André Paulo Ferreira Machado

Methodologies to Improve One-Class Classifier Performance Applied to Multivariate Time Series

Doctoral Thesis submitted to the Post-Graduation Program in Electrical Engineering from the Technological Center of the Federal University of Espírito Santo, as a partial requirement for the degree of Doctor in Electrical Engineering.

Federal University of Espírito Santo – UFES

Electrical Engineering Department

Post-Graduation Program

Supervisor: Dr. Celso José Munaro

Co-supervisor: Dr. Patrick Marques Ciarelli

Brazil

2024

André Paulo Ferreira Machado

Methodologies to Improve One-Class Classifier Performance Applied to Multivariate Time Series

Doctoral Thesis submitted to the Post-Graduation Program in Electrical Engineering from the Technological Center of the Federal University of Espírito Santo, as a partial requirement for the degree of Doctor in Electrical Engineering.

Approved Thesis. Brazil, April 5th, 2024:

Prof. Dr. Celso José Munaro
Supervisor

Prof. Dr. Patrick Marques Ciarelli
Co-supervisor

Dr. Ricardo Emanuel Vaz Vargas
Petróleo Brasileiro, S. A.

Prof. Dr. Ginalber Luiz de Oliveira Serra
Instituto Federal do Maranhão

Prof. Dr. Leandro dos Santos Coelho
Pontifícia Universidade Católica do Paraná e
Universidade Federal do Paraná

Prof. Dr. Sergio Lima Netto
Universidade Federal do Rio de Janeiro

Brazil
2024

I dedicate this work to my family, who have always supported and encouraged me in every moment, especially during the most challenging phases of my life.

Acknowledgements

First and foremost, I express my gratitude to God, the author of life, for guiding my steps and allowing me to reach this point.

To my wife Rafaela, for her love, respect, dedication, companionship, and patience, even during the moments that seemed insurmountable. Her love and belief in me have been the driving force behind my achievements. I am eternally grateful for her unwavering partnership.

To my children, Maria Carolina and João Vicente who continually surprise me and provide inspiration for me to keep moving forward.

To the graduate Program in Electrical Engineering at the Federal University of Espírito Santo and its professors, particularly my supervisor Dr. Celso José Munaro, and co-supervisor Dr. Patrick Marques Ciarelli. I am grateful for their patience, guidance, enthusiasm, and willingness always to assist me. Their guidance and teachings have been invaluable for my academic and professional growth.

"Do you want me to teach you the way to attain true knowledge? It is to know what you know, and to know what you do not know. In truth, this is wisdom."

(Confucius)

Resumo

Essa dissertação propõe metodologias inovadoras para melhorar o desempenho de classificadores One-Class aplicados a dados de séries temporais multivariadas. O método principal se baseia no agrupamento de séries temporais multivariadas. Os conjuntos de dados provenientes de processos reais vêm de sensores disponíveis e são afetados por muitos fatores, tais como o mudança do processo, mudanças na região de operação e mau funcionamento do equipamento. Apesar disso, espera-se que as classes representadas por esses dados tão diversos possam ser reveladas por meio de classificadores treinados. Este trabalho levanta a hipótese de que o desempenho geral pode ser aprimorado treinando conjuntos de classificadores One-Class com subconjuntos de dados agrupados por similaridade, obtidos pela Média da Centroide de Distorção Temporal Dinâmica (DTW Barycenter Averaging - DBA), usada para medir a similaridade entre as séries temporais e de cada grupo. O método proposto é aplicado a classificadores One-Class, pois eles são treinados apenas com a classe alvo, que é agrupada com base na similaridade da série temporal usando Distorção Temporal Dinâmica (DBA) e agrupamento de dados k-médias. Além disso, uma segunda abordagem é proposta, chamada deslocamento temporal de rótulos, para melhorar a diferenciação entre dados normais e defeituosos. Este método é aplicado durante a fase de treinamento e foca em situações específicas envolvendo a transição da normalidade para dados defeituosos, onde os limites são difíceis de diferenciar (dados sobrepostos). Os resultados do deslocamento temporal mostram uma mitigação do efeito dos dados sobrepostos. As vantagens das técnicas são ilustradas por meio de sua aplicação em dois conjuntos de dados públicos: um da indústria de petróleo com instâncias que caracterizam oito classes de dados representadas por cinco séries temporais (conjunto de dados 3W) e outro de um sistema hidráulico para o estudo de falhas típicas de sistemas hidráulicos com cinco classes e dezessete séries temporais (conjunto de dados Monitoramento de condições de sistemas hidráulicos - ICM). Para o conjunto de dados 3W, sete classes são selecionadas para treinar classificadores LSTM (Long Short-Term Memory) usando o agrupamento de séries temporais. Os resultados demonstram que o aumento da similaridade dos dados de treinamento tende a melhorar o desempenho do classificador LSTM, alcançando um aumento de 10% no desempenho geral no conjunto de dados 3W. Em um caso específico, onde o modelo de agrupamento aumentou a similaridade em 84%, o desempenho da classificação melhorou em 21%. Para o monitoramento da condição de dados do sistema hidráulico, o método proposto alcançou uma melhoria significativa de desempenho de mais de 40% em comparação com o modelo base. Notavelmente, no caso específico de falha de vazamento, a melhoria do desempenho da classificação aumentou em 64%.

Palavras-chaves: Agrupamento de series temporais. One-Class Classifier. Dynamic Time Warping. Long Short-Term Memory. Classificação de Series Temporais Multivariadas.

Abstract

This work proposes novel methodologies to improve the performance of one-class classifiers applied to multivariate time series data. The main method is through clustering of multivariate time series. Datasets arising from real processes come from the available sensors and are affected by many factors, such as aging of the process, changes in the operation region, and equipment malfunction. Despite that, one expects that the classes represented by such diverse data can be unveiled via trained classifiers. This work hypothesizes that the overall performance can be improved by training sets of one-class classifiers with subsets of data clustered by similarity, obtained by DTW Barycenter Averaging (DBA) which is used to measure the similarity between the time series and each cluster. The proposed method is applied to one class classifiers since they are trained only with the target class, which is clustered based on time series similarity using Dynamic Time Warping and k-means. Additionally, a second approach is proposed, called time-shift of labels, to improve the differentiation between normal and faulty data. This method is applied during the training phase and focuses on particular situations involving the transition from normality to faulty data, where the boundaries are difficult to differentiate (overlapping data). The time-shift results show a mitigation of the effect of overlapping data. The advantages of the techniques are illustrated through their application to two public datasets one from the oil industry with instances characterizing eight classes of data represented by five time series (3W dataset), and another from a hydraulic system for the study of typical hydraulic system failures with five classes and seventeen time series (Condition monitoring of hydraulic systems - ICM dataset). For the 3W dataset, seven classes are selected to train Long Short Term Memory (LSTM) classifiers using the variables and instances clustered using time series clustering algorithms. The results demonstrate that increasing the similarity of training data tends to improve the performance of the LSTM classifier, achieving an increase of 10% in the overall performance on the 3W dataset. In a specific case, where the clustering model raised the similarity by 84%, the classification performance improved by 21%. For condition monitoring of hydraulic system data, the proposed method achieved a significant performance improvement of over 40% compared to the baseline model. Notably, in the specific case of leakage fault, the classification performance improvement rises by 64%.

Key-words: Time Series Clustering. One-Class Classifier. Dynamic Time Warping. Long Short-Term Memory. Multivariate Time Series Classification.

List of Figures

List of Tables

List of abbreviations and acronyms

BSW	*Basic Sediment and Water*
CSV	*Comma-Separated Values*
DBDES	*Density-Based Dynamic Ensemble Selection*
DBA	*DTW Barycenter Averaging*
DCS	*Dynamic Classifier Selection*
DES	*Dynamic Ensemble Selection*
DHSV	*Downhole Safety Valves*
DT	*Decision Tree*
DTW	*Dynamic Time Warping*
FP	*False Positive*
GRUs	*Gated Recurrent Units*
IIP	*Instances Identification Percentage*
LCSS	*Longest Common Sub-Sequence*
LSTM	*Long Short-Term Memory*
MODES	*One-class Dynamic Ensemble Selection for Multi-class Problems*
MODH	*Modified Hausdorff Distance*
MSC	*Multiple Classifier Systems*
MSE	*Mean Square Error*
MTS	*Multivariate Time Series*
OCSVM	*One Class Suport Vector Machine*
OCC	*One-Class Classification*
PCA	*Principal Component Analysis*
PDF	*Probability Density Function*

RF	*Random Forest*
RNN	*Recurrent Neural Network*
SFS	*Sequential forward selection*
SVDD	*Support Vector Data Descriptor*
SVM	*Support Vector Machine*

Contents

1 Introduction

Time series data can be found in various fields such as finance, economics, engineering, environmental science, and healthcare (CHOI et al., 2021). Examples of time series data include sensors, industrial processes, stock market prices, weather patterns, and patient vital signs. Analyzing time series data can provide valuable insights into the patterns and trends that exist over time (AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015). Some common applications of time series analysis include forecasting future values, detecting anomalies or outliers, and classification of trends and patterns (CHOI et al., 2021).

Classification in time series refers to the task of assigning a label or category to a time series based on its characteristics or features. The goal of time series classification is to classify a time series into a predefined category or class, such as a particular disease diagnosis, a financial market trend, or a type of equipment or industrial process pattern.

Multivariate Time Series Classification task involves analyzing and classifying sequences of data points where multiple variables are measured over time. The classification of multivariate time series adds an extra layer of complexity to an already challenging analytical task. Multivariate time series data is complex to analyze due to the intricate relationships between variables, the simultaneous consideration of temporal dependencies, and the high number of dimensions involved. In real-world applications, examples of some classes may be hard to collect because they represent rare events, or because they are expensive to obtain (e.g. requiring the use of expensive equipment or human expertise). It is common to have missing values or miss the entire signal of the multivariate time series that compose the instance. In this case, there is some careful use of that data. There are two approaches to training models: using all available variables even with limited data (which can be impractical in some cases) or focusing on a smaller set of key variables with a larger dataset. The latter approach, with a reasonable number of variables and sufficient instances, is generally preferred.

Classifying rare events or objects with limited instances or variables is particularly challenging. This is because the classes of interest most are often the ones with the fewest training examples (MOULTON, 2018). However, once this challenge is overcome, valuable insights obtained from the analysis can be useful for decision-making and forecasting in many real-world applications.

Thus, having highly capable classifiers that are well-trained to detect events in datasets with instances containing time series that characterize those events is valuable. One or more time series can be used to characterize each event, labeled as belonging to

a given class. The greater number of instances allows for better training and statistical tests that make the classifier more reliable. However, when rare events are investigated, the scarcity of instances can be an issue in training classifiers (VARGAS et al., 2019).

Moreover, the different realizations of time series associated with a class tend to be similar in more controlled environments. They can be intentionally impacted by factors one wants to analyze, for example, changing the load on a motor under test (KUMAR et al., 2022). However, when data are collected in less controlled environments, the factors that affect the collected data may be unknown or difficult to measure, causing time series that characterize the same event to behave in different ways.

In the context of the One-Class Classification (OCC) problem, the use of such data for training a single classifier tends to generate poor results, due to the difficulty in associating the same class with data with different behaviors. Problems like this are considered by Sharma, Somayaji & Japkowicz (2018), where the so-called complex original domains are divided into subdomains that are easier to model, i.e., learning in the context of sub-concepts rather than over the entire domain. Each sub-concept of the majority class is identified and a classifier is created for each one, creating a group of classifiers, known as an ensemble (DONG et al., 2020).

Ensemble learning has demonstrated its effectiveness in improving predictive accuracy, breaking down complex and challenging learning problems into easier sub-problems (KRAWCZYK et al., 2017). Promising results have been demonstrated by relevant research utilizing one-class classifiers in Multiple Classifier Systems (MSC) (KRAWCZYK; WOŹNIAK; CYGANEK, 2014; KRAWCZYK et al., 2018; FRAGOSO et al., 2021; MOULTON, 2018). Although this technique has demonstrated effectiveness, it remains unexplored whether this method can be applied to the OCC in multivariate time series (MTS). In MTS scenarios, the use of the approach can be tricky, due to the presence of unbalanced data with behaviors affected by diverse and unknown factors (MACHADO et al., 2024).

A multi-class problem can be divided into several easier-to-solve sub-problems. The use of one-class classifiers has several suitable properties for treating such sub-problems. Krawczyk, Woźniak & Cyganek (2014) and later Krawczyk et al. (2018) used approaches of this type. The approach used here is similar, but with a time series clustering step, using Dynamic Time Warping (DTW) and k-means methods to generate the sub-problems (subsets) for each class considered. First, time series clustering followed by instances clustering is applied to generate subsets from the original data. Second, one-class classifiers are trained for each subset, and the best classifiers are selected. Finally, the outputs of the selected classifiers are combined. This methodology, based on a new decomposition strategy, breaks the target classes into sub-concepts. The proposed methodology can be applied to datasets composed of multivariate time series to be handled by one-class classifiers. This is the major contribution of this thesis.

Furthermore, a second method, called the time-shift, is proposed to improve classification for problems where data exhibits overlapping patterns across different classes. This scenario occurs when samples from one instance with distinct classes share similar characteristics, making it challenging for traditional models to distinguish them accurately.

The time-shift method tackles this challenge by strategically excluding transient data points from the training set. This process involves creating shifted versions of the original training data.

1.1 Problem Statement

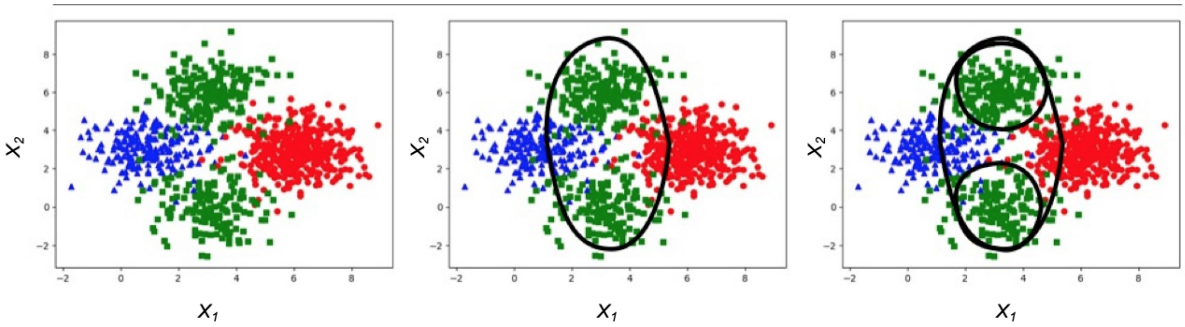
The natural approach to address multi-class problems is by using multi-class classifiers. However, in certain scenarios, such as imbalanced data or a large number of classes, breaking down a multi-class problem into smaller and simpler problems can be more efficient than using multi-class classifiers (DONG et al., 2020). One alternative approach is one-class decomposition, which involves training one-class classifiers (OCCs) for each class. However, it can be challenging for OCCs to fit the data optimally, particularly with complex intra-class distributions. Multiple classifier systems (MSC) offer inherent robustness and are better suited for such situations. Therefore, using multiple OCCs for each class can potentially improve the effectiveness of one-class decomposition (FRAGOSO et al., 2021).

This work addresses the challenge of achieving multi-class classification performance for multivariate time series data. The data exhibits diverse behaviors within the same class due to the influence of various factors. Some common problems in real data are different behaviors for different operation regions, overlapping data, frozen, and missing values related to the collecting process, outliers, and unexpected behavior of variables related to interference from process operators. Thus, the first step in the modeling is to assign proper data in clusters. These clusters can be used to fit different models or even partially discard data when their behaviors have unexpected patterns. Time series clustering is an important tool used here to obtain different clusters of data related to the same class.

The models considered here are one-class classifiers used to discriminate the classes described by a multivariate time series dataset. The strategy adopted consists of clustering data for training classifiers to improve their performance using data with more similarity and considering also the available variables. The dataset should be organized into homogeneous groups where the within-group-object similarity is minimized and the between-group-object dissimilarity is maximized. The hypothesis is that the classifiers trained with data with more similarity will benefit from an increase in their classification performance. However, the process of selecting data must ensure that only data that worsens classifiers' performance is discarded from the training process.

Figure 1 illustrates the clustering stages involved in the proposed method for one-class classification. On the left, three classes of data, blue triangles, red circles, and green squares, from the target class are available, after pre-processing. Each point is an instance (i.e., a multivariate time series represented in 2D). We now discuss the processing of the green class, since each class is processed separately. In the middle, after the training phase, the green square class was modeled by an OCC, and a decision boundary (black line) was estimated, describing the target class. Finally, on the right, the result of the training phase after clustering by similarity: two subsets of instances were grouped by similarity from the original target class. Decision boundaries, describing the target class, were estimated for each subset. As can be seen, the green class was easily separated from the other classes (red and blue) using the combination of the outputs of two classifiers rather than using just one. For this example, no improvement is expected with clustering instances of the other two classes, since a single boundary can represent each one of them.

Figure 1 – Clustering training dataset illustration.



Source: The Author

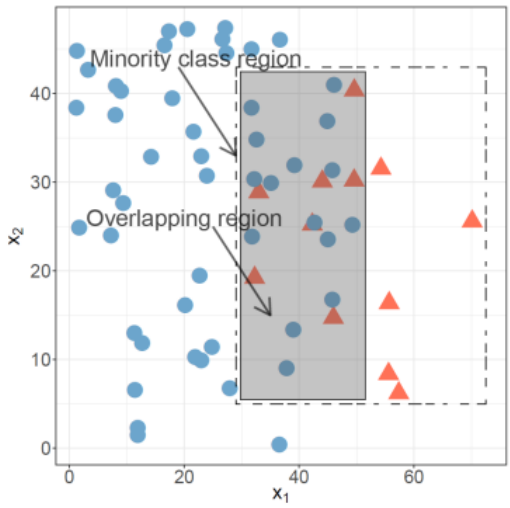
Additionally, in many applications involving classification, there is often occurs that some samples from different classes have very related characteristics which are called overlapping samples for they usually reside in overlapping regions in the feature space (CHATTERJEE et al., 2021).

Multiple authors have identified class overlap as a related issue for machine learning algorithms, focusing on the effect of class overlap by comparing the degradation in performance. Prati, Batista & Monard (2004) studied the role of overlapping between minority and majority classes in imbalanced datasets. They showed that increasing the overlapping ratio was more responsible for decreasing AUC results than decreasing the cardinality of the minority class.

Experimental results in Vuttipittayamongkol, Elyan & Petrovski (2021) are consistent with existing literature and show that the performance of the learning algorithms deteriorates across varying degrees of class overlap whereas class imbalance does not always have an effect. The review emphasizes the need for further research towards handling class

overlap in imbalanced datasets to effectively improve learning algorithms' performance. Figure 2 shows an illustration, in this case, the red triangle characterizes a minority class. Blue circles represent the majority class and the shadow region presents the overlapping. In other words, class overlap occurs when instances of more than one class share a common

Figure 2 – Class Overlapping illustration.



Source: Vuttipittayamongkol, Elyan & Petrovski (2021)

region in the data space. These instances have similarities in feature values although they belong to different classes, and such a complication is a substantial obstacle in classification tasks. The class overlap problem becomes more serious when class imbalance is also present in the data (VUTTIPITTAYAMONGKOL; ELYAN; PETROVSKI, 2021).

The literature discusses the issue of class overlapping from two perspectives, which result in two types of overlapping referred to as "concept overlapping" and "sample overlapping". The concept overlapping perspective takes a broader view, where the concepts of two classes share an overlapping region in the feature space. On the other hand, the sample overlapping perspective focuses on individual samples, where some samples from one class may have similar characteristics or overlap with some samples from the other class in the feature space (XIONG et al., 2013).

In that context, there is a challenge of dealing with an unknown number of overlapping patterns that can be categorized as boundary patterns (TANG; GAO, 2007). These patterns are a subset of data points that lie between two classes. For instance, in a medical dataset, the boundary patterns belonging to the positive class could refer to a subgroup of the population that has recently contracted a specific disease and is still in the early stages of that condition. On the other hand, the remaining boundary patterns of the negative class may indicate a population that is prone to the disease but has not yet developed any symptoms. It is crucial to pay particular attention to these boundary patterns.

1.2 Hypothesis

This study establishes the hypothesis to be defended:

Hypothesis 1: *The OCC performance can be improved by training a set of OCCs with subsets of data clustered by similarity.*

Hypothesis 2: *The performance of OCCs trained exclusively with positive data could be adversely impacted when introducing data that resemble negative instances. This introduction of negative data has the potential to directly interfere with the classifier's performance and undermine its effectiveness.*

1.3 Objectives

The objective of this research is to introduce a novel approach to improve the performance of one-class classifiers (OCC) using time series clustering by similarity. The proposed methodology involves clustering the training data based on the similarity of their time series, generating multiple OCCs for each class, and evaluating how this clustering affects the performance of the OCC.

Moreover, the study examines the effect of variable selection when forming subsets of the training data for clustering. The choice of variables can impact the clustering quality and the performance of the OCC, thus it is essential to evaluate different subsets of variables and their impact on the OCCs performance.

Thus, this research seeks to contribute to the existing literature on OCC by presenting a novel methodology that improves the classifier's performance through time series clustering by similarity and the careful selection of subsets based on available variables and the classifier performance, which can benefit and have practical applications in various fields that use multivariate time series, including, for example, industry, finance, healthcare, and cybersecurity (CHOI et al., 2021).

Furthermore, a secondary study examines the effect of time-shift labels to mitigate the sample overlapping problem to improve the performance of OCC.

The specific objectives are:

- Review the literature about one-class classifiers (OCC)
- Develop a new method for improving the performance of one-class classifiers (OCC);
- Select data bases to apply the methodology;
- Investigate the effectiveness of time-shift labels to improve the performance of OCC;

- Utilize the technique of clustering time series data based on their similarity;
- Evaluate the similarity clustering effects on the performance of the OCC;
- Compare and validate the clustering model results with a baseline model;
- Analyze and evaluate new model fusion approaches.

1.4 Thesis Structure

The remainder of this thesis is organized in the following structure. In Chapter 2 information about One-class classification will be presented. In Chapter 3 analysis techniques will be described time series clustering. In Chapter 4 ensembles of One-class classifiers are present. The proposed methodology is presented in Chapter 5. Results and important observations will be discussed in Chapter 6. Finally, in Chapter 7, the thesis will be concluded with the next steps for carrying out the Thesis Defense.

2 One-class Classification

One-class classification (OCC) is a machine learning algorithm used for binary classification problems where only one class is present in the training data, and the goal is to identify whether a new sample belongs to the same class or is an outlier. In the context of time series data, a one-class classifier can be used to detect unusual behavior in a time-dependent sequence of observations, for example, in stock prices, network traffic, or sensor readings. The algorithm usually models the normal behavior of the time series and flags any instances that deviate from the expected pattern as anomalies. OCC can handle imbalanced data, where the normal class is much more prevalent than the deviant instances. In such cases, traditional binary classification algorithms, such as logistic regression or support vector machines, may not perform well because they are vulnerable to being polarized by the majority class (SELIYA; ZADEH; KHOSHGOFTAAR, 2021). Compared with binary classifiers, the OCC is similar to binary classification, but its training methodology is the main distinguishing factor (KANG, 2022).

According to Khan & Madden (2014), one-class classifiers can be classified into three main categories: density methods, boundary methods, and reconstruction methods.

Density methods are used to compute the probability density function (PDF) of the target class data. This method involves determining the density of the majority class in particular regions of the feature space, which helps to identify whether a test instance originates from a high-density or a low-density area within the majority class.

Boundary methods, on the other hand, attempt to create a boundary that separates the target class from other classes in the data. Boundary methods require less training data than density methods (MOULTON, 2018). During testing, a test instance is evaluated concerning the boundary. If the instance is on the same side of the boundary as the target class, it is classified as belonging to the target class. Within the category of boundary methods, Support Vector Data Descriptor (SVDD) (TAX; DUIN, 2004), One-class Support Vector Machine (OCSVM) (PLATT et al., 1999), and Nearest Neighbor Method (TAX; DUIN, 2000) stand out as remarkable examples (FRAGOSO et al., 2021).

Finally, reconstruction methods attempt to reconstruct the target class data by using a model that represents the behavior of the target class. During testing, a test instance is evaluated by comparing it with the model. A threshold error is then used to determine whether a test instance belongs to the target class. If the instance deviates from the model, it is classified as an anomaly. Self-organizing Maps and Learning Vector Quantization are classified as reconstruction methods (JOVANOVIĆ; HIKAWA, 2022). Another type of this approach is an autoencoder that transforms the input and then tries

to reconstruct it.

An Autoencoder Network is an unsupervised neural network that can be divided into two parts: encoder and decoder. The encoder maps the input data to a lower-dimensional representation and the decoder maps the lower-dimensional representation back to the original input data, attempting to reconstruct the original input data from the lower-dimensional representation. Since the objective of an autoencoder is to learn an encoding that captures the most important features of the input data, the reconstruction error tends to be smaller to data similar to that used in the training phase. Several types of autoencoders have been proposed in the literature. Among these types, the Long Short-Term Memory (LSTM) autoencoder refers to the autoencoder where both the encoder and the decoder are LSTM networks. Their advantage is their ability to handle long sequences of data, where traditional statistical models may not perform well. Additionally, LSTM autoencoders are capable of modeling complex dependencies between observations, making them well-suited for modeling complex time series data (NGUYEN et al., 2021).

Thus, there are several techniques used for OCC, including One-Class Support Vector Machines (OCSVM) and Autoencoder Networks (PERERA; OZA; PATEL, 2021). These techniques differ in their complexity and the level of prior knowledge required to implement them, but they all aim to capture the underlying structure and patterns in the target instances.

Sáez, Krawczyk & Woźniak (2016) presented the challenges of learning from imbalanced multi-class datasets, which are more complex than binary datasets. Krawczyk et al. (2018) used the OCC methods to transform a multi-class problem into several one-class problems. Although OCC is by no means superior to the usage of binary classifiers for decomposing multi-class problems, its properties are superior to tackle problems such as class imbalance, class noise, overlapping, or inner outliers. In addition, to consider the classes separately, data used for training can produce superior results if it is split into subsets used for training more than one classifier for the same class. This approach was considered in (KRAWCZYK; WOŹNIAK; CYGANEK, 2014), divided into three steps: data clustering, classifier training, and fusion of classifier outputs. This work is immersed in this context. First, we consider each class handled separately from the others. Second, for each class, sets of OCC are trained for subsets of the original data clustered using a similarity metric. Finally, the fusion of their outputs produces the desired classification.

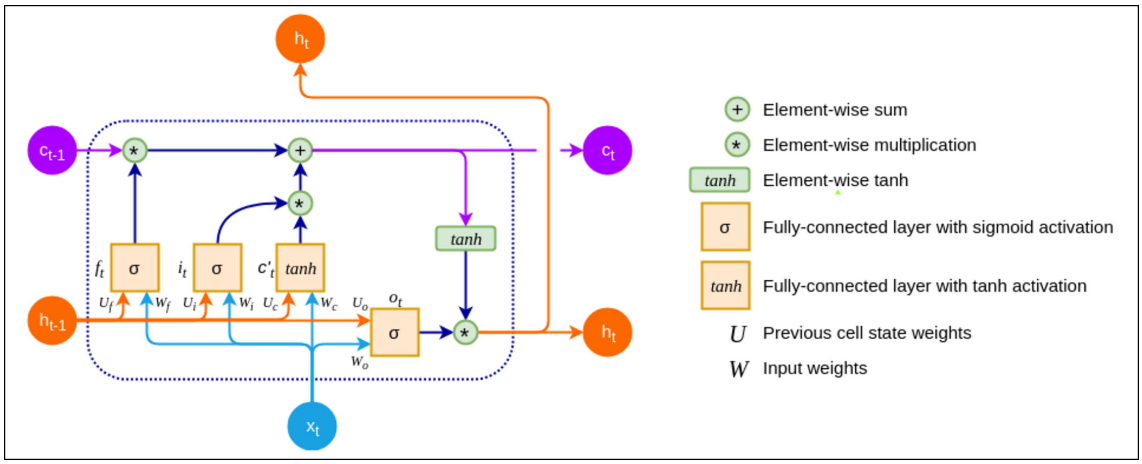
The next sections provide an overview of the OCC models employed in this work. Firstly, the LSTM model, a type of Recurrent Neural Network (RNN) that is well-suited for processing and predicting sequential data. Secondly, the LSTM-Autoencoder, a method that combines the power of LSTMs and autoencoders. Finally, One-class SVM (OCSVM).

2.1 Long Short Term Memory Networks - LSTM

LSTM is a RNN type that can maintain long-term dependencies between data points from previous time steps (HOCHREITER; SCHMIDHUBER, 1997). It consists of a chain of repeating modules of neural networks, each of which includes three control gates: the forget gate, the input gate, and the output gate. Each gate comprises a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layers produce values between 0 and 1, representing the fraction of input information that should be allowed through. When used for time series data, the LSTM reads a sequence of input vectors, where each vector represents m-dimensional readings for variables at a particular time instance (NGUYEN et al., 2021).

An LSTM cell internally has three modules that are responsible for controlling the flow of information, that is, they decide which information should be kept, transmitted, and discarded. These modules, called gates, are: the forget gate, input gate, and output gate. Moreover, the LSTM has an important component called cell state C_t, which is the LSTM memory at state t, and it aggregates data from the previous states. Cell state gives the model longer memory of past events, present in Figure 3.

Figure 3 – The LSTM Memory Cell Basic Architecture.



Fonte: Santos, Pereira & Schirru (2021)

LSTM reads a sequence of input vectors $x = \{x_1, x_2, \ldots, x_t\}$, where $x_t \in \mathbb{R}^d$ represents an d-dimensional vector of readings for d variables at time-instance t. Given the new information x_t in state t, firstly, the forget gate determines what old information should be forgotten or kept. A value within $[0, 1]$, say f_t, is computed by Equation 2.1. (NGUYEN et al., 2021)

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f), \tag{2.1}$$

where σ is the sigmoid function, h_{t-1} is the output in state $t-1$, W_f and b_f are the weight matrices and the bias of the forget gate, respectively.

Then, x_t is processed before storing in the cell state. The value i_t is determined in the input gate (Equation 2.2) along with a vector of candidate values $\widetilde{C}_t$ (Equation 2.3). These two parts will be combined to update the cell state C_t (Equation 2.4)

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i), \tag{2.2}$$

$$\widetilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c), \tag{2.3}$$

and

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t, \tag{2.4}$$

where $tanh$ is the hyperbolic tangent function, (W_i, b_i) and (W_c, b_c) are the weight matrices and the biases of input gate and cell state, respectively. The symbol $*$ corresponds to the element-wise product, formally known as the Hadamard product. Finally, the output gate is defined by

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o), \tag{2.5}$$

$$h_t = o_t * tanh(C_t), \tag{2.6}$$

where W_o and b_o are the weight matrix and the bias of the output gate, respectively. The output gate determines the output of the LSTM cell.

There are several derivatives of LSTM such as gated recurrent units (GRUs) that we have not employed since they typically give similar results. We conducted initial tests using vanilla LSTM, KNN, OCSVM, and LSTM Autoencoder and discovered that the latter was the most appropriate for the dataset. As a result, we chose to use it as a baseline for our experiments.

2.2 LSTM Autoencoder

Autoencoder is an unsupervised neural network that aims to learn the best encoding-decoding scheme from data (NGUYEN et al., 2021). An Autoencoder is trained in order to minimize the distance between input and output data (PERERA; OZA; PATEL, 2021).

Given an input $x \in \mathbf{R}^m$, the encoder compresses it to generate an encoded representation $z = e(x) \in \mathbf{R}^n$. The output $\hat{x} = d(z) \in \mathbf{R}^m$ is generated by the decoder, which utilizes the encoded representation to reconstruct the original input. The autoencoder is trained by minimizing the reconstruction error, given by Equation 2.7

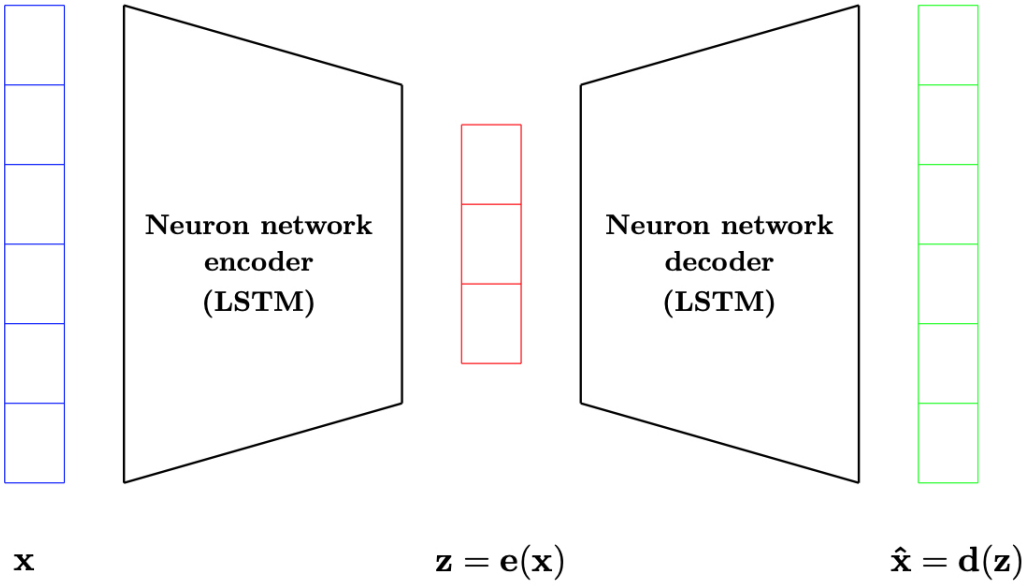
$$L = \frac{1}{2} \sum \|x - \hat{x}\|^2 \tag{2.7}$$

The primary objective of the autoencoder is not merely to duplicate the input as output. Instead, by limiting the size of the latent space to be smaller than the input, the autoencoder is compelled to grasp the most significant aspects of the training data, i.e., a

crucial aspect of the autoencoder's design is that it compresses the data dimensions while preserving the fundamental information of the data structure.

