Improvement of One-Class Classifier Performance via Time Series Clustering: Application to a Hydraulic System

André Paulo Ferreira Machado * **Celso Jose Munaro** *
Patrick Marques Ciarelli *

* *Departamento de Engenharia Elétrica, Universidade Federal do Espírito Santo, ES (e-mails: andrepaulofm@gmail.com, celso.munaro@ufes.br, patrick.ciarelli@ufes.br)*

Abstract: This work proposes a methodology aimed at improving the performance of one-class classifiers through time series clustering. The clustering process involves utilizing DTW Barycenter Averaging (DBA) and k-means to group multivariate time series based on their similarity. The Apriori algorithm is employed to generate subsets of instances, which are used to train and select multiple one-class classifiers for the same class. The effectiveness of the proposed method is evaluated on a dataset from a hydraulic system to study typical failures happening simultaneously and with several intensities. One of the faults was selected and the results showed that increasing similarity in the subset training data led to an improvement of the classifier performance of more than 34%.

Keywords: One-Class Classifier, Dynamic Time Warping, Long Short Term Memory, Multivariate Time Series Classification, Hydraulic System.

1. INTRODUCTION

Datasets collected from real processes come from the available sensors and are affected by many factors, such as aging of the process, equipment state, and changes in the operation region. Despite that, trained classifiers have to distinguish the process failures represented by such diverse data. Changes in operating modes can create uncertainties that affect process monitoring and the performance of the classifiers.

Models trained under OCC approach aim to

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. ~~Considering the OCC approach, the model aims~~ to capture the underlying structure and patterns in the target instances. With unbalanced datasets, binary algorithms are prone to being polarized by the majority class. However, this issue does not happen in OCC, 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 et al. (2014).

Problems like this are considered by Sharma et al. (2018), where the so-called complex original domains are divided into subdomains that are easier to model. This involves the modeling of sub-concepts rather than tackling the entire domain at once. Each sub-concept of the majority class is identified and a classifier is created for each one, forming an ensemble of classifiers. Ensembles have demonstrated their effectiveness in improving predictive accuracy, breaking down complex and challenging learning problems into easier sub-problems (Krawczyk et al., 2017). The utilization of one-class classifiers within Multiple Classifier Systems (MSC) has shown promising results (Krawczyk et al., 2014, 2018; Fragoso et al., 2021; Alghanam et al., 2023). Although this approach has well-known effectiveness, it remains poorly explored 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, since time series require specific clustering methods to handle the time dependency of data.

In this paper, we investigate the impact of classifiers' performance arising from the increase of similarity on cluster subsets of multivariate time series during the training phase of the classifiers. OCC classifiers are trained with and without the clustering method and the achieved performance is compared. The proposed methodology is applied to a hydraulic system dataset (ICM dataset) composed of multivariate time series. In this case, the challenge is to create a clustering model capable of correctly diagnosing cooler fail conditions, despite all other simultaneous hydraulic conditions and failures.

2. METHODOLOGY

In this section, we present a methodology that addresses a multi-class classification problem on multivariate time series by modeling each class with a set of OCCs (Machado et al., 2024). Long Short-Term Memory (LSTM) is a powerful deep learning algorithm to analyze sequential data (Jun et al., 2020). In this work, the 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 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.

An overview of the proposed methodology applied to each class of fault is presented in Figure 1. 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, 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.

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 hyper-parameter 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, $C^1_{x_i}$ and $C^2_{x_i}$. 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 DTW and is given by $DTW(C^j_{x_i}), j = 1, 2$. Therefore, the n variables will result in $2n$ clusters that can be used to train models for each class.

