On the Performance of Machine Learning at the Network Edge to Detect Industrial IoT Faults

Yuri Santo*, Bruno L. Dalmazo†, Roger Immich‡, André Riker*
*Federal University of Pará, Brazil
†Federal University of Rio Grande, Brazil
‡Federal University of Rio Grande do Norte, Brazil
yuri.santo@icen.ufpa.br, dalmazo@furg.br, roger@imd.ufrn.br, ariker@ufpa.br

***Abstract*—Industrial Internet-of-Things (IoT) massively deploys intelligent computing in industrial production and manufacturing environments seeking automation, reliability, and control. Machine Learning models provide intelligent decisions to drive manufacturing systems to the next level of productivity, efficiency, and safety. One of the critical challenges that must be faced is the deployment of Machine Learning models at the network edge to detect data anomalies caused by Industrial IoT hardware failures, since industrial IoT devices are prone to errors and failures. These anomalies can harm the industrial IoT system by producing false alarms, consuming network resources, and affecting productivity. Because of that, it is critical to rely on low latency and high precision detection systems to verify the data received from industrial IoT devices. In light of this, we assessed key performance indicators of five machine learning models running at edge computing, to provide in-depth discussions. The performance results were obtained from an oil refinery scenario using a real industrial IoT dataset. The performance was measured in terms of (a) Accuracy, (b) Precision, (c) Recall, (d) F1 score, (e) Training time, and (f) Response time.**

***Index Terms*—Industrial Internet-of-Things; Machine Learning; Edge Computing.**

I. Introduction

The emergence of the Internet-of-Things (IoT) in the industry is enabling a new industrial revolution [1], since IoT enables smart monitoring [2] with efficient communication [3] [4] [5] for a vast set of applications [6].

Machine learning in IoT is driving the industry to the next generation of smart manufacturing, logistics, control, and distribution. However, industrial IoT environments and systems are sensitive to errors and faults occurring in IoT devices. According to [7], faults are defined as deviations from expected behaviour in the device output. The occurrence of faults results in production halts, accidents, false alarms, and equipment damage. Therefore, it is crucial to have precise and fast mechanisms to detect and diagnose faults in industrial IoT systems.

Edge computing has emerged as a paradigm to increase the computational capability of IoT systems, providing extra storage and processing. Edge computing devices are deployed near the IoT devices to reduce the time necessary to complete computationally demanding tasks.

Given the low latency requirement of industrial IoT applications and the capabilities of edge computing to provide that, the objective of this paper is to assess the performance of machine learning models running in edge computing devices to detect failures in industrial Internet-of-Things. The consider industrial scenario is a petrochemical power plant, where a set of boilers produces energy for an oil refinery. This paper proposes an algorithm to inject faults on the original dataset and also presents key performance indicators applying five machine learning models to identify faults. Besides, this work presents the obtained computational time to train and test these models, using an edge computing device.

The rest of this paper is organized as follows. Section II presents the related work. The description of the industrial scenario, which is a petrochemical power plant, is given in Section III. Section IV presents the details related to the machine learning models. Next, the evaluation, including the performance metrics, settings, and the obtained results are detailed in Section V. At the end, Section VI presents the conclusions and future works.

II. Related Work

There are in the literature, solutions which rely on machine learning models to support smart decisions in industrial IoT environments. Jan et al. [7] [8] design a distributed diagnosis system to detect sensor-fault for Internet-of-Things. The authors consider devices with limited computation resources, such as memory, processing, and energy. This solution is based on a Support Vector Machine (SVM) model, where response time is not the top priority for the application.

Saeed et al [9] propose a solution for fault detection and diagnosis in sensors, considering a set of faults, namely erratic, drift, hard-over, spike, and stuck faults. This work captured real data from a temperature-to-voltage converter and the faults were injected into the original dataset. The detection systems apply a SVM model and the authors present a set of tuning parameters able to improve the performance of fault detection.