Several types of autoencoders have been proposed in the literature. Among these types, the LSTM Autoencoder refers to the autoencoder that both the encoder and the decoder are the LSTM network. Figure 4 presents a representation. LSTM is suitable for time series prediction or anomaly detection due to the ability to learn patterns in data over long sequences (NGUYEN et al., 2021).

Figure 4 – LSTM Autoencoder Representation.



Source: Nguyen et al. (2021)

2.3 One-class SVM

The OCSVM is a variant of the Support Vector Machine (SVM) algorithm that is specifically designed for one-class classification. During training, only positively labeled data is used, and the hyperplane corresponding to the negative class is set to be the origin of the coordinate system. OCSVM utilizes an appropriate feature space to map the data and identify a hyperplane that can effectively separate the majority class instances from the origin (MOULTON, 2018).

When there are samples of a single class, OCSVM can map the input vector to a high-dimensional space through nonlinearity, and this case can be treated as a special binary classification problem. Here, the mapping function is set to Φ, and the high-dimensional space is represented by Θ (MA et al., 2021).

Given a data set $\{x_1, x_2, \ldots, x_i, \ldots, x_N\}, x_i \in \mathbb{R}^d$, the purpose of OCSVM is to find a hyperplane $f(x) = w^T x + b$ in the high-dimensional space, and then maximize

the Euclidean distance $b/\|w\|$ from the origin to the hyperplane, where w represents the normal vector of the hyperplane, and b represents the intercept of the hyperplane. To ease certain restrictions, a slack variable ξ_i is introduced, and the quadratic optimization problem is shown in Equation 2.8 (MA et al., 2021)

$$\begin{cases} \min_{w,\zeta,b} \|w\|^2/2 + \sum_{i=1}^{N} \xi_i/vN - b \\ \text{s.t. } (w \cdot \Phi(x_i)) \geq b - \xi_i, \quad \xi_i \geq 0, \quad \text{i} = 1, 2, \ldots, N \end{cases} \tag{2.8}$$

where parameter $v \in (0, 1]$ is used to regulate the degree of relaxation of the target. To solve Equation 2.8, a Lagrange function is introduced, as shown in Equation 2.9

$$L(w, \xi, b, \theta, \gamma) = \|w\|^2/2 + \sum_{i=1}^{N} \xi_i/vN - b - \tag{2.9}$$

$$\sum_{i=1}^{N} \xi_i \cdot \theta_i - \sum_{i=1}^{N} [(w \cdot \Phi(x_i)) - b + \xi_i] \cdot \gamma_i \tag{2.10}$$

where θ_i and γ_i are both non-negative. Due to the nonlinear mapping from a low-dimensional to a high-dimensional space, according to the kernel function theory, the inner product of the mapped vector is equal to the kernel function of the original space (MA et al., 2021). Given a valid kernel function k, we have

$$(\Phi(x_i), \Phi(x_j)) = k(x_i, x_j). \tag{2.11}$$

Since we have a convex quadratic programming problem, the duality condition is met, as shown in Equation 2.13

$$\begin{cases} w = \sum_{i=1}^{N} \gamma_i \cdot \Phi(x_i) \\ \gamma_i = 1/vN - \theta \\ \sum_{i=1}^{N} \gamma_i = 1 \end{cases} \tag{2.12}$$

$$\downarrow$$

$$\begin{cases} \min_{\gamma_i} \sum_{i=1}^{N} \sum_{i=1}^{N} \gamma_i \cdot \gamma_j \cdot k(x_i, x_j)/2 \\ \text{s.t. } 0 \leq \gamma_i \leq 1/vN, \quad \sum_{i=1}^{N} \gamma_i = 1 \end{cases} \tag{2.13}$$

In the Lagrange function, b can be obtained arbitrarily within $0 \leq \gamma_i \leq 1/vN$. The final decision function is shown in Equation 2.14

$$\text{f}(x) = \text{sgn}\left(\sum_{i=1}^{N} \gamma_i k(x_i, x) - \sum_{i=1}^{N} \gamma_i k(x_i, x_{\text{j}})\right) \tag{2.14}$$

3 Time Series Clustering

The unsupervised learning method known as clustering is widely utilized. Its purpose in time series clustering is to identify similar objects in unlabeled data based on common features, resulting in a grouped structure where data within each cluster share similarities while data in other clusters are dissimilar. Designing effective and efficient clustering and classification algorithms for time series data is particularly challenging due to the presence of noise, high dimensionality, and high feature correlation, which are typically present at a level much higher than in non-temporal data. Therefore, several algorithms have been improved specifically to address the challenges of clustering time series data (JAVED; LEE; RIZZO, 2020).

3.1 Families of Time Series Clustering

There are generally three categories of time series clustering, as outlined by Aghabozorgi, Shirkhorshidi & Wah (2015). The first category is whole time series clustering, which involves clustering a set of individual time series based on their similarities to group them together. The second category is subsequence clustering, in which the time series data is divided at certain intervals using a sliding window technique, and the clustering is performed on the extracted subsequences of the time series (ALI et al., 2019). The third category involves clustering time points based on their temporal proximity and the similarity of their corresponding values. This method shares similarities with time series segmentation, but it diverges in that not all data points are required to be assigned to clusters; certain points are regarded as noise (AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015).

3.2 Time Series Similarity Measures

Similarity measures play an important role in many multivariate methods to determine (dis)similarity between objects characterized by numeric vectors (KARACA et al., 2022). Commonly used similarity measures for datasets with quantitative variables are the Euclidean distance and the Manhattan distance (WARRENS, 2016). However, both distance metrics may not be efficient in dealing with drifts in phase or amplitude. Due to the fixed mapping between points in two-time series, these distance measures become highly susceptible to noise and temporal misalignments (DING et al., 2008). As a result, they cannot manage localized time shifts and treat time series with different lengths.

As a consequence, several distance measures have been developed to specify similarities between time series. The most commonly used distance measurement methods for time series data include the Hausdorff distance, modified Hausdorff (MODH), HMM-based distance, Dynamic Time Warping (DTW), Euclidean distance, Euclidean distance in a PCA subspace, and Longest Common Sub-Sequence (LCSS) (AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015).

DTW is a similarity measure between time series that seeks the temporal alignment that minimizes Euclidean distance between aligned series. Many alternatives to DTW have been proposed, however, (WANG et al., 2013) evaluated eight different distance measures on 38 data sets and found none significantly better than DTW. The DTW measure cannot be considered a distance, since it does not satisfy the triangle inequality condition. This work considers $DTW(u_i, v_j)$ as the similarity measurement between the time series u_i and v_j using DTW.

Consider time series sequences represented by U and V with lengths n and m, respectively:

$$U = u_1, u_2, ..., u_i, ..., u_n. \tag{3.1}$$

$$V = v_1, v_2, ..., v_j, ..., v_m. \tag{3.2}$$

Before obtaining the optimal warping path, an $n \times m$ distance matrix is created where the elements are made up of $D(i, j) = d(u_i, v_j)$ which represents the distance between two points. Dynamic programming is employed to compute the cumulative distance matrix by applying a reiteration as follows:

$$\gamma(i, j) = d(u_i, v_j) + min\{\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1)\}, \tag{3.3}$$

where $\gamma(i, j)$ is a cumulative distance, $d(u_i, v_j)$ is the distance of the current cell and the rest of the formula is the minimum value of the adjacent elements of the respective cumulative distance. A path $W = \{w_1, w_2, ..., w_k, ..., w_c\}$ that minimizes the total cumulative distance between the two sequences is created to get the best alignment. Three conditions must be satisfied to build the warping path, which are boundary conditions, continuity, and monotonicity (ŁUCZAK, 2016):

$$\begin{cases} (i)\ \ w_1 = D(1,1) \ \text{ and } \ w_k = D(n,m) \\ (ii)\ \ w_k = D(i_k, j_k), \ \text{ and } \ w_{k+1} = D(i_{k+1,jk+1}), i_{k+1} - i_k \leq 1 \ \text{ and } \ j_{k+1} - j_k \leq 1, \\ (iii)\ \ i_{k+1} - i_k \geq 0 \ \text{ and } \ j_{k+1} - j_k \geq 0. \end{cases}$$

Therefore, the DTW measure is the path that minimizes the warping cost:

$$DTW(u, v) = min \sum_{k=1}^{c} w_k \tag{3.4}$$

3.3 Time Series Clustering Algorithms

Several clustering algorithms utilize raw time series data, while others employ dimensionality reduction methods before clustering the time series data. Clustering methods can generally be categorized into six groups: partitioning, hierarchical, grid-based, model-based, density-based clustering, and multi-step clustering algorithms (AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015).

The partitioning clustering approach involves dividing n unlabelled objects into k groups, such that each group has at least one object. One commonly utilized algorithm for partitioning clustering is k-means. Employed in this work, k-means uses a prototype for each cluster that represents the mean value of its objects. The primary objective of k-means clustering is to minimize the total distance, typically Euclidean distance, between all objects in a cluster from its cluster center. However, in the case of time series clustering, this can be a challenging issue and is not a trivial matter (NIENNATTRAKUL; RATANAMAHATANA, 2007).

Tavenard et al. (2020) combined DTW with k-means clustering (KEOGH; LIN, 2005) to form a clustering algorithm for time series data. In our approach, we consider m multivariate time series with order n. The DTW similarity is measured between each pair of time series, and the k-means algorithm is used to cluster similar time series into groups. This method aims to identify groups of time series that exhibit comparable patterns, regardless of variations in their lengths or dynamics.

The process of clustering with DTW and k-means typically involves the following steps:

1. Preprocessing data: Normalize the time series data to eliminate any variations in scale;
2. Calculate DTW similarity: Compute the DTW measure between each pair of time series;
3. Initialize k-means: Initialize the k-means algorithm with k cluster centers, where k is the number of clusters desired;
4. Assign time series to clusters: Assign each time series to the cluster with the most similar center, based on the DTW measure;
5. Update cluster centers: Update the cluster centers by taking the mean of all time series assigned to each cluster;
6. Repeat until convergence: Repeat steps 4 and 5 until the cluster centers no longer change or a maximum number of iterations is reached;

7. Output clusters: The final output of the algorithm is k clusters, each containing a similar time series.

3.4 Clustering Multivariate Time Series and Instances

Up until now, our clustering analysis has focused on individual time series, with the aim of identifying patterns and similarities within each series. However, we now need to cluster instances, which comprise multiple time series. The goal is to group instances based on the similarity of their constituent time series.

To achieve this, we must carefully consider which variables to include in our analysis. It is crucial to include only those variables that exhibit similarities across the different instances, as this will ensure that the resulting clusters are based on a meaningful set of variables. By adopting a selective approach, we can ensure that each cluster of instances includes only variables with similar behavior, thus maximizing data similarity in instances intra-clusters while minimizing the similarity in instances inter-clusters. This strategy allows us to capture the essence of the similarities and differences between different groups of instances, and to identify patterns and structures that might not be evident when analyzing individual time series.

Overall, clustering instances based on the similarity of their time series requires a thoughtful and strategic approach that takes into account the specific characteristics of the data and the goals of the analysis. By carefully selecting relevant variables and adopting a selective approach, we can obtain meaningful insights into the patterns and structures underlying complex multivariate time series data.

In this context of clustering instances, it is important to take a strategic approach that combines time series and identifies intersections between instances to generate clusters. One potential method for accomplishing this is to use an algorithm that combines all sets of time series, but the computational effort required for this approach would likely be infeasible. To address this challenge, was choose an association rules technique for data mining. Association rule induction is a powerful method used to find regularities in data trends. By induction of the association rules, sets of data instances that frequently appear together must be found. Such information is usually expressed in the form of rules. An association rule expresses an association between items or sets of items. However, only those association rules that are expressive and reliable are useful (AFLORI; CRAUS, 2007).

Its definition is as follows: Let $I = \{I_1, I_2, ..., I_m\}$ be a set of m binary attributes called items. D is a set of transactions; each transaction t is included in I. X is a set of items from I, t contains X. An associations rule is a pair $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, $X \cap Y = \varnothing$.

The description of the association rules has two concepts of support and confidence. Confidence of rule $X \rightarrow Y$ is c if $c\%$ of the transactions D that contain the set X, contain also the set Y. Support of rule $X \rightarrow Y$ is s if $s\%$ of transactions in D contains the set $X \cup Y$. They are formulated as follows:

$$c(X \rightarrow Y) = P(Y|X) \tag{3.5}$$

$$s(X \rightarrow Y) = P(X \cup Y) \tag{3.6}$$

where $c(.)$ and $s(.)$ represent confidence and support of association rules, respectively. If the preset minimum support thresholds are satisfied, then X and Y can form a valid association rule.

The Apriori algorithm is a most basic data mining technique used for association rule learning in transactional databases. Those associations can be refined as decision rules to be used. After considering many aspects, the Apriori algorithm was chosen to accomplish the association rules of the instances. The Apriori algorithm is an algorithm for mining frequent item sets of Boolean association rules. The so-called frequent item sets are all item sets whose support is greater than the minimum support threshold (MA et al., 2019). Given a support threshold s, the Apriori algorithm identifies the item sets that are subsets of at least s transactions in the database.

An example is shown in Table 1. Considering ten instances composed of three time series process variables, and the time series split into two similar clusters. A binary values matrix is used to represent instance (item set) and variables (items) in one subset, where each row represents the instance and the columns mean the corresponding process variable. Assuming that "1" represents that the variable is assigned in the corresponding instance to be used in the corresponding subset, otherwise it is represented with "0". The number of variables per instance is the sum of each row; the support of the variable is the sum of each column. From the binary matrix, it is clear which variables (item) are in the instance (item set) and which are not. Table 1 presents one example of the frequent individual items.

Defining support threshold s. Then calculate each support (sum of the columns). When getting support greater than or equal to the support threshold s, it obtains frequent subsets. This process is repeated for all combinations, one variable (order 1), two variables (order 2), and so on. In this example, were presented ten instances and three variables. The results of subsets are presented in Table 2, considering the support threshold as 40%. It means that, in this case, when one subset is selected the frequent set must have at least four from ten instances to be part of the subset result.

The same is done for cluster 2. Once the support threshold is defined, we get the frequent items obtaining frequent subsets greater than or equal to the support threshold. A binary values matrix is shown in Table 3, and Table 4 shows the results of the example.

Table 1 – Example Frequent Individual Items - Cluster 1.

Instance	X_1	X_2	X_3
0	1	1	1
1	1	0	1
2	1	0	1
3	1	1	0
4	0	0	0
5	0	0	0
6	0	1	0
7	0	1	0
8	0	0	0
9	1	0	1

Table 2 – Cluster 1 - Frequent Results.

Order	Frequent items
1	{'X_1', 'X_2', 'X_3'}
2	{'X_1 X_3'}
3	{}

Table 3 – Example Frequent Individual Items - Cluster 2.

Instance	X_1	X_2	X_3
0	0	0	0
1	0	1	0
2	0	1	0
3	0	0	0
4	0	0	1
5	1	1	1
6	0	0	0
7	1	0	1
8	0	1	1
9	0	0	0

Table 4 – Cluster 2 - Frequent Results.

Order	Frequent items
1	{'X_2', 'X_3'}
2	{}
3	{}

Each cluster has one binary matrix, but just for visualization purposes. Table 5 shows the cluster 1 matrix and cluster 2 in only one matrix, where 0 means missing values and it is not possible to use that information, 1 represents the variable associated with cluster 1, and 2 represents that the variable is assigned to cluster 2.

Table 5 – Example Frequent Individual Items - Cluster 1 and 2.

Instance	X_1	X_2	X_3
0	1	1	1
1	1	2	1
2	1	2	1
3	1	1	0
4	0	0	2
5	2	2	2
6	0	1	0
7	2	1	2
8	0	2	2
9	1	0	1

Thus, given a support threshold s, the Apriori algorithm determines the item sets that are subsets of at least s transactions in the database.

Thus, $C^1_{x_i,x_j}$ and $C^2_{x_i,x_j}$ are the clusters of instances containing the variables x_i and x_j, respectively, with more similarity that will be evaluated. This procedure follows until all n variables are considered, generating sets of instances with an increasing number of variables with more similarity. As the number of variables is increased, the number of instances resulting from operations of intersection in the clusters is reduced. A criterion based on classifier performance can be used to select the clusters and variables.

The Apriori algorithm is used to perform combinations and generate clusters of instances with an increasing number of variables, with the following steps:

1. Scan all instances to determine the support of each variable x_i. The support of x_i is the number of instances that contain x_i. Define $p = 2$;

2. Generate a set of candidates with p variables, which includes all possible combinations of p variables that have a support value greater than or equal to a specified minimum support threshold;

3. Scan the instances again to determine the support of each candidate of p variables. Any of these candidates that do not meet the minimum support threshold are discarded;

4. Increase p by 1 and repeat steps 2 and 3. Continue this process until p is equal to the number of variables.

5. Output all the frequent sets of variables that were found.

Thus, given a minimum support threshold s, the Apriori algorithm determines the subsets have at least s instances of total instances in the training database. In the application, it will be shown that similarity increases with clustering.

At the end of this step, we will have clusters of instances, each with a different subset of the time series, each one representing a sub-concept of the class. The result of this clustering process is a pool of classifiers that can be applied to different regions of the feature space. However, it is important to note that the decision about which classifiers to select to represent the class is a crucial step in the process. This selection and the fusion of the classifiers will be explored in detail within the methodology.

4 Ensembles of One-Class Classifiers

Time series data associated with a particular class typically exhibit similar patterns across different instances in controlled environments. However, in real-world data, the factors that affect the collected data may be unknown or difficult to measure, leading to variations in the behavior of time series that correspond to the same event.

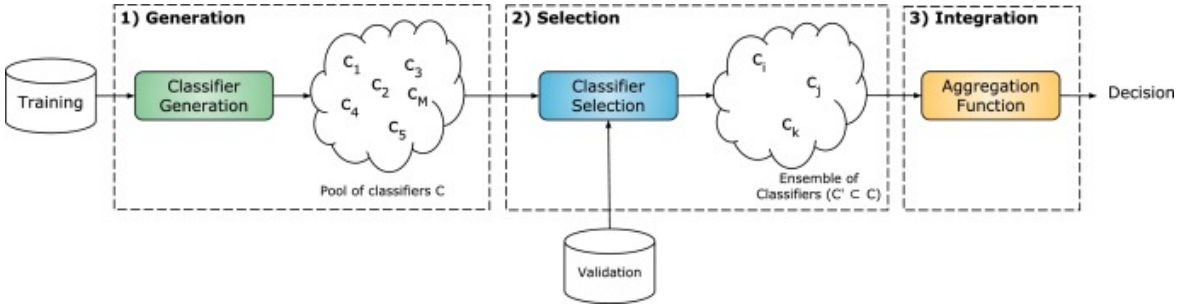
The challenge of associating the same class with data that exhibit different behaviors is particularly relevant in the context of the OCC problem. Training a single classifier using such data often leads to poor results due to this difficulty. To address this issue, recently, in the context of one-class problems, there has been a growing adoption of Multiple Classifier Systems (MSC).

MCS involves combining a group of classifiers to achieve better performance than any single classifier could achieve alone. The process of MCS typically consists of three steps: generation, selection, and fusion/aggregation. During the generation step, a pool of classifiers is created to obtain both accuracy and diversity among the classifiers. This step aims to form sets of classifiers that complement each other, resulting in a pool that contains competent classifiers for different regions of the feature space.

The objective of the selection phase is to identify the most capable classifiers from the available pool. Although it is feasible to create an MCS by bypassing this step and directly merging all classifiers in the pool, the benefits of implementing the selection phase are well-established (KRAWCZYK; WOŹNIAK; CYGANEK, 2014).

In the last phase, the outputs of the base experts are aggregated to give the final decision on the system (CRUZ; SABOURIN; CAVALCANTI, 2018). Figure 5 shows the three possible phases of an MCS system. Within that particular context, several works have been proposed for each of the three phases.

Figure 5 – The three possible phases of an MCS system.



Source: Cruz, Sabourin & Cavalcanti (2018)

In a study by Krawczyk, Woźniak & Cyganek (2014), One-class Support Vector

Machine (SVM) ensembles were created using a clustering-based approach. In this approach, the target class is segmented, and an OCC is trained for each cluster. Unlike other MCSs, this system does not involve a selection step, meaning that all classifiers in the pool are used to classify all test instances.

Krawczyk & Woźniak (2016) introduced a Dynamic Classifier Selection (DCS) approach for One-class Classification (OCC) in binary datasets. This method involves selecting a single classifier at runtime to classify each test instance. Specifically, the technique selects a single OCC from a pool of heterogeneous OCC models. The study demonstrated that dynamic selection techniques are effective in the context of OCC, owing to the generation of a diverse pool of classifiers during the training phase. However, it was designed only for learning in the absence of counterexamples, not for handling multi-class problems.

Krawczyk et al. (2018) proposed a dynamic ensemble selection (DES) system for removing non-competent classifiers before aggregating their outputs in the OCC-based decomposition. The method reassembles the original multi-class problem by combining the decisions of the selected one-class classifiers. The process is performed during the classification phase based on the neighborhood of a new example, allowing for the pre-selection of a subset of classifiers that are more likely to contribute towards the correct decision and reducing the complexity of the combination phase. The authors demonstrated that a reduced number of base classifiers may lead to better functioning of combiners.

Problems like this are considered by Sharma, Somayaji & Japkowicz (2018). The proposed approach takes into account the presence of sub-concepts within the target data, which can result in extreme levels of imbalance that make binary classifiers impractical. Additionally, the data available is often derived from complex distributions, which can lead to issues such as class overlap and multimodality that negatively impact the performance of OCC. The hypothesis presented is that these complexities are due to the presence of sub-concepts within the target data and that the performance of OCC can be significantly improved by utilizing domain-specific knowledge to identify and learn over these sub-concepts. Moulton (2018) used the sub-concept knowledge to improve one-class classifier performance applied to the data stream environment.

Fragoso et al. (2021) presented two methods for multi-class classification using one-class decomposition: One-class Dynamic Ensemble Selection for Multi-class problems (MODES) and Density-Based Dynamic Ensemble Selection (DBDES). Both techniques decompose a multiclass problem into several one-class problems, segment the training data of each class into clusters, and train an OCC for each cluster. The experiments indicated that the adoption of a clustering-based approach for the dynamic ensemble of OCCs selection improves the classification performance in the one-class decomposition scheme.

The effectiveness of this technique has been established, but its applicability to the

OCC in multivariate time series (MTS) remains unexplored. The presence of missing values and imbalanced data influenced by a variety of unknown factors make the implementation of this approach challenging in MTS scenarios.

In that context, we address the multi-class classification performance problem with multivariate time series in datasets with different behaviors for the same class, breaking the majority class into subsets. Also, the methodology considers the possibility of the data stream environment. The strategy adopted involves clustering data for training classifiers to improve their performance using instances with more similarity and considering also the lack of data and the available variables.

5 Methodology

In this chapter, the methodology is presented that addresses a multi-class classification problem on multivariate time series by modeling each class with a set of OCCs (MACHADO et al., 2024). In this work, the Long Short-Term Memory (LSTM) autoencoder is used as an OCC. The target class is defined as the class of data labeled with the fault to be detected and will be called the positive class. All other classes (normal or faulty data) will be considered the negative class. This way, one set of OCCs will be trained for each class of fault.

The strategy adopted consists of clustering data for training classifiers to improve their performance using data with more similarity and considering also the available variables.

In this work, the following definition of Multivariate Time Series (MTS) is adopted, which is similar to that used in Zhou & Chan (2015), Weng (2013), He et al. (2013), Vargas et al. (2019). The dataset DS considered here is a set of m multivariate time series (MTS) $S^i|i = \{1, 2, ..., m\}, \forall m \in \mathbb{Z}$, with $m > 1$, and is defined as $DS = \{S^1, S^2, ..., S^m\}$. Each MTS i is an instance, that is composed of a set of n univariate time series $x^i_j|j = \{1, 2, ..., n\}, \forall n \in \mathbb{Z}$, and $n > 1$ (also referenced just as a variable), and is defined as $S^i = \{x^i_1, x^i_2, ..., x^i_n\}$. Each variable j that composes an MTS i is an ordered temporal sequence of p_i observations taken at the time t ($x^i_{j,t}|t = \{1, 2, ..., p_i\}, \forall p_i \in \mathbb{Z}$, and $p_i > 1$). Thus, each MTS i is viewed in this work as a matrix defined as Equation 5.1

$$S^i = \{x^i_{1,1}, x^i_{2,1}, x^i_{n,1}; x^i_{1,2}, x^i_{2,2}, x^i_{n,2}; ...; x^i_{1,p_1}, x^i_{2,p_1}, ..., x^i_{n,p_1}\}. \tag{5.1}$$

All observations of each instance are labeled according to the class they belong to $c_i = \{c_1, c_2, \ldots, c_q\}$, where q is the number of classes in the dataset.

Note that all instances have a fixed number of variables n, but each instance can be composed of any quantity of observations p_i. It is also important to note that all variables of an instance i have a fixed number of observations p_i, but considering the possibility of missing values, the number of observations available in each variable can be less.

5.1 Pre-Processing

In many real-world monitoring applications, it is common to have missing data, caused by hostile working environments, poor installation, or communication problems that prevent the data from sensors from arriving at the monitoring systems.

Initially, the division of instances of data for training, validation, and testing is performed for each class. Then, a pre-processing step is performed to clear out raw samples and even variables that the algorithms cannot use (unassigned variable), downsample variables with a higher sampling frequency (for use of the same sample rate on training, validation, and test phase), and standardize all instances of data using the mean and the standard deviation of the training data.

A variable is said unassigned in a given instance if the number of frozen or missing samples exceeds a selected threshold. The instances with such a variable cannot be used if this variable is required. If the number of missing values is greater than zero but smaller than the missing values threshold, the values are replaced by linear interpolated values and the instance with such a variable can be used. Considering an instance, one variable is called frozen when its assigned values are immutable for some time. To verify that in each variable, we count unique apparitions of values. Get the most frequently occurring element and verify the proportion of this element related to the entire signal. If the most frequently occurring element is greater than a predefined threshold, this variable is considered frozen. The process is performed for all instances of all classes of the datasets.

5.2 Time Series Clustering by Similarity with DBA and k-means

The k-means algorithm is a classical clustering method that partitions the data into k clusters based on the data's distance from the centroid of each cluster (which is equal to the mean of all data assigned to that cluster), where every data point is assigned uniquely to only one of the clusters (BISHOP, 2006). The Euclidean distance is usually the standard measure.

However, as mentioned previously, Euclidean distance is not adequate for time series. An alternative is to use DTW, which is an elastic distance measure used to find an optimal alignment between two time series using certain restrictions (MAHARAJ; D'URSO; CAIADO, 2019). This similarity measure is likely the most commonly employed for analyzing sequences. However, its employability in methods such as k-means was limited due to the lack of a suitable associated averaging technique. To solve the problems of existing pairwise averaging methods, Petitjean, Ketterlin & Gançarski (2011) introduced a global averaging strategy called DTW Barycenter Averaging (DBA). They have successfully applied the k-means algorithm with DTW and DBA with different tests on standard datasets, showing that the DBA achieves robust behavior and better results on all tested datasets. Therefore, the centroid of each cluster is obtained by DBA which is used to measure the similarity between the time series and each cluster. The time series clustering applied here is based on this approach.

In addition, it is considered real data with different lengths and missing variables,

which prevents the application of many time series clustering algorithms directly to multivariate time series.

This step aims to cluster by similarity each time series from training instances. Consider, after pre-processing, a set of m instances $DS = \{S^1, ..S^m\}$ and a set of n variables $(x_1, x_2, ..., x_n)$ associated with each instance. Considering the training set, let x_i^h be the i-th variable from the h-th instance, where $i = 1, ..., n$ and $h = 1, ..., m$. DTW Barycenter Averaging (DBA) and k-means are applied to each i-th variable (time series) from all available instances $h = 1, ..., m$. The number of groups k in the k-means algorithm is a hyperparameter to be selected, and should consider the amount of data in the dataset: the greater the number of groups, the smaller the amount of data in each group. In this work, we consider that each time series x_i is partitioned into 2 clusters ($k = 2$), that is, $C_{x_i}^1$ and $C_{x_i}^2$. This partition will increase the similarity of the variables within the groups. The similarity of the time series x_i in each of these clusters is computed by DBA and is given by $DBA(C_{x_i}^j), j = 1, 2$. Therefore, the n variables will result in $2n$ clusters.

This procedure ensures that each variable x_i, one at a time, is grouped to increase the similarity. However, the increase in similarity is required for all variables together, since they are all used to train each OCC, therefore, another step is required to select the clusters of instances with sets of variables with higher similarity.

5.3 Clustering Instances and Multivariate Time Series

Consider the set of instances $DS = \{S^1, .., S^m\}$ and one of the variables x_i present in each instance. In this work, we consider that each time series x_i is partitioned into 2 clusters ($k = 2$), say $C_{x_i}^1$ and $C_{x_i}^2$.

If the variable x_i is unassigned in instance S^q, then this instance does not belong either to cluster $C_{x_i}^1$ nor $C_{x_i}^2$, i.e., $C_{x_i}^1 \cup C_{x_i}^2 \subset DS$. If now we consider two variables x_i and x_j, the clusters $C_{x_j}^1$, $C_{x_j}^2$, $C_{x_i}^1$, $C_{x_i}^2$ should be evaluated. We now obtain $C_{x_i,x_j}^1 = C_{x_j}^1 \cap C_{x_i}^1$ and $C_{x_i,x_j}^2 = C_{x_j}^2 \cap C_{x_i}^2$. The other subsets $C_{x_i,x_j}^{12} = C_{x_j}^1 \cap C_{x_i}^2$ and $C_{x_i,x_j}^{21} = C_{x_j}^2 \cap C_{x_i}^1$ can also be obtained to further increase the similarity. However, the complexity is increased and the amount of data in the clusters is decreased, and for these reasons, they are not explored in this work.

Thus, C_{x_i,x_j}^1 and C_{x_i,x_j}^2 are the clusters of instances containing the variables x_i and x_j, respectively, with more similarity that will be evaluated. This procedure follows until all n variables are considered, generating sets of instances with an increasing number of variables with more similarity. As the number of variables is increased, the number of instances resulting from operations of intersection in the clusters is reduced. A criterion based on classifier performance can be used to select the clusters and variables.

The Apriori algorithm (MA et al., 2019) is used to perform combinations and generate clusters of instances with an increasing number of variables, with the following steps:

1. Scan all instances to determine the support of each variable x_i. The support of x_i is the number of instances that contain x_i.

2. Define $p = 2$; Generate a set of candidates with p variables, which includes all possible combinations of p variables that have a support value greater than or equal to a specified minimum support threshold A are considered;

3. Scan the instances again to determine the support of each candidate of p variables. Any of these candidates that do not meet the minimum support threshold is discarded;

4. Increase p by 1 and repeat steps 2 and 3. Continue this process until p is equal to the number of variables;

5. Output all the candidates (frequent sets of variables) that achieved the minimum support threshold.

Since the time series x_i are partitioned into 2 clusters, $C^1_{x_i}$ and $C^2_{x_i}$, these steps are applied separately for $C^1_{x_i}, i = 1, ..., n$ and for $C^2_{x_i}, i = 1, ..., n$.

To illustrate the application of Apriori to a dataset, consider the binary values matrix shown in Table 6 used to represent 10 instances and 2 variables (x_1 and x_2) that resulted from pre-processing and time series clustering. We consider in this example the first cluster of the variable x_1, $C^1_{x_1}$, which resulted from pre-processing and clustering the variable x_1. "0" indicates that x_1 was either unassigned in the instance corresponding to the line or belongs to the other cluster, $C^2_{x_1}$, not shown in the table. The same applies to x_2. Column $C^1_{x_1,x_2}$ is the intersection of $C^1_{x_1}$ and $C^1_{x_2}$. Thus, $C^1_{x_1}$ is a subset of the original 10 instances, associated with the analysis of variable x_1. The support of $C^1_{x_1}$ is 60%, meaning that 60% of the instances could be used if this subset is selected and only variable x_1 is used. If one wants to use both x_1 and x_2 for modeling, the support of the subset $C^1_{x_1} \cap C^1_{x_2}$ is 20%. The other subsets, $C^1_{x_1} \cap C^2_{x_2}$, $C^2_{x_1} \cap C^1_{x_2}$, $C^2_{x_1} \cap C^2_{x_2}$ can also be checked to improve support.

Times series clustering generates $2n$ clusters, $C^1_{x_i}$ and $C^2_{x_i}$, $i = 1, ..., n$. The Apriori algorithm does the combinations of the first cluster for all variables, $C^1_{x_i}$, $i = 1, ..., n$, and then for the second cluster for all variables, $C^2_{x_i}$, $i = 1, ..., n$. Results from Apriori the subsets of instances SS^i_1, $i = 1, ..., P_1$, and SS^i_2, $i = 1, ..., P_2$, as can be seen in Figure 7, that have more similar variables, and have also a different number of variables, ranging from 1 to n.

Table 6 – Example: Application of Apriori for clustering 1 group of 2 variables.

