Unsupervised Machine Learning Algorithms for Edge Novelty Detection

April 2024

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

***Abstract*— Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

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 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 Condition Based (CB) or PM strategies, coupled with a lack of knowledge about the modelling or behaviour of a failing system.

According to a recent survey by U.S. Department of Commerce, the top 25% of establishments relying on reactive maintenance were associated with 3.3 times more downtime than those in the bottom 25%.

B. Objective

The goal of this project is to design, develop and test a *degradation* based CB framework that performs ND, Fault Detection (FD) and PM, 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 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 deployed for both PC and Edge Computing: the former developed in Python, the latter in C.

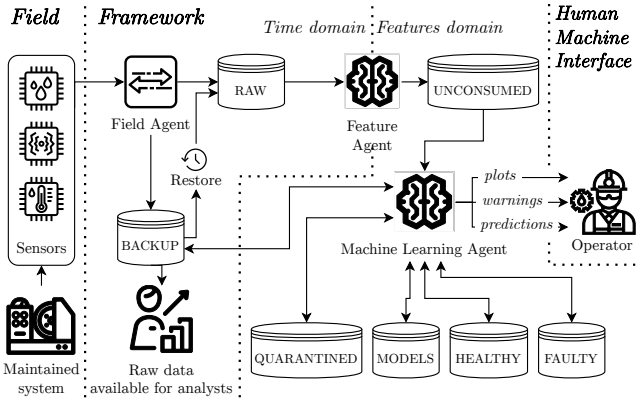

Fig. 1. The structure of the proposed framework

II. PROPOSED FRAMEWORK

The solution developed in this project 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 enough to characterize all the modes of operation of the maintained system, the UML model can be trained. Finally, the framework continues collecting new data, extracting the features, and evaluating the new data. The framework now produces a Novelty Metric (NM) that quantifies the novelty of new data. This phase can last indefinitely, when the NM overshoots a certain threshold, a warning is issued to the maintenance team. They can then decide to perform a maintenance action or to continue monitoring the system. If they declare the system as healthy, the framework can be retrained with the new data, to update the UML model. Otherwise, a second UML model can be trained to characterize the newly discovered fault and perform FD in the future.

A. software agent

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.
- **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

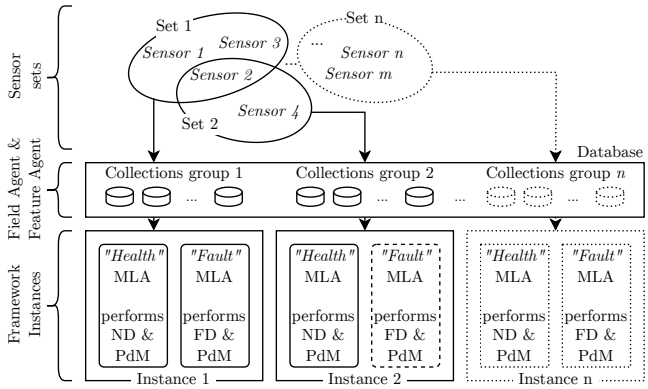

Fig. 2. Multiple instances implementation of the framework

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: *Raw* containing the time series, *Unconsumed* containing the features to be evaluated by the MLA, *Quarantined, Healthy and Faulty* containing the features that have been flagged as novelty, normal or faulty, respectively, and *Models* containing the UML models. The collection *Backup* is general purpose.

C. Multiple Instances

As shown in Fig. 2, 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 sensors that a single instance of the framework is connected to, the more difficult it is to isolate the anomaly.

III. FEATURE EXTRACTION

The framework is developed to acquire time series data from an arbitrary configuration of sensors. The Field Agent (FiA) is responsible for the *synchronous* sampling of the data. Once a time series is available, the Feature Agent (FA) extracts the features from the time-domain data. Every time series is linked to a specific set of features to be extracted, configurable in the `.config` file of the framework.

The considered features are divided into two categories:

- **Time domain features**: Mean, Standard deviation, Peak-to-peak value, RMS, Skewness and Kurtosis.
- **Frequency domain features**: Energy of the Wavelet Packet Decomposition (WPD) coefficients, FFT coefficients. The WPD is based on the PyWavelets[1] library, for Python, and on the Wavelib[2] library, for C.

The time domain features are computed in the form corrected 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.

A. 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 step can be performed. The scaling is done by multiplying the features by a weight array. The weights can be computed by performing a Random Forest training or using the SelectKbest library method. Alternatively, the weights can be used to remove the less informative features, reducing the dimensionality of the feature space. This procedure is shown in Fig. 3.

[1] `https://github.com/PyWavelets/pywt.git`
[2] `https://github.com/rafat/wavelib.git`

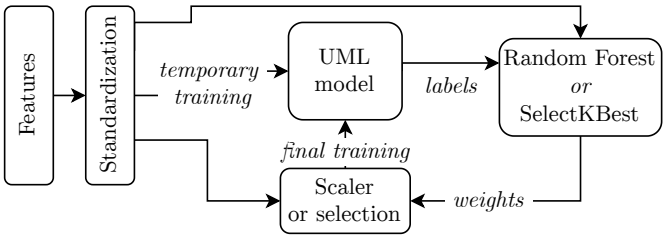

Fig. 3. Feature scaling and selection

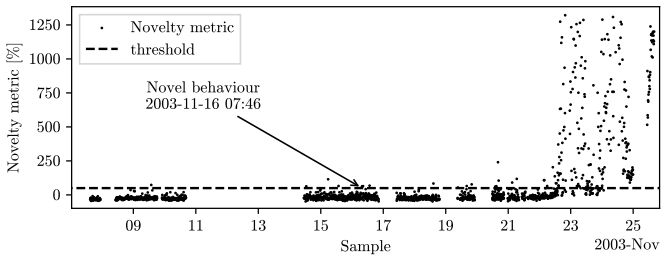

Fig. 4. Results of ND on a public dataset

IV. VALIDATION

A. On public datasets

B. Laboratory tests

V. CONCLUSION

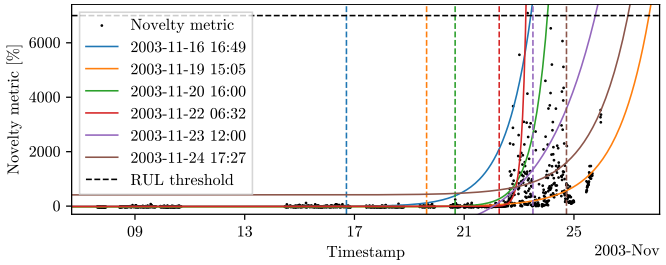

Fig. 5. Results of RUL predictions on a public dataset

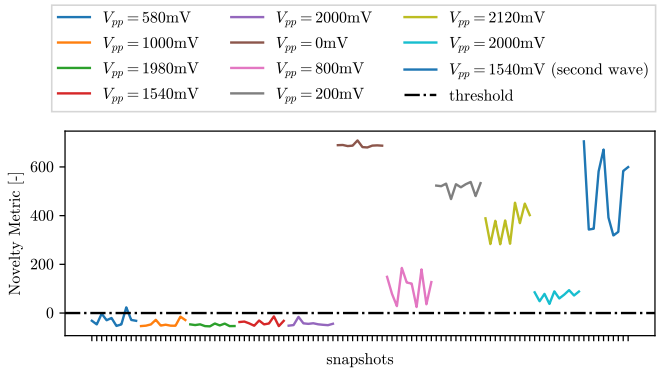

Fig. 6. Results of the shaker laboratory test