Javaid et al [10] apply machine learning to detect a particular type of fault, called drift. In this faulty state, the output of the device keeps increasing or decreasing linearly from the normal state. The dataset used in this work was collected from a digital relative temperature/humidity sensor (DHT22) and the drift faults were inserted using an Arduino controller. The authors compare the performance of the machine learning models in terms of precision, recall, f1-score, and total accuracy.

Zidi et al [11] propose a SVM model to detect faults in Wireless Sensor Networks. This work uses as the original dataset a data collection from the University of South at Greensboro [12] and injected the following type of faults: Offset, Gain, Stuck-at, Out of bounds, and random fault.

Naimi et al [13] consider a nuclear power plant scenario, where industrial IoT devices are prone to errors and faults. The authors propose fault detection and diagnosis based on neural networks and the K-nearest neighbour algorithm. The proposed solution first detects the fault based on the neural

Preprint 2022 IEEE

network, then the K-nearest neighbour algorithm determines the fault type. This work considers the following types of faults: bias, drift, actuator saturation, and actuator offset.

Khodabaksh et al [14] present a comprehensive approach for petrochemical power plants of an oil refinery. The authors made available their dataset in [15]. This work uses ARMA time-series as modelling technique and generates synthetic datasets. Besides, it classifies four fault types of industrial IoT devices, namely called Bias, Drift, Precision Degradation and Failure using a complex decision tree, neural network, and k-nearest neighbour algorithms. These faults were injected into the original dataset and the performance of the detection was measured in terms of Precision and Recall. This work also evaluates the computational performance of the detection algorithms, taking into consideration memory usage and time.

Summing up, the detection of a fault in Industrial IoT is a time-sensitive application, so delays must be minimized. However, many works related to fault detection in industrial scenarios use SVM as their main building block and SVM requires higher computation time. This is the case for Jan et al. [7] [8] and Zidi et al [11]. Naimi et al [13] also do not consider the necessary time to apply a two-phase solution, based on SVM and K-nearest neighbour algorithm. Therefore, the works in the literature do not evaluate the machine learning models considering their performance in edge computing hardware. The performance of the classifiers is important and low computation time is also crucial for industrial IoT scenarios.

III. Description of the Scenario: A Petrochemical Power Plant

Industrial Internet-of-Things (IIoT) aims to interconnect people, machines, and things. In IIoT, intelligent systems must support the full connection among all devices, the entire industry chain, and the entire value chain [16]. Oil & gas businesses are one of the IIoT scenarios, demanding monitoring of thousands of sensors inside and around their physical systems [14]. In these environments, IIoT devices continuously measure temperature, pressure, flow rate and O_2, located in drills, heaters, turbines, boilers, pumps, compressors, and injectors.

Figure 1 illustrates the considered IIoT scenario, which is a power plant located in an oil refinery. The major component of this IIoT environment is a set of boilers and turbines. The IIoT devices deployed in the power plant are sensors, servers, processing and storage devices. Typically, a power plant with 8 boilers produces 80 megawatts of electricity, which is consumed by the rest of the refinery. The process of electricity production is conducted by turning hot water into super-heated and highly-pressurized vapor.

The produced vapor from the boilers is oriented to the set of turbines. Each turbine has an alternator that transforms thermo-kinectic energy into electrical energy. The flow rate in the input and output of the boilers are monitored by a set of IIoT devices, e.g. *F_in#1* and *F_out#1*, as can be observed in Figure 1.

Figure 2 illustrates a power plant boiler and its IIoT devices. The boiler receives as input water, air, fuel oil and fuel gas. The boiler can control the desired stream pressure levels (low, high, very-high). It also controls heating, cooling, recycling, and condensing modes [14]. The implanted devices measure the temperature, pressure, O_2, and flow in the various critical points of the boiler.