Cluster 1			
Instance	$C^1_{x_1}$	$C^1_{x_2}$	$C^1_{x_1,x_2}$
0	1	1	1
1	1	0	0
2	1	0	0
3	1	1	1
4	0	0	0
5	0	1	0
6	1	0	0
7	0	1	0
8	0	0	0
9	1	0	0
Support	60	40	20

The great advantage of using the Apriori algorithm is that it will directly ignore all pairs that contain any of the non-frequent items (support smaller than the threshold), and the number of combinations to scan is reduced significantly. The combination of clusters $C^1_{x_i}$ and $C^2_{x_i}$ would generate more subsets and the support could be improved. Due to the computational effort to handle all these combinations, it will not be considered in this work.

As the number of candidate variables increases, the support tends to decrease. On the other hand, a classification algorithm could have difficulty discriminating the classes in the data set if fewer variables are used. This decision comes in the next step when the classification performances obtained with the subsets will be evaluated. This will be discussed in the next section.

5.4 Training Classifiers

This step aims to train the subsets from those returned by the Apriori algorithm.

For every individual subset, $SS^i_1, i = 1, ..., P_1$ and SS^i_2, $i = 1, ..., P_2$, the given multivariate time series is divided into a sequence of smaller segments using a fixed-size sliding window of size L, that moves incrementally through the time series without overlaps. The data inside the sliding window is considered as a segment (w_1, w_2, ..., w_n), see Figure 6.

To construct the LSTM autoencoder model, the input data was transformed into a 3D tensor, organized based on samples, time steps, and variables, which means w_nxLxxn. This tensor structure was essential for building the LSTM autoencoder architecture and enabling effective data processing. In this case, the time steps or window size were configured to $L = 10$. At this point, each of these subsets becomes prepared for the training

phase.

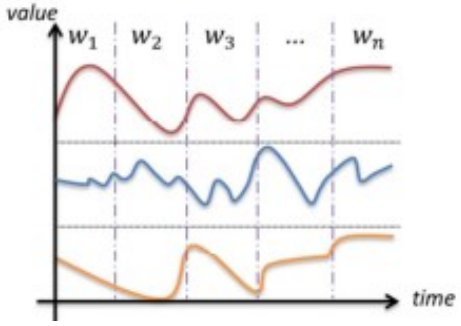


Figure 6 – Multivariate time series divided into a sequence of segments.

considering $SS_1^i, i = 1, ..., P_1$ and SS_2^i, $i = 1, ..., P_2$ those subsets are use to train LSTM models M_{1i}, $i = 1, ..., P_1$ and M_{2i}, $i = 1, ..., P_2$ respectively. After that, one model, from SS_1^P and SS_2^P, will be selected from the set of models, discussed in the next section.

5.5 Selection of Classifiers

This step aims to select the models with the best classification performance. Thus, considering the $SS_1^i, i = 1, ..., P_1$ used to create and train LSTM models M_{1i}, $i = 1, ..., P_1$, those models are evaluated according to the classification performance, and the best model M_1 are selected using a given performance metric and validation data. Similarly, each subset SS_2^i, $i = 1, ..., P_2$, are used to train LSTM models M_{2i}, $i = 1, ..., P_2$, and the model M_2 are selected with the best performance from M_{2i}, $i = 1, ..., P_2$. (see Figure 7).

We define C_1 as the dataset selected to train M_1 and C_2 as the dataset selected to train M_2. Once selected the models M_1, and M_2 the next step is to evaluate them with the test set and compare their performance against a baseline model M_b, to be discussed in the next section.

5.6 Evaluating the Effect of Clustering by Similarity

To quantify the improvement of the proposed methodology, and evaluate the effect of clustering, a baseline model M_b was trained considering all variables and instances without performing the clustering method, i.e., C^b.

The baseline dataset is given by $C^b = DS - C^0$, i.e., the original dataset DS excluding those instances with missing variables.

Finally, the comparison of models M_b, M_1, and M_2 will be performed using the same metric with the same test data (20% of the dataset).

Assuming that the increase in similarity of data used for training improves the performance of the one-class classifier applied to the dataset, the measurement of such

an increase in similarity is required to confirm this hypothesis. It is discuss now the measurement of similarity for the clusters of instances.

First, let us consider the baseline cluster of instances $C^b_{x_i} = DS - C^0_{x_i}$ in this case, for variable x_i. The application of k-means with DBA produces the two clusters $C^1_{x_i}$ and $C^2_{x_i}$ with increased similarity. The similarity of the cluster of instances $C^b_{x_i}$ is given by the average DTW similarity measure of all time series in this cluster to the center. Thus, Equations 5.2 and 5.3 can be used to measure the increase of similarity among the time series related to the variable x_i after clustering,

$$IS^{01}_{x_i} = 100(1 - \frac{DTW(C^1_{x_i})}{DTW(C^b_{x_i})}) \tag{5.2}$$

$$IS^{02}_{x_i} = 100(1 - \frac{DTW(C^2_{x_i})}{DTW(C^b_{x_i})}) \tag{5.3}$$

Second, a measurement of the similarity of the time series that belong to the cluster of instances used to train the one-class classifier considering all variables is required. The improvement in the similarity in the clusters of instances C_1 and C_2 is given by Equations 5.4 and 5.5

$$IS^{01} = 100(1 - \frac{\mu(DTW(C^1_{x_i}), i = 1, ..., n_1)}{\mu(DTW(C^b_{x_i}), i = 1, ..., n_0)}) \tag{5.4}$$

$$IS^{02} = 100(1 - \frac{\mu(DTW(C^2_{x_i}), i = 1, ..., n_2)}{\mu(DTW(C^b_{x_i}), i = 1, ..., n_0)}) \tag{5.5}$$

where $\mu(.)$ denotes the mean value.

These equations are used in the application to relate the increase in performance of the classifier with the increase in similarity (IS) of data used for training.

5.7 Overview of the Methodology

An overview of the proposed methodology is presented in Figure 7. After splitting data into training, validation, and test sets, the pre-filtering is performed to find frozen and missing values in the time series. The time series in the training data are clustered by similarity via DBA k-means generating the subsets of time series $C^1_{x_1}, C^1_{x_2}, ...$ and $C^2_{x_1}, C^2_{x_2}, ...$ These instances are then clustered using the Apriori algorithm to keep subsets of time series $S^1_1, S^1_2, ...$ and $S^2_1, S^2_2, ...$ that are more similar with different numbers of variables. LSTM models are trained with data labeled as faulty for each of these subsets of instances, and those with the highest performance (M_1 and M_2) on validation data are selected. The evaluation of the models is finally performed using test data.

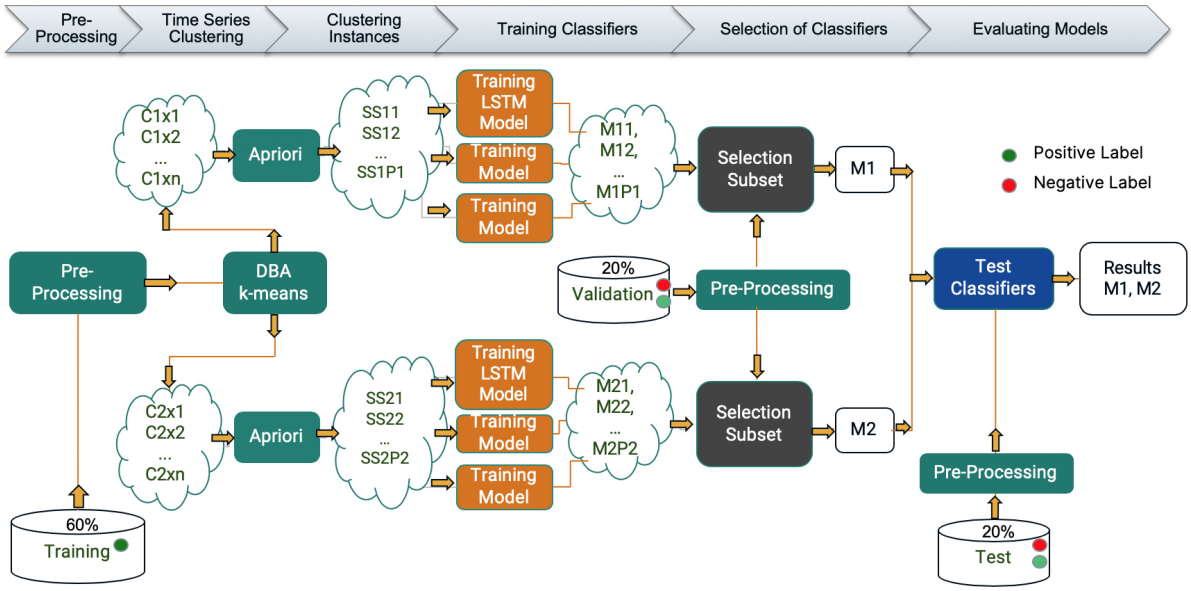


Figure 7 – Methodology flowchart.

5.8 Alternative Approaches to Fusion Classifiers

The fusion of classifier outputs proved to be a good alternative to increase the classifier performance (KRAWCZYK et al., 2018). According to (KUNCHEVA; BEZDEK; DUIN, 2001), selecting a subset from the pool may be an option for improving the classification performance instead of using all of the classifiers. This section presents alternatives to improve the proposed method through classifier fusion. Three approaches are proposed.

The first fusion approach is to combine the outputs of the selected models with the highest performance (M_1 and M_2) on validation data (KRAWCZYK; WOŹNIAK; CYGANEK, 2014), using the logical operation OR, and create the model $M3$. The performance of this additional model is also evaluated with the test dataset. This is not a classifier, but a combination of the output of models M_1 and M_2. The purpose of using M_3 is to make the fusion of the specialized knowledge acquired by each specific model. Figure 8 illustrates the flowchart of the approach #1.

The fusion approach #2 is based on sequential forward selection (SFS), in which features are sequentially added to an empty candidate set until further features do not decrease the criterion (KUMAR; MINZ, 2014).

The fitted models M11 to M1P2 are ranked according to their performance. The model with the highest performance, denoted as MO_0, is chosen as the starting point. Next, the performance of the fusion of MO_0 with each remaining model (M11 to M1P2), one each time, is performed. If the performance is increased, the model is retained, otherwise, it is discarded and the fusion with the next model is evaluated. This process continues

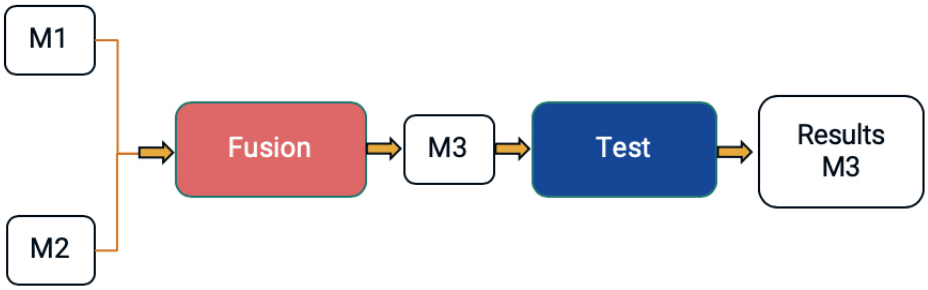


Figure 8 – Fusion Approach #1 flowchart.

until all remaining models have been evaluated in combination with MO_0 and the retained models.

As a sequential search algorithm, the fusion approach #2 adds models from a candidate subset while evaluating the criterion. Since an exhaustive comparison of the criterion value at all subsets is typically infeasible (depending on the size of P and the cost of objective calls), sequential searches move in only one direction, always growing the candidate set. Figure 9 presents the flowchart fusion approach #2.

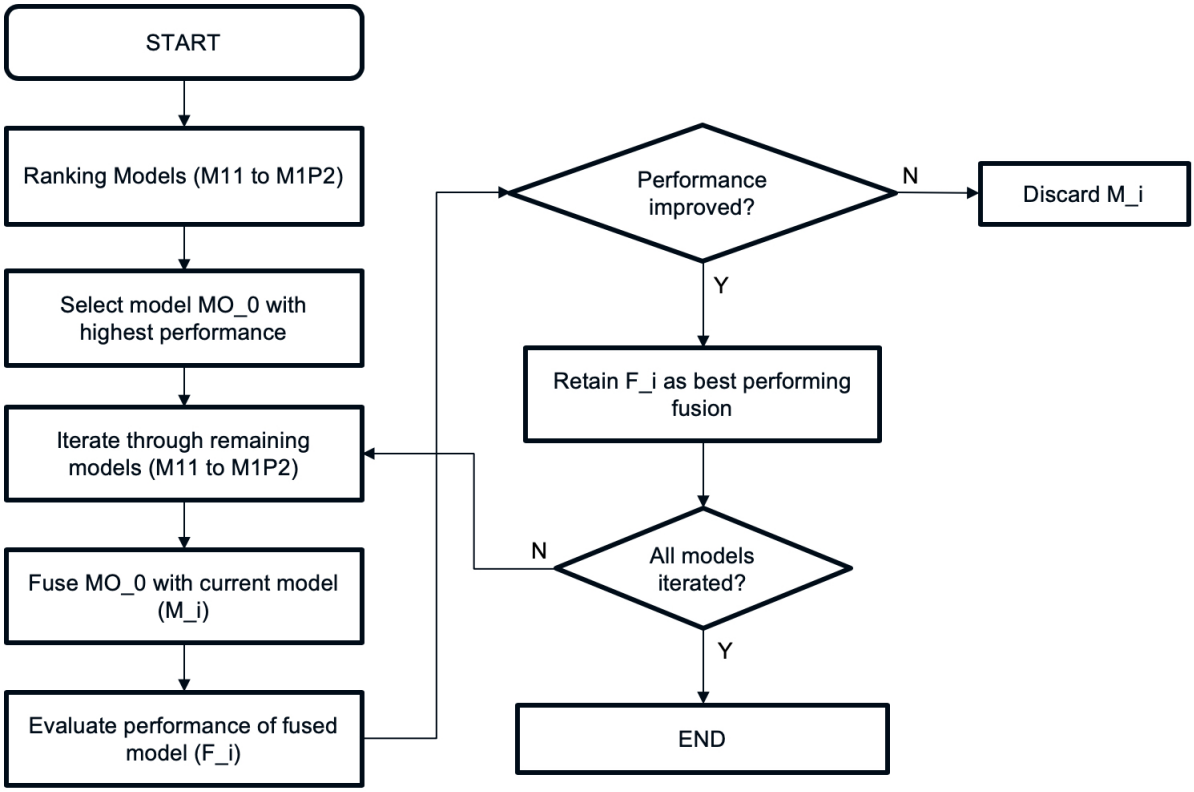


Figure 9 – Fusion Approach #2 flowchart.

The fusion approach #3 is illustrated by the flowchart in Figure 10. Initially, the selected models with the highest performance (M_1 and M_2) are chosen. Considering C_1

as the subset of instances selected to train M_1 and C_2 as the subset selected to train M_2, we use the complement of the subsets C_1 and C_2, C_1' and C_2', respectively, to train new models, i.e., the instances that were excluded from the clustering process to train the models M_1 and M_2.

A new model M_1' is trained with data from C_1' using the same variables selected for M_1. Similarly, a new model M_2' is trained with the data from C_2' using the same variables selected for M_2. Then, the fusion is performed between M_1 and M_1' and M_2 and M_2', creating two new classifiers: $M11'$ and $M22'$. Models $M11'$ and $M22'$ will also be evaluated with the test set and their performance will be compared.

The set of instances C_1' has a smaller number of instances if compared to C_1, and the same applies to C_2'. In addition, we have no information about the similarity of the selected variables from the instances of these complementary sets of instances C_1' and C_2'. Thus, although a poor performance is expected from the models M_1' and M_2' individually, their fusion to provide $M11'$ and $M22'$ can improve performance since neglected behaviors of the time series are included.

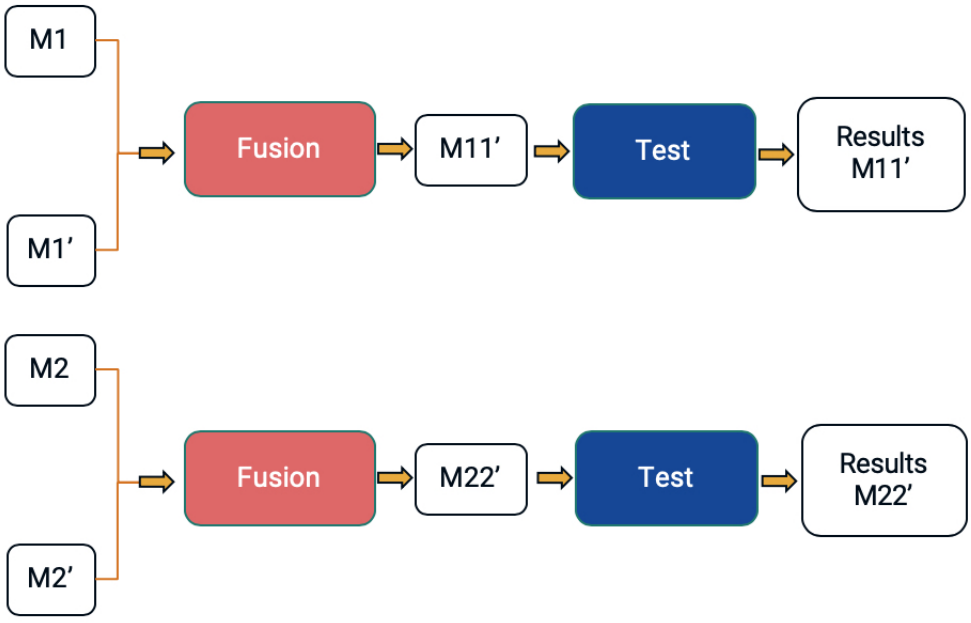


Figure 10 – Fusion Approach #3 flowchart.

5.9 Time-shift to Improve Performance of Classifiers

In the context of overlapping classes in multivariate time series, a second approach to improving the performance of the OCC system was developed. An analysis was performed using a data interval that falls between negative and positive classes. During this analysis, was observed that some data which was originally annotated as positive was very similar

to negative data because of the class overlapping. The study was conducted to analyze the influence of overlap classes on the performance of OCC. This interference could negatively impact the overall results of the OCC system.

The proposed solution to mitigate the sample overlapping involves a time-shift of the original positive labels only on the training data. This means that the labels of the training data are adjusted by a certain amount of time to reduce the overlap in the original labels.

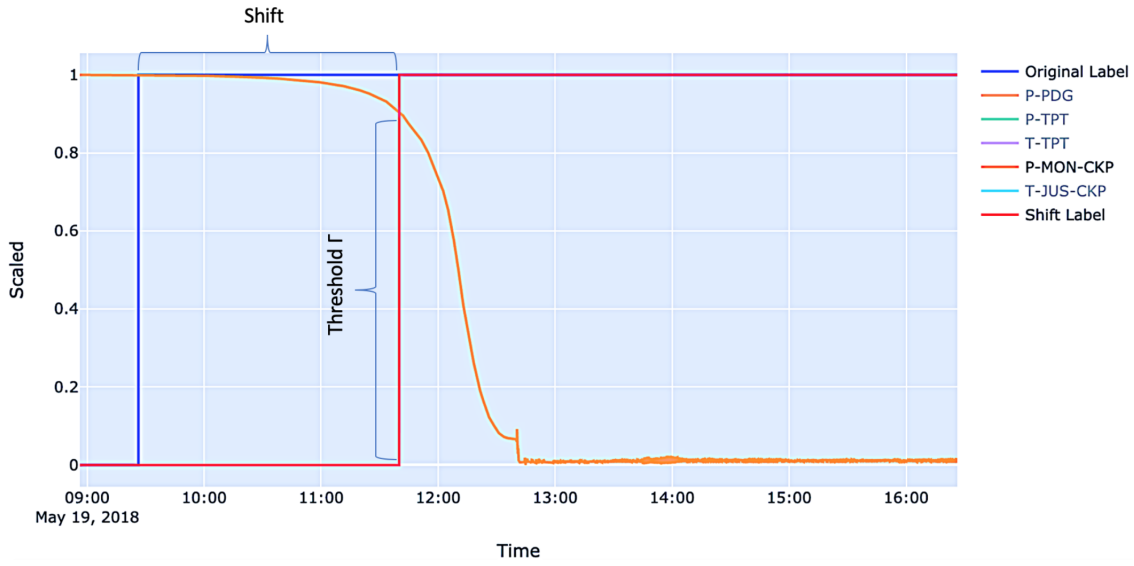
For each class, the labels for training instances are time-shifted. The time-shifted labels are then used to train the model. Once the training is complete, a test is performed to evaluate the performance of the model. A range of time-shift is evaluated. This test indicates the influence of the time-shifted labels on the classification performance. It is a search problem in the overlap region and there should be one effective time-shift adjustment that maximizes the performance of the classifier. To do this, some basis needs to be established.

Thus, we established a criterion for the time-shift of training data set labels. First, we select one of the input variables to apply the time-shift, as follows. Here, r is the variable labeling the samples, assuming values 0 and 1 for negative and positive samples, respectively; y_i $(i = 1, \ldots, 8)$ are the variables under analysis. They are normalized to vary from 0 (at time t_1) to 1 or from 1 (at time t_1) to 0, depending on the direction of the change. Index $I(i) = RMSE(r - y_i)$ is computed if $y_i(t_1) = 0$, and $I(i) = RMSE(r - |y_i - 1|)$ is calculated if $y_i(t_1) = 1$, where t_1 is the instant when r changes from 0 to 1. The variable y_i that results in the smallest I value is selected since it approaches the steady state quickly. This same selected variable will be used for time-shift of all instances of the class. Since the variables have different behaviors in the dataset for instances originating from different sources, the index I is the average of a set of values of indexes I computed for these instances.

Variables that do not change during the fault were excluded from this selection. Once selected the variable, we use the variation percentage of y_i in relation to normality to choose the time-shift for the labels, from now on we associate this variation with the threshold γ. Figure 11 presents an example of a time series overlap problem. Normality labels (blue line) are time-shifted until the selected variable y_i reaches the threshold $\gamma = 90\%$ of its value in relation to normality.

It is important to note that the success of this approach depends on the extent of the overlapping in the original labels and the effectiveness of the time-shift adjustments.

Figure 11 – Time-shift of the Training Data Labels.



Source: The Author

5.10 Metrics

One-class classifiers are evaluated using a validation and test dataset. Therefore, the testing procedure of one-class classifiers is analogous to binary classifiers. Considering a multi-class problem, the data are divided into positive and negative classes, where the positive is the class of interest (target class) of the one-class classifiers, and the negative class all other classes. Metrics for classification and event detection capacity were considered to evaluate the proposed method.

F1-score (harmonic mean of precision and recall) and specificity are adopted as classification metrics. Widely used, the F1-score allows for the mitigation of the effect of unbalanced data. Higher specificity has a lower false positive (FP) indication. This means that a test with 100% specificity will recognize all negative testing samples. Therefore, a positive test result would definitively rule in the presence of the fault, increasing confidence in the OCC prediction.

The classification metrics below are used in this work:

$$F1\_score = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{5.6}$$

$$Precision = \frac{TP}{TP + FP}, \tag{5.7}$$

$$Recall = \frac{TP}{TP + FN}, \tag{5.8}$$

$$Specificity = \frac{TN}{TN + FP}, \tag{5.9}$$

where TP and TN are samples correctly classified as positive and negative, respectively; FP and FN are samples incorrectly classified as positive and negative, respectively.

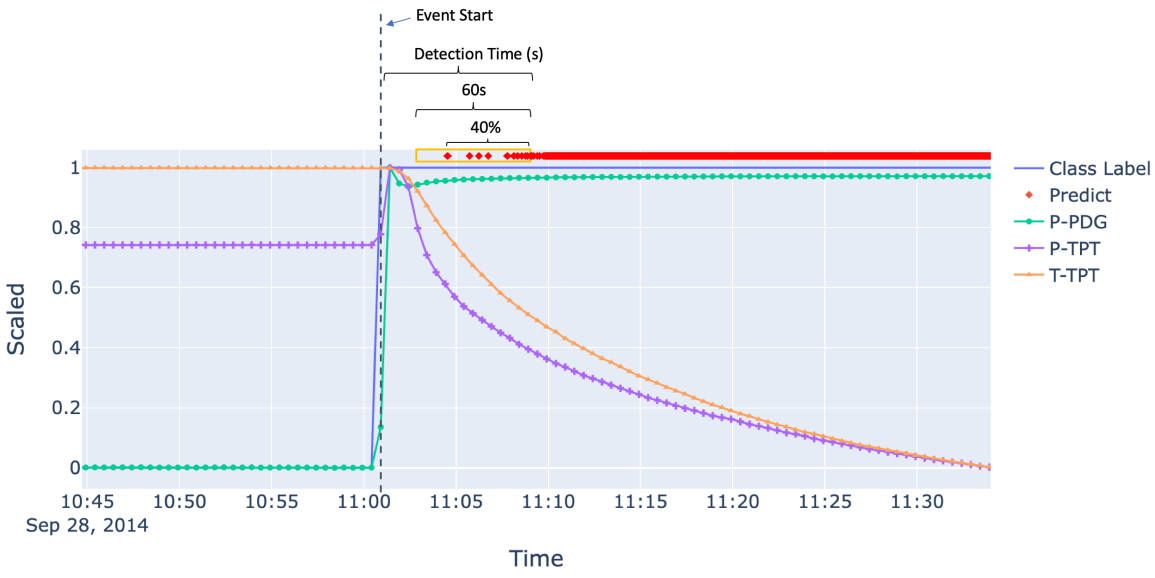
5.10.1 Detection Time Metric

The detection time metric was utilized in conjunction with the label time-shift method to evaluate the model's capability in detecting failures during the early stages, even when there is a shift in the labels during the training phase.

In time series classification, the correct identification of a fault is as important as quick fault detection. To measure detection time, a sliding window with $L = 60$ samples is used. If at least 40% of the samples in this window are identified as faulty, the detection time is the timestamp of the first faulty sample in this window. The detection time metric is the average detection time of all instances of the test set.

Figure 12 exemplifies how this metric works using an instance of class 2, with the model prediction indicated in red and the actual classification in blue. The yellow sliding window stops when the model identified 40% of the faulty data, and the detection time is the time of the first sample of this window.

Figure 12 – Illustration of Time Detection Metric.



Source: The Author

5.10.2 Event Detection Capacity

Instances Identification Percentage (IIP) measures the capacity to detect the fault event at one target class instance. It is only calculated when the model is tested against instances of the same class of the target class. We recall that all instances of the modeled event have the first-time interval with normal samples as negative, followed by a time interval with faulty positive samples (see Figure 13). This metric is the percentage of the total instances of classes in the test set in which the classifier was able to identify positive and negative samples in one instance, which means, both phases: the initial normal phase preceding the fault and the fault phase. Thus, the model needs to correctly identify at least one fault sample and at least 10% of normal samples for the instance to be included as identified.

An IIP value of 1.00 indicates that the model identifies the occurrence of normality and fault in all target classes tested instances. This metric prevents the inclusion of those instances whose model output only indicates faulty samples for the target class.

5.10.3 Performance Classification Metric

The F1-score, Specificity, and IIP (Instances Identification Percentage) are performance metrics and were used in Machado et al. (2022), Marins et al. (2021) and Turan & Jäschke (2021) to evaluate the performance of classifiers applied to the same dataset used in this work. Care must be taken when comparing the performance of one-class and multi-class classifiers since data used for training and testing in these methods are quite different (HEMPSTALK; EIBE, 2008). It was observed that good results measured by F1-score or only by specificity sometimes hide some weaknesses, like the inability to discriminate fault and normality in the tested instances measured by IIP. Each one of these metrics has particular benefits but, to evaluate and select the models in a simple way, only one metric must be adopted. Besides, only one metric should be used to select the classifier with the best performance.

In this work, an average metric μ_M is considered as the performance classification metric to evaluate the models:

$$\mu_M = \frac{F1 + Specificity + IIP}{3}. \tag{5.10}$$

The metrics used to compute μ_M are all important to ensure the proposed methods will provide good results in real applications. The three individual metrics (F1-score, Specificity, and IIP) are also shown in the results (Section 6.1.1.6) to clarify the composition of μ_M.

6 Applications and Results

The methods proposed in this work were originally developed for the 3W dataset (MACHADO et al., 2022), (MACHADO et al., 2024). It was later realized that they could be applied to broader scenarios. In this chapter, the application to the 3W dataset is initially shown, exploring the improvements derived from time series clustering and label time-shift. Later, the time series clustering is applied to data used for condition monitoring of a hydraulic system (ICM dataset), a public dataset composed of 17 variables generated during supervised tests. In contrast to the 3W dataset, the ICM dataset does not contain multiple labels per instance, rendering the application of the label time-shift method unnecessary. Another characteristic of the ICM dataset is that all variables are consistently available, while the faults are introduced simultaneously, representing a different challenge. The one-class classifier evaluated for time series clustering was the LSTM Autoencoder, while both the LSTM Autoencoder and OCSVM were employed for the time-shift label method.

6.1 3W Application

The 3W dataset was proposed by Vargas et al. (2019) and is composed of multivariate time series obtained from offshore naturally flowing wells, containing data under normal operation and data associated with eight kinds of undesirable events (classes) characterized by eight process variables. Every instance of each class is represented by a comma-separated values (CSV) file, according to their source: real, simulated, and hand-drawn. Real events were extracted from a plant information system. Simulated data were obtained from an oil and gas simulation software called OLGA (SCHLUMBERGER, 2014), using common parameters for naturally flowing wells. Hand-drawn instances include experts' knowledge of time series characterizing the undesirable events considered. Table 7 summarizes the instances of all classes of the dataset.

Table 7 – Quantities of Events that Compose the 3W Dataset.

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

In this work, the proposed methodology was applied to this dataset for automatically detecting and classifying different event types as follows:

1. **Class 1 - Abrupt BSW Increase**

   Marked by an elevation in Basic Sediment and Water (BSW), which is described as the proportion between the liquid flow rates and water and sediment. Its effects pertain to flow assurance, decreased oil production, oil lifting, scaling, industrial plant processing, and the recovery factor.

2. **Class 2 - Spurious DHSV Closure**

   Specified by the spurious closing of Downhole Safety Valves (DHSV). These valves are intended to shut down wells in the event of a physical disconnection between the production unit and the well or surface equipment failure. They are placed within the production tubing of wells and their closure can be difficult to detect from the surface. Therefore, identifying the event promptly can help minimize the loss of production.

3. **Class 3 - Severe Slugging**

   Characterized by periodic signals in production line sensors, which can stress or even damage well and/or processing plant equipment.

4. **Class 4 - Flow Instability**

   Characterized by irregular, relevant changes in at least one of the production line signals.

5. **Class 5 - Rapid Productivity Loss**

   Characterized by changes to the properties of naturally flowing wells, unprofitable production, and production reduction or ceasing. Prompt detection of this event may allow changing the operating point of the well to maintain its productivity.

6. **Class 6 - Quick PCK Restriction**

   Characterized by the spurious operation of Production Choke (PCK) valves used for surface control of well flow.

7. **Class 8 - Hydrates in Production Lines**

   The occurrence of hydrate formation, which is the development of crystal compounds resulting from the combination of natural gas and water, resembling ice, is a significant challenge in the oil industry. The obstruction of production tubing due to hydrate can result in days or even weeks of production shutdown.

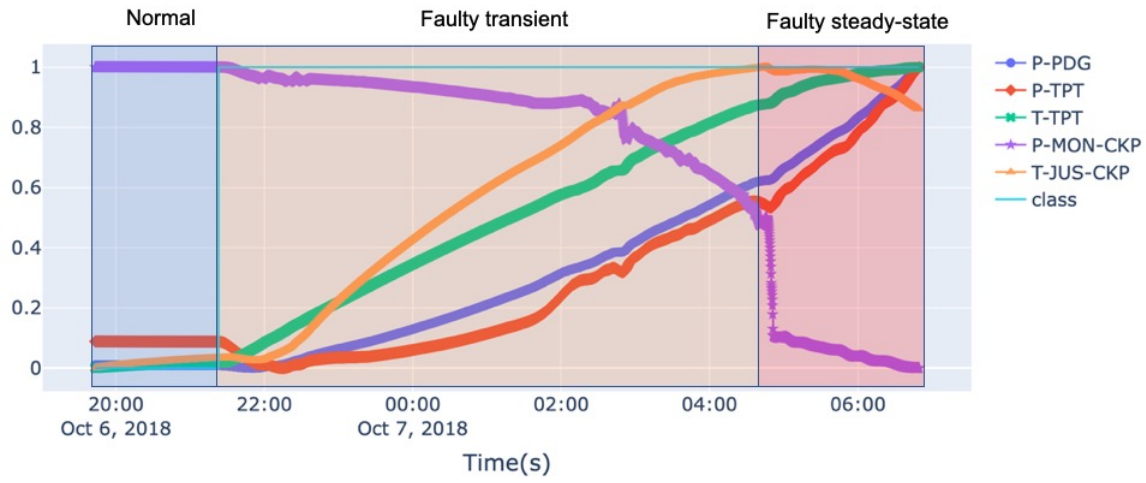
The proposed methodology was applied to this dataset for automatically detecting and classifying the different event types. The results for classes 1, 2, and 8 were presented in detail.

Class 1 has all three types of instances: real, simulated, and hand-drawn. Class 2 is the only one that has a similar proportion of simulated and real instances, while in classes 1 and 8 the majority of the instances were obtained via simulation. Furthermore, class 2 has the fastest dynamic among the eight classes, in the order of a few minutes from the transient period to the steady fault. Finally, class 8 has the slowest dynamic, occurring over hours. These three different scenarios allow a good range of evaluation of the methodology.

Only the overall performance improvement is shown for classes 3, 4, 5, and 6. Time series in classes 3 and 4 do not start with samples labeled as normal, and one of the proposed performance metrics cannot be applied. Finally, class 7 was not evaluated since only 4 real instances are available, and hand-drawn instances are not suitable for machine learning algorithms.

