment characterized as *fault versus normal operation*. In this experiment, each classifier is fit on data from normal operation (class 0) and their respective fault. The only change in any hyperparameter relates to the subsampling factor. In this case, fault events with less than 20 real instances apply a subsampling factor of 1 for real instances. This is with the intent to balance the distribution ratio, between simulated and real instances. Moreover, the list of window sizes of each classifier consists of 900, 400, 1200 for classes 1, 2, and 8, respectively; and, 300 for the remaining classes. These specific window sizes represent the best performance for their given classifier, which is evident in figure 5.1.

The test results of the 3W dataset for the transitional states and overall accuracy can be seen in Table 5.2. In this table, 'Transitional ACC' denotes the overall accuracy for the transitional states, i.e., initial normal (normal operation preceding an anomaly event), faulty transient state, and faulty steady-state. Additionally, 'Overall ACC' represents the accuracy for all transitional states combined with samples from class 0. The reason for showing the overall and transitional accuracy separately is because approximately 30% of all instances from the complete 3W dataset belongs to Class 0. Thus, the overall accuracy is not sufficient alone and may misrepresent the performance of the classifier. More details can be seen in Table 5.3, which shows real and simulated test results for class 0 and each transitional state.

TABLE 5.2: Overall and transitional test results of the classification scenario fault versus normal operation, including real and simulated instances for each classifier.

Fault	Window Size	Type	Transitional ACC	Overall ACC
Class 1	900	Real	0.214	0.989
		Simulated	0.999	0.999
Class 2	400	Real	0.986	0.999
		Simulated	0.996	0.996
Class 3	300	Real	0.998	0.999
		Simulated	0.998	0.998
Class 4	300	Real	0.967	0.986
		Simulated	-	-
Class 5	300	Real	0.888	0.992
		Simulated	0.993	0.993
Class 6	300	Real	0.972	0.999
		Simulated	0.951	0.951
Class 8	1200	Real	0.892	0.999
		Simulated	0.956	0.956

TABLE 5.3: Test results of 'Normal Operation' (class 0) and each transitional state of the classification scenario fault versus normal operation, including real and simulated instances for each classifier.

Fault	Window Size	Type	Normal (Class 0)	Initial Normal	Transient State	Steady State
Class 1	900	Real	1.000	0.779	0.098	0.011
		Simulated	-	0.999	0.999	0.999
Class 2	400	Real	0.999	0.952	0.996	1.000
		Simulated	-	1.000	0.970	1.000
Class 3	300	Real	0.999	-	-	0.998
		Simulated	-	-	-	0.998
Class 4	300	Real	0.990	-	-	0.967
		Simulated	-	-	-	-
Class 5	300	Real	0.999	0.514	0.904	-
		Simulated	-	0.134	0.999	1.000
Class 6	300	Real	0.999	0.981	0.882	1.000
		Simulated	-	0.876	0.955	0.956
Class 8	1200	Real	0.999	0.000	1.000	1.000
		Simulated	-	0.258	0.996	0.999

Analyzing these two tables, one can notice that almost all classifiers outperform the models from the grid search. This is a result of a limited dataset in regards to the amount of data at disposal. The grid search was fit on only real samples (of which are very few) and with cross-validation, which prevents the classifier to be fit on all samples at a time. Only classes 3 and 4 show a similar performance between these two

results, which is a consequence of being among the classes that have larger datasets. Thus, the classifier can afford to split the training data into a larger amount of subsets. To conclude further on this, the imbalance between the distribution ratio of the 3W dataset reflects on the performance of each classifier. This is evident in the case of class 1 and 8, where both classifiers seem to struggle on real data if one omits the performance of class 0 (normal operation). These two classes have less than 10 real instances combined, in contrast to classes 2, 3, and 4, which have 22, 34, and 344 instances, respectively. One can also observe that class 5 has difficulties correctly classifying samples of the initial state. This applies to the case of both real and simulated instances. On the other hand, the classifier achieves satisfactory accuracy in correctly classifying samples belonging to normal operation and the transient state. In this case, the results show an accuracy of over 90% for the latter two events.

5.4.1 Event-based evaluation

This section reviews the system in a real-world scenario as a CBM system. To be a reliable CBM system, the system must have the capability to detect any event as soon as possible. Table 5.4 shows how many events each classifier was able to detect correctly. An event is considered to be detected correctly if the classifier can correctly predict all its samples with an accuracy of 90%. In this event-based evaluation, only real events will be analyzed and reviewed. Class 4 wrongly predicted seven steady-state events of all 103 events. However, there was only one event in which the classifier was not able to predict any correct samples. For the remaining events, the classifier was able to correctly predict all samples for four events with approximately 70% accuracy, or more and the other two events with 41.6% and 59.5% accuracy.

Outwardly, the accuracy numbers of the wrongly classified events from Class 4 may seem impressive. But, the classifier in these events shows an inconsistent behavior. This is evident in Figure 5.2, which shows the rapid fluctuations in the classification between event values '0' and '1'. In Class 5, one event yields inconsistent behavior with fluctuating system classifications. This can be seen in Figure 5.3, which shows the system classification for all transitional states of a real event. Except for this given example of Class 5, inconsistent system classifications appear to be constrained to the events that have been wrongly classified in Class 4. This may be due to the high accuracy Class 4 achieves on each event that is categorized as wrongly classified. Except for this given example of Class 5, inconsistent system classifications appear to be constrained to the events that have been wrongly classified in Class 4. This may be due to the high accuracy Class 4 achieves on each event that is categorized as wrongly classified. This is clear in the examples shown in Figures 5.4, 5.5, and 5.6. These are events that have been wrongly classified by their respective classifiers but still show a consistent classification pattern.

TABLE 5.4: Event-based analysis of classification scenario fault versus not normal, showing the events that were correctly classified with an accuracy of 90%. Designated numbers in parenthesis show total number of events.

Fault	Type	Initial Normal	Transient	Steady-State
Class 1	Real	1(2)	0(2)	0(2)
	Simulated	34(34)	34(34)	34(34)
Class 2	Real	5(7)	7(7)	3(3)
	Simulated	5(5)	5(5)	5(5)
Class 3	Real	-	-	10(10)
	Simulated	-	-	22(22)
Class 4	Real	-	-	96(103)
	Simulated	-	-	-
Class 5	Real	1(3)	3(4)	-
	Simulated	18(132)	132(132)	132(132)
Class 6	Real	1(2)	1(2)	2(2)
	Simulated	57(65)	61(65)	61(65)
Class 8	Real	0(1)	1(1)	1(1)
	Simulated	23(24)	24(24)	19(19)

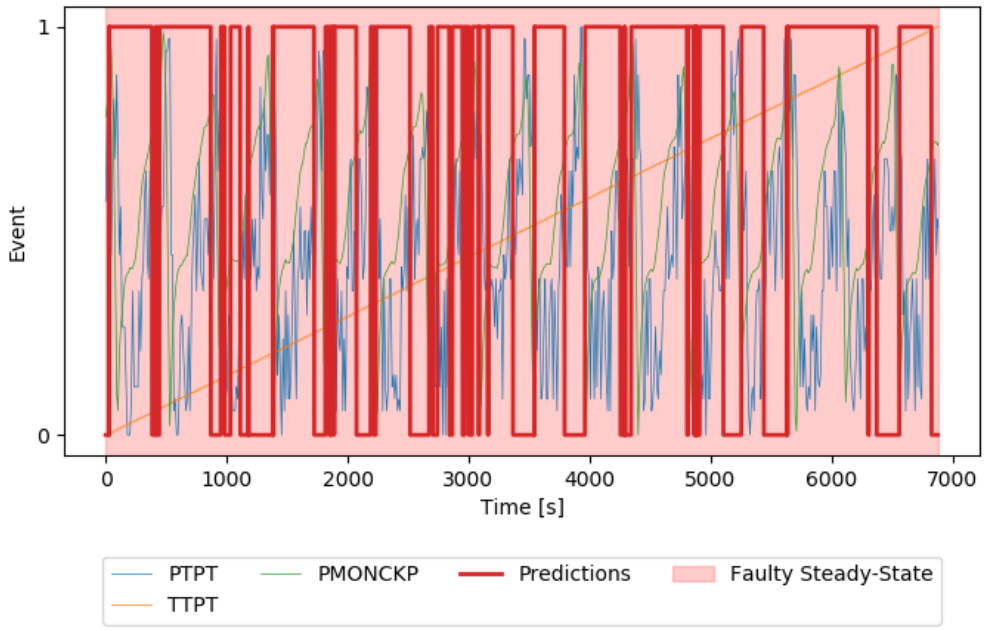


FIGURE 5.2: Example of a real instance of Class 4 showing the inconsistency of the system classification, where only 70% of the samples were correctly classified. Event values '1' and '0' denote normal and faulty states.