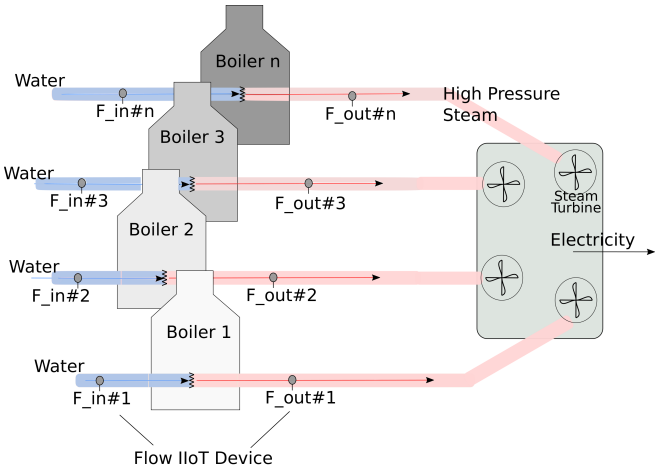


Fig. 1: IIoT Scenario: A Power Plant in a Oil Refinery [14].

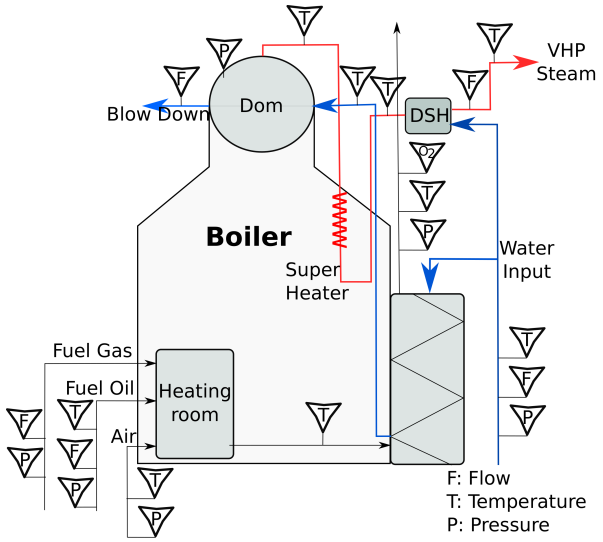


Fig. 2: Boiler in a power plant [14].

IV. Machine Learning for IIoT Device Failure Detection

IIoT devices are prone to errors and failures. These anomalies can harm the IIoT system by producing false alarms, since the failure of an IIoT device can indicate erroneous data. The production is halted, based on these anomalous detections. This work proposes the use of machine learning to detect IIoT device failures. Section IV-A presents the dataset from Turkish Petroleum Refineries Inc. (TUPRAS) that contains real IIoT measurements. Section IV-B presents how the failures were injected in the original dataset. Section IV-C describes the applied machine learning models.

A. Dataset with Real Measurements

In this paper, we use an IIoT dataset provided by Khodabakhsh et al. [14], which provides real measurements of more than 1,000 devices deployed in the Turkish Petroleum Refineries Inc. (TUPRAS) power plant. This dataset is a sample of real data measured every minute and it is made available for academic use in [15].

Data used in these experiments are 200,000 records from (Water, De-Super Heater, Vapor) flow sensors sampled every 60 seconds for about 5 months in the TUPRAS power plant. This data was replicated 5 times to form 1 million lines to better represent the real sensor loads. Every line has records of 3 flow sensors in the power plant dataset and 17 flow sensors in the petrochemical dataset.

B. Injection of Hardware Faults

According to [7], faults are defined as deviations from expected behaviour in the device output. The faults are a data corruption behaviour and it is related to the physical defects of the devices and their operational conditions.

In this paper, a single type of fault has been considered, namely Spike Fault. As can be observed in Figure 3, a spike fault is an effect observed as a large-amplitude value occurring at time intervals in sensor output. These errors occur due to vibrations from other parts of the system affecting the produced output.

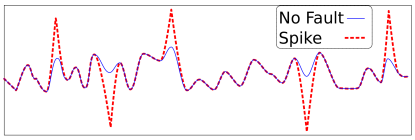


Fig. 3: Spike Fault.

The original dataset with real measurements does not identify, i.e. label, any faults or errors. However, machine learning models using supervised techniques must have labeled data identifying the faults. Based on statistical analysis performed on the dataset, we developed an algorithm to inject fault in the dataset, inserting the respective labels for train and test.

Algorithm 1 presents the pseudo-code used to insert spike faults in the original dataset.