Some instances may have missing variables, and this is a common problem when data comes from sensors working in hostile environments. In this case, these instances are not included neither to cluster $C^1_{x_i}$ nor $C^2_{x_i}$, but to $C^0_{x_i}$. Thus, after the application of DTW Barycenter Averaging (DBA) and k-means to all variables, to each variable x_i we have the initial cluster of instances DS split in clusters so that $DS = C^1_{x_i} \cup C^2_{x_i} \cup C^0_{x_i}$ (see Figure 1). We define C_0 as the subset of instances that have missing variables. This procedure ensures that the instances in $C^1_{x_i}$ and $C^2_{x_i}$, $i = 1, ..., n$, are grouped to increase the similarity of each variable x_i. 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. If now we consider two variables x_i and x_j, the clusters $C^1_{x_j}$, $C^2_{x_j}$, $C^1_{x_i}$, $C^2_{x_i}$ should be evaluated. We now obtain $C^1_{x_i,x_j} = C^1_{x_j} \cap C^1_{x_i}$ and $C^2_{x_i,x_j} = C^2_{x_j} \cap C^2_{x_i}$. 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 (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. Define $p = 2$.
(2) Generate a set of candidates with p variables, which includes all possible combinations of p variables. Only variables with a support 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 with 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.

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. For example, 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 paper. Therefore, given a minimum support threshold A, the Apriori algorithm determines the instances that are subsets of at least $A\%$ instances in the dataset. When there are more candidate variables, the support may decrease. However, if there are fewer variables used in a classification algorithm, it could become difficult for it to distinguish between the classes in the dataset.

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 1, that have more similar variables, and have also a different number of variables, ranging from 1 to n. Each subset $SS^i_1, i = 1, ..., P_1$ will be used to train an LSTM model M_{1i}, $i = 1, ..., P_1$, and the best model M_1 will be selected using a given performance metric and validation data. Similarly, each subset SS^i_2, $i = 1, ..., P_2$, will be used to train an LSTM model M_{2i}, $i = 1, ..., P_2$, resulting in the model M_2 with the best performance of these subsets (see Figure 1). The models M_1 and M_2 are then tested against a baseline model M_b, trained with the baseline dataset, using test data. We define C_1 as the dataset selected to train M_1 and C_2 as the dataset selected to train M_2. The baseline dataset is given by $C^b = DS - C^0$, i.e., the original dataset DS excluding those instances with missing variables. Data used for validation and testing must be selected assuring that the set of variables used for training is available since they are the only inputs to the LSTM Models. In the application, it will be shown that similarity increases with clustering.

An alternative to improve the proposed method is to train one classifier for each of the subsets and to perform a fusion of all classifier outputs, like in (Krawczyk et al., 2018). According to Kuncheva et al. (2001), instead of using all of the classifiers, selecting a subset from the pool may be an option for improving the classification performance.

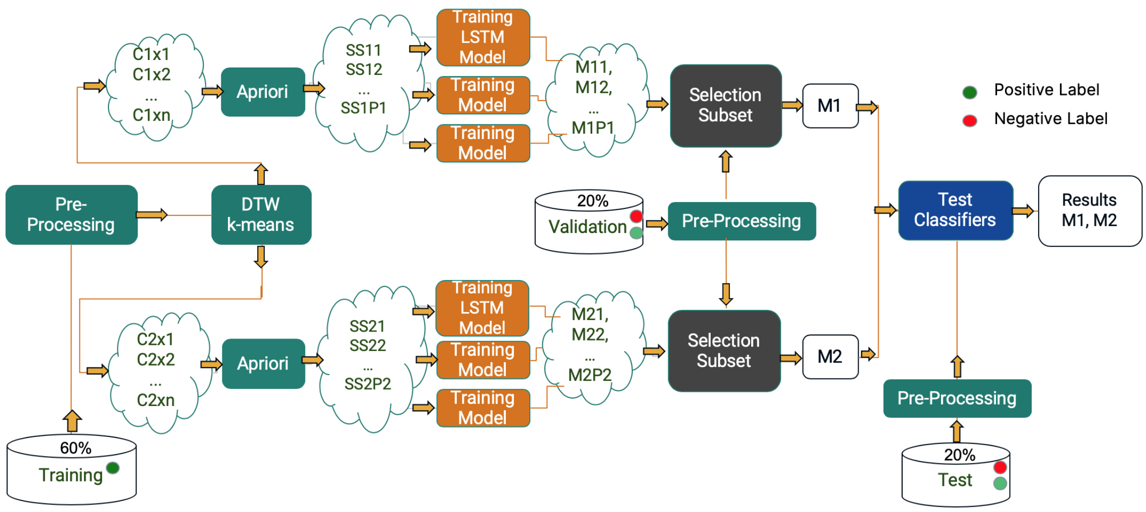

Fig. 1. Methodology flowchart.