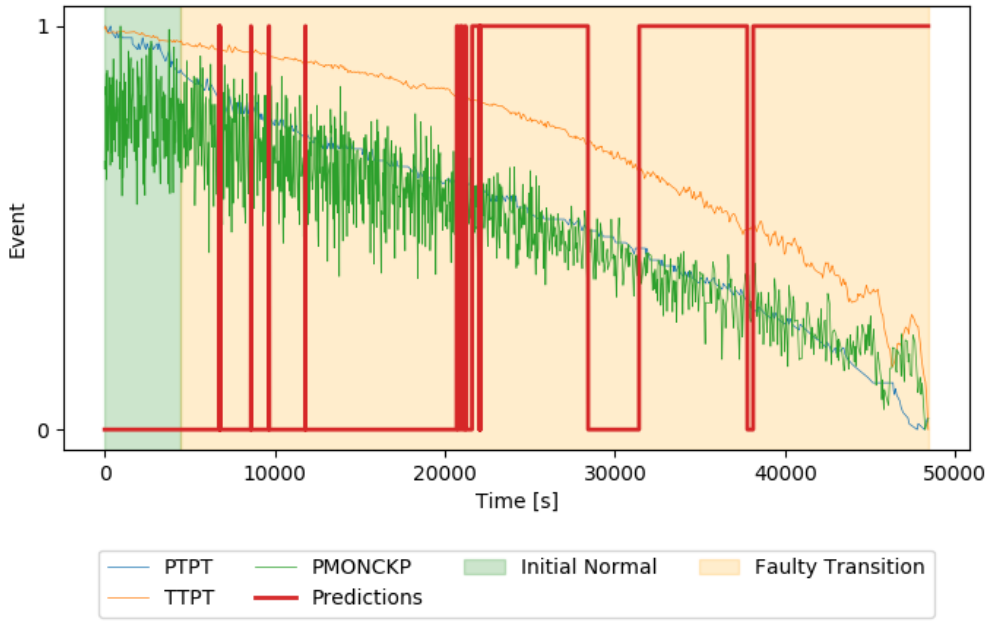


FIGURE 5.3: Example of a real instance of Class 5 showing the inconsistency of the system classification, where only 54% of the samples were correctly classified. Event values '1' and '0' denote normal and faulty states.

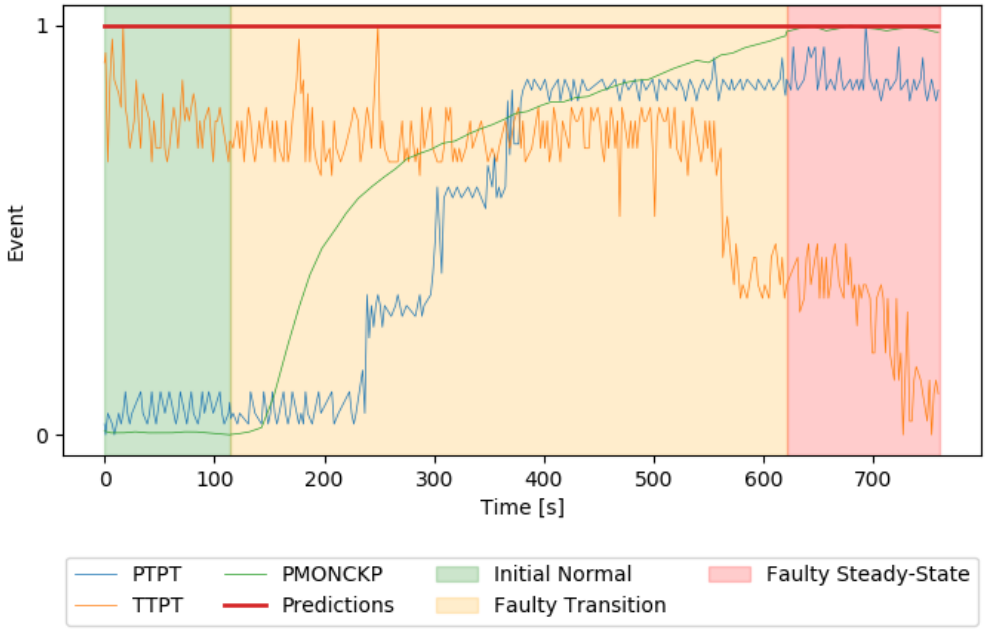


FIGURE 5.4: Example of a real instance of Class 6 along with the system classification, where the system was not able to correctly classify a single sample in the initial normal state. Event values '1' and '0' denote normal and faulty states.

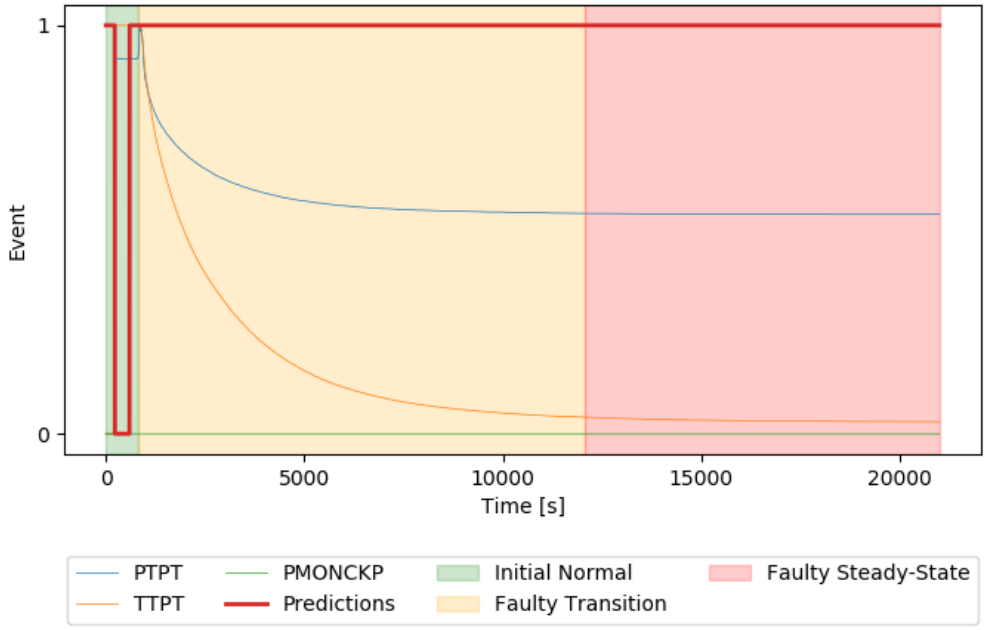


FIGURE 5.5: Example of a real instance of Class 2 along with the system classification, where the system achieved below than 90% accuracy on initial normal. Event values '1' and '0' denote normal and faulty states.