Except for classes 3 and 4, all events start with normal data, change to faulty transient data, and, finally, to faulty steady-state data, as shown in Figure 13. When using the OCC strategy, data labeled as steady-state and faulty transients were considered faulty. Consequently, the classifiers were created to identify transient states as faults, enabling preemptive measures to be taken before they worsen. This approach is crucial to convert the initial problem with three classes into a two-class issue and enable the utilization of binary classifiers.

Figure 13 – Example of class 1 instance, with normalized sensor data.



Source: The Author

The eight process variables that characterize the events are not always available, for

several reasons. Since oil wells are hostile environments, this selection considers situations in which the instrumentation is more reliable.

Three of these variables are not related to naturally flowing wells, so only five variables were used to develop the models. Descriptions of the variables and raw data units are in Table 8.

Table 8 – Selected Variables Description.

Name	Description	Unit
P-PDG	Pressure at permanent downhole gauge	Pa
P-TPT	Pressure at temperature/pressure transducer	Pa
T-TPT	Temperature at temperature/pressure transducer	ºC
P-MON-CKP	Pressure upstream of production choke	Pa
T-JUS-CKP	Temperature downstream of production choke	ºC

Table 9 shows the name of the variables and their corresponding numbers since they are used in many figures for simplicity.

Table 9 – Variables Name and Number.

Variable Name	Variable Number
P-PDG	0
P-TPT	1
T-TPT	2
P-MON-CKP	3
T-JUS-CKP	4

It is notorious that this is a challenging dataset since data was collected in different wells under different conditions, variables with missing values, distinct dynamics events, three state phases (normal, transient, and faulty), and different sources of data collection (real, simulated, and hand-drawn).

6.1.1 Clustering by Similarity Result

In this section, we present the results achieved using the method clustering by similarity methodology applied to the 3W dataset. The OCC model used to evaluate was the LSTM Autoencoder.

6.1.1.1 LSTM Autoencoder Parameters

We provide an overview of the parameters used for the LSTM autoencoder in the proposed method. To construct the LSTM autoencoder model, the input data was transformed into a 3-dimensional matrix, organized by samples, time steps, and variables. The chosen data window size for time steps was set to 10 samples. The implementation of the model was carried out using the Python programming language and the Keras library.

To train the LSTM autoencoder, the following hyperparameters were used:

- Time Steps: 10
- Batch Size: 64
- Number of units in the encoder layer: 32/16
- Number of units in the decoder layer: 16/32
- Epochs: 100
- Learning Rate: 0.0001

Adam (Adaptive Moment Estimation) optimizer and mean square error (MSE) were used as cost functions.

To distinguish the target class from others, a threshold is required for the LSTM autoencoder. Test data samples with an error above this threshold are labeled as negative, while those with errors below it are labeled as positive (the target class). Given that the LSTM was trained on faulty data, a low reconstruction error value indicates the presence of an anomaly in a particular sample at a specific time point.

This approach involves creating a one-class classifier that is responsible for identifying a particular fault while distinguishing it from others. Each classifier corresponds to a single faulty class.

6.1.1.2 Pre-Processing

Pre-processing was performed on each instance for all variables. A preliminary analysis for the three classes and five variables has shown that missing values in one instance either are smaller than 3% or equal to 100% of the total samples. Thus, a variable is said unassigned in a given instance if missing values exceed 3% of the total instance samples or if the number of frozen values exceeds 95% of the instance. The pre-processing was performed for all instances of all classes. When missing values were less than 3%, interpolation was used to replace them. Table 10 shows this pre-processing result. All class variables have at least one instance with missing values or frozen values. Class 2, for example, has 50% of the instances with variable 4 with more than 3% of missing values.

After the treatment of missing and frozen values, for each class, the instances were split into 60% training, 20% validation, and 20% at the test set. The data set was not split at the sample level because of the time dependency of each variable. Data standardization was performed using training data z-score (standard score), which maps data to have a zero mean and a unity standard deviation. The same standardization was applied to the validation and test sets.

Following the treatment, the data were down-sampled by applying a non-overlapping moving average window to reduce its size. The window was moved through each variable

Table 10 – Pre-Processed Variables.

Variable	Class	Unassigned	Frozen	Missing Values
0	1	3.10%	3.10%	0.00%
	2	15.79%	15.79%	0.00%
	8	1.19%	1.19%	0.00%
1	1	0.00%	0.00%	0.00%
	2	0.00%	0.00%	0.00%
	8	1.19%	1.19%	0.00%
2	1	0.00%	0.00%	0.00%
	2	0.00%	0.00%	0.00%
	8	1.19%	1.19%	0.00%
3	1	0.00%	0.00%	0.00%
	2	36.84%	0.00%	36.84%
	8	0.00%	0.00%	0.00%
4	1	0.78%	0.78%	0.00%
	2	52.63%	2.63%	50.00%
	8	3.57%	0.00%	3.57%

with a step size equal to the window size, ensuring that there was no overlap between the windows. It was performed by replacing the window sample values with the average single value. However, before using this technique, an appropriate down-sampling factor must be determined.

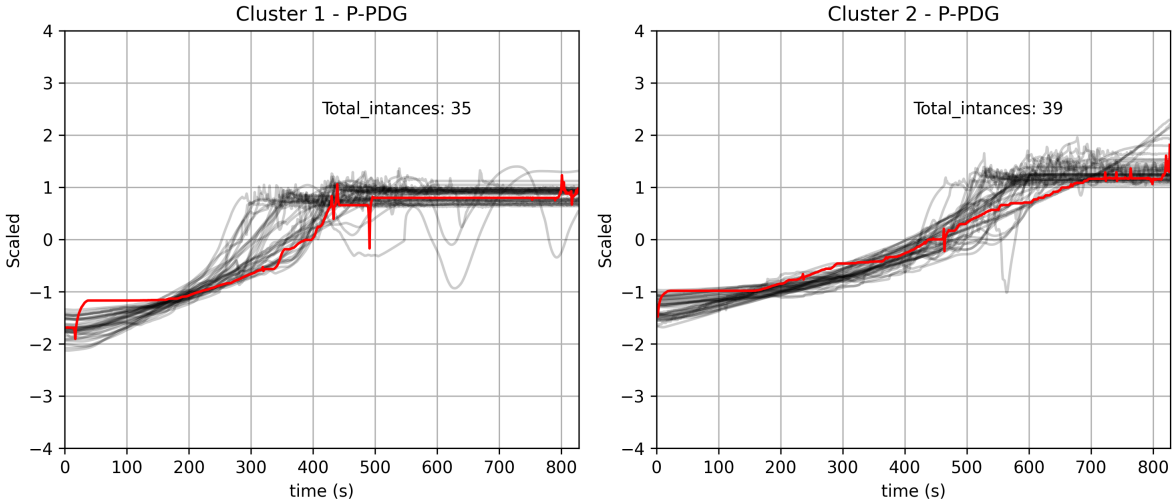
This process is useful to reduce the amount of data while still preserving the overall trend of the variables. As reported by Tyagi & Mittal (2020), in their study on sampling techniques, the application of down-sampling can produce robust performance across a range of metrics. So, to determine the suitable down-sampling factor, an investigation was carried out. The examination evaluated different down-sampling factors of 1, 10, 50, 100, and 200. An LSTM model was trained and its parameters were kept constant, with only the down-sampling factor being modified for training and testing purposes. The performance was evaluated using μ_m, and no statistical difference was observed until a down-sampling factor of 100 was reached. Therefore, this value was selected. By doing so, down-sampled variables had 100 times fewer samples compared to the original signal, and the models could be trained faster, keeping the compromise with the performance.

6.1.1.3 Time Series Clustering

At this step, k-means with DTW metric was applied to split the time series of the training instances into two groups for each variable based on their similarity. The objective was to maximize data similarity within clusters while minimizing the similarity across clusters, resulting in the clusters $C^1_{x_i}$ and $C^2_{x_i}$, $i = 0, ..., 4$, shown in Figures 14 to 18 for class 1 and the five variables. The gray lines represent instances and the red lines represent the centroids of the clusters. Class 1 has a total of 77 instances for training. The

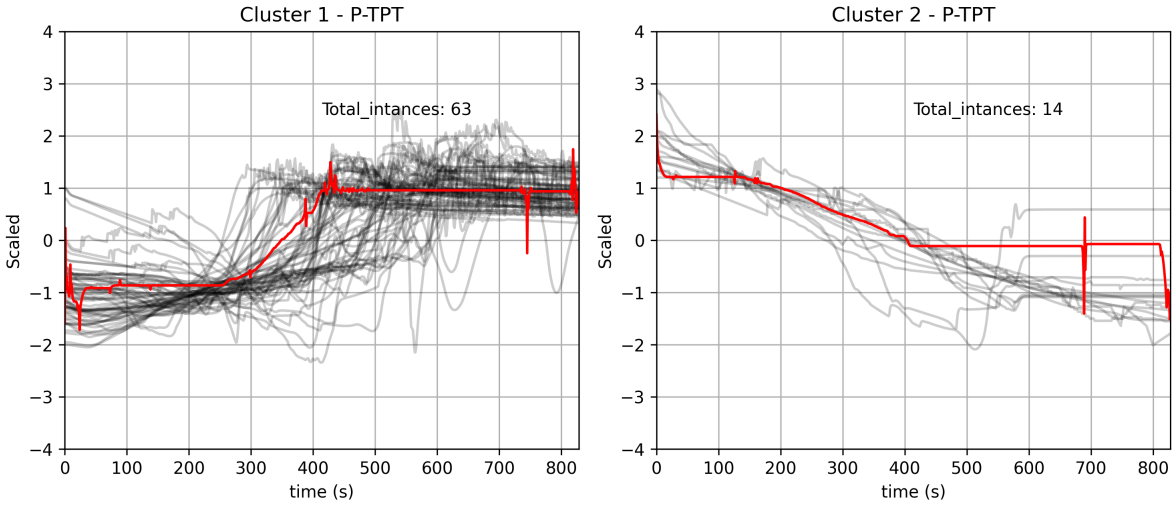
amount of instances in each cluster for each variable is shown in the figures. For example, for variable x_0 (P-PDG), the cluster $C^1_{x_0}$ has 35 instances, and the cluster $C^2_{x_0}$ has 39 instances, and 3 unassigned instances from the pre-processing step. After this process, it is possible to identify that some variables have different behaviors for the same class. It leads to the discovery of interesting patterns in the time series data set. Figure 16, for example, reveals that the variable x_2 (T-TPT) has two opposite behaviors. This affects the instances' similarity.

Figure 14 – Class 1 - P-PDG similarity clustering.



Source: The Author.

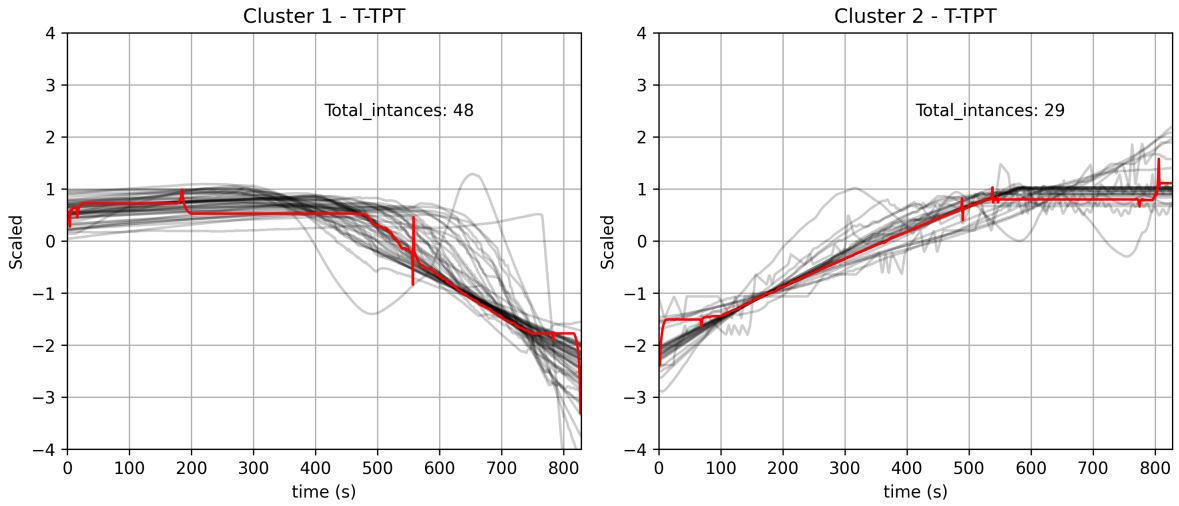
Figure 15 – Class 1 - P-TPT similarity clustering.



Source: The Author.

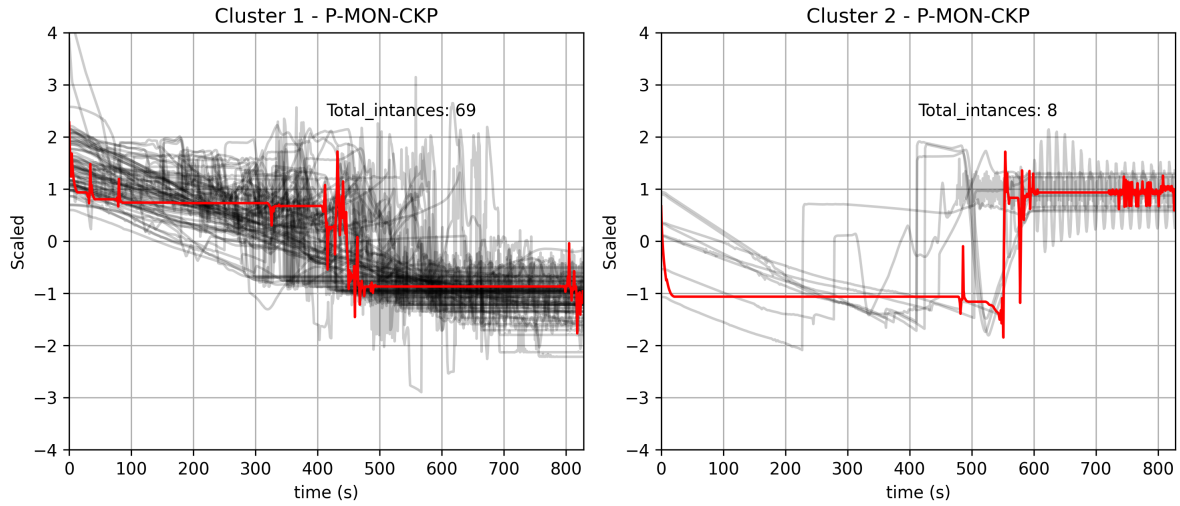
The increase of similarity of data in the clusters after the application of k-means and DBA was calculated using Equations 5.2 and 5.3.

Figure 16 – Class 1 - T-TPT similarity clustering.



Source: The Author.

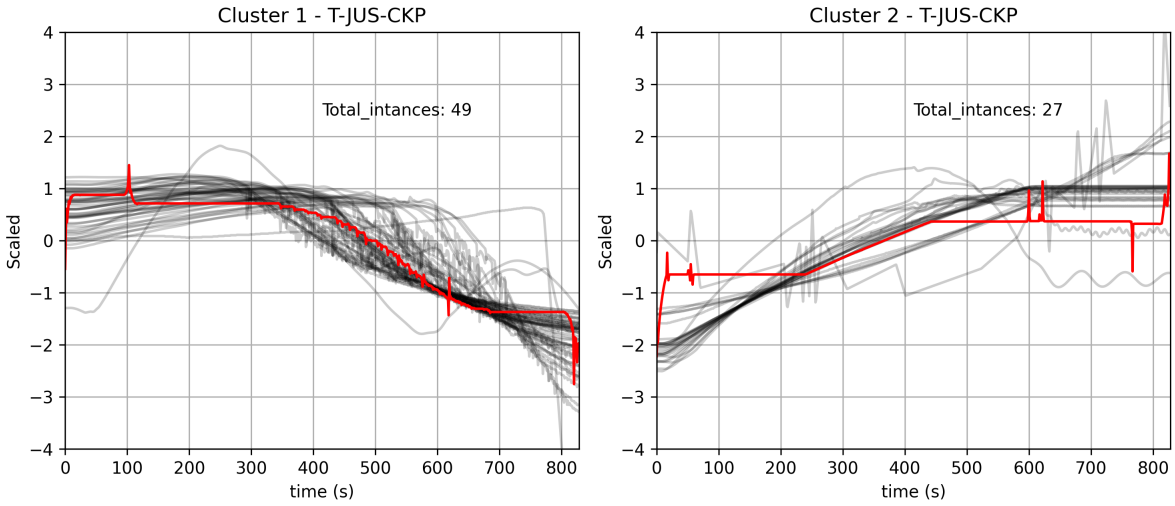
Figure 17 – Class 1 - P-MON-CKP similarity clustering.



Source: The Author.

The increase of similarity for class 1 and all five variables, with respect to cluster C^b, are shown in Figure 19. Cluster 1 means the clusters $C^1_{x_i}, i = 0, .., 4$ and Cluster 2 means the clusters $C^2_{x_i}, i = 0, .., 4$. All variables' similarity increased after clustering. The variable T-TPT, for example, had the highest increase value, for both clusters, by more than 60% related to the same variable in cluster C^b. The explanation for these high values is the behavior of the T-TPT instances. The T-TPT instances are very similar to their centroid, being that centroid 1 decreases and centroid 2 increases over time, as observed in Figure 16, thus maximizing data similarity within clusters and minimizing it across clusters. Figures 20 and 21 present the increase of similarity of each variable for classes 2

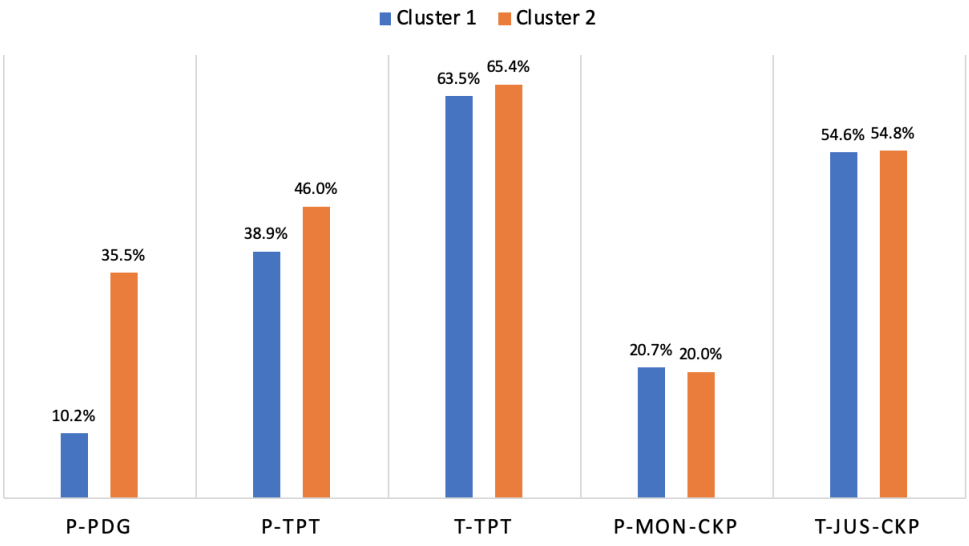
Figure 18 – Class 1 - T-JUS-CKP similarity clustering.



Source: The Author.

and 8, respectively. As expected, the increase of similarity varies among the variables and clusters 1 or 2.

Figure 19 – Class 1 - Increase of similarity of each variable.

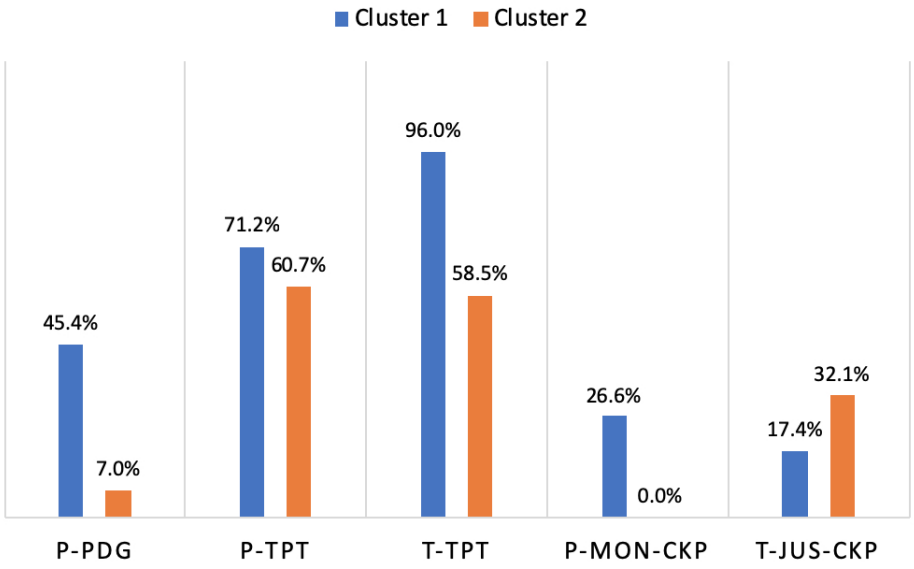


Source: The Author.

6.1.1.4 Apriori Clustering Results

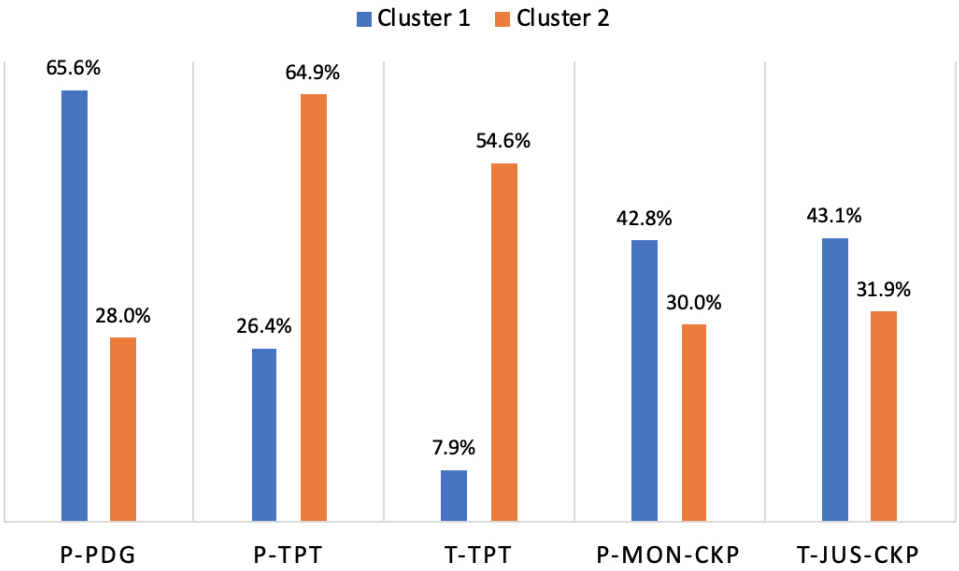
Times series clustering generated two clusters for each variable. The Apriori algorithm did the combinations of cluster 1 for all variables, and then for cluster 2 for all variables. Given a support threshold $s = 0.35$, the Apriori algorithm identified the

Figure 20 – Class 2 - Increase of similarity of each variable.



Source: The Author.

Figure 21 – Class 8 - Increase of similarity of each variable.



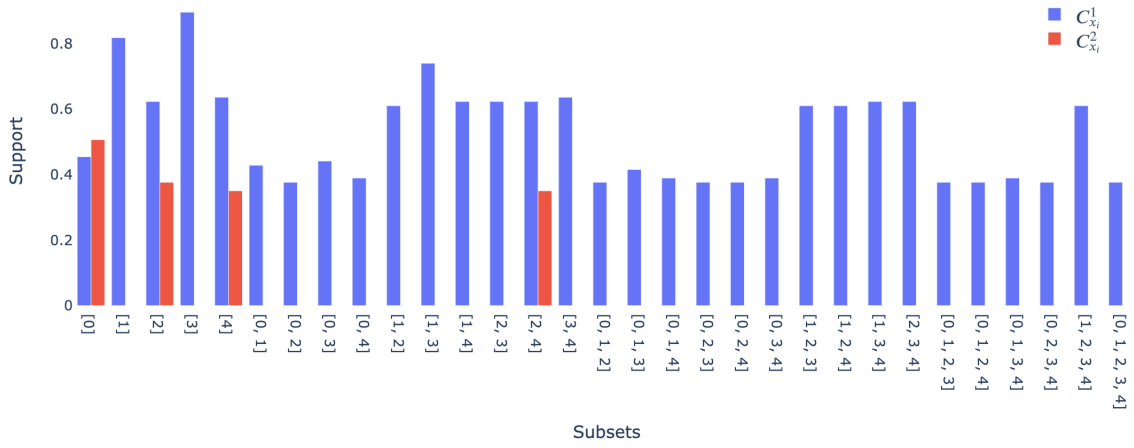
Source: The Author.

combination of variables that appeared in at least 35% of instances in the training data set.

Figure 22 shows subsets of instances of class 1 that satisfy the Apriori support threshold criterion. In this case, 31 subsets were selected from cluster $C^1_{x_i}$, and 4 subsets were selected from cluster $C^2_{x_i}$. The range of variables in the subsets went from one to four.

The next step was selecting the subsets from clusters 1 and 2 shown in this figure.

Figure 22 – Subsets of instances of Class 1 that meet the Apriori support threshold requirement.



Source: The Author.

For cluster C_0, at the end of pre-processing, there was a single group of variables and instances available. The Apriori algorithm was applied to identify the availability of the instances and variables that attended to the minimum support criterion. In this case, the support value was $s = 0.7$, since cluster C_0 was not split into two groups.

6.1.1.5 Selection of Cluster Subsets

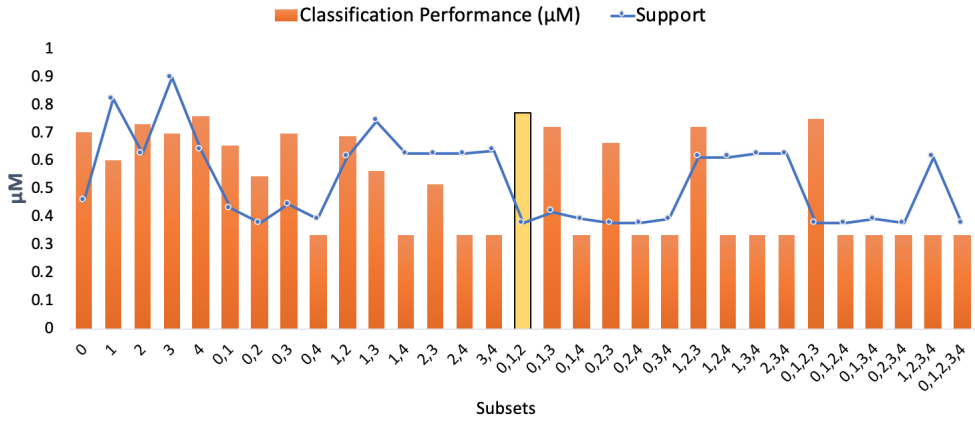
In this phase, for each subset item from the Apriori step, one model was trained and tested to choose the cluster subset. For clusters 1 and 2, the metric μ_M (Equation 5.10) was considered to select the cluster subset that presented the best classification performance. For both clusters, the subset that obtained the best result, using the validation data set, was selected. It is important to emphasize that the validation and test data did not go through the similarity clustering process.

Reminding that cluster C^b variable selection was done only by availability and there was no variable selection by similarity. So, in this single cluster, the subset with a larger number of variables was desired, according to availability. In class 1, the variables {0,1,2,3} were selected for cluster C^b with 96% of training instances.

The models were evaluated and Figures 23 and 24 exhibit clusters 1 and 2 and their subsets results, respectively, for class 1. The bars indicate μ_M the classification performance results and the blue line represents the subsets support value.

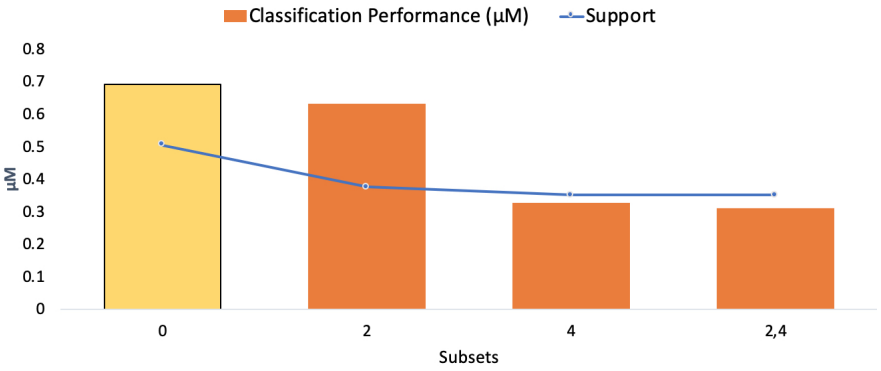
In this case, it is possible to observe highlighted in Figure 23 that the best result was achieved by the subset of variables {0,1,2} with a support value of 38% represented

Figure 23 – Class 1 - Outputs of the subsets of cluster 1.



Source: The Author.

Figure 24 – Class 1 - Outputs of the subsets of cluster 2.



Source: The Author.

by the line. So, the cluster C^1 was composed of the variables {0,1,2} with 37.66% of 77 training instances, and the cluster C^2 had only variable {0} with 50.65% of the 77 training instances, as highlighted in Figure 24. These subsets showed the best classification performance by the validation data set.

Using C^1 for training, the performance was $\mu_M = 0.77$ and using C^2 $\mu_M = 0.69$ was obtained. The variables and instances selected for C^b, C^1, and C^2 are presented in Table 11 for class 1, together with the percentage of instances remaining after the Apriori process. We recall that original data was distributed 60% for training, 20% for validation, and 20% for testing. The percentage shown in the table is the percentage of instances that remained in these sets. For the training set, this corresponds to the instances selected after pre-processing, clustering by similarity, and subset selection using the classifiers. For the validation and test sets, this corresponds to the instances selected after pre-processing and the availability of the variables used for training the selected classifier.

For example, for the cluster C^1, the variables {0,1,2} were selected and, with these variables, this cluster retained 37.66% of the original training set. To have these selected variables in the instances, 77.83% and 74.58% of the original instances remained in the sets for validation and testing, respectively.

Cluster C^b also experienced a decrease in the number of instances since some of them were eliminated due to the unavailability of the selected variables in those instances, keeping 96.10% of the original training set.

Table 11 – Variables and Instances Selected on Validation - Class 1.

Cluster	Selected Variables	Train	Validation	Test
C^b	0,1,2,3	96.10%	69.33%	68.21%
C^1	0,1,2	37.66%	77.83%	74.58%
C^2	0	50.65%	90.00%	89.17%

Table 12 displays the selected variables and instance availability for class 2. The clusters C^1 and C^2 had the same instances for validation and testing because the same variables ({1,2}) were selected for them. However, the training set percentages for C^1 and C^2 were different because the clustering process led to different quantities of similar instances for each cluster.

Table 12 – Variables and Instances Selected on Validation - Class 2.

Cluster	Selected Variables	Train	Validation	Test
C^b	0,1,2	76.19%	76.44%	76.25%
C^1	1,2	42.86%	76.44%	77.08%
C^2	1,2	52.38%	76.44%	77.08%

The same process was repeated for class 8. The selected subsets are presented in Table 13. For cluster C^b and the selected variables ({0,1,2,3}), the original training set was fully available. One can also see that different sets of variables are selected to detect class 8 for the models obtained using C^b, C^1, and C^2.

Table 13 – Variables and Instances Selected on Validation - Class 8.

Cluster	Selected Variables	Train	Validation	Test
C^b	0,1,2,3	100%	68.74%	68.91%
C^1	1,2	48.98%	77.83%	74.86%
C^2	0	48.98%	90%	88.61%

6.1.1.6 Test Results

In the previous section, the training data were selected based on similarity measures to obtain the best performance for the selected one-class classifier, in our case, an LTSM Autoencoder network. In this section, the results are presented after evaluating the trained models M_b, M_1, M_2, and fusion model M_3 using the test set, that was not used before.

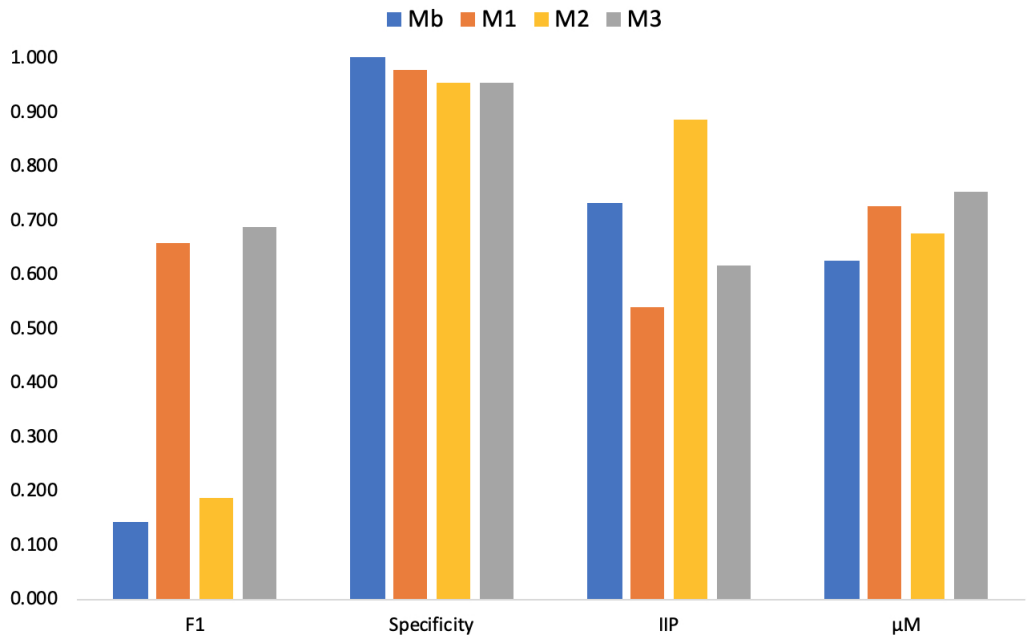
In addition to the performance classification metric μ_M, which was used to select models and compare their performance using test data, the previously discussed metrics (F1-score, Specificity, and IIP) are also shown, for a better understanding of μ_M behavior.

