International Journal of Informatics and Applied Mathematics
e-ISSN:2667-6990 Vol. 7, No. 2, 49-70

Smart Predictive Maintenance for Energy System Performed by Artificial Intelligence

Sahli Nabil[1,2], Kechida Sihem[1], and Merzouki Rochdi[3]

[1] LAIG Laboratory, 8 May 1945 University, Algeria
[2] Sonelgaz Distribution Constantine Region RD, Constantine, Algeria
[3] Lille University, CRISTAL Laboratory, Lille, France
sahli.nabil@univ-guelma.dz

Abstract. In the era of digital transformation, artificial intelligence (AI) is emerging as a foundational technology that is driving efficiency and innovation across many industries. One area where AI has had a significant impact is Smart predictive maintenance (SPM). Industries are gradually moving away from old models of reactive maintenance to proactive methods using AI. This shift helps minimize downtime, reduce costs, and improve operational efficiency. This article explores the many benefits, real-world applications, and techniques through which AI is enabling the implementation of SPM. Multi-agent system-based SPM employing machine learning classifiers has been used combined with deep learning proposed algorithms LSTM for optimizing SPM for energy systems at SONELGAZ Algeria. Using the forecast model and analyzing time-series data, LSTM model has obtained good accuracy with almost 97% accuracy. The experimental results showcase remarkable performance, achieving a Score about of 92% for binary classification and an impressive 97% for multiple classifications. Comparative analysis highlights the superiority of the MAS-LSTM hybrid approach in prediction accuracy. Our solution model, SIPM (Smart energy system, Intelligent, Predictive, and Maintenance), implemented in Python, predicts a device failure probability within 30 days as 0.0046.

Keywords: Artificial Intelligence · Autonomy Robot · Smart Predictive Maintenance · Energy Systems · LSTM · Multi-Agent System.

Received: 16 December 2024 | Revised: 21 December 2024 | Accepted: 20 January 2025
DOI: 10.53508/ijiam.1602468

1 Introduction

Predictive maintenance (PM) is a strategy that directly monitors the condition and performance of equipment during normal operation to reduce the likelihood of failures. SPM attempts to keep costs low by reducing the frequency of maintenance tasks, reducing unplanned breakdowns, and eliminating unnecessary preventive maintenance.

AI-powered predictive maintenance is an advanced technological approach that uses artificial intelligence (AI) to predict and prevent equipment failures within an infrastructure before they occur. As can be seen from Figure 1, there are three main types of maintenance methods.

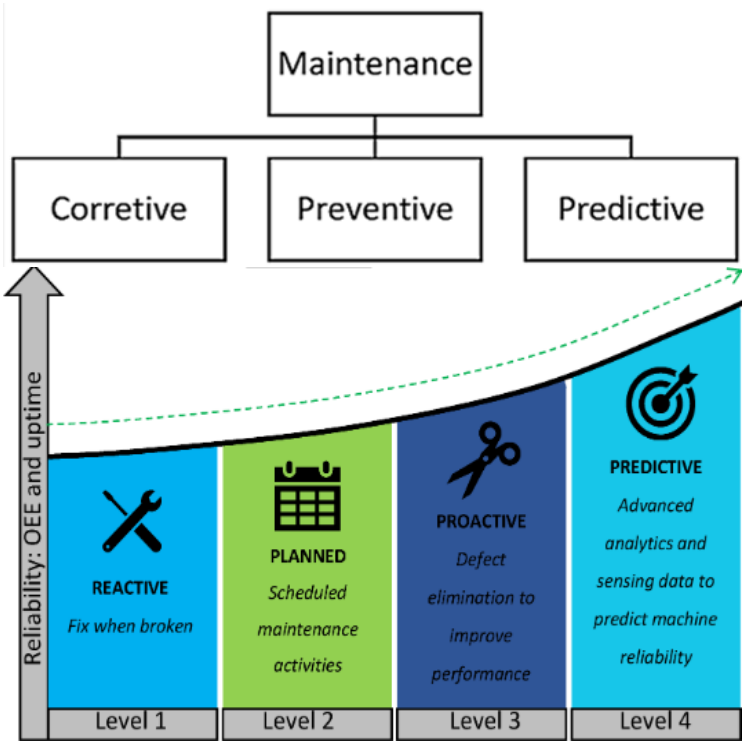

Fig. 1: The maintenance methods and levels [1].