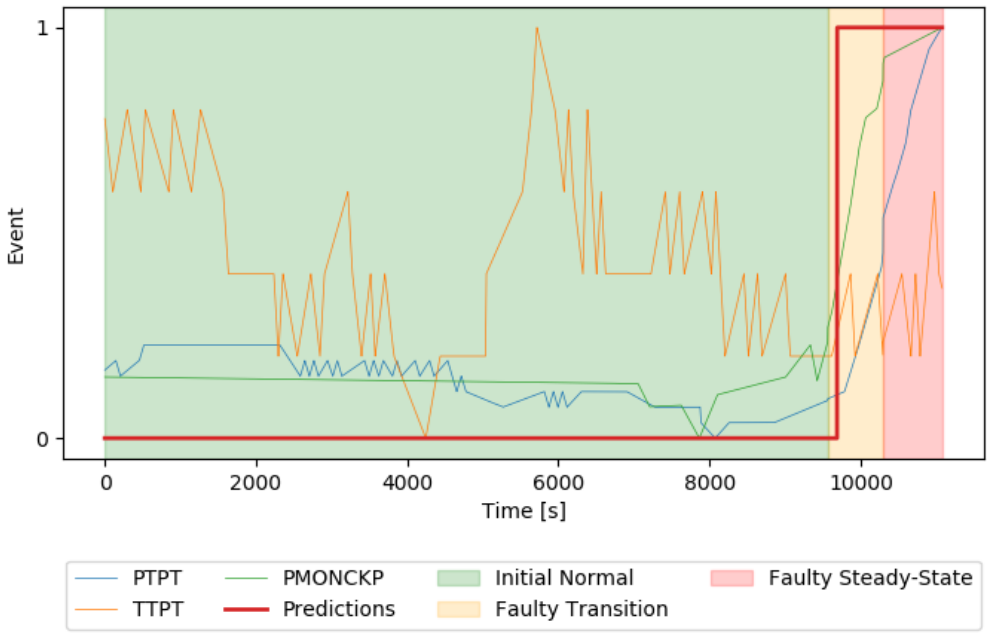


FIGURE 5.6: Example of a real instance of Class 6 along with the system classification. All samples belonging to initial normal and steady-state were correctly classified. However, the transient state only accomplished 80% accuracy. Event values '1' and '0' denote normal and faulty states.

5.4.2 System efficiency and reliability evaluation

To assess how reliable and efficient the system is to anticipate incoming failures, three time-intervals have been applied. These time intervals (in seconds) are defined as following: how fast each classifier is to detect the transient state (time of detection); how

many consecutive correct predictions are made after the time of detection; and how long time to take action before the incoming failure occurs, respectively denoted as t_1, t_2, and t_3. Figure 5.7 illustrates these time intervals on a real instance of fault 1 along with the system classification. This case is of an instance considered as wrongly classified, as the system was only able to correctly classify roughly 12% of the transient state. However, the system roughly used $t_1 = 2460s$ to detect the incoming failure, with $t_2 = 2700s$ consecutive correct predictions, and $t_3 = 18530s$ time left before the condition of faulty-steady state arrives. Taken this into consideration, one can argue that the system was able to detect the transient state with 2700 consecutive correct predictions from time of detection. Table 5.5 shows the average values of the time-intervals t_1, t_2, and t_3 for each classifier and their respective faults, where each number designated in parenthesis is the percentage of the corresponding time-interval concerning the transient state. Classes 0, 3, and 4 are not included as their transient phase is absent in the 3W dataset.

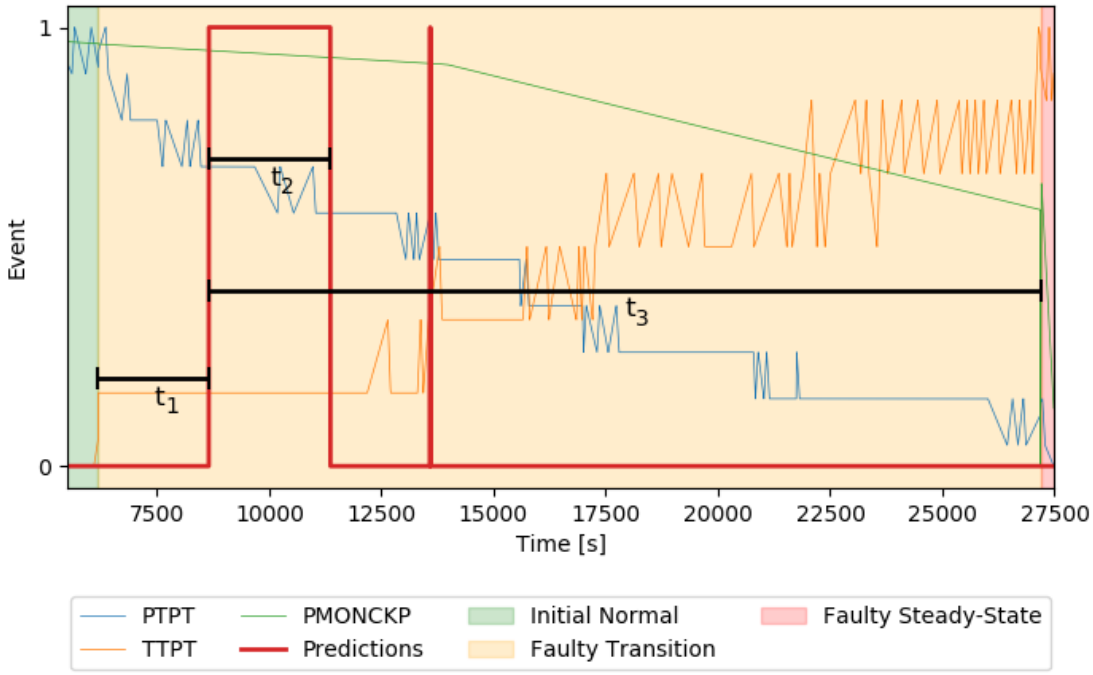


FIGURE 5.7: A real instance of Class 1, predicted sample-wise for all transitional states. Event '0' and '1' denote the initial normal, transient, and steady states, respectively.

TABLE 5.5: Transient analysis of how well each classifier perform. Time-intervals are given in seconds and the designated numbers in parenthesis are the percentage of the corresponding time-interval concerning the total transient state.

Fault	Type	t1 [s]	t2 [s]	t3 [s]
Class 1	Real	2463.0 (11.73%)	2710.0 (12.90%)	18530.0 (88.26%)
	Simulated	4.4 (0.009%)	38545.2 (83.49%)	46160.2 (99.99%)
Class 2	Real	15.5 (0.33%)	4700.1 (99.67%)	4700.1 (99.67%)
	Simulated	103.4 (2.87%)	2795.4 (77.65%)	3496.6 (97.12%)
Class 5	Real	564.2 (1.06%)	41879.0 (78.87%)	52533.7 (98.93%)
	Simulated	1.18 (0.02%)	4708.7 (99.99%)	4708.7 (99.99%)
Class 6	Real	73.5 (11.87%)	546.0 (88.20%)	546.0 (88.20%)
	Simulated	1.5 (0.02%)	6677.1 (92.34%)	7229.8 (99.98 %)
Class 8	Real	1.0 (0.00%)	20078.0 (100%)	20078.0 (100%)
	Simulated	5.3 (0.03%)	7640.3 (43.31%)	17633.9 (99.96)

5.5 Experiment 3: fault versus not fault

This section reviews the experiment and results of the classification scenario fault versus not fault. In this experiment, each classifier is fit on data from all classes. The same settings from the latter classification scenario apply in this case, with regards to hyperparameter selection and training routine. As for the training routine, the subsampling factor had to be increased, since each classifier is fit on data from every class. Each classifier under training applied a subsampling factor of 100 for instances that did not belong to their given class (Not Fault). Also, real instances belonging to their given class were not subsampled, but simulated instances applied a subsampling factor of 10.

The test results of this classification scenario of the 3W dataset can be seen in Tables 5.6 and 5.7. The classification method shows satisfactory results with the classifiers of classes 2, 3, and 4. These classes are correctly classified with an average accuracy of over 90% of Class 0 and all transitional states when it comes to real instances. Simulated instances of these classes indicate to be harder to predict, where the accuracy drops as low as 80% for the transient state and 93% for the steady-state for classes 2 and 3, respectively.

TABLE 5.6: Overall and transitional test results of the classification scenario fault versus not fault, including real and simulated instances for each classifier. Empty entries indicate the absence of data for that given fault type.

Fault	Window Size	Type	Transitional ACC	Overall ACC
Class 1	900	Real	0.213	0.992
		Simulated	0.999	0.999
Class 2	400	Real	0.991	0.999
		Simulated	0.862	0.997
Class 3	300	Real	0.995	0.980
		Simulated	0.936	0.990
Class 4	300	Real	0.977	0.985
		Simulated	-	1.000
Class 5	300	Real	0.373	0.966
		Simulated	0.993	0.996
Class 6	300	Real	0.868	0.999
		Simulated	0.925	0.987
Class 8	1200	Real	0.892	0.999
		Simulated	0.972	0.982

TABLE 5.7: Test results of 'Not Fault' (class 0) and each transitional state of the classification scenario fault versus not fault, including real and simulated instances for each classifier. Empty entries indicate the absence of data for that given fault type.