The results of Class 1 for the models M_b, M_1, M_2, and M_3 are presented in Figure 25 and Table 14. The best performance was achieved by the M_3 model with a value of $\mu_M = 0.753$. On the other hand, the M_b model had the poorest results with a μ_M value of 0.624. The best μ_M was achieved with the best F1 and the third best IIP. The specificity values had a low variation among the four models.

Table 14 – Class 1 - Test Results.

Model	F1	Specificity	IIP	μ_M
M_b	0.14	1.00	0.73	0.624
M_1	0.66	0.98	0.54	0.725
M_2	0.19	0.95	0.89	0.675
M_3	0.69	0.95	0.62	0.753

Figure 25 – Class 1 - Resulting metrics for the 4 models.

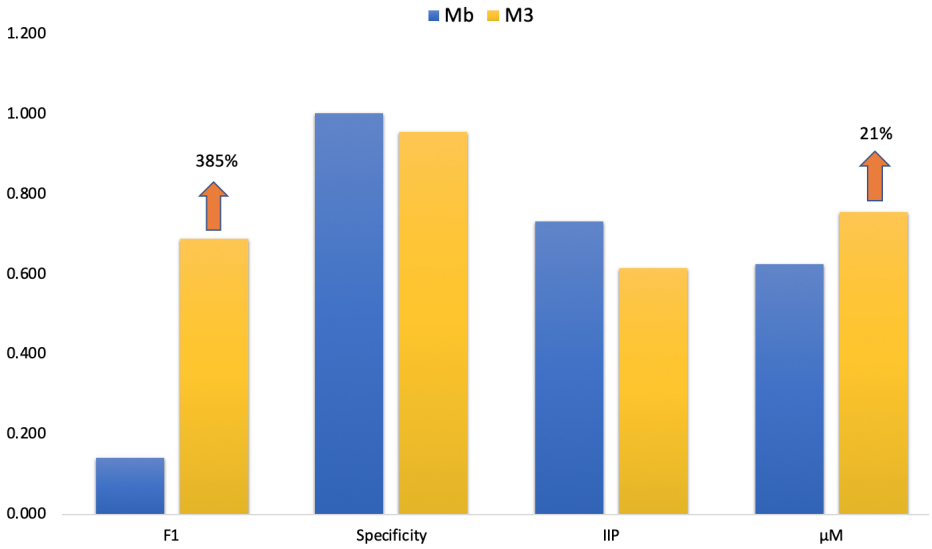


Source: The Author.

Figure 26 shows the comparison of M_3 and M_b models from Class 1. The M_3 model increased 385% F1-score and, comparing the μ_M, this model achieved a 21% of improvement in classification performance.

Figure 27 and Table 15 presents the results of M_b, M_1, M_2, and M_3 models for data from Class 2. The best result was obtained by M_1, which achieved $\mu_M = 0.858\%$.

Figure 26 – Class 1 - Comparing M_3 with M_b.



Source: The Author.

The best μ_M was achieved with the best F1. The specificity values had a low variation among the four models, and IIP was smaller only for M_b.

Table 15 – Class 2 - Test Results.

Model	F1	Specificity	IIP	μ_M
M_b	0.62	1.00	0.50	0.708
M_1	0.76	0.99	0.83	0.858
M_2	0.34	0.94	0.83	0.706
M_3	0.55	0.93	0.83	0.770

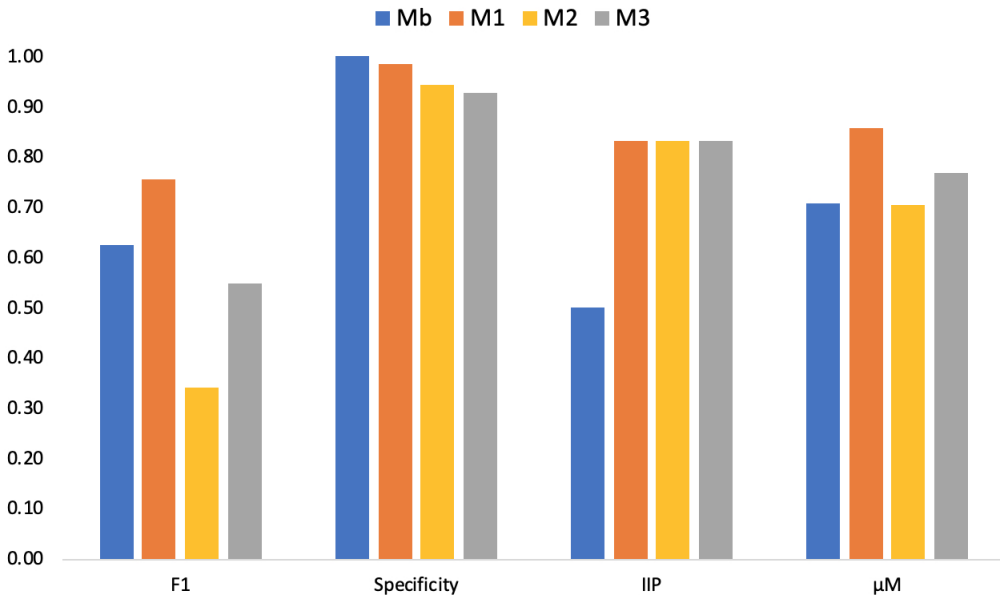
Figure 28 shows the comparison of M_1 and M_b models from Class 2. The M_1 model increased 23% of the F1-score and 67% of the IIP. Comparing μ_M, this model achieved a 21% of improvement in classification performance.

The results of models M_b, M_1, M_2, and M_3, when applied to Class 8 data, are shown in Figure 29 and Table 16. Model M_1 had the best performance with a classification score of 0.93. Again, the best μ_M was achieved with the best F1, the specificity had low variations, and IIP was the same for all models.

Table 16 – Class 8 - Test Results.

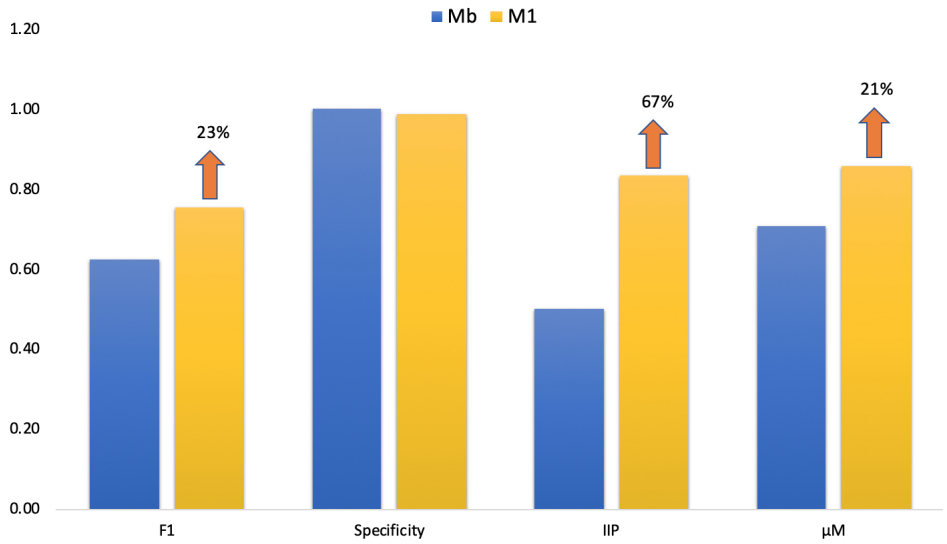
Model	F1	Specificity	IIP	μ_M
M_b	0.75	0.99	1.00	0.91
M_1	0.80	0.98	1.00	0.93
M_2	0.71	0.89	1.00	0.87
M_3	0.78	0.88	1.00	0.89

Figure 27 – Class 2 - Resulting metrics for the 4 models.



Source: The Author.

Figure 28 – Class 2 - Comparing M_1 with M_b.

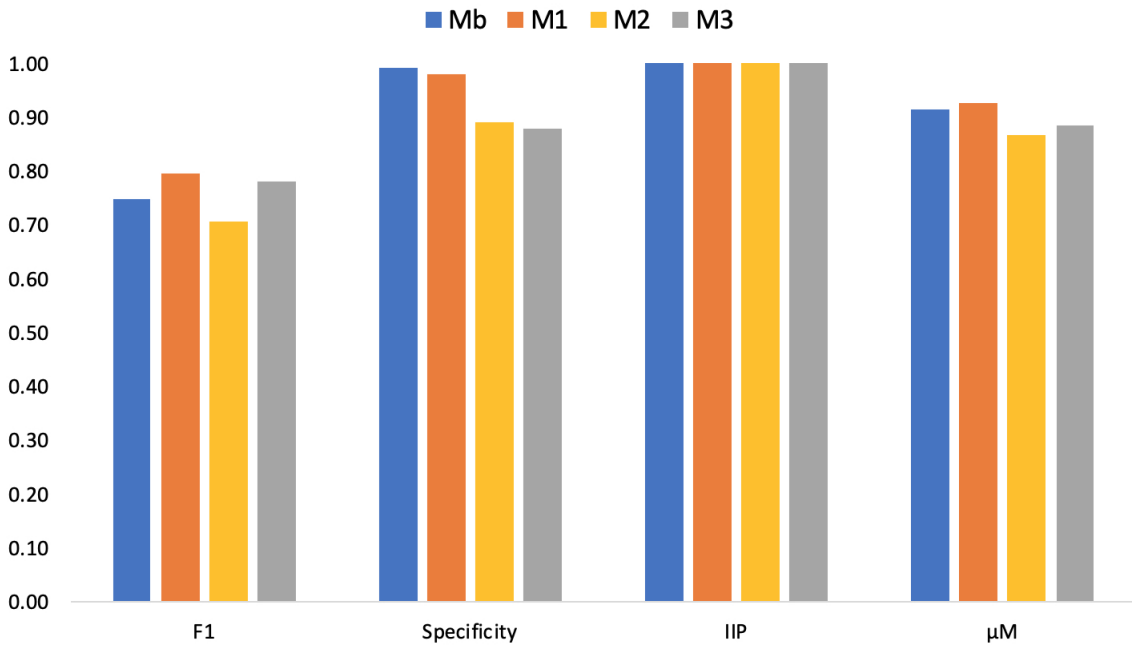


Source: The Author.

Figure 30 shows the comparison of M_1 and M_b models from Class 8. The M_1 model increased 7% F1-score. In terms of μ_M, the M_1 model exhibited a 2% improvement in classification performance compared to M_b.

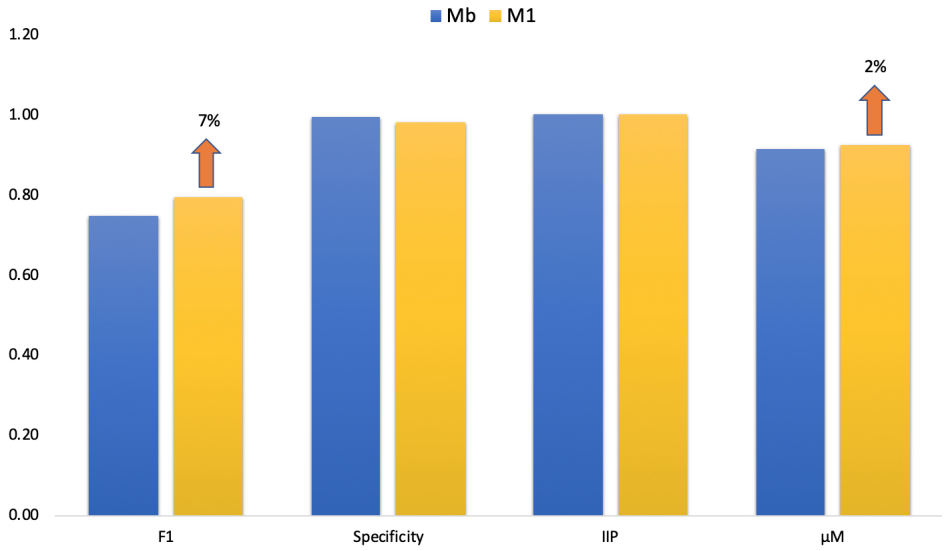
As observed across all classes, the model that achieved the highest performance at μ_M also obtained the highest F1-score, along with a slight reduction in specificity,

Figure 29 – Class 8 - Resulting metrics for the 4 models.



Source: The Author.

Figure 30 – Class 8 - Comparing M_1 with M_b.



Source: The Author.

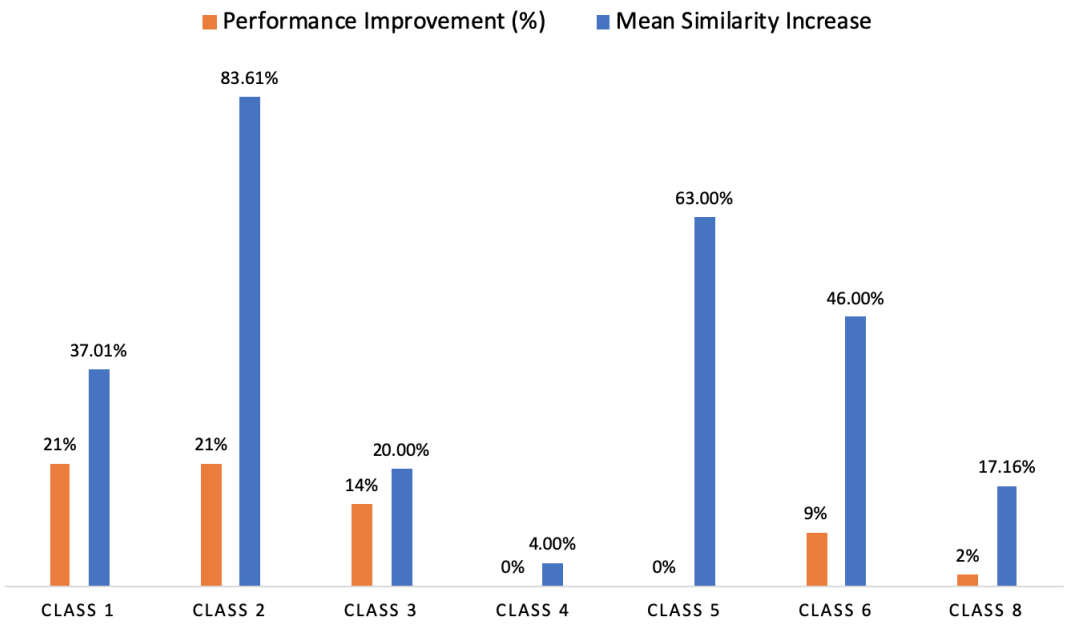
ultimately resulting in an overall performance improvement.

Figure 31 presents the increased similarity and the classification performance improvement μ_M using clustering models for the three classes considered. M_3 clustering model increased the similarity by 37% and the classification performance by 21% in Class

1. In Class 2, the M_1 clustering model increased the similarity by almost 84% and the classification performance by 21%. Finally, Class 8 got a raise of 17% at similarity and 2% on classification performance with M_1. Class 2 was the one that obtained the highest increase in similarity. This class has a total of 38 instances, 22 real and 16 simulated.

The significant increase in similarity in Class 2 can be attributed to the largest proportion of real instances concerning simulated ones. For this dataset, the real instances are affected by factors such as reservoir depletion, pressure, flow rate decline, and diverse characteristics for different oil wells. On the other hand, simulated data depends on the models and parameters used to generate the data. For this dataset, apparently, only mild changes were made. The existence of more instances with dissimilarity allowed the improvement of similarity in the selected clusters.

Figure 31 – Increased similarity and classification performance improvement using clustering models for the three classes.



Source: The Author.

6.1.2 Application to the other classes of 3W dataset

The proposed methodology was also applied to classes 3, 4, 5, and 6, but only the increased similarity and the classification performance improvement μ_M were presented for them. For classes 3 and 4, μ_M was computed as the mean of F1 and specificity since IIP cannot be computed for these classes. Class 7 was not evaluated since only 4 real instances are available, and hand-drawn instances are not suitable for machine learning algorithms. The results for all evaluated classes are shown in Figure 31 and Table 17.

An increase in similarity in data used for training was observed in all classes. The cause of the dissimilarity can be associated with different parameters used for simulation (in simulated instances). Real instances are affected by factors such as reservoir depletion, pressure, flow rate decline, and diverse characteristics of different oil wells. The increase in similarity was accompanied by an increase in the classification performance for classes 1, 2, 3, 6, and 8. No improvements were noticed in class 4 since this event (flow instability) is characterized by the absence of any temporal pattern. Class 5 has 12 real instances and 439 simulated instances, and this class has no improvement in performance despite the great increase of similarity in data used for training. An overall performance improvement of 10% was obtained for the 7 classes evaluated.

Table 17 – Classification Performance for all 3W classes

	Model	F1	Specificity	IIP	μ_M
Class 1	M_b	0.14	1.0	0.73	0.62
	M_1	0.66	0.98	0.54	0.73
	M_2	0.19	0.95	0.89	0.68
Class 2	M_b	0.62	1.0	0.50	0.71
	M_1	0.76	1.0	0.83	0.86
	M_2	0.34	0.94	0.83	0.71
Class 3	M_b	0.60	1.0	NA	0.80
	M_1	0.64	0.70	NA	0.67
	M_2	0.83	1.0	NA	0.91
Class 4	M_b	0.81	1.0	NA	0.91
	M_1	0.80	1.0	NA	0.90
	M_2	0.68	0.80	NA	0.73
Class 5	M_b	0.92	1.0	0.01	0.64
	M_1	0.89	0.90	0.1	0.63
	M_2	0.83	0.76	0.12	0.57
Class 6	M_b	0.77	0.97	0.31	0.68
	M_1	0.61	0.98	0.64	0.74
	M_2	0.69	0.90	0.1	0.56
Class 8	M_b	0.75	0.99	1.0	0.91
	M_1	0.80	0.98	1.0	0.93
	M_2	0.71	0.89	1.0	0.87

6.1.3 Selection of hyperparameters in the LSTM

The same 6 hyperparameters described in Section 6.1.1.1 were used to train all LSTM models (see Figure 7). However, the results could be eventually improved if the hyperparameters were tuned for each LSTM model. For analysis, a grid search for the hyperparameters used to train the LSTM models for class 2 was performed. For the selection of the hyperparameter values of the models M_b, M_1, and M_2, the following hyperparameters were tuned using the validation data set: Epochs: [10, 20, 30], Learning Rate: [0.01, 0.001, 0.0001], and Bath Size: [16, 32, 64]. Figure 32 shows the boxplot of the

metric μ_M of the models across the range of the parameters. This analysis confirms that, in this case, in both models obtained via the proposed methodology (M_1 and M_2), the median of μ_M is statistically higher than M_b.

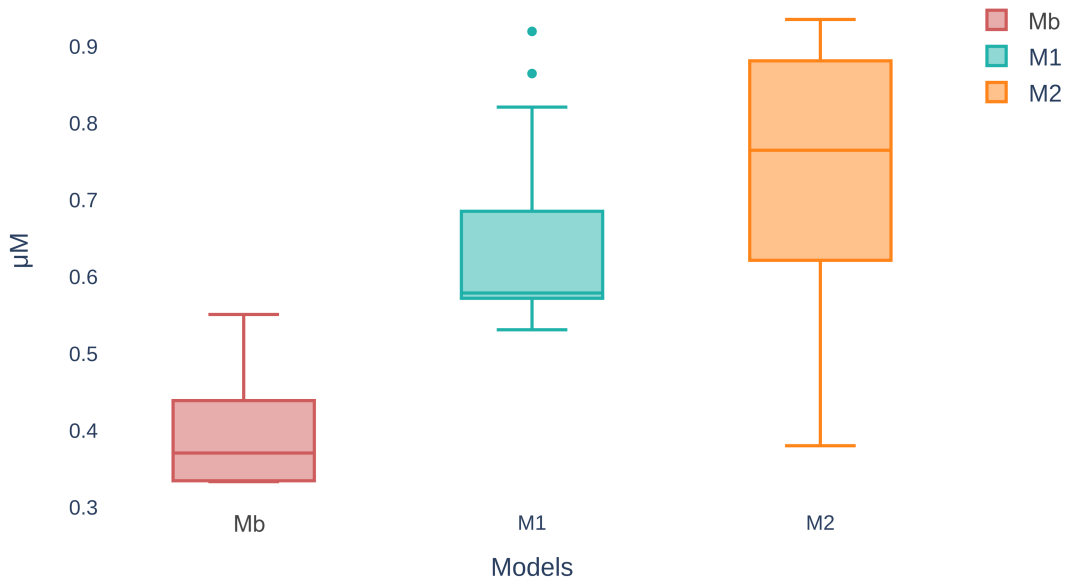


Figure 32 – Hyperparameters Analysis

After selecting the hyperparameters for each model, the models with adjusted hyperparameters were tested. The new hyperparameters and the resulting improvement in performance are shown in Table 18. Compared to Table 15, the grid search slightly reduced the performance of M_b, kept the performance of M_1, and increased the performance of M_2.

Table 18 – Performance after hyperparameters selection - Class 2

Models	epochs	lr	Batch Size	μ_M	Improvement
M_b	30	0.0001	64	0.69	-2.5%
M_1	20	0.0001	64	0.86	0%
M_2	30	0.001	64	0.92	30%

6.1.4 Crossvalidation Analysis

A cross-validation analysis was conducted for Class 2. Models M_b, M_1, and M_2 were executed 12 times with distinct training sets. Figure 33 shows the boxplot of the metric μ_M of the models.

Two statistics tests were conducted: Wilcoxon and Paired t-test. Considering the $\alpha = 0.05$ in both tests, the results from M_2 are distinct and superior compared to M_b. Table 19 shows the Wilcoxon and Paired t-test for Class 2. This analysis confirms that, in

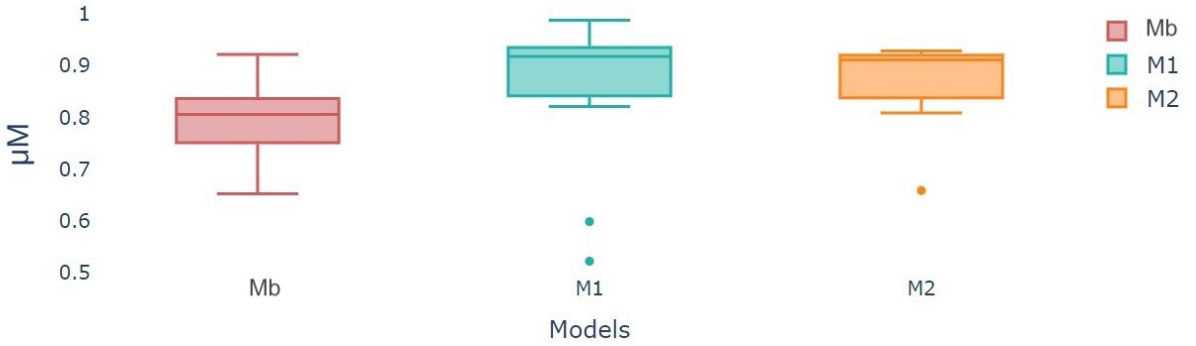


Figure 33 – Cross-validation Analysis - Class 2

Source: The Author

	Wilcoxon	T-Test
M_b M_1	0.17627	0.221109
M_b M_2	**0.052246**	**0.040414**
M_1 M_2	0.791016	0.638503

Table 19 – Wilcoxon and Paired T-Test - Class 2

this case, the model M_2 obtained via the proposed methodology, has the median of μ_M statistically higher than the baseline model, M_b.

6.1.5 Time-shift Results

The time-shift approach is now applied to the data to evaluate the improvement of the OCC, with the results obtained preceding those of time series clustering. The application of a time-shift was implemented to tackle the overlapping problem. In the time series fault events, an expected overlapping issue occurs around the transitional between label classes, due to the related behavior of normal and faulty samples of the process variables at the beginning of the fault. The time-shifting of labels in the training phase would mitigate the effect of overlapping normal attributes on the faulty samples.

In this case, the application of OCC to detect anomalies in oil wells was considered for two types of faults with different dynamics: Spurious DHSV closure (Class 2) and Hydrate in the Production Line (Class 8). Two one-class classifiers, LSTM autoencoder, and OCSVM, were applied and the effect of the time-shifting of labels on classifier performance metrics was evaluated.

The classifier performances were computed using F1, specificity, IIP, and μ_M. As accuracy is influenced by data balancing, but it is a popular metric, it is presented in the results of time-shift only for comparison purposes with other studies. The proposed methodology was applied and the results with the original data were compared to those varying the time-shift of the labels, associated with different values of γ. The original data,

i.e., no time-shift, corresponds to $\gamma = 1$.

Classifier performance was evaluated at different time-shift, associated with values of $\gamma \in]0, 1]$ in each class. The search of γ values was performed for all instances used for training the two classification methods. We separated the instances of each class into two subsets: 80% for training and 20% for the test set in class 2; 50% for training and 50% for the test set in class 8, given its greater number of instances.

The test set comprised 20% of class 2 instances, 50% of class 8 (not used in training), and random instances from the other classes in the 3W dataset. Approximately, the same amount of samples from each class were selected. Table 20 shows Test Data #1 to evaluate the classifiers trained for events of class 2 and the quantitative samples for each class.

Table 20 – Test Data #1 - DHSV - Number of Samples of Each Class.

Class	Negative	Positive	Total Instances
0	207,277	–	14
1	266,365	–	3
2	18,061	12,220	7
3	244,859	–	6
4	206,750	–	29
5	205,016	–	7
6	41,658	–	5
7	388,780	–	2
8	215,904	–	8
Total	1,794,670	12,220	81

Table 21 shows Test Data #2 with the number of samples for each class to evaluate the classifiers modeled for events of class 8.

Table 21 – Test Data #2 - Hydrate - Number of Samples of Each Class.

Class	Negative	Positive	Total Instances
0	210,976	–	14
1	266,365	–	3
2	201,516	–	7
3	244,859	–	6
4	206,651	–	29
5	205,016	–	7
6	41,658	–	5
7	388,780	–	2
8	69,771	982,761	39
Total	1,835,592	982,761	112

6.1.5.1 LSTM Autoencoder Parameters

In this section, we provide an overview of the parameters of the LSTM autoencoder used in the proposed method. The input data was transformed into a 3-D matrix, structured

according to samples, time steps, and variables. A data window size of 10 samples in time with overlapping was employed. The implementation of the model was done using the Python programming language and the Keras library.

To train the LSTM autoencoder, the following hyperparameters were used:

- Time Steps: 10
- Batch Size: 64
- Number of units in the encoder layer: 32/16
- Number of units in the decoder layer: 16/32
- Epochs: 100
- Learning Rate: 0.0001

Adam optimizer and mean square error (MSE) were used as cost functions.

Given that the LSTM was trained on faulty data, a low reconstruction error value indicates the presence of an anomaly in a particular sample at a specific time point.

6.1.5.2 One-class SVM Parameters

One-class SVM (OCSVM) is a special case of Support Vector Machine (SVM). In the case of one-class classification, only positively labeled data is used during training, and hyperplanes corresponding to the negative class are set to be the origin of the coordinate system (PERERA; OZA; PATEL, 2021). Therefore, OCSVM attempts to learn a decision boundary that achieves the maximum separation between the points and the origin (AMER; GOLDSTEIN; ABDENNADHER, 2013).

As with the LSTM method, the one-class SVM classifier also underwent training using faulty data. Different from the LSTM autoencoder, the one-class SVM is a classifier that provides a -1 or 1 output depending on the class, being unnecessary to define a model threshold. The following parameters were used for the scikit-learn library:

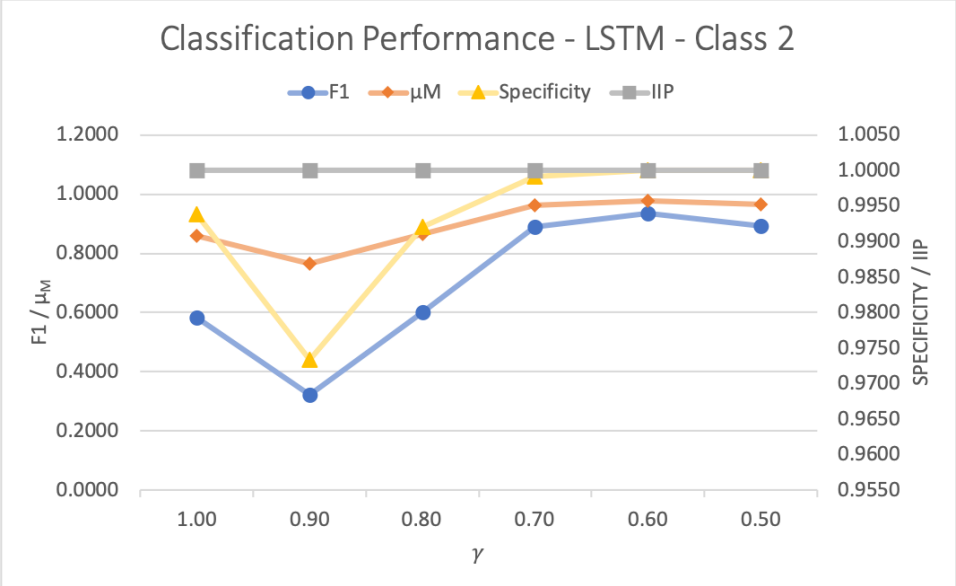
- nu: 0.1
- Gamma: auto
- Kernel: RBF

6.1.5.3 Results for Class 2

- LSTM Autoencoder

  Figure 34 shows the results obtained by the LSTM Autoencoder classifier for Class 2 on the Test Data #1 (Table 20). It is important to warn the reader of the different left and right scales used in this figure and the next ones, better to notice the effect of the time-shift on the metrics.

Figure 34 – Class 2 - LSTM.



Source: The Author

All classifier metrics are improved for values of γ in the interval $\gamma \in [0.6, 0.8]$. F1 increases its value from 0.5829 for $\gamma = 1$ to 0.9360 for $\gamma = 0.6$. The numerical values are shown in Table 22. The numbers in parentheses are the percentages of the corresponding detection times concerning the total time of the faulty transient period. In this case, detection time increases from 123 seconds on the original label to 191 seconds with $\gamma = 0.6$. This increase represents less than 2% of the total transient period. One can see that accuracy is not a good metric in this case, since it is very high for any value of γ (see Table 22).

The average metric μ_M is increased by 14% when γ is reduced from 1.0 to 0.7.

Therefore, from Figure 34, one can see that the time-shifting of labels in training data used in the LSTM autoencoder significantly improved the classification performance. F1 and Specificity were improved, as well as the proposed average metric μ_M.

- One-class SVM

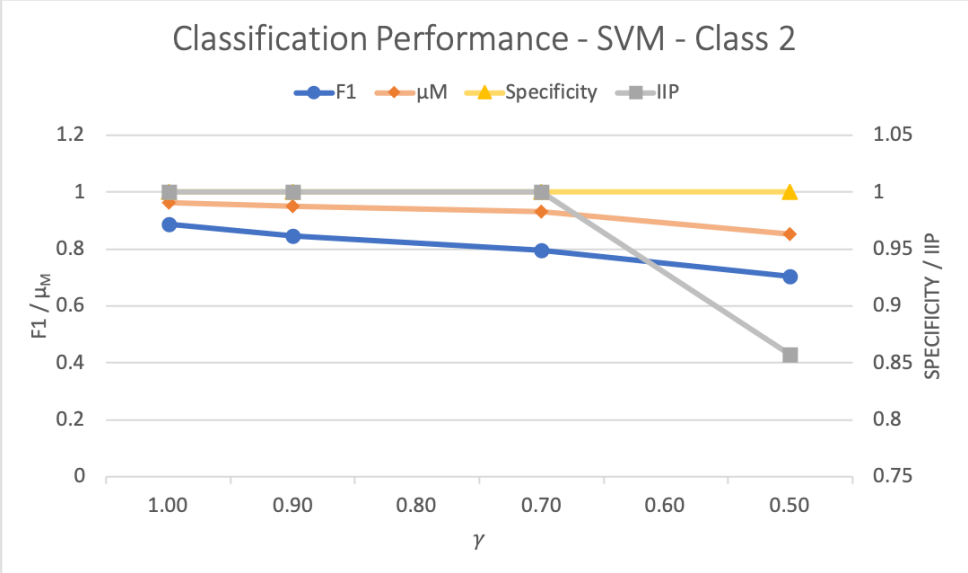
Table 22 – Class 2 - LSTM Results.

Threshold γ	F1	Specificity	IIP (%)	μ_M	Detection Time	Accuracy
1.00	0.5829	0.9938	100	0.8579	123s (2.84%)	0.9924
0.90	0.3229	0.9733	100	0.7654	114s (2.63%)	0.9731
0.80	0.6009	0.9921	100	0.8643	147s (3.39%)	0.9917
0.70	0.8887	0.9992	100	0.9626	207s (4.78%)	0.9985
0.60	0.9360	1.0000	100	0.9787	191s (4.41%)	0.9992
0.50	0.8938	1.0000	100	0.9646	347s (8.01%)	0.9987

As can be seen in Figure 35, the proposed methodology did not improve the performance metrics for one-class SVM applied to data from Class 2. The performance is similar for $1 \leq \gamma \leq 0.7$, and is reduced for $\gamma < 0.7$.