Smart Predictive Maintenance (SPM) is the practice of using Big Data analytics tools and techniques to detect anomalies and predict failures in Energy System equipment before they occur. It is a concept where the optimal time to perform maintenance is determined based on real-time, large-scale data. This differs from planned preventive maintenance, which is performed regardless of the actual condition of the equipment. By ensuring that maintenance is performed

only when necessary, this strategy maximizes resource utilization and minimizes downtime. The use of AI techniques and autonomous robots equipped with intelligent sensors and actuators facilitates the implementation of such Energy System maintenance strategies.

1.1 AI-based Predictive Maintenance in Energy Sector

In the energy sector, AI-based predictive maintenance ensures the reliability and efficiency of power distribution and generation systems. AI algorithms are used in wind farms to evaluate sensor data and anticipate component issues, such as a bearing or gearbox failure, before they cause downtime. With these predictive capabilities, energy companies like SONELGAZ in Algeria can maintain high levels of energy production and avoid the significant expense of unplanned outages.

1.2 The Use of AI and Autonomy Robots for Predictive Maintenance

AI is revolutionizing predictive maintenance by leveraging technologies like advanced data analytics, pattern recognition, and predictive modeling. Here's how AI is improving each area:

Advanced data analytics, where AI systems examine vast amounts of information generated by industrial operations, such as maintenance logs, sensor data, and operating logs. These algorithms can provide deep insights into operational efficiency and equipment health by detecting patterns and correlations that human analysts might miss.

Maintenance Schedule Optimization, where AI's predictive capabilities enable dynamic optimization of maintenance schedules. By ensuring that maintenance tasks match the actual needs of the equipment, it is possible to extend its lifespan and eliminate unnecessary downtime, as presented in Figure 2.

Fig. 2: AI and Autonomy Robots (AR) at the service of energy systems predictive maintenance [2].

Anomaly Detection, where AI systems monitor Smart Devices in real-time and detect any unusual deviations from standard operating parameters [3]. AI can respond quickly to prevent small issues from becoming major problems by identifying these anomalies before they occur.

Our proposed LSTM model is designed to determine the conditions and the problems which are about to occur and when the maintenance should be performed to overcome the problems. Our solution model SIPM is presented, where S : Smart energy system, I : Intelligent, P : Predictive and M : Maintenance. The implementation of our (LSTM and Multi-Agent System) hybrid solution in Python language, with the probability that Smart Device will fail within 30 days, is 0.004578595. We combine LSTM model for optimizing failure detections in energy platform and multi-agent for optimizing communication between (UAV, sensors, actuators, and remote terminal unit (RTU)) composed the Smart Grids and smart gas pipelines platforms.

2 Related Work

Predictive Maintenance benefits include reduced costs, prolonged Smart Device lifetime, higher product quality and plant safety [4]. However, PM in real factories faces many challenges due to the need to integrate the various technologies such as AI, AR, PM, and MAS. Key challenges for the implementation of PM in energy systems include the processing of a large amount of sensory data from various monitored processes such as production and logistics and model development that will permit timely Smart Device energy system monitoring for robust energy systems [5]. Energy systems of IMs are a main research discipline in energy system management [6].

AI-driven methods of IMs have enhanced the reliability and efficiency of PM due to the development of different methods based on NN, fuzzy logic, neural-fuzzy systems, expert systems and MAS, among others. ML methods have been widely used for Energy System, yet more recently, various deep learning (DL) methods driven by advanced in DL have been developed, in order to implement Energy System in industrial equipment, enhancing PM reliability. DL uses long causal chains of NN layers, where each layer transforms non-linearly the activation of the network into a higher, more abstract representation based on the development of computational models of the real complex system. DL methods have improved the state of the art in energy systems of IMs, overcoming drawbacks of traditional data-driven energy systems methods, including their limited ability to learn from raw data and the dependence on experts for feature selection and extraction [7]. Figure 3 shows the flows of the energy systems sub-processes of ML and DL based methods for IMs [8]. Big Data analyze for developing predictive maintenance solution integrated with artificial intelligence and autonomy robots [9,10].