Algorithm 1 Fault Injection Algorithm

```
1:  Start
2:      function FAULTVALUE(datasetValues, iStart)
3:          windowSize ← 500;
4:          data[windowSize];
5:          for (i = iStart; i < iStart+windowSize; i++) do
6:              data[i] ← datasetValues[i];
7:          end for
8:          mean ← data.mean();
9:          min ← data.min();
10:         max ← data.max();
11:         maxVariance ← (mean - min) * (max - mean);
12:         faultValue ← max+maxVariance;
13:         return faultValue;
14:     end function
15:     indexNum ← datasetValues.size();
16:     for (j = 0; j < indexNum*percentageFaults; j++) do
17:         spike ← faultValue(datasetValues, indexNum);
18:         datasetValues[random(0,indexNum)] ← spike;
19:     end for
20: End
```

The input of this algorithm is the original values sensed by the IIoT devices and its output is either the same original value or a fault. The proportion of inserted faults can be controlled. The faultValue function returns the value for a single spike fault in a particular time. The faultValue function takes into account the sum of the maximum variance (maxVariance) and the maximum value (max) of a window with 500 values (windowSize). The mean and the minimum value of the 500-window are considered, as equation 1 shows. The fault is inserted in the random index taken from a list, i.e. faultsList. The original dataset value is replaced by a new spike value.

$$\sigma^2_{max} = (mean - min)(max - min) \quad (1)$$

C. Machine Learning Models

To detect the hardware faults of IIoT devices, the following set of classifiers were used: Supported Vector Machine (SMV), Decision Tree, Random Forest (RF), Gaussian Naive Bayes (GNB), and Logistic Regression (LR).

The machine learning models are based on supervised learning, using a training set to adjust their models to compute the output. SVM, Decision Tree, GNB, and RF are classifiers that seek to recognize input within the dataset and compute how those entities must be labelled or defined. LR is a regression-based model, which means that it seeks to find the relationship between dependent and independent variables.

V. Performance Assessment

The goal of this section is to evaluate the machine learning models using a comprehensive set of metrics, to present the evaluation settings and environment, and to detail the obtained results. This section is divided as follows: Section V-A presents the metrics defined to evaluate the machine learning performance. The performance related to edge computing is shown in section V-B. Section V-C presents the settings of the evaluation environment. Section V-D presents the obtained results.

A. Metrics for Machine Learning Performance

The following set of metrics has been defined to measure the performance of the machine learning models:

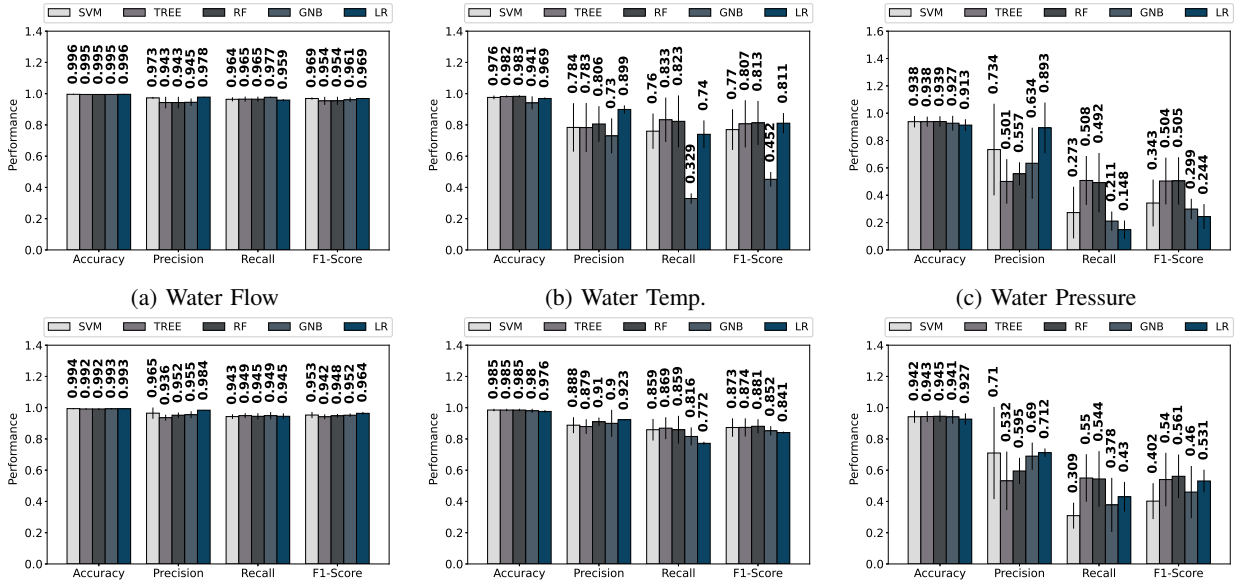
- **Accuracy**: Indicates the number of correct predictions divided by the total number of predictions.
- **Precision**: It is represented by the ratio between the number of correct positive classifications and the number of total positives.
- **Recall**: It represents the number of true positives divided by true positives and false negatives.
- **F1 Score**: It constitutes a harmonic mean between precision and recall. In this metric, 1.0 means excellent performance.