Fault	Window Size	Type	Not Fault (Class 0)	Initial Normal	Transient State	Steady State
Class 1	900	Real	1.000	0.779	0.098	0.001
		Simulated	0.999	0.999	0.999	0.999
Class 2	400	Real	0.999	0.970	0.998	1.000
		Simulated	0.999	1.000	0.821	0.848
Class 3	300	Real	0.979	-	-	0.995
		Simulated	0.998	-	-	0.936
Class 4	300	Real	0.987	-	-	0.977
		Simulated	1.000	-	-	-
Class 5	300	Real	0.999	0.396	0.372	-
		Simulated	0.997	0.103	0.999	0.999
Class 6	300	Real	1.000	0.997	0.384	0.150
		Simulated	0.999	0.813	0.936	0.930
Class 8	1200	Real	0.999	0.000	1.000	1.000
		Simulated	0.983	0.939	0.961	0.999

5.5.1 Event-based evaluation

This classification scenario shows similar problems related to inconsistent system classifications when it comes to Class 4, as in the previous experiment. Furthermore, similar performance is seen in classes 1, 2, 3, 4, and 8 with the same number of correct

classified events. This can be seen in Table 5.8, which shows how many events were classified correctly with a threshold of 90%. However, Class 5 is only able to correctly classify one of the total four transient events, and not a single event of the initial normal state was correctly classified. As a result, the system yields rapid fluctuations in its classifications, which can be seen in Figures 5.8 and 5.9. Class 6 shows similar performance but indicate different behavior compared to Experiment 2 (fault versus normal). Figure 5.10 shows the same instance that the previous system in Experiment 2 wrongly classified all samples in the initial normal event. However, in this case, the system was able to correctly classify roughly 80% of all samples of the same event.

TABLE 5.8: Event-based analysis of classification scenario fault versus not fault, showing the events that were correctly classified with an accuracy of 90%. Designated numbers in parenthesis show total number of events.

Fault	Type	Initial Normal	Transient	Steady-state
Class 1	Real	1(2)	0(2)	0(2)
Class 2	Real	5(7)	7(7)	3(3)
Class 3	Real	-	-	10(10)
Class 4	Real	-	-	96(103)
Class 5	Real	0(3)	1(4)	-
Class 6	Real	1(2)	1(2)	1(2)
Class 8	Real	0(1)	1(1)	1(1)

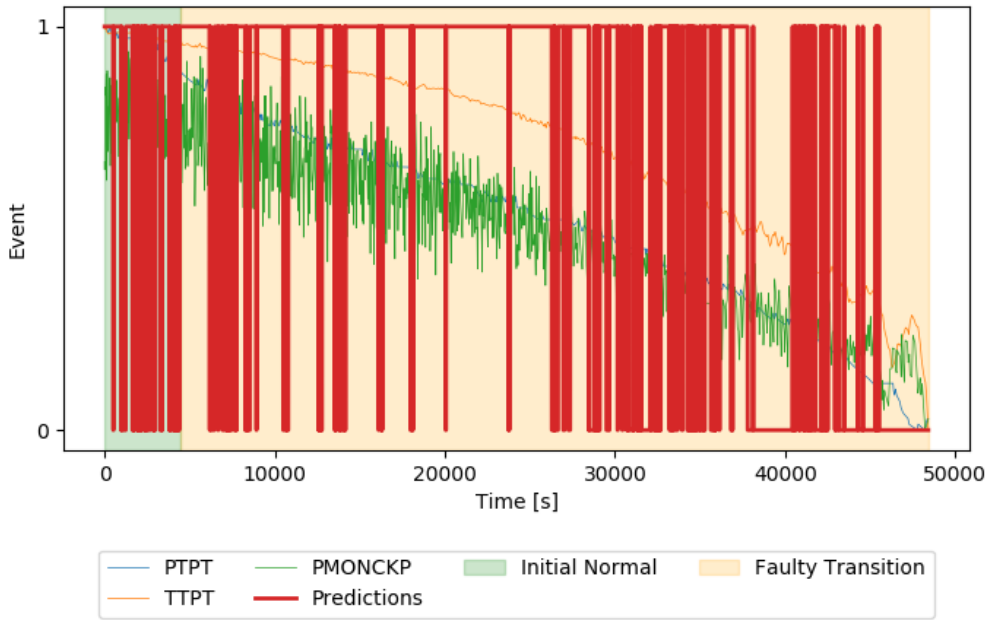


FIGURE 5.8: Example of a real instance of Class 5 along with the rapid inconsistent system classifications. Event values '1' and '0' denote normal and faulty states.

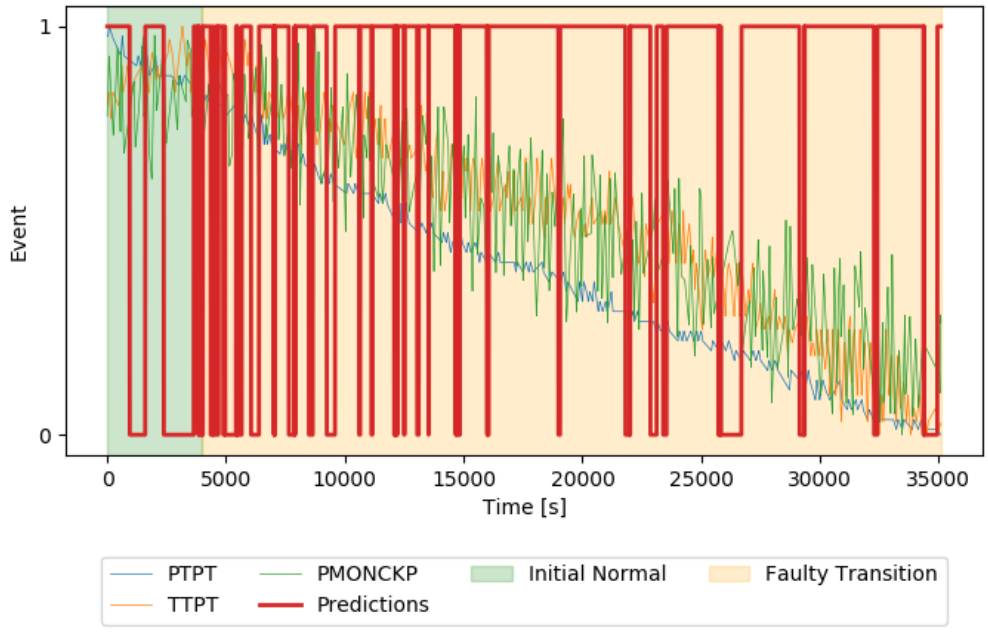


FIGURE 5.9: Example of a real instance of Class 5 along with the rapid inconsistent system classifications. Event values '1' and '0' denote normal and faulty states

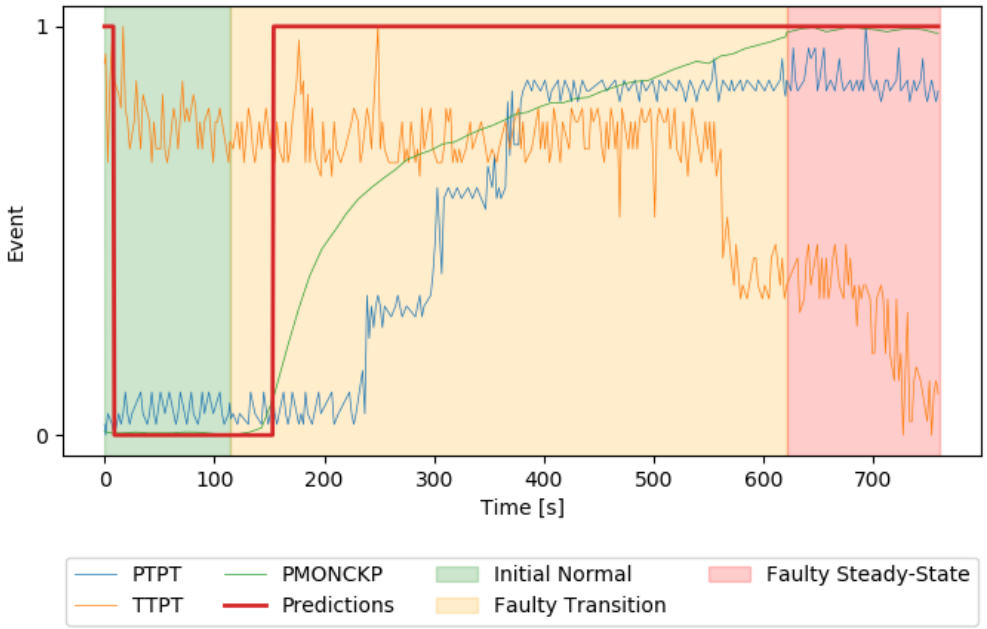


FIGURE 5.10: Example of a real instance of Class 6 along with the system classification, where the system achieved below than 90% accuracy on initial normal. Event values '1' and '0' denote normal and faulty states.