The selection of the OCC models is based on the mean of F1-score and Specificity, μ_m. 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 give more reliance on the existence of a fault, increasing confidence in the OCC prediction.

To quantify the improvement of the proposed methodology, a baseline model M_b was trained considering all variables and instances without performing the clustering method, i.e., C^b. 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. We 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 DTW 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 1 and 2 can be used to measure the increase of similarity among the time series related to the variable x_i after clustering,

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

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

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 3 and 4.

$$IS^{01} = 100(1 - \frac{\mu(DTW(C^1_{x_i}), i = 1, ..., n_1)}{\mu(DTW(C^b_{x_i}), i = 1, ..., n_0)}) \tag{3}$$

$$IS^{02} = 100(1 - \frac{\mu(DTW(C^2_{x_i}), i = 1, ..., n_2)}{\mu(DTW(C^b_{x_i}), i = 1, ..., n_0)}) \tag{4}$$

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.

3. CASE STUDY

The experimental application is based on a dataset produced by Helwig et al. (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 dataset was experimentally obtained with a hydraulic test rig, depicted in Figure 2. This test rig consists of a primary working and a secondary cooling-filtration circuit which are connected via the oil tank (Helwig et al., 2015a), (Helwig et al., 2015b).

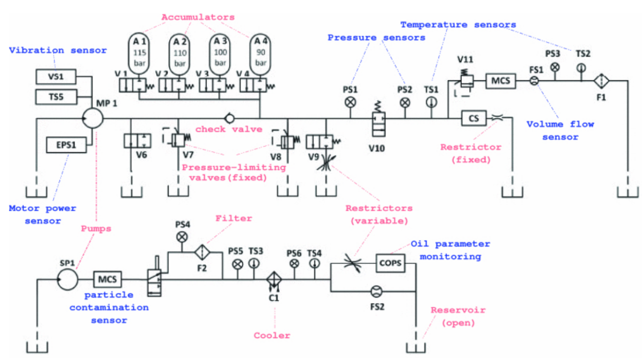

Fig. 2. Test rig hydraulic system

The test system is equipped with multiple sensors that measure various process values. These values include pressure (PS1 - PS6), flow (FS1, FS2), temperature (TS1 - TS5), 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 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 et al., 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 setpoints are defined as 100%, 90%, 80%, and 73% of the nominal value. The internal pump leakage levels are caused by cascaded orifices, measuring 0.2 mm and 0.25 mm. The pre-charge pressure steps of the accumulator are set at 115, 110, 100, and 90 bar (Helwig et al., 2015a). The sensor data are described in Table 1.

Table 1. 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	ºC	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. 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 five class labels, with each label having one severity level representing different states. These classes are assigned to 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 2.

Figure 3 illustrates the labeling of each of the 2205 instances in the dataset with one of five class labels and 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].

Table 2. Summary of the 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

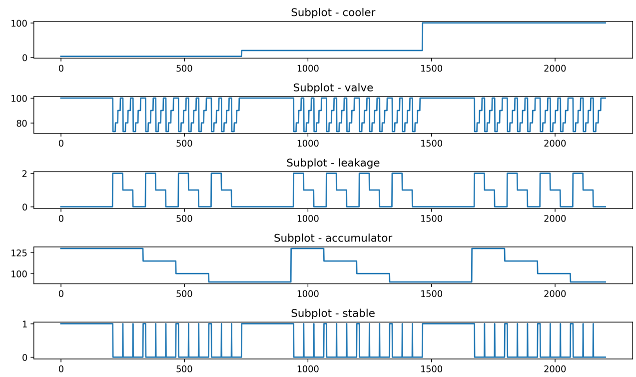

Fig. 3. Each of the 2205 instances in the dataset is labeled with five class labels. Each class label is associated with a severity level that represents distinct states.

Our main objective is to showcase the practical implementation of the clustering by similarity methodology on the ICM dataset. 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.