In general, the time-shift reduces the amount of FP. In this case, no improvement was achieved, since the original model did not classify any sample as FP. Detection time increases from 355 seconds on the original label to 452 seconds with $\gamma = 0.9$, which represents less than 2% of the total faulty transient period.

Figure 35 – Class 2 - One-class SVM.



Source: The Author

In Table 23 the lowest detection time was 355 seconds, obtained for $\gamma = 1.0$. In this table, one can also see that accuracy was very high with low sensitivity to different values of γ.

Table 23 – Class 2 - One-class SVM Results.

Threshold γ	F1	Specificity	IIP	μ_M	Detection Time	Accuracy
1.00	0.8880	1.0000	100.00%	0.9627	355s (8.20%)	0.9986
0.90	0.8467	1.0000	100.00%	0.9489	452s (10.44%)	0.9982
0.70	0.7952	1.0000	100.00%	0.9317	574s (13.26%)	0.9977
0.50	0.7024	1.0000	85.71%	0.8531	805s (18.59%)	0.9969

6.1.5.4 Results for Class 8

The same methodology was applied to obtain binary classifiers for class 8. This fault differs from the fault related to class 2 due to the slow dynamic of the variables associated with the fault.

- LSTM Autoencoder

  The results for LSTM are shown in Figure 36. The trained classifier was unable to distinguish between normality and fault in any of the tested events using original data. That is why we see the IIP index equal to 0% for $\gamma = 1$. This metric increases to 1 for $\gamma \leq 0.85$. Similarly, Specificity increases from 0.948 to 1 for the previous values of γ, indicating that the model managed to reduce the number of FP to zero. Only F1 was decreased, from 0.90 ($\gamma = 1$) to 0.61 ($\gamma = 0.85$). However, the good result for F1 would be useless, since $IIP = 0$ in this case ($\gamma = 1$).

  The numerical values are shown in Table 24. Accuracy ranged from 0.9339 to 0.7918, as γ varied from 1 to 0.5. However, the higher values of accuracy correspond to IIP = 0%. The highest value for the average metric μ_M is achieved for $\gamma = 0.8$, an increase of 41% compared to $\gamma = 1$. One can see that the detection time decreased from 23,046 seconds with $\gamma = 1.0$ to 13,416 seconds with $\gamma = 0.85$.

Figure 36 – Classe 8 - LSTM.

Source: The Author

- One-class SVM

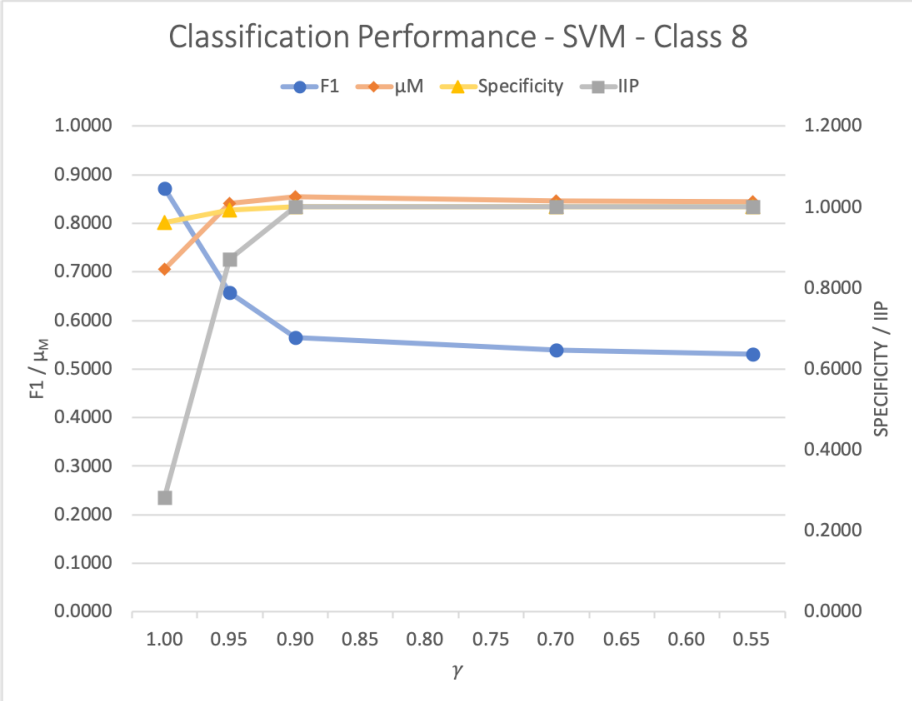
  The model trained with the original label showed F1 = 0.8715, but with an IIP of approximately 28%, while for $\gamma = 0.9$, F1 was reduced to 0.5645, but IIP increased

Table 24 – Class 8 - LSTM Results.

Threshold γ	F1	Specificity	IIP	μ_M	Detection Time	Accuracy
1.00	0.9055	0.9480	0.00%	0.6178	23046s (100%)	0.9339
0.90	0.9531	0.9479	0.00%	0.6337	23046s (100%)	0.9657
0.85	0.6073	1.0000	100.00%	0.8691	13416s (58.21%)	0.8033
0.80	0.6136	1.0000	100.00%	0.8712	13676s (59.34%)	0.8056
0.70	0.5942	1.0000	100.00%	0.8647	14026s (60.86%)	0.7987
0.50	0.5744	1.0000	100.00%	0.8581	14921s (64.74%)	0.7918

to 100%. Besides improving the IIP, this time-shift reduced the number of FP to zero, with Specificity going from 0.9626 to 1 (Figure 37).

Figure 37 – Class 8 - One-class SVM.



Source: The Author

One can observe in Table 25 that $\gamma = 0.9$ achieved the best performance, with $\mu_M = 0.8548$, i.e., an increase of 21% compared to $\gamma = 1$. In this case, detection time decreases from 20,211 seconds, with $\gamma = 1.0$, to 14,868 seconds.

Table 25 – Class 8 - One-class SVM Results.

Threshold γ	F1	Specificity	IIP	μ_M	Detection Time	Accuracy
1.00	0.8715	0.9626	28.20%	0.7054	20211s (87.70%)	0.9150
0.95	0.6578	0.9931	87.17%	0.8409	18391s (79.80%)	0.8199
0.90	0.5645	1.0000	100.00%	0.8548	14868s (64.51%)	0.8895
0.70	0.5387	1.0000	100.00%	0.8462	18997s (82.43%)	0.8850
0.55	0.5306	1.0000	100.00%	0.8435	19167s (83.17%)	0.8836

6.1.5.5 Defining criteria for selection of parameter γ and classifier

The use of three different metrics shows important effects of the methodology and of the parameter γ, which was reduced to increase the time-shift of the labels. The average values of F1, Specificity, and IIP were computed and the maximum value was selected from the set of γ values evaluated in the tests. The average values obtained for both test data and for LSTM and One-class SVM classifiers were compared with the original data, i.e., $\gamma = 1$, and the percentage variation is shown in Figure 38. The time-shift improved this average metric for both classes when using LSTM. For One-class SVM, the result was improved for class 8 and was similar to the original data for class 2.

The improvements in the average metric μ_M are summarized in Figure 39. For class 2, LSTM achieved the best value (0.98) with $\gamma = 0.7$. LSTM also achieved the best value for class 8 with $\gamma = 0.8$. Thus, the decision to use LSTM classifier for classes 2 and 8 is straightforward if the average metric μ_M is considered.

The detection time metric was also summarized in Figure 40. The detection time was divided by the transient time of the faults. Thus, 100% means that the classification method required all transient time to detect the fault. The results presented in Figure 40 allow the comparison of detection time metrics for faults with different dynamics. Observing Figure 40, one concludes that a good reduction was achieved by LSTM and One-class SVM methods when applied to class 8 (slower dynamic, in red) for a given γ.

For class 2, with a faster dynamic, the detection time was very low compared to the transient time of the fault. One can see that less than 11% of the transient time was required for all values of γ. Thus, for class 2, the difference between the two methods is not relevant for real situations. The similar detection time for both methods for class 2 shows the decision should come from other metrics.

Figures 39 and 40 allow the selection of the most suitable method for each class. For classes 2 and 8, LSTM is the best choice using the criteria shown in both figures.

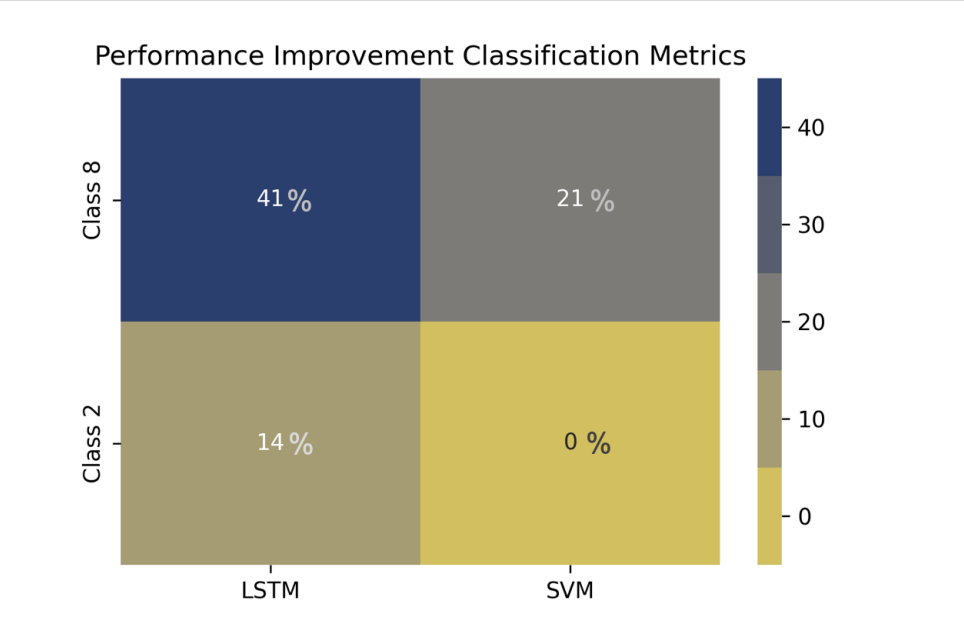
6.1.5.6 Comparison with other methods

In this work, only LSTM autoencoder and one-class SVM were considered, since they can both be trained similarly, defining each fault as the target class. The time-shift of the labels allowed us to infer the improvements in their performances.

We now compare the achieved results with previous works from the literature, using Random Forest (RF) (MARINS et al., 2021) and Decision Tree (DT) (TURAN; JÄSCHKE, 2021) classifiers. This comparison is important to check if the improvements here achieved are competitive with other methods.

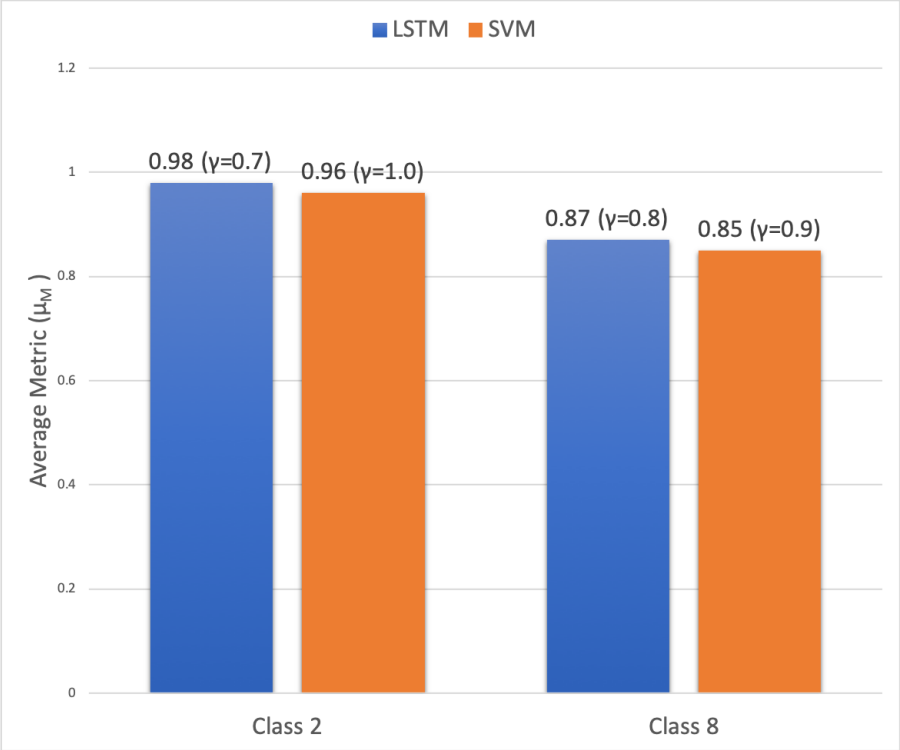
In the evaluation of the RF results (MARINS et al., 2021) was not defined the IIP metric, as presented in this work. However, the authors counted the number of instances

Figure 38 – Average of Metrics μ_M Relative to No Label Shift.



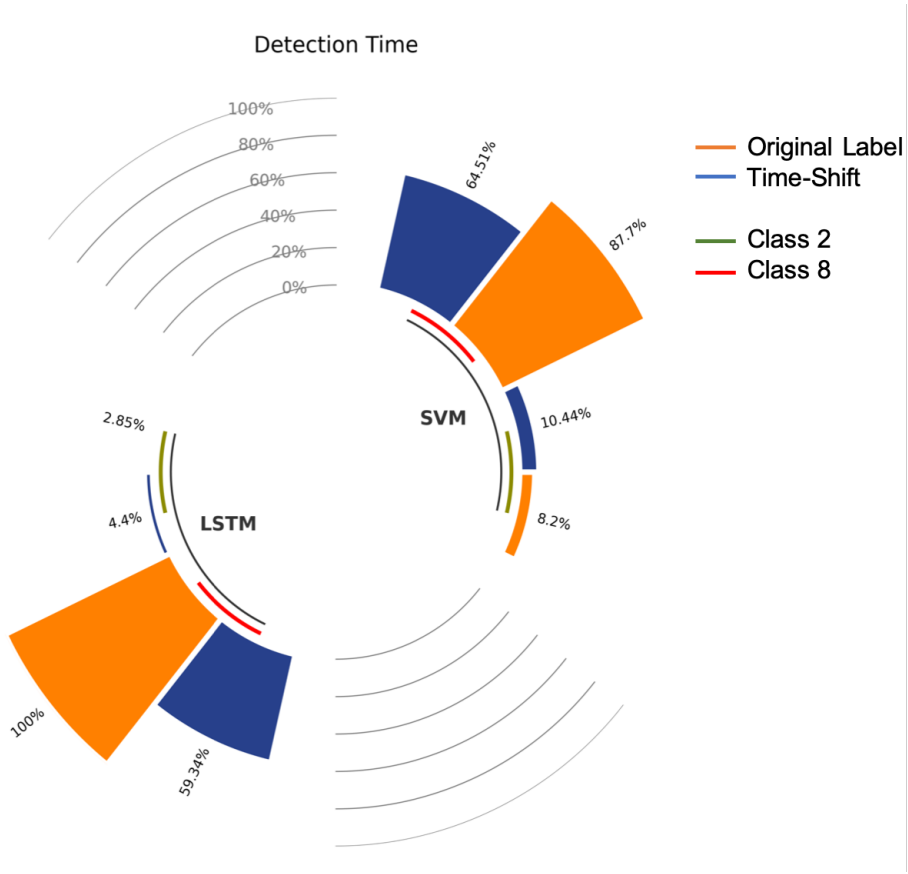
Source: The Author

Figure 39 – Comparative Average Metric.



Source: The Author

Figure 40 – Detection Time Relative to Transient Time.



Source: The Author

in which the model correctly identified the normal phase before the fault one, similar to the IIP metric and they are perfectly comparable. Table 26 summarizes the performance of three classifiers for faults of classes 2 and 8, where RF results are from (MARINS et al., 2021) and DT results are from (TURAN; JÄSCHKE, 2021). Finally, LSTM corresponds to the better results obtained in this work using LSTM and the proposed average metric.

The results in this table show the performance of these methods to isolate these two faults from normality and all other faults. The metrics used are accuracy, F1, and IIP. A blank entry indicates a lack of the metric in the cited reference. IIP was not computed for DT, and F1 was not computed in RF. In these three methods, the instances for training and testing were randomly selected.

For class 2, the LSTM method achieved the best results for the three metrics using $\gamma = 0.6$. The poor results for this class using DT were justified by the smaller number of instances (TURAN; JÄSCHKE, 2021). IIP value was also low using RF.

For class 8, the best accuracy was for LSTM with $\gamma = 0.9$. However, IIP $= 0$ in this case, i.e., the classifier does not discriminate between fault and normality. For $\gamma = 0.8$,

Table 26 – Comparison with Other Methods.

Class 2			
Classifiers	Accuracy	F1	IIP
RF	0.8708	-	0.58
DT	0.6000	0.4900	-
LSTM ($\gamma = 0.6$)	0.9992	0.9360	1.00
Class 8			
Classifiers	Accuracy	F1	IIP
RF	0.8359	-	0.89
DT	0.9000	0.8700	-
LSTM ($\gamma = 0.9$)	0.9657	0.9531	0.00
LSTM ($\gamma = 0.8$)	0.8056	0.6136	1.00

IIP = 1, but accuracy was reduced to 0.8056. The results with RF gave an accuracy of 0.8359 but with IIP = 0.89. The accuracy with DT was 0.9000 but with no information about IIP. Using the average of accuracy and IIP as a criterion, LSTM had a better result (0.9028) with $\gamma = 0.8$ when compared to RF (0.8629). Using the average of accuracy and F1 as a criterion, LSTM had also a better result (0.9594) with $\gamma = 0.9$ when compared to DT (0.8850).

This comparison highlights the importance of selecting criteria that allow a wider analysis of performance metrics that best unveil the strengths and weaknesses of a given classifier and methodology. Table 26 shows the approach here proposed is a good candidate under different requirements.

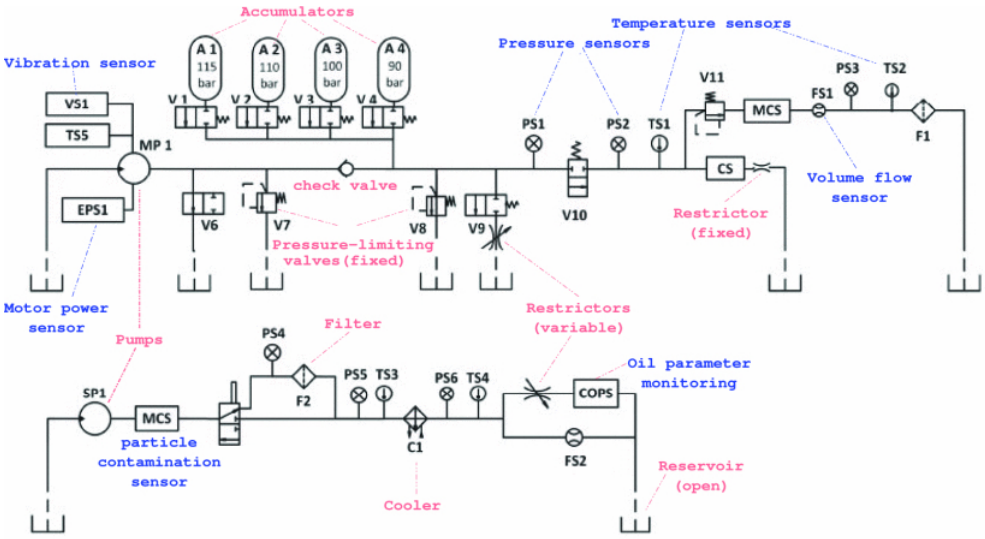
6.2 Condition Monitoring of a Hydraulic System

The second experimental application is based on a dataset produced by Helwig, Pignanelli & Schütze (2015a). The authors provide a publicly available dataset from one hydraulic system for the study of typical hydraulic system failures based on sensor readings.

The data set was experimentally obtained with a hydraulic test rig, depicted in Figure 41. This test rig consists of a primary working and a secondary cooling-filtration circuit which are connected via the oil tank (HELWIG; PIGNANELLI; SCHÜTZE, 2015a), (HELWIG; PIGNANELLI; SCHÜTZE, 2015b).

The test system is equipped with multiple sensors that measure various process values. These values include pressure (PS1 - PS6), flow (FS1, FS2), temperature (TS1 - TS4), electrical power (EPS1), and vibration (VS1). The sensors are connected to a data acquisition system using standard industrial 20 mA current loop interfaces. Additionally, the system integrates sensors for particle contamination (CS and MCS) and oil parameter monitoring (COPS) with digital EIA232 and EIA-485 bus interfaces. The sampling rates

Figure 41 – Test rig hydraulic system



Source: Keleko et al. (2023)

for the sensors vary depending on the dynamics of the physical values being measured, ranging from 100 Hz for pressure to 1 Hz for temperature. During runtime, the sensor data are collected and buffered on a PLC (Beckhoff CX5020) and then transferred to a PC via EtherCAT, where they are stored (HELWIG; PIGNANELLI; SCHÜTZE, 2015a).

The oil temperature, which is influenced by the duty cycle of the cooler fan, varies across different values. At a 100% duty cycle, the temperature is 44 °C, at a 20% duty cycle, it is 55 °C, and at a 3% duty cycle, it reaches 66 °C. The viscosity of the oil is exponentially related to temperature, so the condition of the dominant cooler strongly affects the characteristics of each fault. The valve current set-points are defined as 100%, 90%, 80%, and 73% of the nominal value. The internal pump leakage levels are caused by cascading 0.2 mm and 0.25 mm orifices. The pre-charge pressure steps of the accumulator are set at 115, 110, 100, and 90 bar (HELWIG; PIGNANELLI; SCHÜTZE, 2015a). The sensor data are described in Table 27.

Table 27 – Sensors Used in the Hydraulic Test Rig.

Name	Description	Unit	Rate (Hz)
PS1-PS6	Pressure	bar	100
EPS1	Motor Power	W	100
FS1, FS2	Volume Flow	l/min	10
TS1-TS4	Temperature	°C	1
VS1	Vibration	mm/s	1
CE	Cooling efficiency (virtual)	%	1
CP	Cooling power (virtual)	kW %	1
SE	Efficiency factor	%	1

The system repetitively performs fixed load cycles lasting 60 seconds and collects data on process variables such as pressures, volume flows, and temperatures. At the same time, it systematically varies the quantitative condition of four hydraulic components (cooler, valve, pump, and accumulator). The dataset consists of 2205 instances, corresponding to the measurements taken during these load cycles. Each instance is labeled with one of five class labels, with each label having 2-4 severity levels representing different states. These classes are assigned numeric values and coexist simultaneously. However, class number 5, referred to as the stable flag, signifies degradation processes over time and does not indicate distinct categories. One challenge is to accurately differentiate one class from the others, particularly considering the concurrent failures. Additionally, managing time series data with varying sampling frequencies, as well as distinct underlying periodicity and trends, presents another challenge. The target variables and the associated states for each component severity level are described in Table 28.

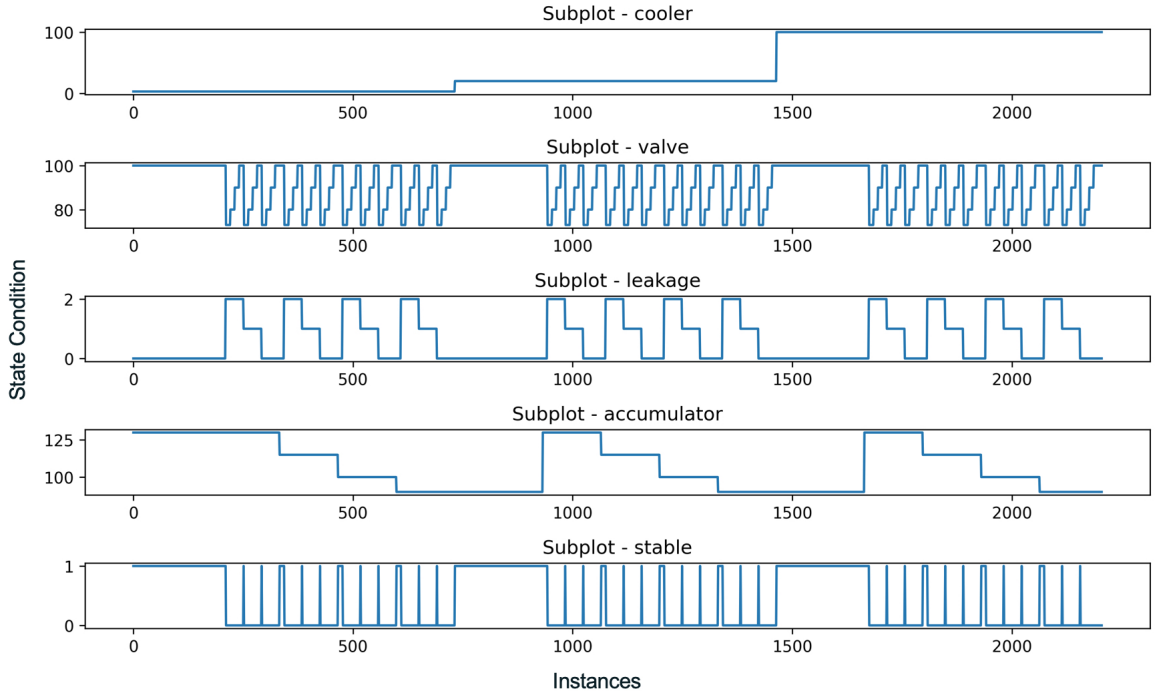
Table 28 – Summary categorical variables of the hydraulic test bench, with each variable representing a multi-state operating condition of the system.

Condition Variable	State	Instances	Target Value
Cooler condition (%)	- Close to total failure	732	3
	- Reduced efficiency	732	20
	- Full efficiency	741	100
Valve condition (%)	- Optimal switching behavior	1125	100
	- Small lag	360	90
	-Severe lag	360	80
	- Close to total failure	360	73
Internal pump leakage	- No leakage	1221	0
	- Weak leakage	492	1
	- Severe leakage	492	2
Hydraulic accumulator	- Optimal pressure	599	130
	- Slightly reduced pressure	399	115
	- Severely reduced pressure	399	100
	- Close to total failure	808	90
Stable flag	- Conditions were stable	1449	0
	- Conditions may be unstable	756	1

Figure 42 illustrates the labeling of each of the 2205 instances in the dataset with one of five class labels. Each class label is associated with a severity level that represents distinct states. Each class label is associated with a severity level that represents distinct states. For example, instance zero has the labels [3, 100, 0, 130, 1]

Drawing a comparison between the 3W dataset and the ICM dataset, the five faulty classes in the ICM dataset correspond to the eight undesirable events in the 3W dataset. In the ICM dataset, the designated states for each class comprise two, three, or four different conditions, while in the 3W dataset, the three predefined states are: normal, transient, and faulty. In the ICM dataset, each instance is assigned a single label that

Figure 42 – Each of the 2205 instances in the dataset is labeled with five class labels. Each class label is associated with a severity level that represents distinct states.



Source: The Author

includes multiple concurrent faults, whereas in the 3W dataset, each instance consists of a single event composed of the three labels: normal, transient, and fault. 3W has three different sources of data collection (real, simulated, and hand-drawn), while in the ICM the data set was experimentally obtained with a hydraulic test rig.

6.2.1 Clustering by Similarity Result

In this section, our main objective is to showcase the practical implementation of the clustering by similarity methodology on the ICM dataset. The purpose is to establish the feasibility of the methodology across various types of datasets and its potential to enhance the performance of one-class classification (OCC) models. By applying the clustering by similarity methodology to the ICM dataset, we seek to provide evidence of its effectiveness in improving the performance of OCC models. Through our investigation of the ICM dataset, we aim to emphasize the generalizability and potential benefits of employing the clustering by similarity methodology in the field of OCC. The OCC model used to evaluate was the LSTM Autoencoder.

It is important to assess the feasibility of applying the methodology to this dataset. The four faulty classes were investigated. Considering the multiple states of the faulty conditions, with OCC it is necessary to define the state(s) as a target. Table 29 presents the target states of each Class. For example, for the cooler condition class, the states of

"Close to total failure (3)" and "Reduced efficiency (20)" were identified as the target class, while "Full efficiency (100)" was designated as the negative class. The criteria used for this was normal condition as negative and all other states were considered as targets, thus the models will be able to detect the faulty in the early stages.

Table 29 – Target states of the Classes.

Class	Target state	Negative state
Cooler	3, 20	100
Valve	73, 80, 90	100
Leakage	1, 2	0
Accumulator	90, 100, 115	130

6.2.1.1 Pre-Processing

This step ensures that the data is properly structured and organized, providing a solid foundation for subsequent processing and modeling tasks. The dataset used in this case study was complete, without missing values. To facilitate further analysis, the initial step was to organize the files based on instances, ensuring a systematic arrangement of the data. Subsequently, the sensor data were downsampled to maintain a consistent frequency rate of 1Hz. This standardization simplifies the comparison and analysis of different instances across time, enabling a more straightforward evaluation of their similarities and differences. It allows for more effective instance comparison, facilitating comprehensive insights and enabling accurate modeling for subsequent stages of the study.

Following the initial steps, the positive instances (target) were partitioned into three sets: 60% for training, 20% for validation, and 20% for testing. The negative instances were split into 50% for validation and 50% for testing. Tables 30 to 33 show the distribution of instances of the Cooler, Valve, Leakage, and Accumulator Classes, respectively. Data standardization was performed using the z-score transformation. This technique involves mapping the data to have zero mean and unit standard deviation. The same standardization process, based on the statistics derived from the training data, was applied to the validation and test sets.

Table 30 – Number of Instances for training, validation, and test - Cooler.

Label	Training	Validation	Test
Positive	878	293	293
Negative	0	370	371
Total	878	663	664

Now that the data standardization is complete, we can move on to the next step which is time series clustering. By using k-means and DBA on the training data, we can detect and extract patterns and similarities within the time series data. This will allow us to create models and perform analysis in the following steps.

Table 31 – Number of Instances for training, validation, and test - Valve.

Label	Training	Validation	Test
Positive	648	778	779
Negative	0	562	563
Total	648	1340	1342

Table 32 – Number of Instances for training, validation, and test - Leakage.

Label	Training	Validation	Test
Positive	590	807	808
Negative	0	610	611
Total	590	1417	1419

Table 33 – Number of Instances for training, validation, and test - Accumulator.

Label	Training	Validation	Test
Positive	963	620	622
Negative	0	299	300
Total	963	919	922

6.2.1.2 Time Series Clustering

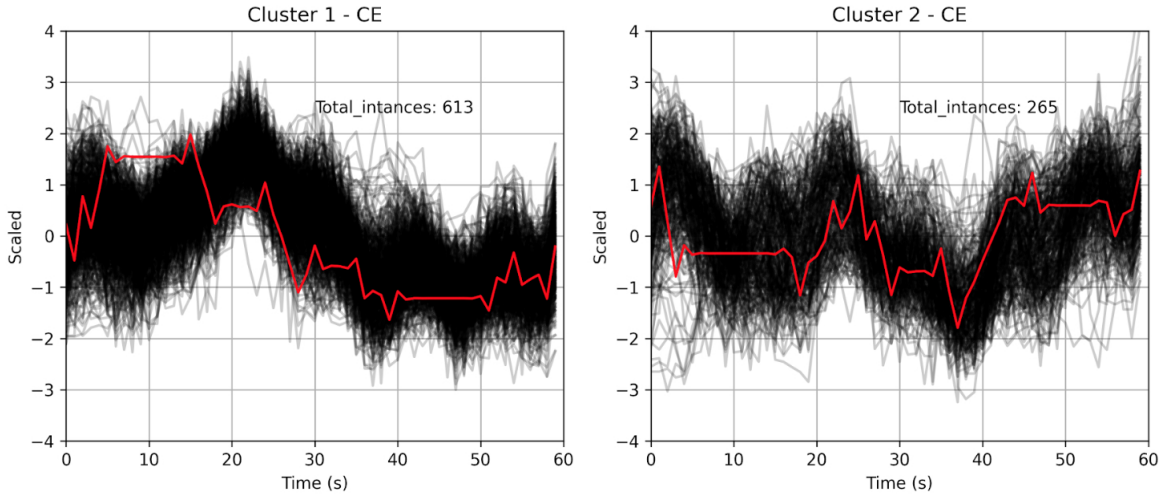
At this step, k-means with DBA was applied to split the time series of the training instances into two groups for each variable based on their similarity. The objective was to maximize data similarity within clusters while minimizing the similarity across clusters, resulting in the clusters $C^1_{x_i}$ and $C^2_{x_i}$, $i = 0, ..., 16$. Figures 43 to 59 show the results of the clustering of the time series of the training set for the Cooler case. The gray lines represent instances and the red lines represent the centroids of the clusters. The amount of instances in each cluster for each variable is shown in the figures. After this process, it is possible to identify that some variables have different behaviors for the same class. It leads to the discovery of interesting patterns in the time series data set.

6.2.1.3 Apriori Clustering Results

Times series clustering generated two clusters for each variable. The Apriori algorithm did the combinations of cluster 1 for all variables, and then for cluster 2 for all variables. Given a support threshold $s = 0.7$, the Apriori algorithm identified the combination of variables whose intersection of their clusters corresponded to at least 70% of instances in the training data set.