B. Metrics for Edge Computing Performance

The performance of the machine learning running at the edge computing is assessed by the following metrics:

- **Train Time**: This is the time to complete the machine learning train in a particular edge computing hardware.
- **Response Time**: This is the time taken by the machine learning algorithm to compute 13393 outputs, i.e. 30% of the dataset.

It is important to notice that the training time is an important metric, since the machine learning models must be updated frequently in order to learn new behaviors. It means that the machine learning models are likely to be trained over time.



(a) Water Flow (b) Water Temp. (c) Water Pressure

(d) Steam Flow (e) Steam Temp. (f) Steam Pressure

Fig. 4: Machine Learning performance.

C. Evaluation Settings

Table I shows the settings used to run the tests. The original dataset has 44643 data values for six data types. The test performance considers four proportions of injected spike faults, as follows: 446 (1%), 2232 (5%), 4464 (10%), and 6696 (15%). The dataset was divided in 70% for training and 30% for tests.

TABLE I: Parameters and Settings.

Parameter	Value
Original Dataset	44643 data values. TUPRAS [14]
Data type	Water flow, water temperature, water pressure Steam flow, steam temperature, steam pressure
Type of Fault	Spike
Injected Faults	446 (1%), 2232 (5%), 4464 (10%), 6696 (15%)
ML Models	SVM, Decision Tree, RF, GNB, and LR
Implementation	Python, scikit-learn library
Edge Computing	Hardware: i7-8700 3.2GHz, 12GB RAM;

The machine learning models were implemented in Python, using the scikit-learn library and running on Linux Operating System. The hardware used for the tests seeks to reflect edge computing hardware. For these tests, the following hardware was used: i7-8700 3.2GHz, 12GB RAM.

D. Obtained Results

Figure 4 presents the results for the five machine learning models and their performance in terms of Accuracy, Precision, Recall, and F-1 score. As can be observed, all five machine learning models have high performance for Water Flow and Steam Flow, since all the models achieved more than 0.90, considering all the performance metrics. In comparison, the performance for Water Pressure and Steam Pressure shows that the models have high performance in terms of accuracy. However, their performance in terms of Precision, Recall, and F1 Score is poor. This occurs because Accuracy is a percent value of correct predictions, it is not recommended for an unbalanced dataset. For instance, in a dataset with 1% of faults, the Accuracy would be 99% if the model predicts zero faults.