5.5.2 System efficiency and reliability evaluation

The assessments of each classifier based on the time-intervals t_1, t_2, and t_3 can be seen in Table 5.9. The system shows interesting results in class 2, where the time of detection has decreased by over six seconds compared to Experiment 2. As mentioned and showed in the last section, Class 5 is influenced by very inconsistent classifications, and therefore, it might seem like it quickly detects the transient state. This is not the case

and becomes evident with the figures showed earlier, which shows the inconsistency. The inconsistency applies to all the transitional states, and this makes the system very unreliable.

TABLE 5.9: Transient analysis of how well each classifier perform. Time-intervals are given in seconds and the designated numbers in parenthesis are the percentage of the corresponding time-interval concerning the total transient state.

Fault	Type	t1 [s]	t2 [s]	t3 [s]
Class 1	Real	2463.0 (11.73%)	2707.0 (12.90%)	18530.0 (88.26%)
Class 2	Real	9.3 (0.19%)	4706.4 (99.81%)	4706.4 (99.81%)
Class 5	Real	1.0 (0.00%)	822.3 (1.55%)	53097.3 (99.9%)
Class 6	Real	318.5 (51.5%)	237.5 (38.4%)	300.5(48.54%)
Class 8	Real	1.0 (0.00%)	20078.0 (100%)	20078.0 (100%)

5.6 Discussion

The results achieved from Experiment 2 and Experiment 3 show that the system is capable of correctly classifying faults but also capable of predicting incoming faults from their transient state. These incoming faults are often predicted in an early stage, such that it is possible to take necessary preventive action. The amount of data at disposal for each class tends to reflect the system performance for the given classifier. This applies to both classification methods, and in particular, to real instances of classes 1 and 8. Neither of the two classification methods can correctly classify any sample related to the initial normal state of Class 8. Classes 2, 3, and 4 achieves great accuracy for their respective transitional states and Class 0, for both classification scenarios. On the other hand, the fault versus normal classification scenario accomplishes better results for classes 5 and 6 compared to its counterpart, fault versus not fault.

Comparing these results with the work of Marins et al. [21], it is noticeable that their binary classification method with fixed window size achieved higher accuracy in classifying the initial normal state for each class on real instances. However, their multiclass classification method with a fixed window size achieved a lower average accuracy for real instances over all transitional states for most classes. Their multiclass classification method achieved 50.8%, 88.8%, 79.1% 95.4%, 83.3%, 71.1%, and 0% accuracy for class 1 to 8, respectively. In comparison to the results achieved by the 'fault versus normal' system with individual window sizes, their multiclass classification system only achieved a higher average accuracy over all transitional states for class 1. Keep in mind that it is expected that the binary classifiers in this work achieve higher accuracy for each class compared to the multiclass classification system. This is because using a single classification system to identify all classes imposes a significantly more challenging machine-learning problem than using multiple binary classifiers.

5.6.1 Initial normal

The results have shown that the normal samples that belong to Class 0 (normal operation) are easily classified compared to normal samples preceding an abnormal event (initial normal). These samples vary a lot when compared to each other. A reason for

the discrepancy between the classification accuracy of Class 0 samples and initial normal samples for real instances may be related to inactive sensors and frozen/missing variables. An example of this is that all sensor readings of the sensor referred to as "QGL" are approximately frozen 70% of the time and inactive the remaining 30% of the time for Class 0 (normal operation). Furthermore, in the case of Class 8, all sensor readings of the sensor "QGL" are available for all initial normal samples. Conclusion on this, the result from Experiment 2 (fault versus normal) suggests that the classifier identifies normal operation solely based on whether the sensor "QGL" is inactive or not, which is why not a single initial normal sample is correctly classified for Class 8.

5.6.2 Inconsistency

Besides the poor performance related to Classes 1 and 8, the limitations of this system are tied to the fluctuations in system classifications for Classes 4 and 5. A few instances from those two classes have shown inconsistent classifications for both systems in Experiments 1 and 2. However, in those situations, the classifiers have often shown high accuracy, which indicates that measures can be taken to decrease these inconsistencies. In this work, we suggest a simple filter to smoothen these fluctuations that occur during inconsistent classifications, referred to as "time-consistency filter." The filter strides over the system classifications with a window and removes the class with the fewest output classifications inside that given window. Figure 5.11 shows an example of a real instance of Class 4, along with the system classifications and the outputs of the time-consistency filter. In this case, the window size of the time-consistency filter is 120 samples. It is evident that the filter is not able to prevent all classification oscillations, but it does smoothen out the majority. The filter may be more efficient if the window size is increased, but this will increase the delay, which should be prevented.

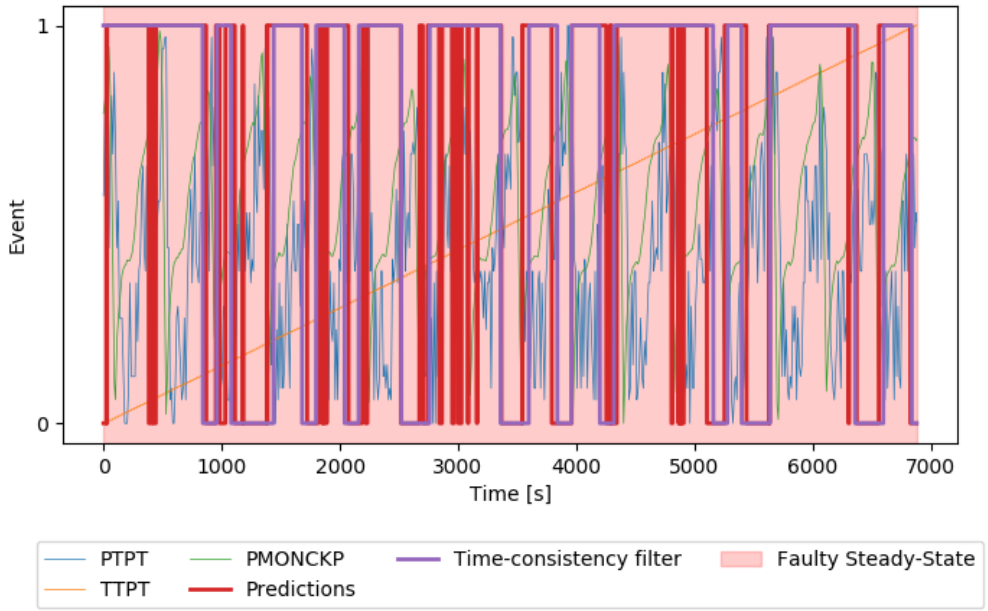


FIGURE 5.11: Example of a real instance of Class 4 along with the rapid inconsistent system classifications and with the time-consistency filter. Event values '1' and '0' denote normal and faulty states.

5.7 Summary

This chapter has reviewed the implementation and results of the proposed system. Two classification scenarios have been implemented, where a strategy of individual window sizes have been applied. Initial experiments have been done with these window sizes through the grid search to find the best hyperparameters. Furthermore, an in-depth analysis of the results related to the classification scenario is provided, where the system is evaluated as a CBM system in a real-world scenario, where factors such as efficiency and reliability are addressed. This chapter has also reviewed the limitations related to the system and provided an analysis of the challenges related to the initial normal samples, and given a method to smoothen out oscillations during inconsistent classifications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The development of innovative systems considered as "Smart Technology" is one of the hottest topics of modern engineering. These technologies apply the current state-of-the-art in sensor-technology and communication systems, known as IoT systems, and data-driven applications for analysis. This work is concerned with fault detection and identification in particular, and I had to explore fields related to condition-based monitoring and predictive maintenance, where complex machine learning tools are the current state-of-the-art.

