

**Candidate:** Ariel Priarone

**Supervisor:** Marcello Chiaberge **Co-supervisors:** Umberto Albertin, Gianluca Dara

## I. INTRODUCTION

Predictive Maintenance (PM) and Novelty Detection (ND) are important topics in modern industrial engineering, aimed at proactively identifying equipment failures before they affect system functionality. Embracing these practices is crucial for reducing equipment downtime and optimizing maintenance efforts. PM ~~aims~~ attempts to quantify and forecast the state of degradation of a system. A quite novel frontier is the direct implementation of PM algorithms within the maintained device, using the principles of Edge Computing.

### A. Motivation

Despite the Fourth Industrial Revolution, the maintenance approach remained unchanged in many industrial applications. The primary factor impeding the advancement of the maintenance approach is the significant expense associated with implementing PM strategies, coupled with a lack of knowledge about the modelling or behaviour of a failing system.

According to a recent survey by the U.S. Department of Commerce, establishments ~~relying on fixing failures are associated with focusing on preventing equipment failures experience~~ 3.3 times ~~more downtime than those actively preventing failures less downtime compared to those only fixing issues after they occur.~~

### B. Objective

The goal of this project is to design, develop and test a *degradation* based framework that performs ND using one or several Unsupervised Machine Learning (UML) algorithms. The structure of the framework is thought to be modular and general-purpose to ease the implementation into different systems. ~~It is~~ The framework has been developed following an unsupervised approach to overcome the common lack of physical models and labelled data of the maintained device. ~~This framework has to be~~ It has been deployed for both PC and Edge Computing.

## II. PROPOSED FRAMEWORK

The solution developed in this thesis is thought to be set up on a new device, and linked to the sensors of the most informative quantities of the system. In the first phase of commissioning, the framework collects the data, extracts the features and stores them. When the ~~data collected is~~ collected data are enough to characterize all the modes of operation of the maintained system, the UML model can be trained. Finally, the framework continues collecting new ~~datatime series~~, extracting the features ~~, and evaluating the new data. The framework now and evaluating these new samples. Now, the framework~~ produces a Novelty Metric (NM) that quantifies how ~~abnormal-anomalous~~ the new data

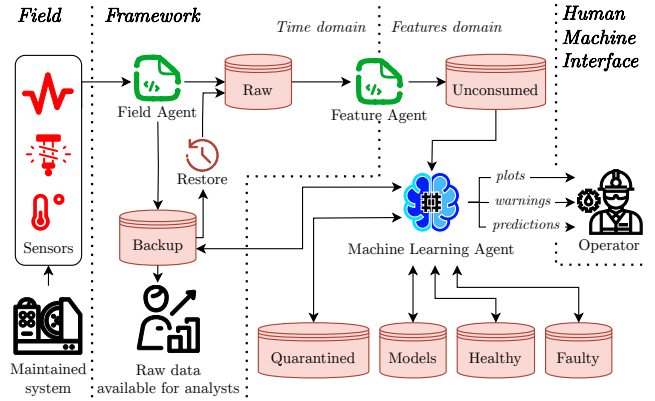


Fig. 1. The structure of the proposed framework. The Field Agent collects the time series, the Feature Agent extracts the features, and the Machine Learning Agent evaluates the health of the maintained device. The agents are connected to collections of a common database. The icon represents a collection of a database that every agent can read and write.

are. This phase ~~can last indefinitely~~ lasts indefinitely (until the maintenance team performs a model update). When the NM overshoots a certain threshold, a warning is issued to the maintenance team. Then, the team can ~~then~~ decide to perform a maintenance action or to continue monitoring the system. If ~~they declare the~~ the team declares the system as healthy, the framework can be retrained with the new data, to update the UML model. Otherwise, ~~a second UML model the framework~~ can be trained to characterize also the newly discovered fault ~~and in order to~~ perform Fault Detection (FD) in the future.

### A. Software Agents

The proposed framework is based on software agents. Each agent is autonomous and performs a specific task. The developed agents, as shown in Fig. 1, are:

- **Field Agent (FiA):** responsible for the synchronous sampling of the data. It provides the time series records and ensures a correct sampling frequency.
- **Feature Agent (FA):** extracts the features from the time series.
- **Machine Learning Agent (MLA):** trains the UML algorithms and then performs ND, FD and PM. It reports the results to the user.

### B. Database

All the Agents are connected to a common database. In the case of the PC implementation, MongoDB has been used. In the case of the Edge implementation, the data are stored directly in the microcontroller's memory. Regarding the structure shown in Fig. 1, the MongoDB database is composed of seven collections. Every collection has a specific role in the framework. For example, the *Quarantined*, *Healthy* and *Faulty* collections contain the features that have been flagged as novelty, normal or faulty, respectively.