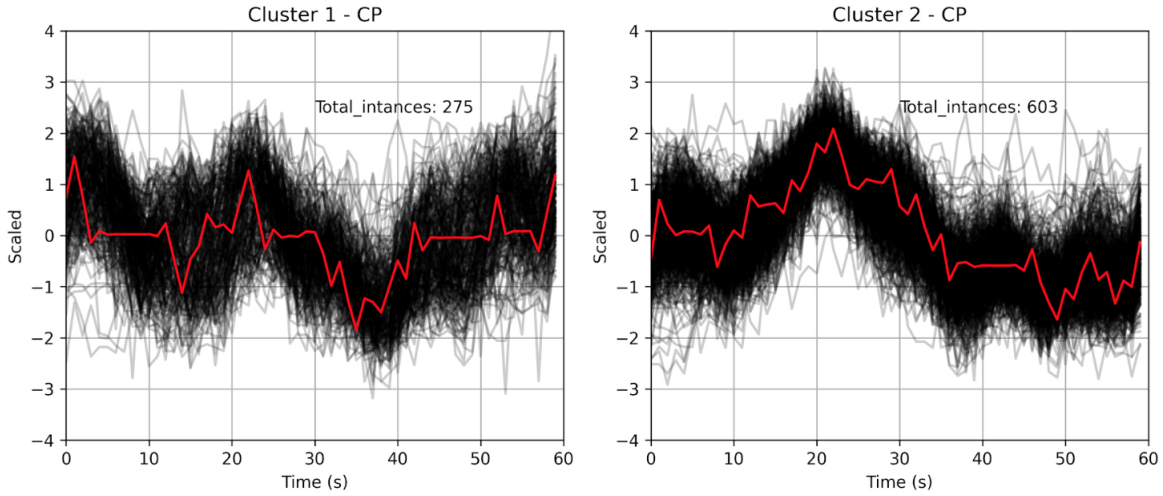
Figure 60 shows subsets of instances that satisfy the Apriori support threshold criterion for the Cooler. In this case, 60 subsets were selected from cluster $C^1_{x_i}$, and 90 subsets were selected from cluster $C^2_{x_i}$. The range of variables in the subsets went from 0 to 16. The next step was to select the subsets from clusters 1 and 2 shown in this figure.

Figure 43 – Cooler - CE similarity clustering



Source: The Author

Figure 44 – Cooler - CP similarity clustering



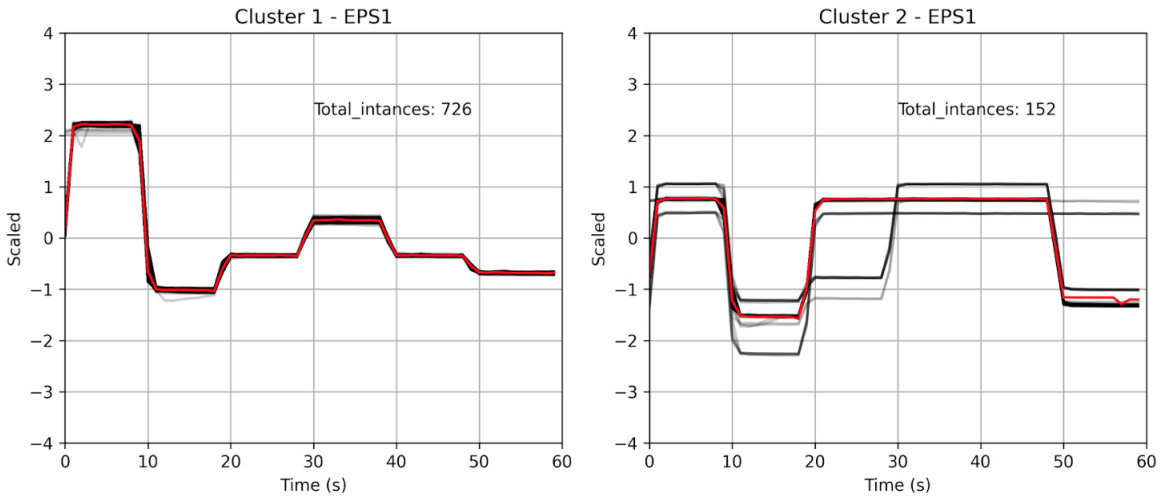
Source: The Author

6.2.1.4 Selection of Cluster Subsets

For each subset item from the Apriori step, one model was trained and validated to choose the cluster subset. To construct the LSTM autoencoder model, the input data is transformed into a 3-D matrix, organized based on samples, time steps, and variables. This matrix structure was essential for building the LSTM autoencoder architecture and enabling effective data processing. In this case, the time steps were specifically configured to encompass 10 consecutive samples, defining the data window size.

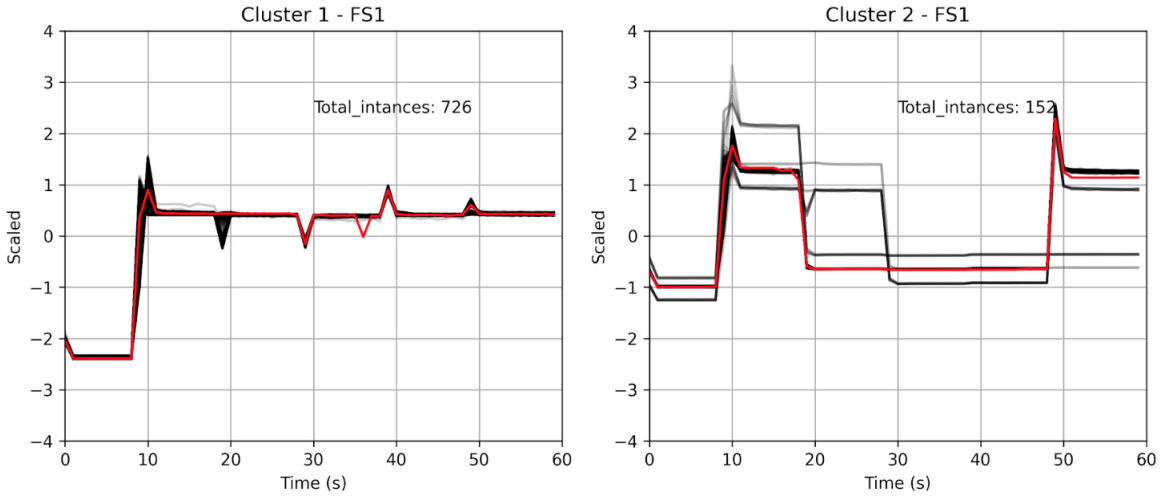
The implementation of the LSTM autoencoder model was executed using the Python programming language. The Keras library, which is a popular deep-learning framework in Python, was utilized for model development and training. To train each

Figure 45 – Cooler - EPS1 similarity clustering



Source: The Author

Figure 46 – Cooler - FS1 similarity clustering

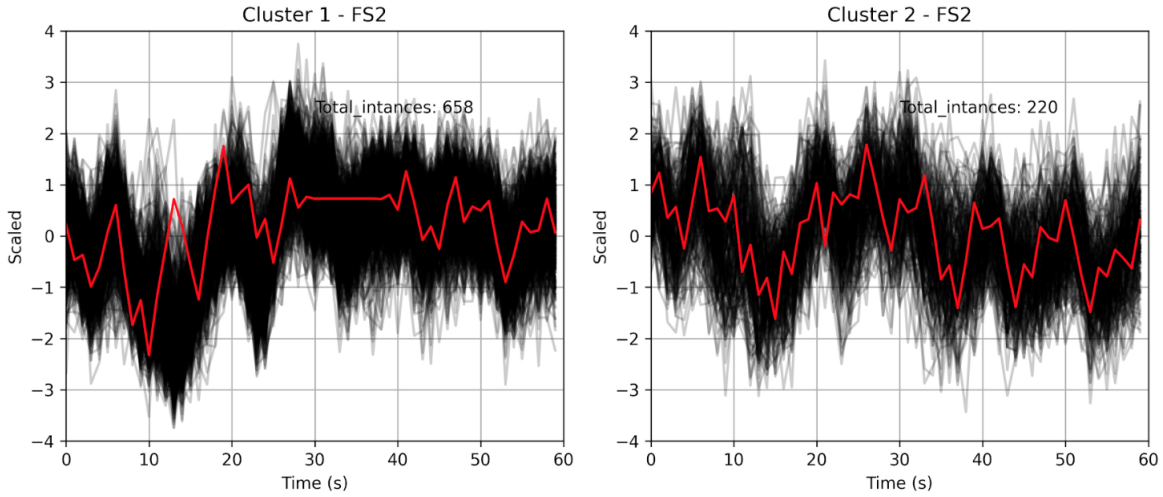


Source: The Author

LSTM autoencoder, the following hyperparameters were used:

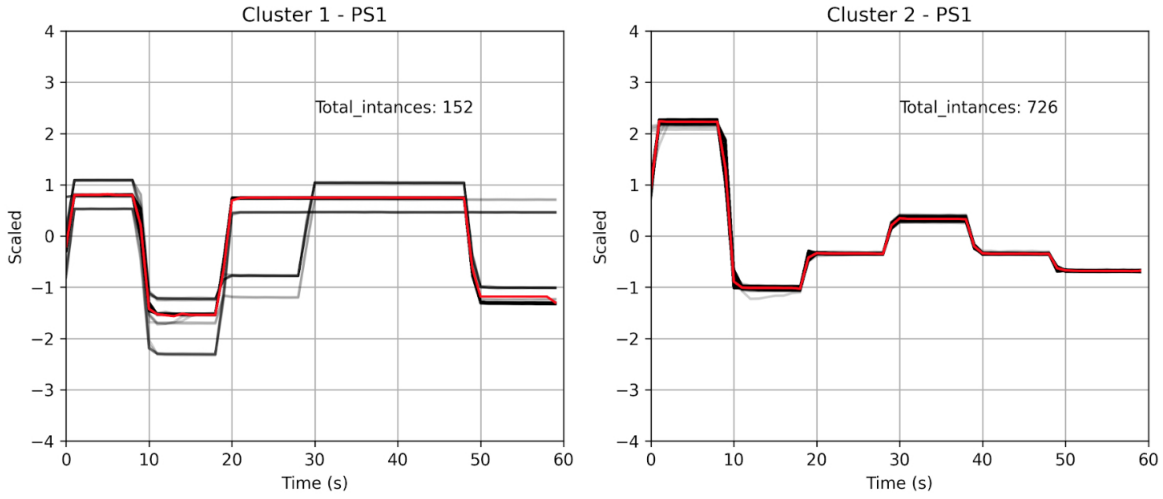
- Time Steps: 10
- Batch Size: 2048
- Number of units in the encoder layer: 32/16
- Number of units in the decoder layer: 16/32
- Epochs: 100
- Learning Rate: 0.0002

Figure 47 – Cooler - FS2 similarity clustering



Source: The Author

Figure 48 – Cooler - PS1 similarity clustering

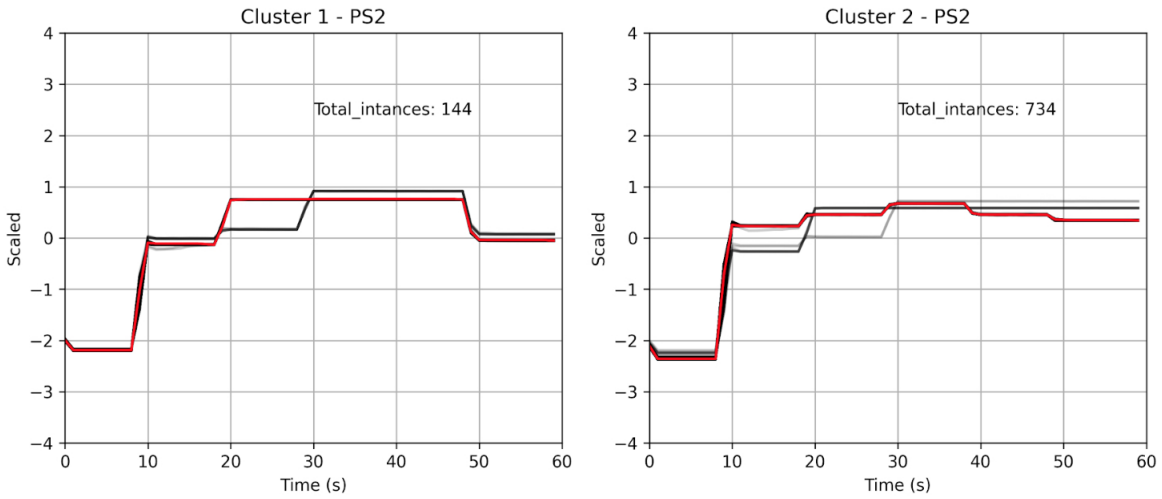


Source: The Author

The Adam algorithm was used as the optimizer and mean square error (MSE) was used as the cost function. A threshold was employed to distinguish between the target class and the others. Samples with test data exhibiting reconstruction errors exceeding this threshold were labeled as negative, while those with errors below it were labeled as positive (target class). As the LSTM was trained on faulty data, a low reconstruction error value indicates the presence of an anomaly in a particular sample at a specific time point.

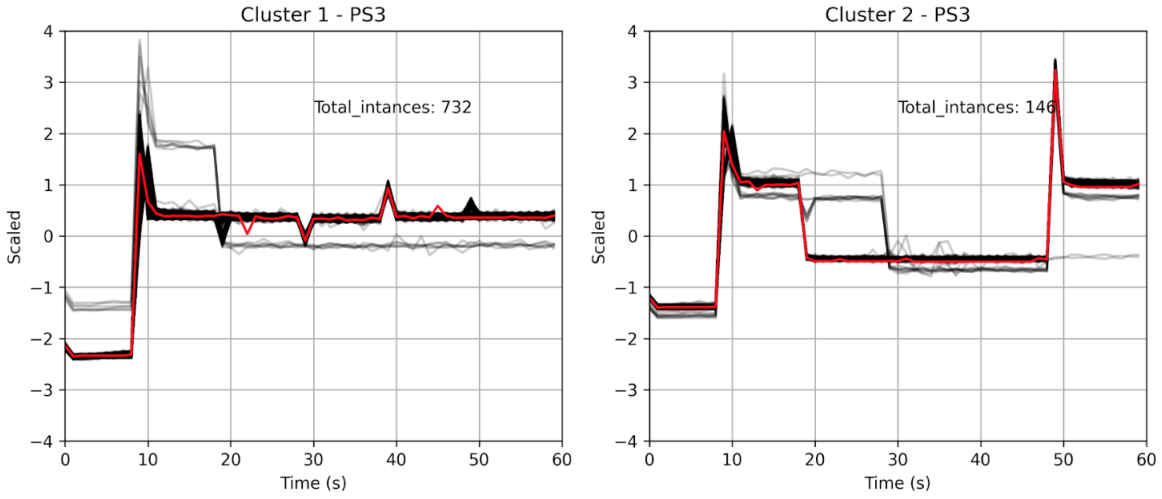
For clusters 1 and 2, the metric F1-score was considered to select the cluster subset that presented the best classification performance. For both clusters, the subset that obtained the best result, using the validation data set, was selected. It is important to emphasize that the validation and test data did not go through the similarity clustering process.

Figure 49 – Cooler - PS2 similarity clustering



Source: The Author

Figure 50 – Cooler - PS3 similarity clustering

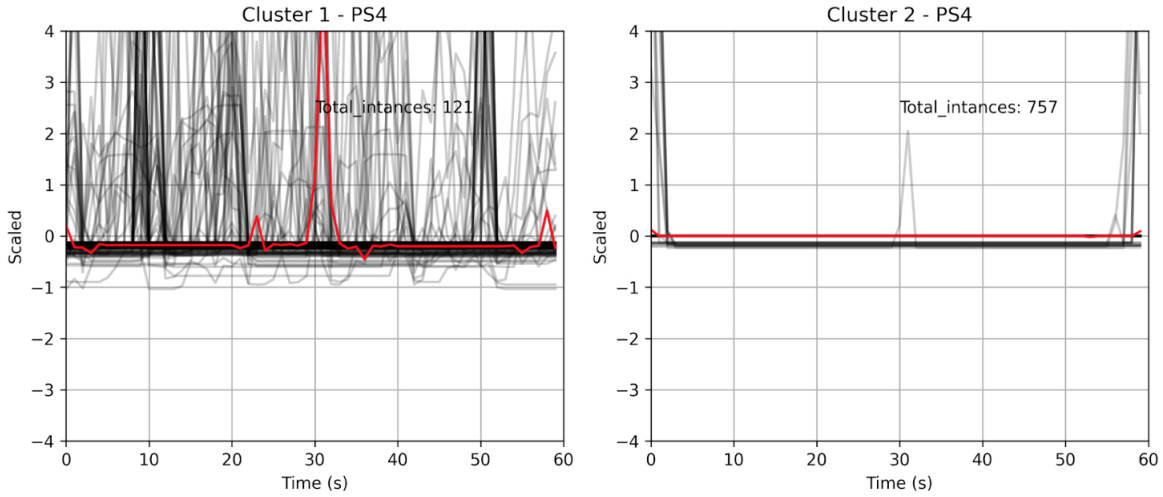


Source: The Author

Reminding that cluster C^b variable selection was done only by availability and there was no variable selection by similarity. So, in this single cluster, the subset with a larger number of variables was desired, according to availability. In this case, there are no missing values, thus all variables were selected for cluster C^b with 100% of training instances.

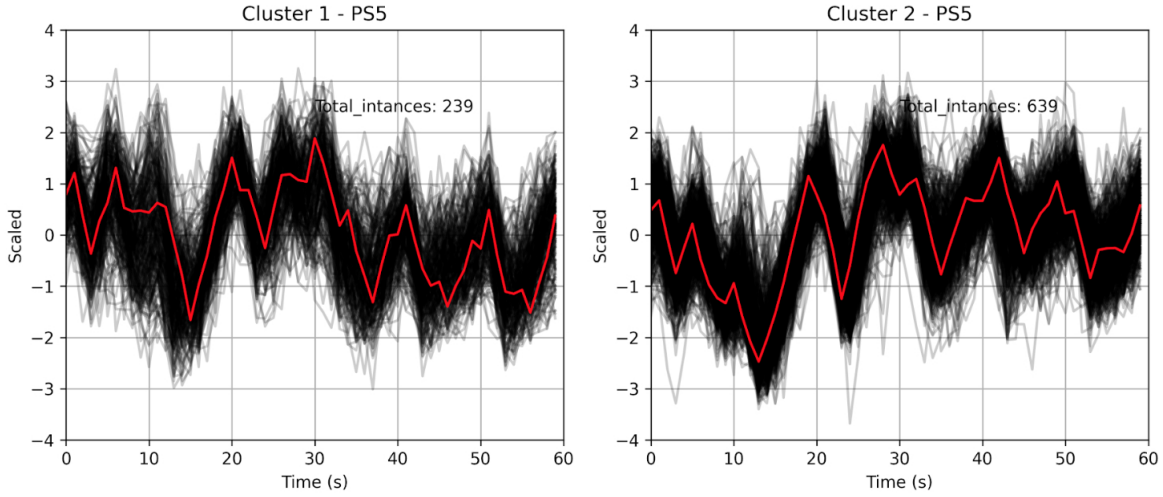
Tables 34 to 37 show the results of the cluster subset selection for Cooler, Valve, Leakage, and Accumulator, respectively. . For the case of the Cooler condition, the selected cluster C^1 had only variable {FS2} with 75% of training instances, and an increase of similarity (IS) of 43%. Cluster C^2 was composed of the variables {PS5, PS6} with 70% of the 878 training instances and IS with 53%. These subsets showed the best classification performance on the validation data set.

Figure 51 – Cooler - PS4 similarity clustering



Source: The Author

Figure 52 – Cooler - PS5 similarity clustering



Source: The Author

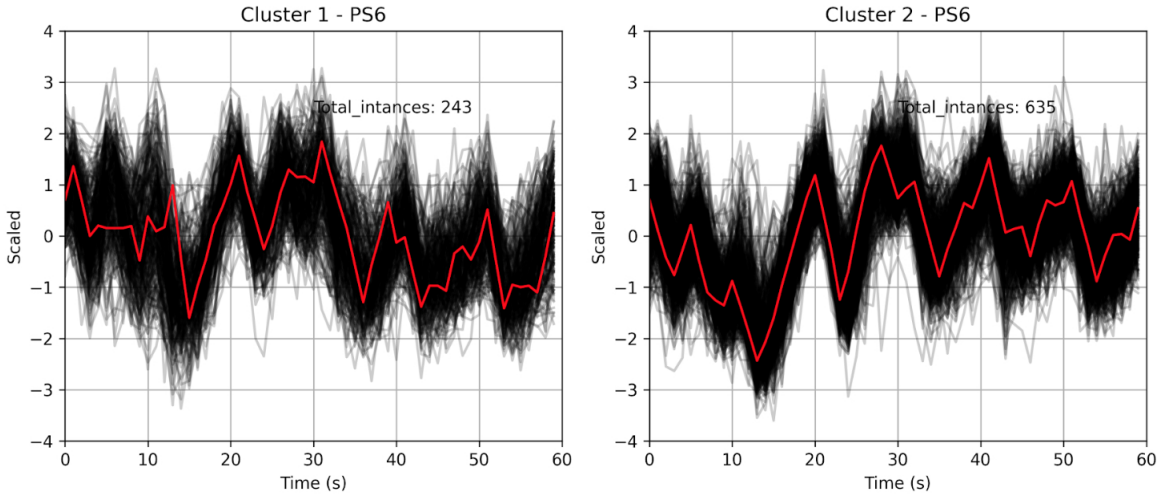
Table 34 – Selection of clusters subset - Cooler.

Cluster	Variables	Support	IS
C^b	All variables	1.0	–
C^1	FS2	0.75	43%
C^2	PS5, PS6	0.70	53%

Table 35 – Selection of clusters subset - Valve.

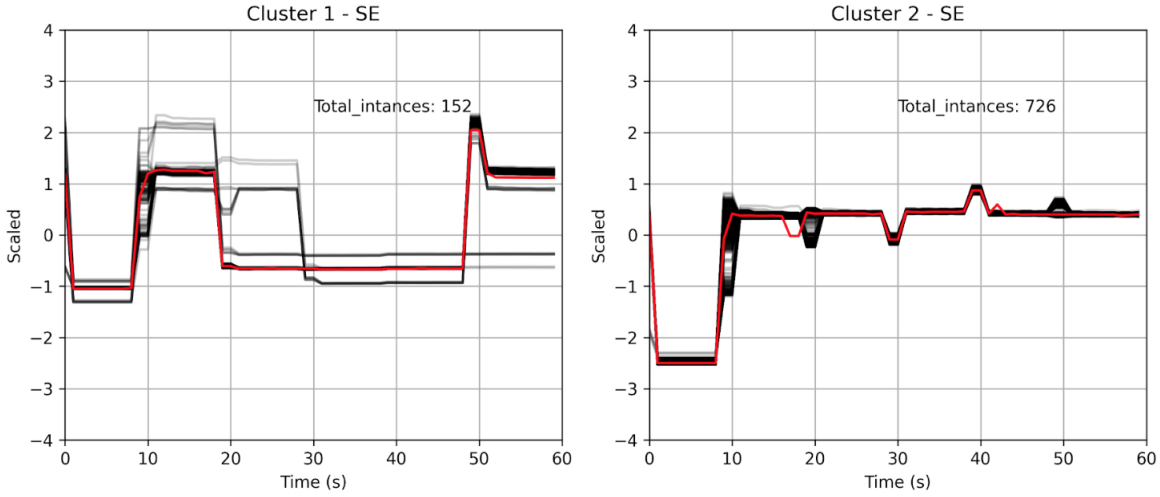
Cluster	Variables	Support	IS
C^b	All variables	1.0	–
C^1	TS1,PS1	0.84	49%
C^2	PS2	0.85	63%

Figure 53 – Cooler - PS6 similarity clustering



Source: The Author

Figure 54 – Cooler - SE similarity clustering



Source: The Author

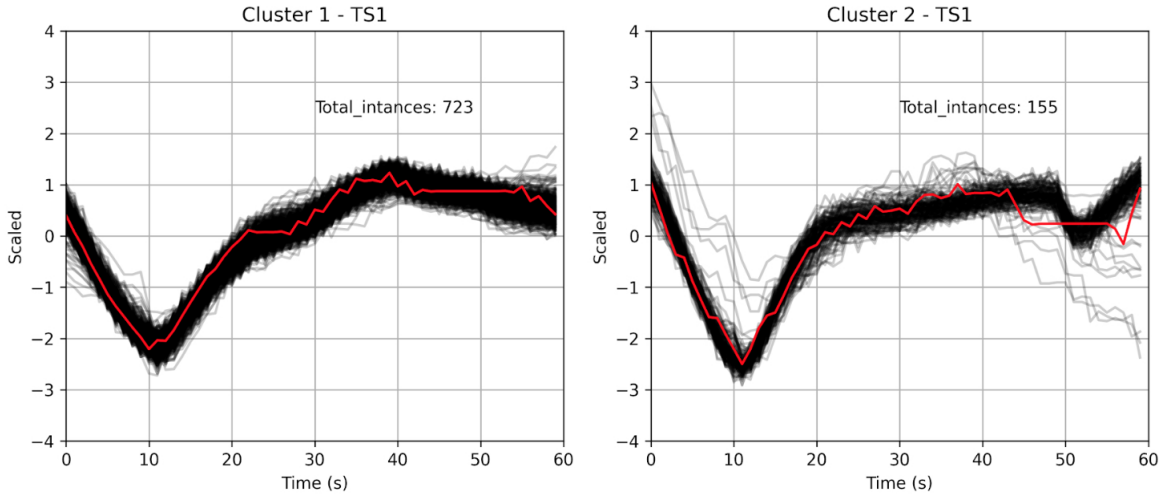
Table 36 – Selection of clusters subset - Leakage.

Cluster	Variables	Support	IS
C^b	All variables	1.0	–
C^1	SE	0.77	29%
C^2	TS1,PS2,PS3	0.77	66%

Table 37 – Selection of clusters subset - Accumulator.

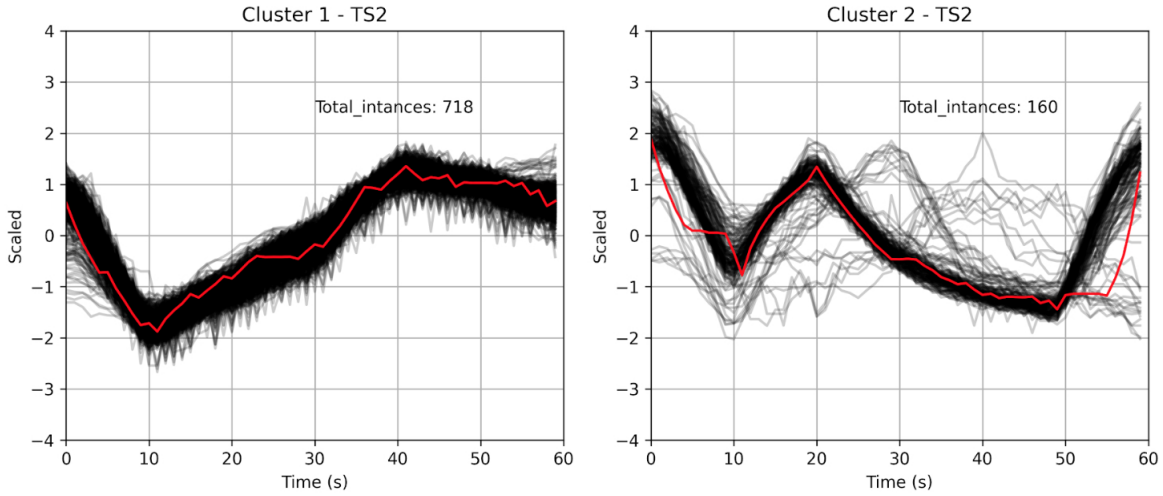
Cluster	Variables	Support	IS
C^b	All variables	1.0	–
C^1	EPS1,PS3	0.84	69%
C^2	SE,TS1	0.83	33%

Figure 55 – Cooler - TS1 similarity clustering



Source: The Author

Figure 56 – Cooler - TS2 similarity clustering



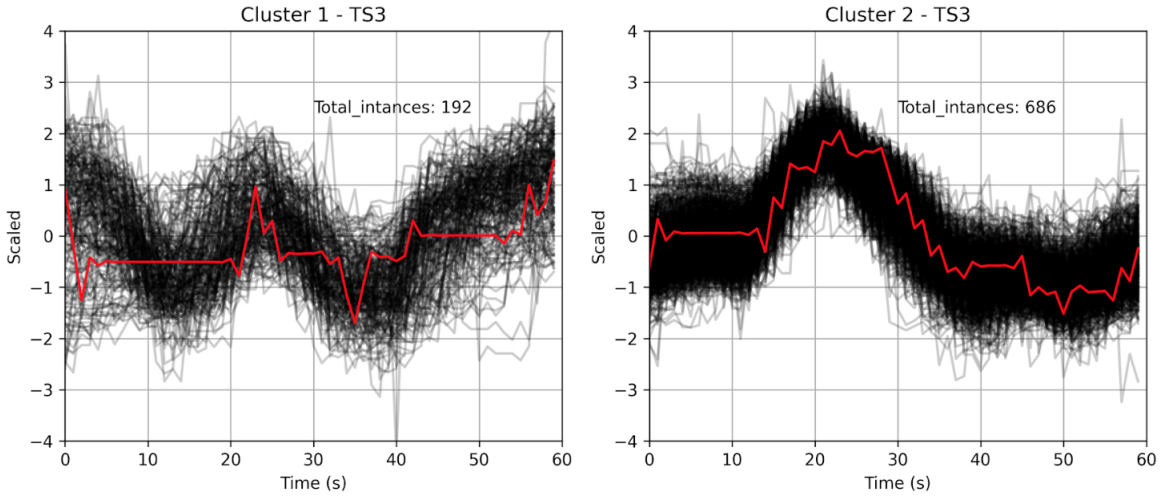
Source: The Author

6.2.1.5 Test Results

In the previous section, the subset was selected to obtain the best classification performance for the selected one-class classifier. In this section, the results for the four classes are presented after evaluating the trained models M_b, M_1, and M_2 using the test set.

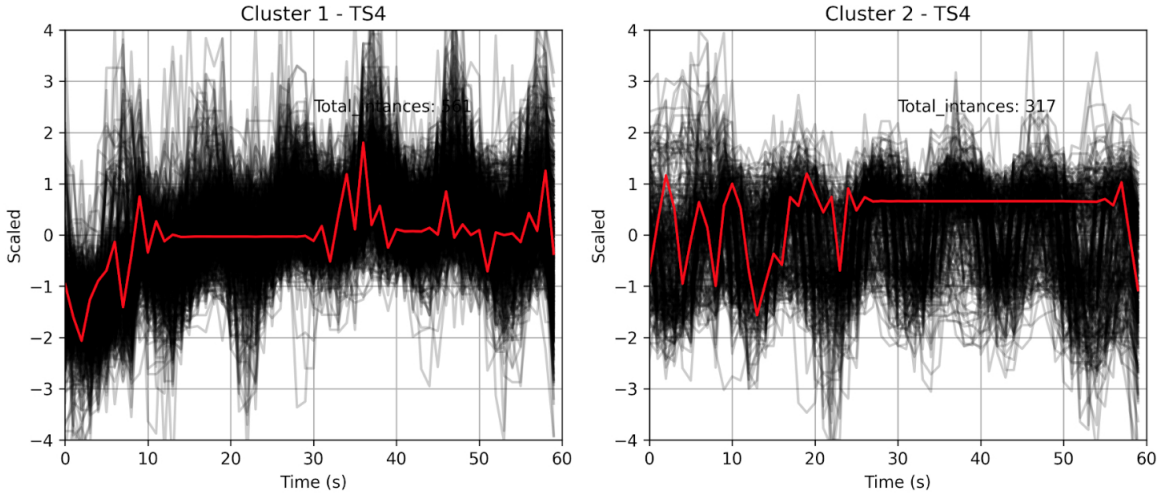
Table 38 showcases the results of the Cooler condition obtained for the models M_b, M_1, and M_2. The IIP is not applicable in the hydraulic dataset, thus μ_M is the mean value of the F1-score and specificity. Among these models, the highest performance was observed in the M_2 model, yielding a remarkable μ_M value of 0.999. However, the M_b model exhibited the weakest results, with a μ_M value of 0.87 and an F1-score of 0.74.

Figure 57 – Cooler - TS3 similarity clustering



Source: The Author

Figure 58 – Cooler - TS4 similarity clustering



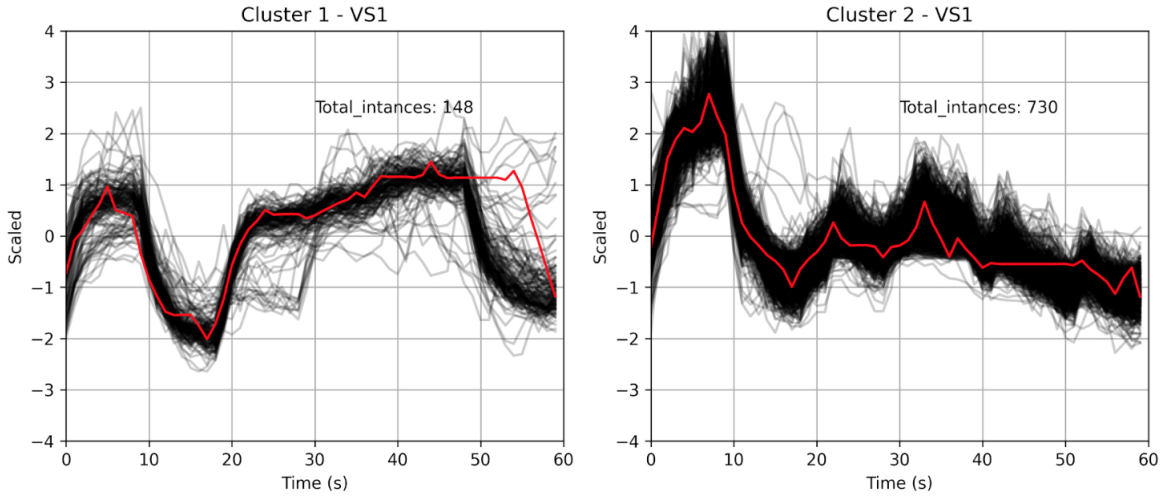
Source: The Author

The models M_1 and M_2 both exhibited high performance, surpassing the M_b model in terms of μ_M and F1-score. These results suggest that these models successfully strike a balance between precision and recall, enabling accurate detection of anomalies while minimizing the occurrence of false positives.