To assess the feasibility of applying the methodology to this dataset, the cooler condition class was selected for analysis, specifically focusing on the states of close to total failure (3) and reduced efficiency (20) as the target class (positive class). The state full efficiency (100) was designated as the negative class.

4. RESULTS

We address the multi-class classification problem on multivariate time series with different behaviors for the same class. The dissimilarity of data for the same class is due to the different levels of intensity for the same fault and different faults happening concurrently. Our challenge in this work is to create a clustering model capable of correctly classifying and diagnosing cooler fail conditions, despite all other hydraulic conditions happening simultaneously.

4.1 Pre-Processing

The sensor data were downsampled to maintain a consistent frequency rate of 1 Hz. This standardization simplifies the comparison and analysis of different instances across time, enabling a more straightforward evaluation of their similarities and differences.

The total of instances is 2205, being 1464 positive instances, considering numbers 3 and 20 the states of "close to total failure" (3) and "reduced efficiency" (20) as target class, and 741 negative instances designated as full efficiency represented by (100). The 1464 positive class instances were partitioned into three sets: 60% for training, 20% for validation, and 20% for testing. The 741 negative class instances were split into 50% for validation and 50% for testing. Table 3 shows the distribution of instances. Data standardization was performed using the z-score transformation. This technique involves mapping the data to have zero mean and unit standard deviation. The validation and test sets were standardized using the mean and standard deviation of the training data.

Table 3. Number of instances for training, validation, and test.

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

4.2 Time Series Clustering

At this step, k-means with DTW metric was applied to split the time series of the 878 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 $C^2_{x_i}$, $i = 1, ..., 17$, For example, the result of clustering variable $VS1$ (vibration sensor) is shown in Figure 4. The gray lines represent instances and the red lines represent the centroids of the clusters. The amount of instances in each cluster is shown in the graphics of the figure. After this process, using Equations 3 and 4, the similarity of $C^1_{x_i}$ was increased by 21.3% and $C^2_{x_i}$ in 20.4% related to $C^b_{x_i}$. We identified that some variables had different behaviors for the same class, as can be seen in the example of Figure 4. It leads to the discovery of interesting patterns in the time series dataset that affect the instances' similarity.

4.3 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 $A = 0.7$, the Apriori algorithm identified the combination of variables that appeared in at least 70% of instances in the training data set. In this case, 60 subsets of features were selected from cluster $C^1_{x_i}$, and 90 subsets were selected from cluster

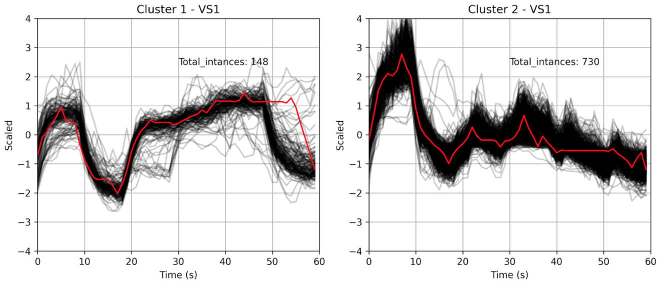

Fig. 4. VS1 (vibration sensor) similarity clustering.

$C^2_{x_i}$. The number of variables in the subsets ranged from 1 to 17. The next step was to select the subsets from clusters 1 and 2.

4.4 Selection of Cluster Subsets

For each subset item from the Apriori step, one model was trained and validated to choose, for each cluster, one subset of features from those returned by the Apriori algorithm. To construct the LSTM autoencoder model, the input data was transformed into a 3D tensor, organized based on samples, time steps, and variables. This tensor 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, defined by the data window size.

The LSTM autoencoder model was implemented 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 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: 20
- Learning Rate: 0.0001

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

For clusters C^1 and C^2, the mean μ_m of ~~the metrics~~ F1-score and specificity were 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.

Only variable {FS2} was selected for Cluster C^1, keeping 75% of training instances. For cluster C^2, the variables

{PS5, PS6} were selected keeping 70% of the 878 training instances. These subsets showed the best classification performance by the validation data set. We highlight that all variables and training instances were used for the baseline model C^b.

4.5 Test Results

In this section, the results are presented after evaluating the trained models M_b, M_1, and M_2 using the test set.