Recent works in the domain of using UAV and autonomy drones in predictive maintenance as “Autonomous predictive maintenance of quadrotor UAV with multi-actuator degradation” [12], and “Emerging Technologies in Secure Data

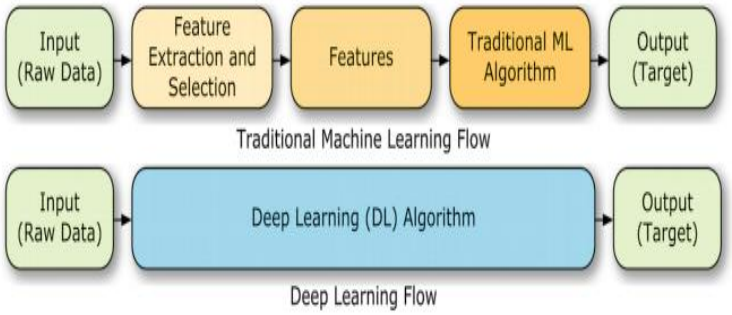

Fig. 3: Energy systems predictive maintenance flows of traditional ML and DL based methods [11].

Sharing, Predictive Maintenance, and Autonomous Systems in 5G Networks" [13], finally "Robotics and Artificial Intelligence (AI) for Maintenance" [14], and the work Predictive maintenance as an artificial intelligence service: a study of value creation [15]. Finally, the use of Internet of Things-Based Control of Induction Machines: Specifics of Electric Drives and Wind Energy Conversion Systems [16] for developing predictive maintenance in Electric Drives and Wind Energy Conversion Systems.

3 Deep Learning and Multi Agent System– A Brief Overview

In the case of predictive maintenance, AI solutions can analyze current operational conditions and look for indications that a piece of Smart Grid equipment may fail in the future — even if it hasn't displayed any overt operational issues. By assessing current machine performance against baseline data, AI tools can pinpoint small reductions in efficiency that may suggest the need for maintenance. Maintenance Teams are then alerted to these needs and can replace specific parts before failure occurs.

The learning capacity of AI and more particularly of data mining allows us in certain cases to move from a preventive maintenance logic, based on compliance with manufacturers' recommendations and/or on feedback from operators, to a predictive maintenance logic taking into account a large amount of information or measurements available relating in particular to the equipment installed on the network.

In this context, the automation of tasks allows asset management which, coupled with AI techniques, significantly increases the availability of equipment and opens up prospects for reducing costs, whether economic or environmental.

In low voltage, for example, France has two million departures. Based on the history of replacements, but also on a number of exogenous variables (humidity, nearby work, etc.), a Smart Device learning algorithm can calculate the probability of a cable failing based on its characteristics and its environment. Similar

applications are being developed for medium-voltage cables and transformers in source substations.

AI techniques of a different kind are also being used on overhead cables. This is the case, for example, for overhead lines inspected each year by helicopter or drone. Image recognition here makes it possible to optimize the programming of the renovation of technical equipment. Indeed, rather than spotting faults with the naked eye, based on the expertise of field agents, the scheduled renovation of networks is now triggered by automatic diagnostics based on the analysis of photographs. The ML subclass studied are presented in Figure 4. These techniques used for failure prediction in energy systems are diverse and rely on several scientific and technological approaches:

- *Time series analysis*: Evaluation of data collected over time to detect anomalies.
- *Neural networks*: Used to model complex relationships between different flight parameters.
- *Classification algorithms*: To identify and categorize the different anomalies encountered.