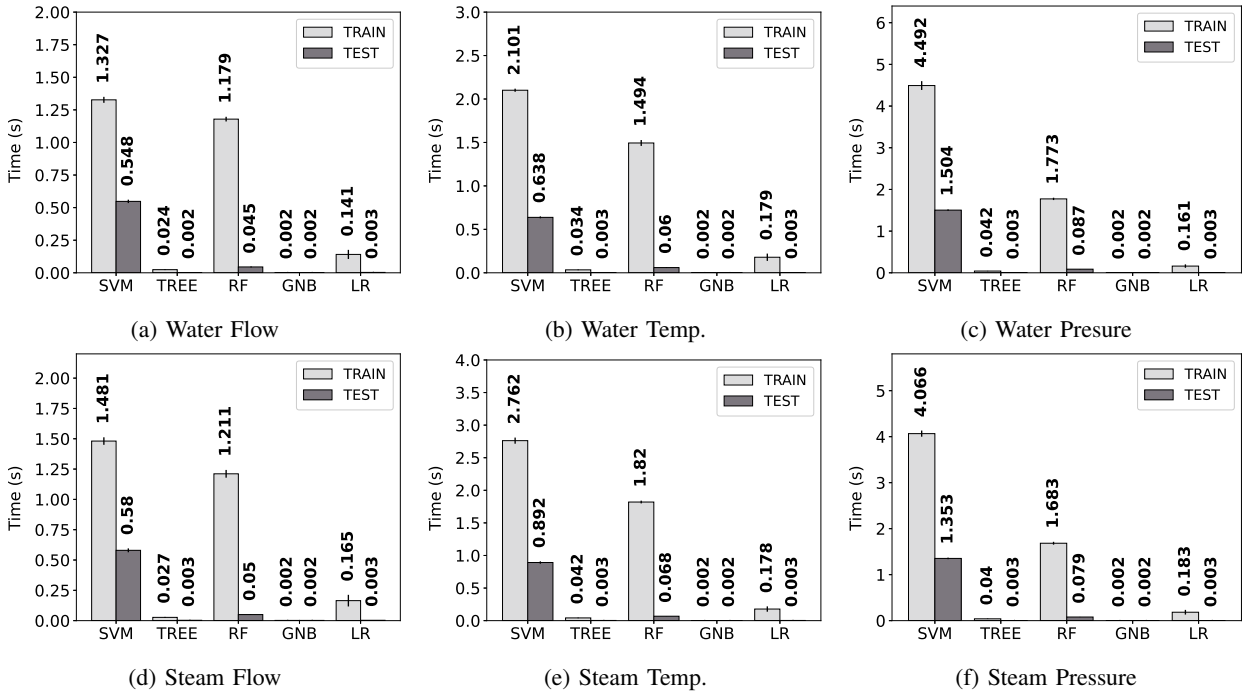
The difference in performance between Flow and Pressure is noticeable. It is important to observe that Water and Steam Flow have a higher variance in their values than Water and Steam Pressure. As the spike injection algorithm takes into account the variance of the data, the spike fault for Flow is a value more distance from the normal behavior than Pressure. As the Pressure data is more stable, the spike fault, in this case, is not so distant from the normal data.

Figure 5 presents the training and response time, considering edge computing with i7-8700 3.2GHz, 12GB RAM. It is possible to observe that SVM and Random Forest are the machine learning models with the highest train time. Decision Tree, GNB, and LR are the fastest approaches. It is possible to observe that Decision Tree and GNB are more than ten times fast than SVM. The fastest approach is able to save more than 4 seconds in training, and 1.5 second in response time (see Fig. 5.c).

Taking into account the performance and time of all classifiers involved, GNB and LR would be possible choices to be deployed on typical edge hardware. Both provide precise and fast classification. In terms of machine learning performance, despite the GNB not having the best classification performance in Water Temperature, its performance is one of the best. In terms of time, GNB and LR show the best performance for edge computing hardware. Based on the obtained results, if there is a need to deal with new training and testing processes, both would deal with it in a shorter time and the precision of the predictions would be at a high level.

VI. Conclusion and Future Works

The revolution caused by the Internet-of-Things and machine learning algorithms applied to industrial environments has gained relevance over the last years. Industrial Internet-of-Things can benefit from machine learning by adding intelligence to the continuous monitoring and controlling process executed on manufacturing systems. In this context, edge



(a) Water Flow (b) Water Temp. (c) Water Presure

(d) Steam Flow (e) Steam Temp. (f) Steam Pressure

Fig. 5: Train and Response Time on Edge Computing.

computing is another trend, mainly due to the necessity to achieve low latency on computational tasks in industrial IoT applications.