Table 4 shows the results obtained by the models M_b, M_1, and M_2. The support values represent the percent of instances used for training the model. The IS values are calculated using Equations 3 and 4 which represent the percentage of increase of similarity compared with the model M_b. Among these models, the highest performance was observed with the M_2 model, yielding a μ_M value of 0.999. Conversely, the M_b model exhibited the worst results of F1-score and μ_M, respectively, 0.74 and 0.87.

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 5. 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 cooler fail detection capabilities.

Table 4. 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	42.68%	0.995	0.995	0.995
M_2	PS5, PS6	0.70	52.72%	0.998	1.0	0.999

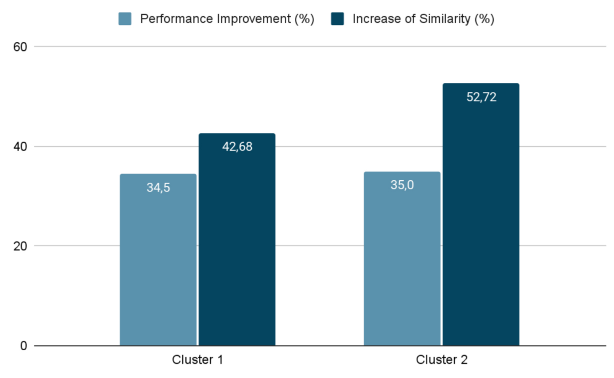

Fig. 5. Performance Improvement

5. CONCLUSION

In this work, a methodology to improve the performance of One-Class Classifiers through time series clustering was applied for condition monitoring of data from a hydraulic system. This is a challenging dataset since faults happen simultaneously and with different intensities. The performance of One-Class Classifiers trained with data obtained using the proposed method was improved by more than 34% for the fault evaluated in this work, compared with the classifier trained with original data and using the F1-score as a reference metric. For future work, we intend to apply the same methodology to the other 5 faults of this system and also to consider the fusion of the best classifiers to achieve even higher performances.

REFERENCES

Alghanam, O.A., Almobaideen, W., Saadeh, M., and Adwan, O. (2023). An improved pio feature selection algorithm for iot network intrusion detection system based on ensemble learning. *Expert Systems with Applications*, 213, 118745.

Fragoso, R.C., Cavalcanti, G.D., Pinheiro, R.H., and Oliveira, L.S. (2021). Dynamic selection and combination of one-class classifiers for multi-class classification. *Knowledge-Based Systems*, 228, 107290.

Helwig, N., Pignanelli, E., and Schütze, A. (2015a). Condition monitoring of a complex hydraulic system using multivariate statistics. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, 210–215. IEEE.

Helwig, N., Pignanelli, E., and Schütze, A. (2015b). D8. 1-detecting and compensating sensor faults in a hydraulic condition monitoring system. *Proceedings SENSOR 2015*, 641–646.

Jun, K., Lee, D.W., Lee, K., Lee, S., and Kim, M.S. (2020). Feature extraction using an rnn autoencoder for skeleton-based abnormal gait recognition. *IEEE Access*, 8, 19196–19207.

Krawczyk, B., Galar, M., Woźniak, M., Bustince, H., and Herrera, F. (2018). Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition*, 83, 34–51.

Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37, 132–156.

Krawczyk, B., Woźniak, M., and Cyganek, B. (2014). Clustering-based ensembles for one-class classification. *Information sciences*, 264, 182–195.

Kuncheva, L.I., Bezdek, J.C., and Duin, R.P. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, 34(2), 299–314.

Ma, D., Wang, L., Zhang, D., Sun, Y., and Shi, X. (2019). Application of a-priori algorithm in vibration fault diagnosis for rotary machinery. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, 672–675. IEEE.

Machado, A.P.F., Munaro, C.J., Ciarelli, P.M., and Vargas, R.E.V. (2024). Time series clustering to improve one-class classifier performance. *Expert Systems with Applications*, 243, 122895.

Sharma, S., Somayaji, A., and Japkowicz, N. (2018). Learning over subconcepts: Strategies for 1-class classification. *Computational Intelligence*, 34(2), 440–467.