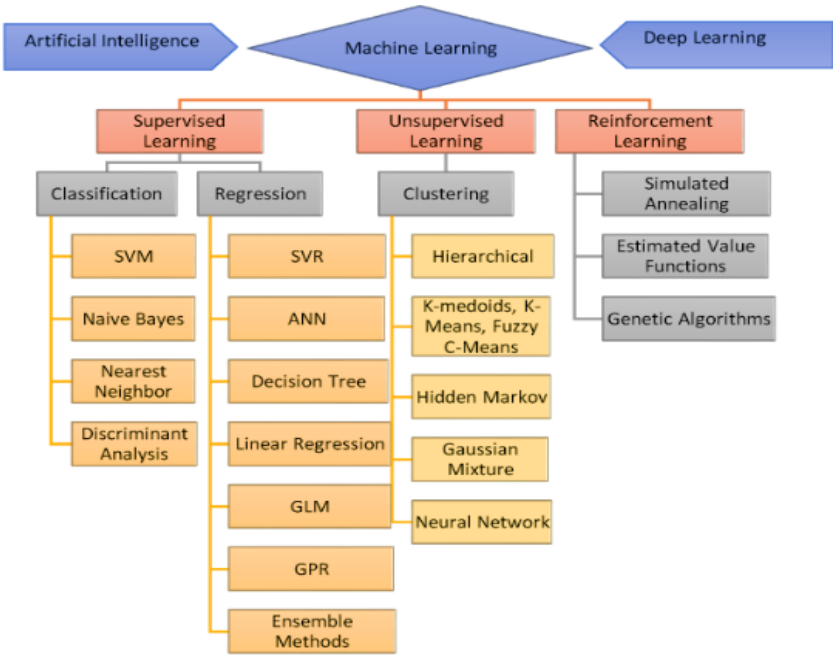

Fig. 4: ML Classification Techniques [17].

For example, the use of Random Forest can allow an efficient classification of potential failures thanks to its supervised learning ensemble approach. DL ap-

proaches are proven to achieve state-of-the-art performance in predictive maintenance because of their capability to transform complex information and Big Data databases fusion problems into supervised multi-label classification tasks [18].

Convolutional Neural Networks (CNNs) A typical CNN architecture consists of alternating and stacking convolution and pooling layers followed by fully connected layers. Convolutional layers use trainable filters called kernels that propagate the input data with a continued or windowed sliding technique.

Recurrent Neural Networks (RNNs) are networks that perform the same computation for every element in a set of sequential input data. Even though their typical architecture is simple, consisting of an input layer, a hidden layer and an output layer, they are difficult to train and subject to a lot of fine-tuning.

Auto-Encoders (AEs) The AE is a particular type of neural network that compresses the input data into a low-dimensional code and attempts to reconstruct the data from this latent-space representation. Specifically, AE training aims to converge to the identity function, so that the output X' is similar to input X. Traditionally, AE architecture consists of three core parts. Several variants of AE have been developed, Auto Encoder (DAE) is an extension version of the basic auto encoder, which is trained to reconstruct the stochastically corrupted input data by adding isotropic Gaussian and by forcing the hidden layer to explore robust features; Sparse Auto Encoder (SAE) imposes sparse constraints on the hidden layers while imposing activation closing to zero. This is used for fault diagnosis and intelligent fault signature. Variation Auto Encoder (VAE) is a deep generative architecture constructing a probability distribution for a latent variable as presented in Figure 5.

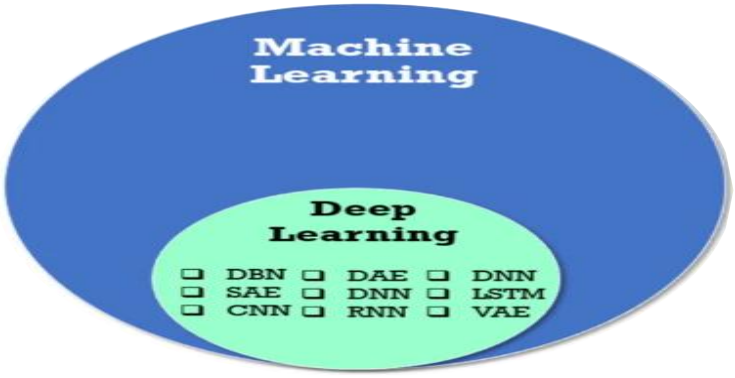

Fig. 5: Deep learning methods.

To overcome this fundamental limitation and to solve such complex and dynamic decision processes, a MAS is required to provide intelligent energy management and home automation. The MAS is a distributed artificial intelligence

system that allows the intercommunication and the interaction between intelligent agents. MAS have been defined using several terms going by one of the most common definitions, MAS is made up of various interacting agents, where an agent is a computer system with the ability to perform autonomous actions in order to accomplish certain goals in some environments. An intelligent agent achieves its goals by perceiving its environment and utilizing observations to decide on which action(s) to perform in the global predictive maintenance solution.

4 Proposed Solution and Experimental Results

Our proposed SPM pyramid is presented in Figure 6 use LSTM (Long Short-term Memory) Deep Learning solution, where we propose the use of fly robots, different intelligent sensors (Automatic, Adaptive, Energy saving, Wireless, Low cost, Real-time, Robust and Miniaturized), with the use multisource network of sensors (M), vibration and temperature sensors (V), Big data analytic tolls (B), interoperability challenges (I), decentralization of control (D) and virtual and Nano sensors (N), that trends MVBIDN.