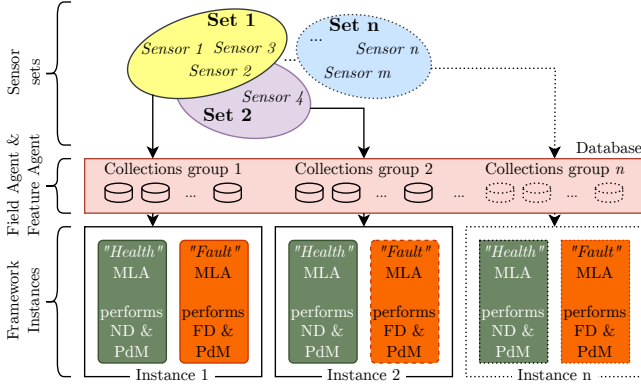


Fig. 2. Multiple instances implementation of the framework. Every instance is linked to a different set of sensors, to monitor different parts of the system.

### C. Multiple Instances

As shown in Fig. 2, multiple instances of the framework can be implemented in multiple instances. This is useful to better isolate the location of the anomaly in a complex system. The larger the set of sensors that a single instance of the framework is connected to, the more difficult it is to isolate the anomaly. On the other hand, configuring a large group of sensors allows the detection of complex anomalies.

For example, a shaft with two bearings can be monitored by two instances of the framework, one for each bearing vibration signal. Another instance can be linked to the signals of both bearings to detect more complex novelty patterns. The latter instance can exploit the case where the bearings signals are normally correlated (e.g. proportional in amplitude) to detect a novelty that is not detectable by analyzing the signals separately.

### D. Unsupervised Machine Learning Models

The ~~considered~~ UML implemented in the framework are: *K-means*, *DBSCAN*, *Gaussian Mixture Model (GMM)*, *Isolation Forest (IF)*, *Local Outlier Factor (LOF)*, *One-Class Support Vector Machine (ν-SVM)*.

K-means and DBSCAN are traditionally clustering algorithms, so a custom NM has been developed for these algorithms. For example, if K-means is used, the NM is defined as the distance of a sample from the closest centroid, normalized by the cluster radius. The other models are already commonly used for ND, so the NM has been linked to the “score” provided by the library functions.

If the UML is trained with faulty data ~~to perform FD~~, the instead of normal ones, then it performs FD. The NM measures “how not faulty” the new data are. In this case, the value is transformed into a Fault Metric (FM) with the function  $FM = -\ln(NM + 1)$  to preserve the coherence.

## III. FEATURE EXTRACTION

The framework is developed to acquire time series data from an arbitrary configuration of sensors. When a time series is available, the FA extracts the features. ~~Every~~ Each time series is linked to a specific set of features to be extracted.

### A. Feature set

The considered features are divided into two categories:

- **Time domain features:** Mean, Standard deviation, Peak-to-peak value (P2P), Root Mean Square value (RMS), Skewness and Kurtosis.
- **Frequency domain features:** Energy of the Wavelet Packet Decomposition (WPD) coefficients, Fast Fourier

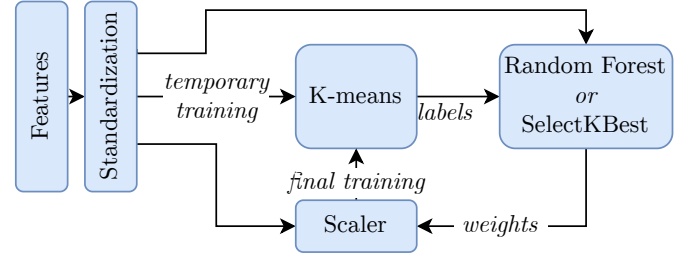


Fig. 3. Feature scaling and selection procedure. The features are standardized and then scaled by a weight array or selected to reduce the feature space dimensionality, discarding the less informative features.

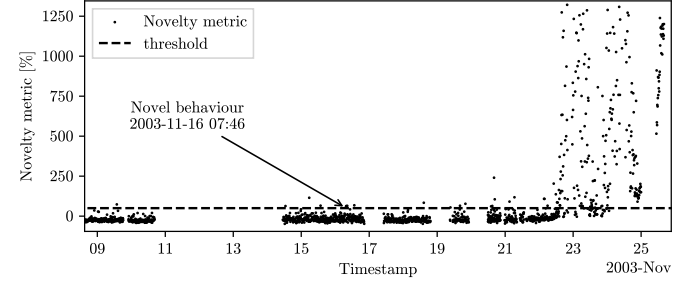


Fig. 4. Results of ND on the IMS dataset. When the Novelty Metric crosses the threshold, a warning is issued. The malfunction happens at the end of the dataset.