Industrial IoT environments are prone to faults occurring in IoT devices. These faults represent a risk for the factory, since they can result in dangerous events and can stop production. So, it is essential to rely on precise and fast systems to detect the occurrence of IoT faults.

This paper proposes the use of five machine learning models to detect IoT device faults, using edge computing hardware to run the classifiers. It uses a dataset with real industrial data and proposes an algorithm to inject faults in the original dataset. The performance analysis is conducted using six metrics, measuring the precision of the machine learning models and the computational time to train and produce the outputs.

In future work, the authors intend to use a comprehensive set of edge computing hardware and implement communication policies in case of fault detection.

REFERENCES

[1] R. Marino, C. Wisultschew, A. Otero, J. M. Lanza-Gutierrez, J. Portilla, and E. de la Torre, "A machine-learning-based distributed system for fault diagnosis with scalable detection quality in industrial iot," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4339–4352, 2020.

[2] R. Mota, A. Riker, and D. Rosário, "Adjusting group communication in dense internet of things networks with heterogeneous energy sources," in *Anais do XI Simposio Brasileiro de Computacao Ubiqua e Pervasiva*. SBC, 2019.

[3] M. Silva, A. Riker, J. Torrado, J. Santos, and M. Curado, "Extending energy neutral operation in internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7510–7524, 2021.

[4] A. Riker, R. Mota, D. Rośario, V. Pereira, and M. Curado, "Autonomic management of group communication for internet of things applications," *International Journal of Communication Systems*, p. e5200, 2022.

[5] L. Barbosa, B. L. Dalmazo, W. Cordeiro, R. Immich, A. Abelém, and A. Riker, "Deconn: Combining minimum and neutral energy consumption strategies in iot networks," in *2022 14th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2022, pp. 101–107.

[6] J. Moraes, N. Matni, A. Riker, H. Oliveira, E. Cerqueira, C. Both, and D. Rosário, "An efficient heuristic lorawan adaptive resource allocation for iot applications," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–6.

[7] S. U. Jan, Y. D. Lee, and I. S. Koo, "A distributed sensor-fault detection and diagnosis framework using machine learning," *Information Sciences*, vol. 547, pp. 777–796, 2021.

[8] S. U. Jan, Y.-D. Lee, J. Shin, and I. Koo, "Sensor fault classification based on support vector machine and statistical time-domain features," *IEEE Access*, vol. 5, pp. 8682–8690, 2017.

[9] U. Saeed, S. U. Jan, Y.-D. Lee, and I. Koo, "Machine learning-based real-time sensor drift fault detection using raspberry pi," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2020, pp. 1–7.

[10] A. Javaid, N. Javaid, Z. Wadud, T. Saba, O. E. Sheta, M. Q. Saleem, and M. E. Alzahrani, "Machine learning algorithms and fault detection for improved belief function based decision fusion in wireless sensor networks," *Sensors*, vol. 19, no. 6, p. 1334, 2019.

[11] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through svm classifier," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 340–347, 2017.

[12] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *2010 sixth international conference on intelligent sensors, sensor networks and information processing*. IEEE, 2010, pp. 269–274.

[13] A. Naimi, J. Deng, S. Shimjith, and A. J. Arul, "Fault detection and isolation of a pressurized water reactor based on neural network and k-nearest neighbor," *IEEE Access*, vol. 10, pp. 17 113–17 121, 2022.

[14] A. Khodabakhsh, I. Ari, M. Bakír, and A. O. Ercan, "Multivariate sensor data analysis for oil refineries and multi-mode identification of system behavior in real-time," *IEEE Access*, vol. 6, pp. 64 389–64 405, 2018.

[15] "Tupras dataset," accessed 01 October 2022. [Online]. Available: https://www.openml.org/d/41170

[16] Y. Liu, C. Chi, Y. Zhang, and T. Tang, "Identification and resolution for industrial internet: Architecture and key technology," *IEEE Internet of Things Journal*, 2022.