This work has built a complete CBM system to successfully identify and detect real abnormal events in offshore oil wells related to the 3W dataset. The implemented CBM system is capable of preprocessing raw-sensor data, extracting features, reducing dimensionality, and classifying with the popular random forests algorithm. A new classification scenario and strategy has been introduced to the problem. The new strategy includes the use of individual window sizes for each class fault during the feature extraction process. The use of individual window sizes has been applied to two classification scenarios, which is the "fault versus normal operation" and the new classification scenario referred to as "fault versus not fault." These two classification scenarios include the use of binary classifiers, one for each class, but attempt to identify and classify fault events in a different manner. The two methods do not only show that they are capable of correctly classifying faults but also to anticipate incoming faults.

Analysis of the results indicates that there are certain limitations related to the system. These limitations arise in the minority of the number of situations for Classes 4 and 5 and are associated with rapid fluctuations in system classification. In the few cases it happens, the classifiers show an inconsistent behavior in classifications but usually achieve an overall high accuracy of predictions. However, these behavioral patterns occur mainly during faulty states, which can be desirable. This is due to the reasoning that many false positives can cause distrust for an operator, and the system will be viewed as unreliable. Consequently, if these fluctuating classifications should appear in faulty states, they would still not be desirable but would be able to detect a fault and notify an operator.

6.2 Future work

This work has successfully implemented a CBM system to detect and identify normal and faulty events in subsea oil wells. The CBM system applies state-of-the-art solutions, such as machine learning techniques and algorithms. Despite this, the experiment still has uncovered areas. The following issues would be worth investigating further:

- There are many opportunities in feature analysis and application for this experiment. This could be done by investigating the features that have been applied, where interesting subjects such as feature importance can uncover knowledge of which features are most rich in information. Furthermore, introduce new features. Until now, only statistical first-order features have been applied. Interesting research would be to explore second-order features applied in fields such as signal processing.
- Implementing more powerful machine learning classification algorithms, where XGBoost is an interesting algorithm that attempts to exploit the advantages of random forests and gradient boosting. Another widely used and popular machine learning algorithm that can be explored is the Light Gradient Boosting Machine (LGBM), which also has a similar architecture to random forests and XGBoosts, which is that is built of trees.

Appendix A

Experiments

A.1 Grid Search on rolling window size

This appendix shows the results for Experiment 1, where the grid search was applied to find the best window size for each class. In this experiment, only real samples from each class have been used, and the classification method "fault versus normal." The seven separate figures in this appendix show the mean test accuracy for every window size $N \in 100, 200, 300, ..., 1200$, for each fault. The best window size for each fault is shown with a red cross, marking the local maximum.

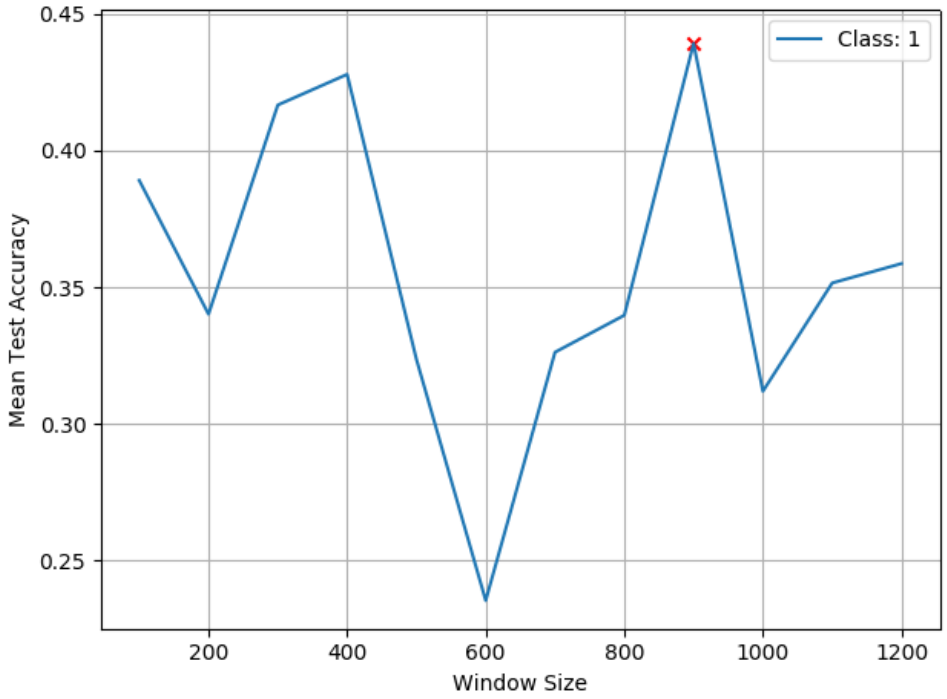


FIGURE A.1: Grid search result showing mean test accuracy for the different window sizes for fault 1.

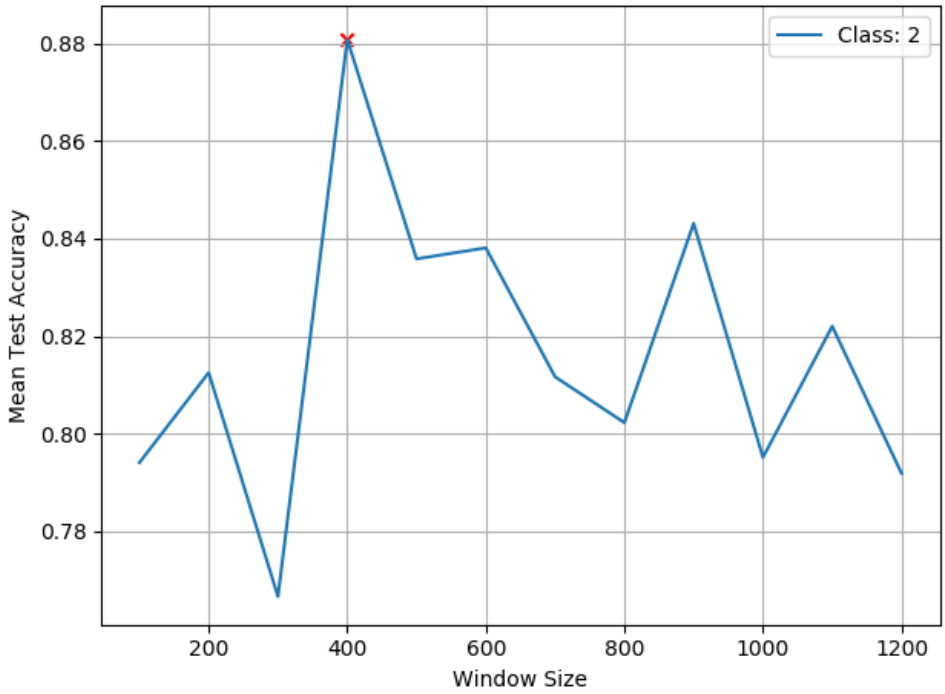


FIGURE A.2: Grid search result showing mean test accuracy for the different window sizes for fault 2.

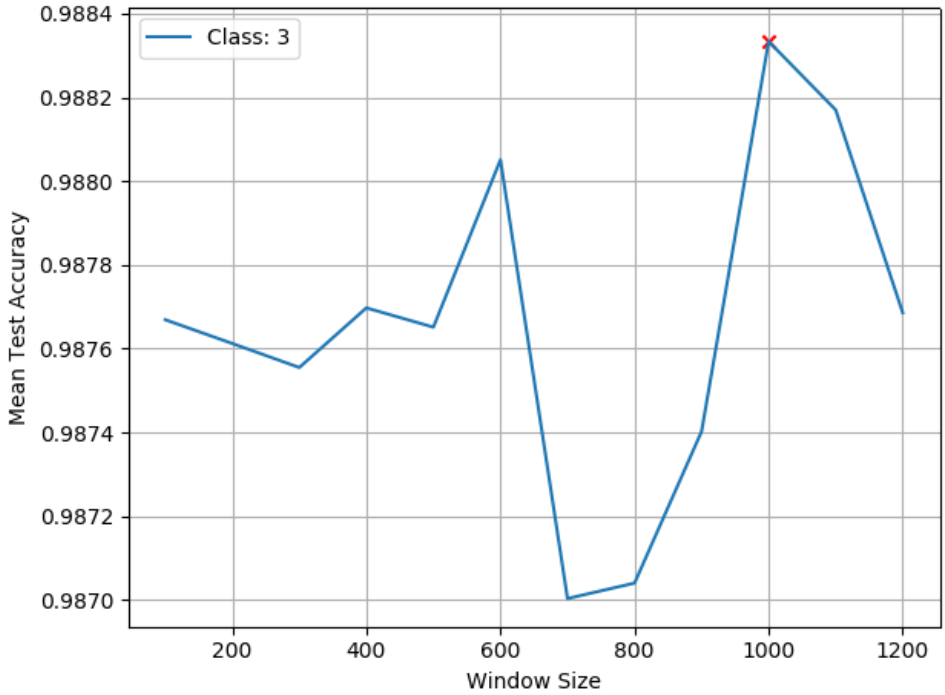


FIGURE A.3: Grid search result showing mean test accuracy for the different window sizes for fault 3.