Transform (FFT) coefficients. The WPD is based on the PyWavelets<sup>1</sup> and Wavelib<sup>2</sup> libraries.

The time domain features are computed in the corrected form for sampled data. In the frequency domain, the WPD is preferred in this work, because it reduces the dimensionality of the feature space. The features are standardized along the training dataset so that the mean and standard deviation are 0 and 1, respectively. This is done to ease the training of the UML algorithms.

### B. Scaling and selection

Despite the standardization, during the experimental validation, it has been observed that some features are more informative than others. To reduce the impact of the less informative features, an optional feature scaling and selection step can be performed, as shown in Fig. 3. The weights used for scaling the features can be computed by performing a Random Forest training or using the SelectKBest library method.

## IV. VALIDATION

### A. On bearings vibration datasets

The PC implementation of the framework has been thoroughly tested on a publicly available dataset collected by the Center for Intelligent Maintenance Systems (IMS). The dataset contains time series of ~~vibrations of bearings~~. It contains bearings vibrations collected during three separate “run to failure” experiences.

The ND capability of the framework has been tested using all the cited UML models. The Lead Time (LT) elapsed between the ND event and the actual fault is used to compare the models. The results are compared in table I. The evolution of the NM, using a K-means model on a signal of the test N°1, is shown in Fig. 4.

To perform PM, the framework has to estimate the Remaining Useful Life (RUL) of the component. The RUL

<sup>1</sup><https://github.com/PyWavelets/pywt.git>

<sup>2</sup><https://github.com/rafat/wavelib.git>

TABLE I

COMPARISON OF THE RESULTS FOR THE TEST N°1 OF IMS DATASET.

Algorithm	ND event	LT [min]	LT [days]
K-means	2003-11-16 07:46	<b>13913</b>	<b>9.6</b>
DBSCAN	2003-11-22 15:06	4833	3.3
GMM	2003-11-22 03:47	5513	3.8
BGMM	2003-11-22 03:45	5514	3.8
$\nu$ -SVM	2003-11-22 14:56	4844	3.3
IF	2003-11-16 10:08	13771	9.6
LOF	2003-11-16 07:48	13912	9.6
P2P without any ML	2003-11-22 16:06	4774	3.3

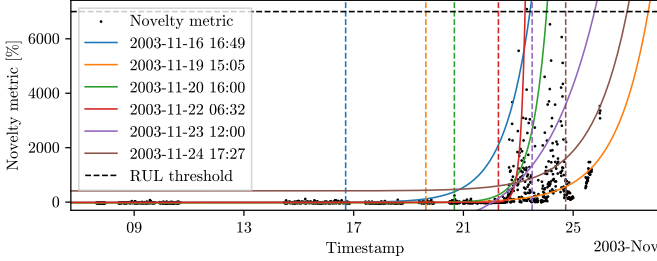


Fig. 5. Results of Remaining Useful Life predictions on the IMS dataset. The Vertical lines indicate the time when the predictions are performed. The RUL is the time remaining before the fitted curve crosses the threshold.

is predicted by a curve fitted to the data. The type of curve to fit is configurable, ~~in~~ in this work  $y = a \cdot e^{b \cdot x} + c$  has been used. The results of the RUL predictions are shown in Fig. 5.

Since the second and third tests of the IMS dataset share the same type of fault, the framework has been trained to perform FD with the faulty data of the second test and then it has been evaluated on the third test. Analogously to the ND, after when the FM crosses a certain threshold, a warning is issued and RUL predictions are performed. The results of the FD are shown in Fig. 6.

### B. Laboratory tests

K-means, IF and LOF are the three models that performed the best on the dataset. ~~K-means performed equally or only slightly better than IF and LOF.~~ Notably, the K-means model has advantages such as minimal parameter storage requirements and low computational costs for calculating the

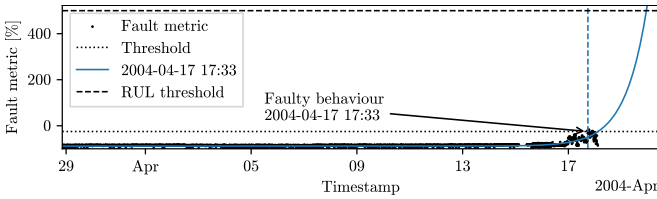


Fig. 6. Results of FD on the IMS dataset. The malfunction is detected when the Fault Metric crosses the threshold. The RUL is the time remaining before the fitted curve crosses the threshold.