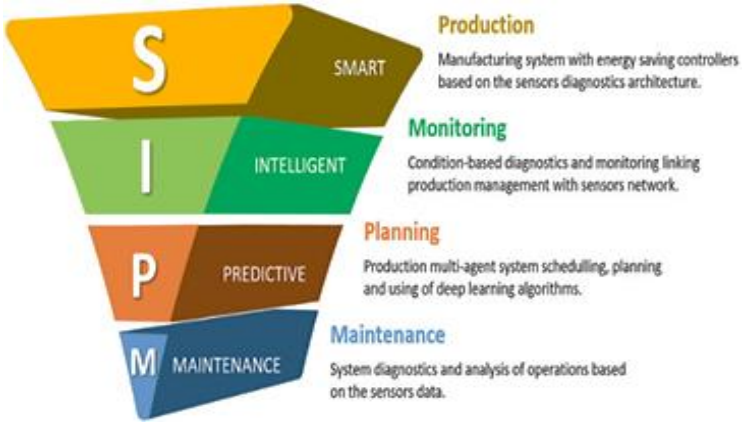

Fig. 6: Smart Predictive Maintenance pyramid.

The proposed framework in Figure 4 represents a Smart Device learning approach adapted to the building context. The framework is composed of five steps: data collection, data processing, model development, fault notification and model improvement, as presented in Figure 7.

Our work is related to the 3 W dataset and downloadable from the UCI website at the link `https://archive.ics.uci.edu/ml/datasets/3W+dataset`. The dataset, containing observations related to off-shore oil wells, comprises 1984-time series divided into nine classes: good-predictive-maintenance (class 0) and eight error classes (bad-predictive-maintenance). However, one of the errors

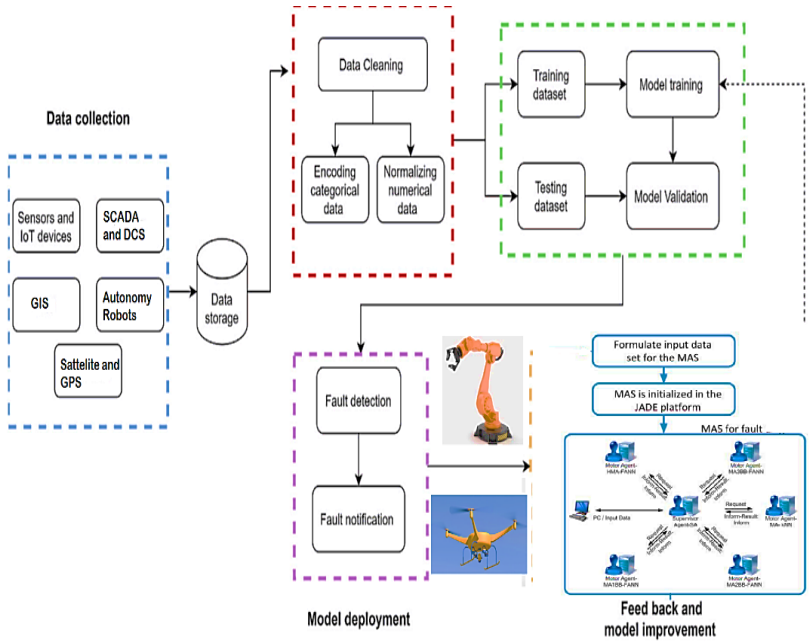

Fig. 7: The proposed framework for energy systems predictive maintenance based on MAS and LSTM.

is poorly represented in the dataset, so we consider only the error types from 1 to 6 and 8. Furthermore, we use a train-validation-test split, as shown in Figure 8.

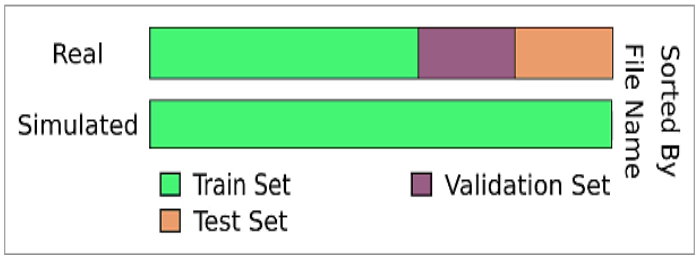

Fig. 8: Dataset for predictive maintenance test results.