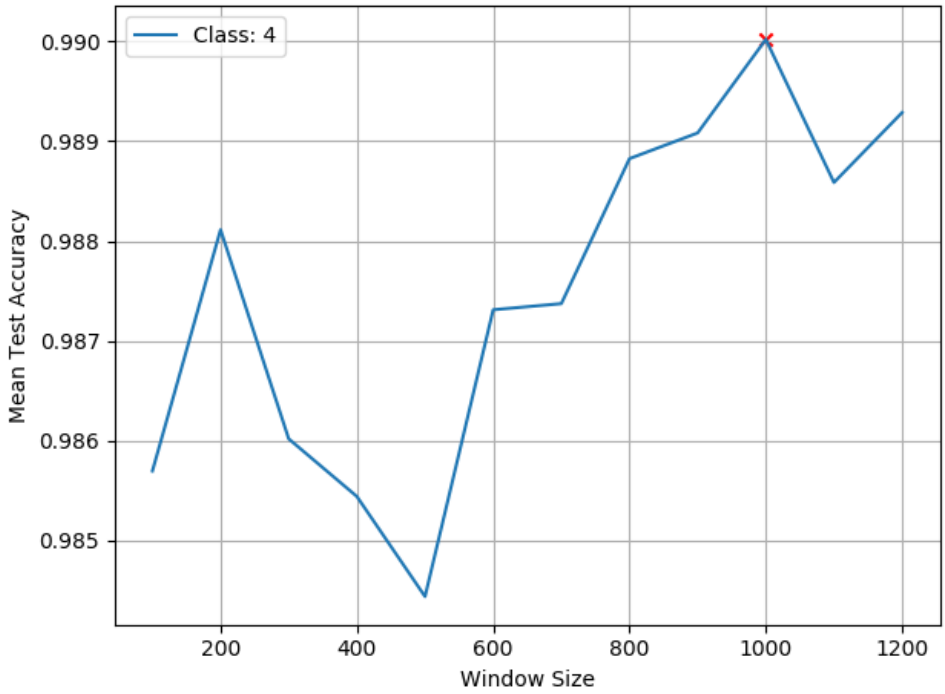


FIGURE A.4: Grid search result showing mean test accuracy for the different window sizes for fault 4.

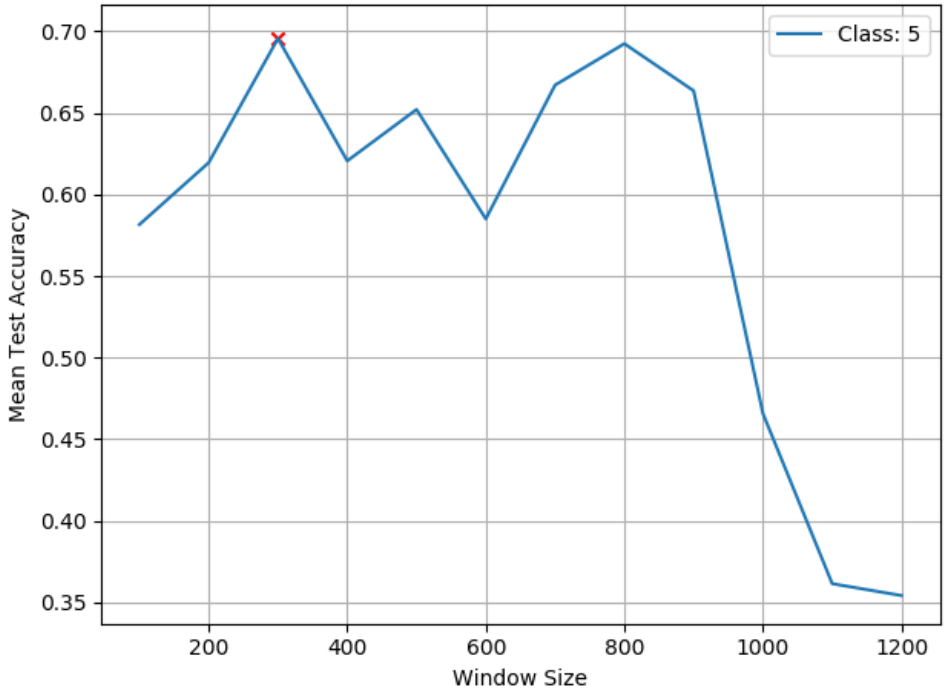


FIGURE A.5: Grid search result showing mean test accuracy for the different window sizes for fault 5.

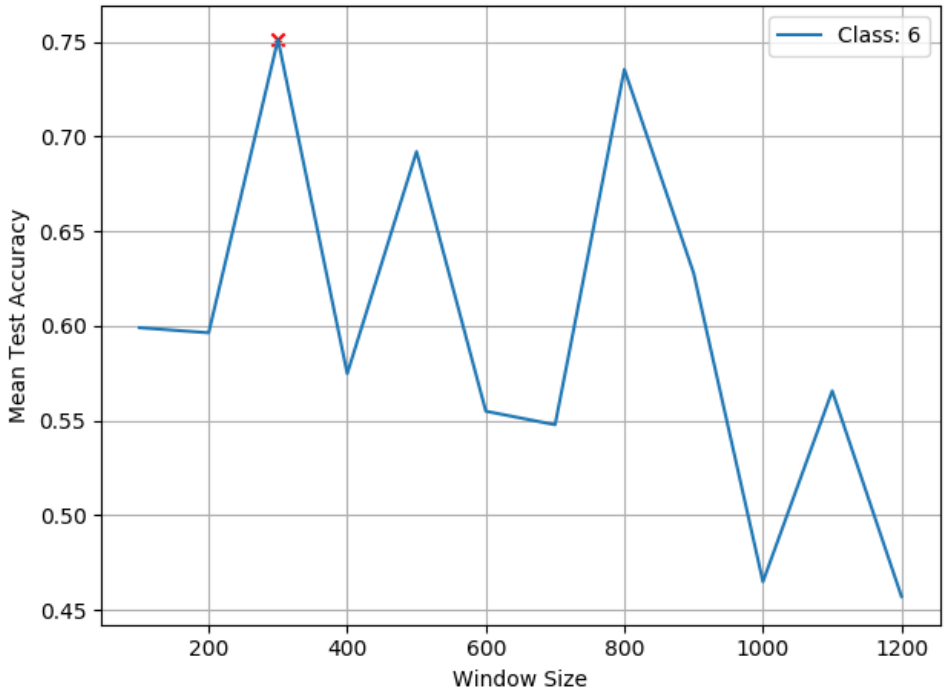


FIGURE A.6: Grid search result showing mean test accuracy for the different window sizes for fault 6.