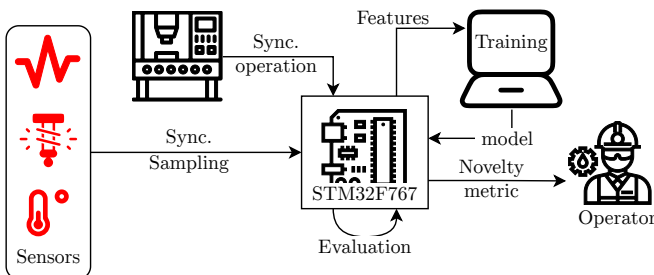


Fig. 7. Structure of the Edge implementation. The microcontroller autonomously evaluates the status of the maintained system. Only during the training phase, the PC implementation computes the model for the Edge.

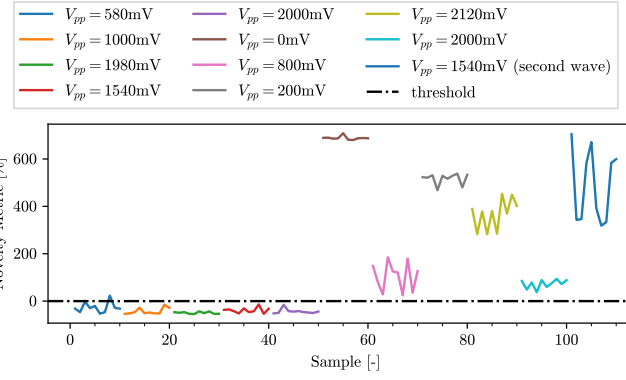


Fig. 8. Results of laboratory test on the shaker. The first five lines are evaluations of known vibrations, correctly flagged as normal samples. The remaining lines are the successful detections of the novelty of new vibrations.

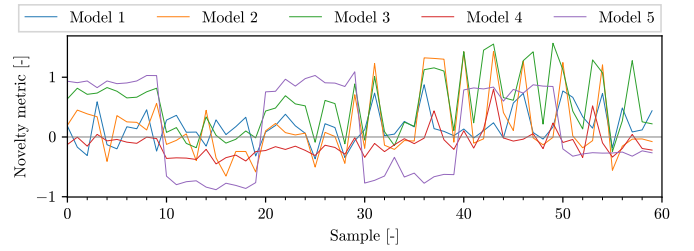


Fig. 9. Results of laboratory test on linear actuator acceleration profiles. The dimensionality of the feature space has been reduced to train Model 5, which performs a clear and sharp detection of the alternating pattern between known and unknown movement profiles.

NM. For these reasons, the K-means model has been chosen for the Edge implementation.

In the first phase, the framework gathers the data and extracts the features. The UML model is trained on a PC and then included in the Edge. After, the microcontroller autonomously evaluates the status of the maintained system. The structure of the Edge implementation is shown in Fig. 7.

Extensive tests have been performed on a laboratory shaker, simulating the vibrations generated by a generic mechanical system. The tests have been carried out to evaluate the sensitivity to variations in both amplitude and frequency content. The results of the ND are shown in Fig. 8.

A second series of tests has been performed mounting the accelerometer on a linear actuator. The axis of the accelerometer has been aligned with the direction of the movement of the actuator, to sense more the acceleration actuated actuated acceleration rather than the vibrations. A set of predefined movement profiles has been used for training, while another set for testing. This series of tests exploited a high number of non-significant features. This is because the WPD gave high resolution on over a wide range of frequency content, but the actuator excited only a few, and very low, frequencies. The remaining features were almost all noise, and the standardization procedure had the side effect of amplifying them to the same level as the significant ones. An effort has been made to fine-tune the model to reduce the impact of the noise (see Fig. 9, models 1 to 4). The best result, however, has been obtained by removing the less informative features and using a reduced feature space. The benefit of this approach is evident in Fig. 9.

## V. CONCLUSION AND FUTURE WORK

The proposed solution has been proven effective in performing Novelty Detection in both public datasets and laboratory tests. The flexibility of the framework allows it to be easily adapted to different systems.

Some laboratory tests on the Edge implementation have shown that the standardization of the features is not always enough to make the clusters in the feature space meaningful. When this happened, ~~in the PC implementation,~~ Isolation Forest and Local Outlier Factor ~~Models~~models were tested in the PC implementation and, being not cluster-based, ~~were tested and resulted as~~proved to be more effective. As future work, the deployment of these models also in the Edge implementation would improve the performance of the Edge framework in some specific applications.