The results of the F1-score comparison for clustering models with M_b are illustrated in Figure 61. The similarity was increased by 43% for model M_1 and 53% for model M_2, and the classification performance between clustering models and M_b was improved by 35%.

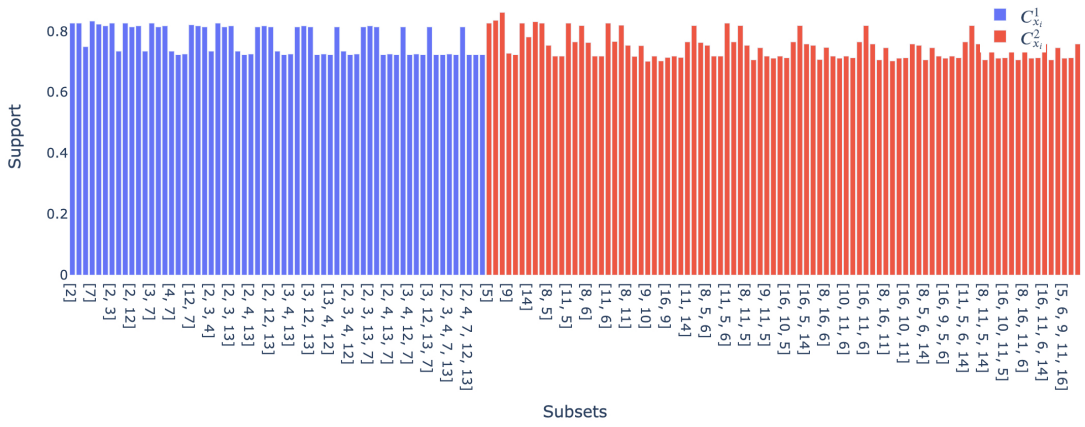
The M_2 model, which utilized only two variables and 70% of the training instances, achieved maximum performance. Similarly, the M_1 model, relying on just one variable and 75% of the training instances, also demonstrated superior anomaly detection capabilities.

Figure 59 – Cooler - VS1 similarity clustering



Source: The Author

Figure 60 – Cooler - Apriori Clustering Subsets



Source: The Author

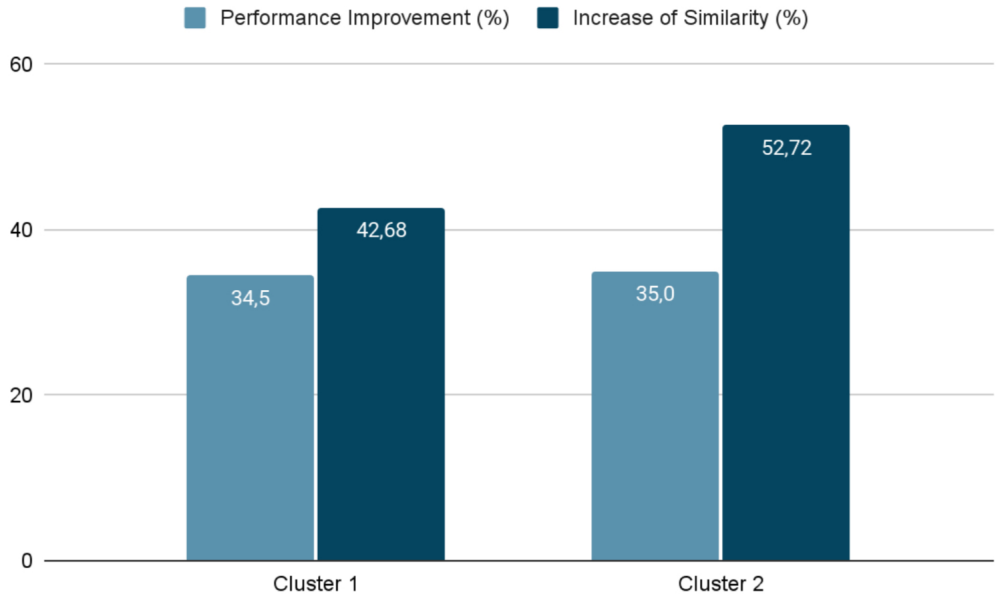
Table 38 – ICM Dataset - Cooler Models Test Results.

Model	Variables	Support	IS	F1	Specificity	μ_M
M_b	All variables	1.0	–	0.74	1.0	0.87
M_1	FS2	0.75	43%	0.995	0.995	0.995
M_2	PS5, PS6	0.70	53%	0.998	1.0	0.999

Table 39 shows the results of the Class Valve obtained for the models M_b, M_1, and M_2 on the test set. The highest performance was observed in the M_2 model, yielding μ_M value of 0.92, F1-score of 0.85, and specificity value of 1.0 with only one variable $PS2$ and 85% of the training set instances. Compared with both M_b and M_1, the M_2 model exhibited the best F1-score and specificity. It is important to emphasize that the similarity of C^2, utilized to train M_2, was increased by 63%.

The results of the Class Leakage are presented in Table 40. Model M_1 achieved the

Figure 61 – Cooler - Performance Improvement



Source: The Author

Table 39 – ICM Dataset - Valve Models Test Results.

Model	Variables	Support	IS	F1	Specificity	μ_M
M_b	All variables	1.0	–	0.47	0.46	0.46
M_1	TS1,PS1	0.84	49%	0.54	0.74	0.64
M_2	PS2	0.85	63%	0.85	1.0	0.92

best result of all classification metrics μ_M of 0.93, F1-score value of 0.87, and specificity of 0.99. These results were attained with just one variable, SE, and 77% of the training set instances. The IS (increase of similarity) value of M_1 was 29%.

Table 40 – ICM Dataset - Leakage Models Test Results.

Model	Variables	Support	IS	F1	Specificity	μ_M
M_b	All variables	1.0	–	0.53	0.61	0.57
M_1	SE	0.77	29%	0.87	0.99	0.93
M_2	TS1,PS2,PS3	0.77	66%	0.82	0.97	0.90

Table 41 presents the results of the Class Accumulator. Model M_2 achieved the highest result of μ_M. Model M_1 obtained the best F1-score with an increase of similarity (IS) value of 69%. M_1 and M_2 achieved superior classification results compared with M_b, reducing the number of variables from 17 to 2, and using 84% of training instances. Models M_1 and M_2 presented IS of 69% and 33%, respectively.

Table 42 displays the F1-score results for the four classes across the three models, the increase of similarity (IS), as well as the improvement in classification performance relative to M_b. Across all classes, both clustering models, M_1 and M_2, increased the similarity and outperformed M_b. Notably, in the Cooler class, both models achieved an

Table 41 – ICM Dataset - Accumulator Models Test Results.

Model	Variables	Support	IS	F1	Specificity	μ_M
M_b	All variables	1.0	–	0.74	0.66	0.70
M_1	EPS1,PS3	0.84	69%	0.78	0.69	0.74
M_2	SE,TS1	0.83	33%	0.73	0.84	0.78

F1-score of 1.0. However, the Accumulator condition posed the greatest challenge for classification. The Leakage Class exhibited the most significant performance improvement of 64%, while the Accumulator Class showed a minor improvement. In addition, this approach achieved significant reductions in the number of variables and instances required for training models and event identification. The hydraulic system dataset was successfully compressed from 17 variables to as few as 3 or even 1.

Table 42 – All Classes Test Results - F1-Score.

Class	M_b	M_1	M_2	IS	Improvement
Cooler	0.74	1.0	1.0	53%	35%
Valve	0.47	0.54	0.85	63%	57%
Leakage	0.53	0.87	0.82	29%	64%
Accumulator	0.74	0.78	0.73	69%	5%

6.2.1.6 Classifiers Fusion Results

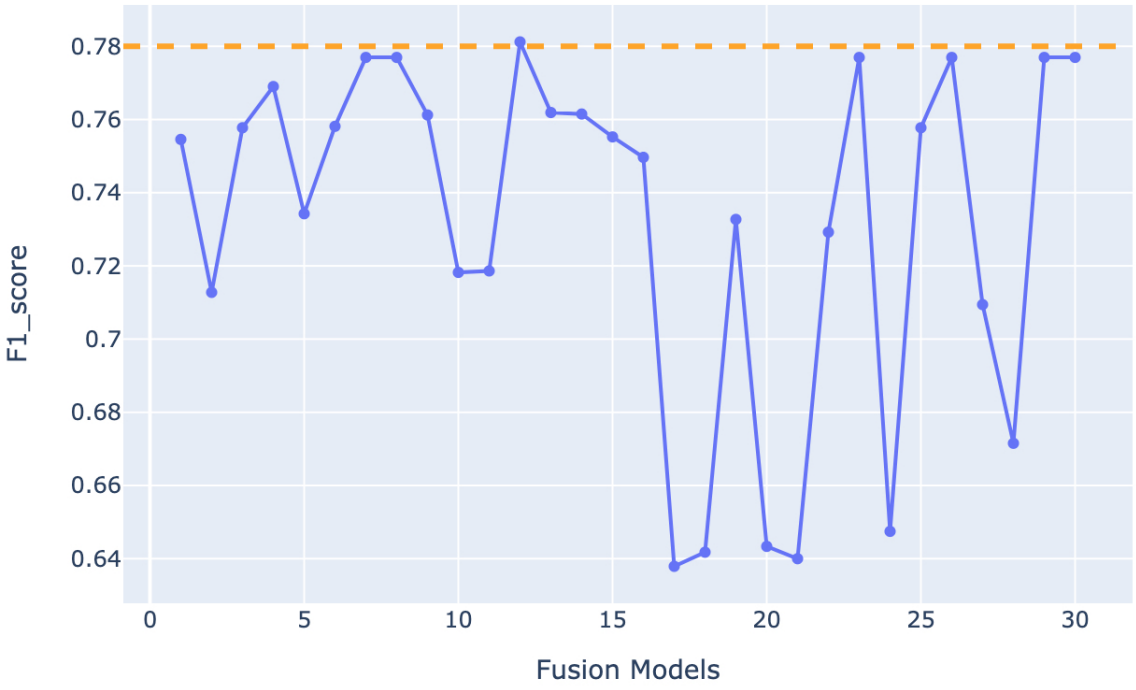
After conducting tests on the models and confirming gains within the methodology, a fusion strategy for combining the models was implemented to enhance the proposed method. This fusion involved employing approaches detailed in Section 5.8. This section aims to show the results of the fusion strategies.

Fusion approach #2 was evaluated; however, it did not show any improvement compared to individual models M_1 and M_2. Figure 62 shows the result of the fusion #2 applied to the Class Accumulator. The x-axis represents the number of fusion models and the y-axis F1-score. For Class Accumulator, M_1 achieved the best F1-score of 0.78 (refer to Table 41). The F1-score for Fusion #2 did not surpass the M_1 represented by the dashed line.

The outcomes of fusion approaches #1 and #3 are provided in Tables 43 and 45, respectively. Table 43 shows the results of fusion #1. While the fusion model M_3 achieves an F1-score of 1.0 for the Class Cooler, maintaining the performance of models M_1 and M_2, it does not outperform the individual models for the other three classes (see Table 44).

Table 43 compares the F1-score of model M_3 against M_1 and M_2. While M_3 continues to outperform model M_b, it does not surpass the best performance achieved among the models.

Figure 62 – Result Fusion Approach #2 - Accumulator



Source: The Author

Table 43 – ICM Dataset - Fusion #1 Test Results.

Class	Model	F1	Specificity	μ_M
Cooler	M_3	1.0	0.99	0.99
Valve	M_3	0.62	0.74	0.68
Leakage	M_3	0.84	0.97	0.90
Accumulator	M_3	0.76	0.60	0.68

Table 44 – Test Results Comparing Models with Fusion #1 - F1-Score.

Class	M_b	M_1	M_2	M_3
Cooler	0.74	1.0	1.0	1.0
Valve	0.47	0.54	0.74	0.62
Leakage	0.53	0.87	0.82	0.84
Accumulator	0.74	0.78	0.73	0.76

Table 45 shows the classification results of the fusion approach #3. Table 46 compares the F1-score of fusion models that achieve the best performance against the models M_1 and M_2. The fusion model $M22'$ maintains the maximum performance for the Cooler. In the Class Valve, the model $M22'$ surpasses all other models with an F1-score value of 0.89 and achieved the most significant performance improvement of 89% related to M_b. Model $M11'$ outperforms the classification performance for Class Leakage with an improvement of 85%. The accumulator condition still poses the challenge for classification, however, the model $M11'$ improves the performance at 12% with an F1-score of 0.83.

Table 46 presents the best model's performance and the improvement achieved

Table 45 – ICM Dataset - Fusion #3 Test Results.

Class	Model	F1	Specificity	μ_M
Cooler	$M11'$	0.85	0.73	0.79
Cooler	$M22'$	1.0	1.0	1.0
Valve	$M11'$	0.61	0.70	0.65
Valve	$M22'$	0.89	0.95	0.92
Leakage	$M11'$	0.98	0.99	0.98
Leakage	$M22'$	0.91	0.95	0.93
Accumulator	$M11'$	0.83	0.69	0.76
Accumulator	$M22'$	0.79	0.78	0.79

Table 46 – Test Results Comparing Models with Fusion #3 - F1-Score.

Class	M_b	M_1	M_2	$M11'$	$M22'$	Improvement
Cooler	0.74	1.0	1.0	0.85	1.0	35%
Valve	0.47	0.54	0.74	0.61	0.89	89%
Leakage	0.53	0.87	0.82	0.98	0.91	85%
Accumulator	0.74	0.78	0.73	0.83	0.79	12%

compared with M_b.

Table 47 – All Classes Best Models Results - F1-Score.

Class	M_b	F1-Score (Model)	Improvement
Cooler	0.74	1.0 ($M22'$)	35%
Valve	0.47	0.89 ($M22'$)	89%
Leakage	0.53	0.98 ($M11'$)	85%
Accumulator	0.74	0.83 ($M11'$)	12%

Based on the results, the fusion approach #3 emerges as the most effective strategy. This approach demonstrably improves classification performance across all classes, with particularly significant improvements observed in the Valve and Leakage classes. The Leakage class nearly achieves the maximum possible classification value.

6.3 Advantages, Disadvantages, and Limitations

The time series clustering method proved to be suitable for clustering data by similarity to training OCC models. The decisions about the selection of subsets of instances for training classifiers considered the problem of a small amount of data. More clusters can be used if more data are available. Three fusion approaches of classifiers were proposed and performed, however, more complex forms to combine the decisions can be proposed (KRAWCZYK; WOŹNIAK; CYGANEK, 2014). More classifiers can be also considered in the phase of training and fusion of their outputs. In addition, other one-class classifiers can be used, since the purpose of the method is to select clusters of training data of the target class to improve the performance of the classifiers.

The application of this approach to multi-class classifiers is not straightforward. Since they must be trained with instances of positive and negative classes, the clustering by similarity should be applied to all classes. For the negative class, each class should be submitted to clustering, generating subsets. The combination of the subsets for each class should be evaluated using the classifier to select the subset of all instances of negative classes that together with the positive class provide the best performance. The computational effort would be prohibitive. Thus, further studies are required to consider a methodology based on clustering by the similarity that can be applied to multi-class classifiers.

7 Conclusion

This work proposes methodologies aimed at improving the performance of one-class classifiers through time-shift labels and time series clustering. The clustering process involves utilizing DBA and k-means to group multivariate time series based on their similarity. To handle the distribution of time series instances, the Apriori algorithm is employed to generate subsets of instances, which are used to train multiple one-class classifiers for the same class. The effectiveness of the proposed method is evaluated on two public datasets: one comprising data from different oil wells under varying conditions, and the other containing data from a hydraulic system to study typical failures.

The results show that increasing the training data's similarity leads to enhanced classifier performance. This improvement is assessed by comparing LSTM Autoencoder models with and without clustered data. The methodology applied to the 3W dataset, in specific cases, models achieving an 84% increase in similarity yielded a 21% improvement in classification performance. Similarly, a 63% increase in similarity led to a 57% improvement in classification, resulting in a 10% overall performance gain.

For condition monitoring of hydraulic system data, the proposed method achieved a significant performance improvement of over 40% compared to the baseline model. Notably, in the specific case of leakage fault, the classification performance improvement rises by 64%.

Moreover, the fusion approaches of classifier outputs are a good alternative to increase classifier performance. Three fusion approaches were proposed and investigated in the hydraulic system. We can highlight approach #3, which improved by 89% the performance of valve condition, achieved an F1-score value of 0.98 for the classification of leakage Class, and improved by 55% the overall classification performance. In addition, the methodology successfully reduced a hydraulic system dataset from 17 variables to as few as 3 or even 1, while still achieving better classification performance.

Additionally, the effect of the time-shifting of labels on classifier performance metrics was evaluated on the 3W dataset. During the training phase, thresholds were used to shift the labels of process variables, reducing the impact of overlapping data. To measure performance, a metric was proposed that considers Specificity, F1, and instance identification performance, and is less affected by unbalanced data. For Class 8 (Hydrates in Production Lines), the performance of LSTM improved by 41%, and SVM by 21%. For Class 2 (Spurious DHSV Closure), LSTM improved by 14%, and no improvement for SVM.

Finally, the findings reveal that the proposed methodologies lead to enhanced OCC

performance and can be extended beyond this study, as they can be adapted to diverse applications, and integrated with a variety of clustering algorithms and OCC methods.

7.1 Future Works

Further studies can consider some topics, such as:

- Evaluate the effect of combining the clustering method with time-shift. This investigation could explore how time-shift might impact clustering outcomes and ultimately classification performance;
- Investigate alternative criteria for clustering data to potentially improve the effectiveness of the approach. This could involve exploring methods that consider not only similarity but also other factors, such as data distribution;
- Explore new criteria for selecting one-class classifiers for fusion;
- Research new functions for merging classifier outputs;
- Extending the proposed approach to accommodate multi-class classification scenarios. This would allow the methodology to address situations with multiclass models;
- While the current study showcases the proposed methods' effectiveness, further research could involve comparing their performance against other established one-class classifiers and time series clustering techniques. This would provide a broader understanding of the approach's relative strengths and weaknesses.

References

AFLORI, C.; CRAUS, M. Grid implementation of the apriori algorithm. Advances in engineering software, Elsevier, v. 38, n. 5, p. 295–300, 2007. Cited on page 40.

AGHABOZORGI, S.; SHIRKHORSHIDI, A. S.; WAH, T. Y. Time-series clustering–a decade review. Information systems, Elsevier, v. 53, p. 16–38, 2015. Cited 4 times on pages 23, 37, 38 e 39.

ALI, M.; JONES, M. W.; XIE, X.; WILLIAMS, M. Timecluster: dimension reduction applied to temporal data for visual analytics. The Visual Computer, Springer, v. 35, n. 6-8, p. 1013–1026, 2019. Cited on page 37.

AMER, M.; GOLDSTEIN, M.; ABDENNADHER, S. Enhancing one-class support vector machines for unsupervised anomaly detection. In: Proceedings of the ACM SIGKDD workshop on outlier detection and description. [S.l.: s.n.], 2013. p. 8–15. Cited on page 85.

BISHOP, C. M. Pattern Recognition and Machine Learning. [S.l.]: Springer, 2006. Cited on page 50.

CHATTERJEE, S.; DAS, A. K.; GHOSH, K.; BANERJEE, A.; BHATTACHARJEE, M.; BANERJEE, S. Performance improvement of artificial neural networks by addressing class overlapping problem. In: SPRINGER. Advances in Smart Communication Technology and Information Processing: OPTRONIX 2020. [S.l.], 2021. p. 229–237. Cited on page 26.

CHOI, K.; YI, J.; PARK, C.; YOON, S. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. IEEE access, IEEE, v. 9, p. 120043–120065, 2021. Cited 2 times on pages 23 e 28.

CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. Information Fusion, Elsevier, v. 41, p. 195–216, 2018. Cited on page 45.

DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; WANG, X.; KEOGH, E. Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment, VLDB Endowment, v. 1, n. 2, p. 1542–1552, 2008. Cited on page 37.

DONG, X.; YU, Z.; CAO, W.; SHI, Y.; MA, Q. A survey on ensemble learning. Frontiers of Computer Science, Springer, v. 14, p. 241–258, 2020. Cited 2 times on pages 24 e 25.

FRAGOSO, R. C.; CAVALCANTI, G. D.; PINHEIRO, R. H.; OLIVEIRA, L. S. Dynamic selection and combination of one-class classifiers for multi-class classification. Knowledge-Based Systems, Elsevier, v. 228, p. 107290, 2021. Cited 4 times on pages 24, 25, 31 e 46.

HE, G.; DUAN, Y.; QIAN, T.; CHEN, X. Early prediction on imbalanced multivariate time series. In: Proceedings of the 22nd ACM international conference on Information & Knowledge Management. [S.l.: s.n.], 2013. p. 1889–1892. Cited on page 49.

HELWIG, N.; PIGNANELLI, E.; SCHÜTZE, A. Condition monitoring of a complex hydraulic system using multivariate statistics. In: IEEE. 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. [S.l.], 2015. p. 210–215. Cited 2 times on pages 93 e 94.

HELWIG, N.; PIGNANELLI, E.; SCHÜTZE, A. D8. 1-detecting and compensating sensor faults in a hydraulic condition monitoring system. Proceedings SENSOR 2015, p. 641–646, 2015. Cited on page 93.

HEMPSTALK, K.; EIBE, F. Discriminating against new classes: One-class versus multi-class classification. In: 21st Australasian Joint Conference on Artificial Intelligence Auckland. [S.l.: s.n.], 2008. p. 325–336. Cited on page 62.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. Neural computation, MIT press, v. 9, n. 8, p. 1735–1780, 1997. Cited on page 33.

JAVED, A.; LEE, B. S.; RIZZO, D. M. A benchmark study on time series clustering. Machine Learning with Applications, Elsevier, v. 1, p. 100001, 2020. Cited on page 37.

JOVANOVIĆ, S.; HIKAWA, H. A survey of hardware self-organizing maps. IEEE Transactions on Neural Networks and Learning Systems, IEEE, 2022. Cited on page 31.

KANG, S. Using binary classifiers for one-class classification. Expert Systems with Applications, Elsevier, v. 187, p. 115920, 2022. Cited on page 31.

KARACA, M.; ALVARADO, M. M.; GAHROOEI, M. R.; BIHORAC, A.; PARDALOS, P. M. Frequent pattern mining from multivariate time series data. Expert Systems with Applications, Elsevier, v. 194, p. 116435, 2022. Cited on page 37.

KELEKO, A. T.; KAMSU-FOGUEM, B.; NGOUNA, R. H.; TONGNE, A. Health condition monitoring of a complex hydraulic system using deep neural network and deepshap explainable xai. Advances in Engineering Software, Elsevier, v. 175, p. 103339, 2023. Cited on page 94.

KEOGH, E.; LIN, J. Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowledge and information systems, Springer, v. 8, p. 154–177, 2005. Cited on page 39.

KHAN, S. S.; MADDEN, M. G. One-class classification: taxonomy of study and review of techniques. The Knowledge Engineering Review, Cambridge University Press, v. 29, n. 3, p. 345–374, 2014. Cited on page 31.

KRAWCZYK, B.; GALAR, M.; WOŹNIAK, M.; BUSTINCE, H.; HERRERA, F. Dynamic ensemble selection for multi-class classification with one-class classifiers. Pattern Recognition, Elsevier, v. 83, p. 34–51, 2018. Cited 4 times on pages 24, 32, 46 e 56.

KRAWCZYK, B.; MINKU, L. L.; GAMA, J.; STEFANOWSKI, J.; WOŹNIAK, M. Ensemble learning for data stream analysis: A survey. Information Fusion, Elsevier, v. 37, p. 132–156, 2017. Cited on page 24.

KRAWCZYK, B.; WOŹNIAK, M. Dynamic classifier selection for one-class classification. Knowledge-Based Systems, Elsevier, v. 107, p. 43–53, 2016. Cited on page 46.

KRAWCZYK, B.; WOŹNIAK, M.; CYGANEK, B. Clustering-based ensembles for one-class classification. Information sciences, Elsevier, v. 264, p. 182–195, 2014. Cited 5 times on pages 24, 32, 45, 56 e 111.

KUMAR, D.; MEHRAN, S.; SHAIKH, M. Z.; HUSSAIN, M.; CHOWDHRY, B. S.; HUSSAIN, T. Triaxial bearing vibration dataset of induction motor under varying load conditions. Data in Brief, Elsevier, v. 42, p. 108315, 2022. Cited on page 24.

KUMAR, V.; MINZ, S. Feature selection: a literature review. SmartCR, v. 4, n. 3, p. 211–229, 2014. Cited on page 56.

KUNCHEVA, L. I.; BEZDEK, J. C.; DUIN, R. P. Decision templates for multiple classifier fusion: an experimental comparison. Pattern recognition, Elsevier, v. 34, n. 2, p. 299–314, 2001. Cited on page 56.

ŁUCZAK, M. Hierarchical clustering of time series data with parametric derivative dynamic time warping. Expert Systems with Applications, Elsevier, v. 62, p. 116–130, 2016. Cited on page 38.

MA, D.; WANG, L.; ZHANG, D.; SUN, Y.; SHI, X. Application of a-priori algorithm in vibration fault diagnosis for rotary machinery. In: IEEE. 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM). [S.l.], 2019. p. 672–675. Cited 2 times on pages 41 e 52.

MA, Z.; LU, S.; ZHANG, R.; YU, H.; WANG, X. Outlier detection method of three rate value based on one class svm. In: IEEE. 2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC). [S.l.], 2021. p. 334–337. Cited 2 times on pages 35 e 36.

MACHADO, A. P. F.; MUNARO, C. J.; CIARELLI, P. M.; VARGAS, R. E. V. Time series clustering to improve one-class classifier performance. Expert Systems with Applications, Elsevier, v. 243, p. 122895, 2024. Cited 3 times on pages 24, 49 e 63.

MACHADO, A. P. F.; VARGAS, R. E. V.; CIARELLI, P. M.; MUNARO, C. J. Improving performance of one-class classifiers applied to anomaly detection in oil wells. Journal of Petroleum Science and Engineering, Elsevier, v. 218, p. 110983, 2022. Cited 2 times on pages 62 e 63.

MAHARAJ, E. A.; D'URSO, P.; CAIADO, J. Time Series Clustering and Classification. [S.l.]: CRC Press, 2019. Cited on page 50.

MARINS, M. A.; BARROS, B. D.; SANTOS, I. H.; BARRIONUEVO, D. C.; VARGAS, R. E.; PREGO, T. d. M.; LIMA, A. A. de; CAMPOS, M. L. de; SILVA, E. A. da; NETTO, S. L. Fault detection and classification in oil wells and production/service lines using random forest. Journal of Petroleum Science and Engineering, Elsevier, v. 197, p. 107879, 2021. Cited 3 times on pages 62, 90 e 92.

MOULTON, R. H. Clustering to Improve One-Class Classifier Performance in Data Streams. Tese (Doutorado) — Université d'Ottawa/University of Ottawa, 2018. Cited 5 times on pages 23, 24, 31, 35 e 46.

NGUYEN, H.; TRAN, K. P.; THOMASSEY, S.; HAMAD, M. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. International Journal of Information Management, Elsevier, v. 57, p. 102282, 2021. Cited 4 times on pages 32, 33, 34 e 35.

NIENNATTRAKUL, V.; RATANAMAHATANA, C. A. Inaccuracies of shape averaging method using dynamic time warping for time series data. In: SPRINGER. Computational Science–ICCS 2007: 7th International Conference, Beijing, China, May 27-30, 2007, Proceedings, Part I 7. [S.l.], 2007. p. 513–520. Cited on page 39.

PERERA, P.; OZA, P.; PATEL, V. M. One-class classification: A survey. arXiv preprint arXiv:2101.03064, 2021. Cited 3 times on pages 32, 34 e 85.

PETITJEAN, F.; KETTERLIN, A.; GANÇARSKI, P. A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition, Elsevier, v. 44, n. 3, p. 678–693, 2011. Cited on page 50.

PLATT, J. C.; SHAWE-TAYLOR, J.; SMOLA, A. J.; WILLIAMSON, R. C. et al. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99–87, Microsoft Research (MSR), Citeseer, 1999. Cited on page 31.

PRATI, R. C.; BATISTA, G. E.; MONARD, M. C. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: SPRINGER. MICAI 2004: Advances in Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, April 26-30, 2004. Proceedings 3. [S.l.], 2004. p. 312–321. Cited on page 26.

SÁEZ, J. A.; KRAWCZYK, B.; WOŹNIAK, M. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognition, Elsevier, v. 57, p. 164–178, 2016. Cited on page 32.

SANTOS, M. C.; PEREIRA, C. M.; SCHIRRU, R. A multiple-architecture deep learning approach for nuclear power plants accidents classification including anomaly detection and "don't know" response. Annals of Nuclear Energy, Elsevier, v. 162, p. 108521, 2021. Cited on page 33.

SCHLUMBERGER, O. Dynamic Multiphase Flow Simulator. [S.l.]: Version, 2014. Cited on page 63.

SELIYA, N.; ZADEH, A. A.; KHOSHGOFTAAR, T. M. A literature review on one-class classification and its potential applications in big data. Journal of Big Data, SpringerOpen, v. 8, n. 1, p. 1–31, 2021. Cited on page 31.

SHARMA, S.; SOMAYAJI, A.; JAPKOWICZ, N. Learning over subconcepts: Strategies for 1-class classification. Computational Intelligence, Wiley Online Library, v. 34, n. 2, p. 440–467, 2018. Cited 2 times on pages 24 e 46.

TANG, Y.; GAO, J. Improved classification for problem involving overlapping patterns. IEICE TRANSACTIONS on Information and Systems, The Institute of Electronics, Information and Communication Engineers, v. 90, n. 11, p. 1787–1795, 2007. Cited on page 27.

TAVENARD, R.; FAOUZI, J.; VANDEWIELE, G.; DIVO, F.; ANDROZ, G.; HOLTZ, C.; PAYNE, M.; YURCHAK, R.; RUSSWURM, M.; KOLAR, K.; WOODS, E. Tslearn, a machine learning toolkit for time series data. Journal of Machine Learning Research, v. 21, n. 118, p. 1–6, 2020. Cited on page 39.

TAX, D. M.; DUIN, R. P. Data description in subspaces. In: IEEE. Proceedings 15th International Conference on Pattern Recognition. ICPR-2000. [S.l.], 2000. v. 2, p. 672–675. Cited on page 31.

TAX, D. M.; DUIN, R. P. Support vector data description. Machine learning, Springer, v. 54, p. 45–66, 2004. Cited on page 31.

TURAN, E. M.; JÄSCHKE, J. Classification of undesirable events in oil well operation. In: IEEE. 2021 23rd International Conference on Process Control (PC). [S.l.], 2021. p. 157–162. Cited 3 times on pages 62, 90 e 92.

TYAGI, S.; MITTAL, S. Sampling approaches for imbalanced data classification problem in machine learning. In: SPRINGER. Proceedings of ICRIC 2019: Recent Innovations in Computing. [S.l.], 2020. p. 209–221. Cited on page 68.

VARGAS, R. E. V.; MUNARO, C. J.; CIARELLI, P. M.; MEDEIROS, A. G.; AMARAL, B. G. do; BARRIONUEVO, D. C.; ARAÚJO, J. C. D. de; RIBEIRO, J. L.; MAGALHAES, L. P. A realistic and public dataset with rare undesirable real events in oil wells. Journal of Petroleum Science and Engineering, Elsevier, v. 181, p. 106223, 2019. Cited 3 times on pages 24, 49 e 63.

VUTTIPITTAYAMONGKOL, P.; ELYAN, E.; PETROVSKI, A. On the class overlap problem in imbalanced data classification. Knowledge-based systems, Elsevier, v. 212, p. 106631, 2021. Cited 2 times on pages 26 e 27.

WANG, X.; MUEEN, A.; DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P.; KEOGH, E. Experimental comparison of representation methods and distance measures for time series data. Data Mining and Knowledge Discovery, Springer, v. 26, p. 275–309, 2013. Cited on page 38.

WARRENS, M. J. Inequalities between similarities for numerical data. Journal of Classification, Springer, v. 33, n. 1, p. 141–148, 2016. Cited on page 37.

WENG, X. Classification of multivariate time series using supervised neighborhood preserving embedding. In: IEEE. 2013 25th Chinese Control and Decision Conference (CCDC). [S.l.], 2013. p. 957–961. Cited on page 49.

XIONG, H.; LI, M.; JIANG, T.; ZHAO, S. Classification algorithm based on nb for class overlapping problem. Appl. Math, v. 7, n. 2L, p. 409–415, 2013. Cited on page 27.

ZHOU, P.-Y.; CHAN, K. C. A feature extraction method for multivariate time series classification using temporal patterns. In: SPRINGER. Pacific-Asia Conference on Knowledge Discovery and Data Mining. [S.l.], 2015. p. 409–421. Cited on page 49.