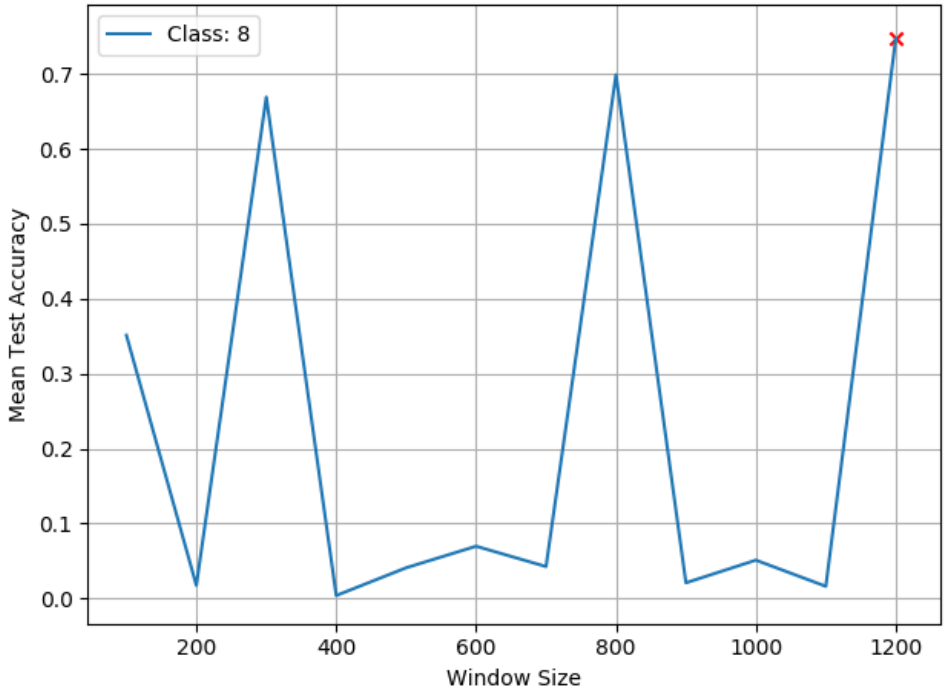


FIGURE A.7: Grid search result showing mean test accuracy for the different window sizes for fault 8.

Bibliography

[1] K. K. Agrawal, G. N. Pandey, and K. Chandrasekaran. "Analysis of the condition based monitoring system for heavy industrial machineries". In: *2013 IEEE International Conference on Computational Intelligence and Computing Research*. 2013, pp. 1–4. DOI: 10.1109/ICCIC.2013.6724183.

[2] Ahmadi et al. "Well-testing model identification using time-series shapelets". In: *Journal of Petroleum Science and Engineering* 149 (2017), pp. 292 –305. ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2016.09.044. URL: http://www.sciencedirect.com/science/article/pii/S0920410516305071.

[3] R. N. Anderson. "'Petroleum Analytics Learning Machine' for optimizing the Internet of Things of today's digital oil field-to-refinery petroleum system". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 4542–4545.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[5] María Blanca et al. "Skewness and Kurtosis in Real Data Samples". In: *Methodology European Journal of Research Methods for the Behavioral and Social Sciences* 9 (May 2013), pp. 78–84. DOI: 10.1027/1614-2241/a000057.

[6] Leo Breiman. "Bagging Predictors". In: *Mach. Learn.* 24.2 (Aug. 1996), 123–140. ISSN: 0885-6125. DOI: 10.1023/A:1018054314350. URL: https://doi.org/10.1023/A:1018054314350.

[7] Antonio Criminisi, Ender Konukoglu, and Jamie Shotton. *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*. Tech. rep. MSR-TR-2011-114. 2011. URL: https://www.microsoft.com/en-us/research/publication/decision-forests-for-classification-regression-density-estimation-manifold-learning-and-semi-supervised-learning/.

[8] Chris von Csefalvay. "Predictive maintenance helped win a war". In: (Sept. 2019). URL: https://medium.com/starschema-blog/predictive-maintenance-helped-win-a-war-now-it-can-help-you-outpace-the-competition-597ebfeab546.

[9] R Goldberg et al. "The Importance of Features for Statistical Anomaly Detection". In: *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*. Santa Clara, CA: USENIX Association, July 2015. URL: https://www.usenix.org/conference/hotcloud15/workshop-program/presentation/goldberg.

[10] Philip Higgs et al. "A Survey on Condition Monitoring Systems in Industry". In: vol. 3. Jan. 2004. DOI: 10.1115/ESDA2004-58216.

[11] Gareth James et al. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. ISBN: 1461471370.

[12] A. K. S. Jardine, D. Lin, and D. Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical Systems and Signal Processing* 20 (2006), pp. 1483–1510.

[13] Ian Jolliffe. *Principal component analysis*. New York: Springer Verlag, 2002.

[14] Sukumar Laik. *Offshore Petroleum Drilling and Production*. Boca Raton: CRC Press, 2018. ISBN: 9781315157177. DOI: https://doi.org/10.1201/9781315157177.

[15] Heiner Lasi et al. "Industry 4.0". In: *Business & Information Systems Engineering* 6 (Aug. 2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[16] Jay Lee, Behrad Bagheri, and Hung-An Kao. "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems". In: *SME Manufacturing Letters* 3 (Dec. 2014). DOI: 10.1016/j.mfglet.2014.12.001.

[17] Cuiwei Liu, Yuxing Li, and Minghai Xu. "An integrated detection and location model for leakages in liquid pipelines". In: *Journal of Petroleum Science and Engineering* 175 (2019), pp. 852 –867. ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2018.12.078. URL: http://www.sciencedirect.com/science/article/pii/S0920410518311872.

[18] Y. Liu et al. "Semi-supervised Failure Prediction for Oil Production Wells". In: *2011 IEEE 11th International Conference on Data Mining Workshops*. 2011, pp. 434–441.

[19] Gilles Louppe. *Understanding Random Forests: From Theory to Practice*. 2014. arXiv: 1407.7502 [stat.ML].

[20] Matheus A. Marins. "Machine learning tecniques applied to hydrate failure detection on production lines". In: 2018.

[21] Matheus A. Marins et al. "Fault detection and classification in oil wells and production/service lines using random forest". In: *Journal of Petroleum Science and Engineering* (2020), p. 107879. ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2020.107879. URL: http://www.sciencedirect.com/science/article/pii/S0920410520309372.

[22] Florent [Di Meglio] et al. "Stabilization of slugging in oil production facilities with or without upstream pressure sensors". In: *Journal of Process Control* 22.4 (2012), pp. 809 –822. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2012.02.014. URL: http://www.sciencedirect.com/science/article/pii/S0959152412000637.

[23] Kevin Patrick Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[24] R. Lyman Ott and Micheal T. Longnecker. *An Introduction to Statistical Methods and Data Analysis, Seventh Edition*. USA, 2016. ISBN: 978-1-305-26947-7.

[25] Om Patri et al. "Predicting Compressor Valve Failures from Multi-Sensor Data". In: Jan. 2015. DOI: 10.2118/174044-MS.

[26] Cauligi Raghavenda et al. "Global Model for Failure Prediction for Rod Pump Artificial Lift Systems". In: (Apr. 2013). DOI: 10.2118/165374-MS.

[27] Sensor-Works. "A Short History of Predictive Maintenance". In: (June 2018). URL: http://sensor-works.com/a-short-history-of-predictive-maintenance/.

[28] Lindsay I Smith. *A tutorial on principal components analysis*. Tech. rep. Cornell University, USA, 2002. URL: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

[29] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. USA: California Technical Publishing, 1997. ISBN: 0966017633.

[30] Hewei Tang et al. "Time series data analysis for automatic flow influx detection during drilling". In: *Journal of Petroleum Science and Engineering* 172 (Sept. 2018). DOI: 10.1016/j.petrol.2018.09.018.

[31] Hewei Tang et al. "Time series data analysis for automatic flow influx detection during drilling". In: *Journal of Petroleum Science and Engineering* 172 (2019), pp. 1103 –1111. ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2018.09.018. URL: http://www.sciencedirect.com/science/article/pii/S0920410518307733.

[32] Ricardo Emanuel Vaz Vargas et al. "A realistic and public dataset with rare undesirable real events in oil wells". In: *Journal of Petroleum Science and Engineering* 181 (2019), p. 106223. ISSN: 0920-4105. DOI: https://doi.org/10.1016/j.petrol.2019.106223. URL: http://www.sciencedirect.com/science/article/pii/S0920410519306357.

[33] Venkat Venkatasubramanian et al. "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods". In: *Computers & Chemical Engineering* 27.3 (2003), pp. 293 –311. ISSN: 0098-1354. DOI: https://doi.org/10.1016/S0098-1354(02)00160-6. URL: http://www.sciencedirect.com/science/article/pii/S0098135402001606.

[34] Lin Xie et al. "Operational data-driven prediction for failure rates of equipment in safety instrumented systems: A case study from the oil and gas industry". In: *Journal of Loss Prevention in the Process Industries* 60 (2019), pp. 96 –105. ISSN: 0950-4230. DOI: https://doi.org/10.1016/j.jlp.2019.04.004. URL: http://www.sciencedirect.com/science/article/pii/S0950423018309434.