To handle the large amount of information provided by the multiple assets of the factories, it's an intelligent solution to establish a multi-agent system composed of multiple intelligent agents interacting with each other to solve difficult and impossible problems to solve individually. The structure of this system could

consist of several agents, each of which is modeled as a virtual digital shell of each asset in the production line and collects the data generated by the asset, and a central agent (middle-ware running on the server) to which it's implemented a learning algorithm to identify abnormal behavior. MAS are a branch of Distributed Artificial Intelligence that involve a group of autonomous agents working together to achieve a common goal. Agents can cooperate or compete, share knowledge with each other, and operate in a loosely connected environment. The key elements of an agent system include the entity, environment, parameters, and action. The entity refers to the type of agent, which can be software, hardware, or a combination of both. The environment refers to the place where the agent is located and can include various features such as accessibility, determinism, dynamism, and continuity. The parameters refer to the different data types that an agent can sense from the environment. The benefits of using MAS include increased efficiency, low cost, flexibility, reliability, scalability, and reusability for optimizing detecting failures in energy platforms.

The contribution of multi-agent system technology with the LSTM model for predictive maintenance of predictive platforms and optimization of communications between the different intelligent components of these platforms as well as the management of a multi-generation system of performance of the prediction from generation 1 to generation N, where N evolves over time. for each generation of the prediction of operating conditions without imposed by the system, according to the evolution of the technical constraints of the energy platforms. The costs of degradation of the platforms which are an important factor in the decision to replace faulty equipment are supported by agents of MAS. The decisions to be supported in the context of predictive maintenance from generation 1 to generation N are also supported by the MAS, as presented in Figure 9. All is coupled with the system (SCADA- Supervisory Control And Data Acquisition).

Smart Predictive Maintenance as a Service (SIPMaaS), the rise of cloud computing and subscription-based models will pave the way for SIPMaaS offerings. SIPMaaS providers will offer scalable, cost-effective predictive maintenance solutions hosted on cloud platforms, allowing organizations to access advanced analytics capabilities, predictive models, and expertise without the need for extensive infrastructure investment or in-house data science resources. Advancements in AI, Mobility Robotics (MR), and autonomous systems will enable the development of autonomous maintenance systems capable of self-diagnosis, self-repair, and self-optimization. These systems will leverage AI-based algorithms to continuously monitor energy systems and energy platforms, detect anomalies, and perform smart maintenance tasks autonomously, reducing the need for human intervention and improving overall operational efficiency.

Part of the model instance of the MAS Python implementation for the (N) generation from generation 1 to generation N is presented:

```
import random
class Agent:
    def __init__(self, id):
        self.id = id
```

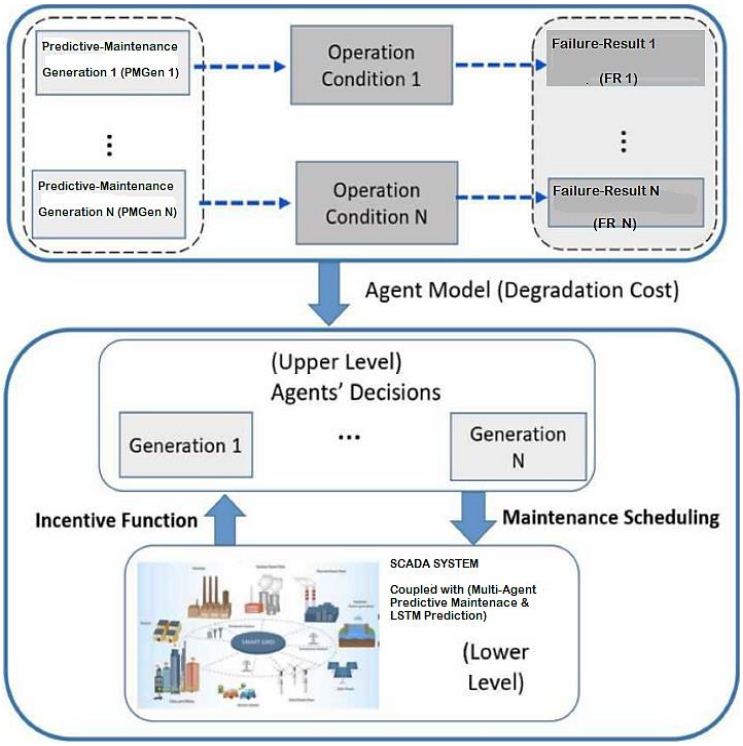

Fig. 9: Predictive maintenance modeling MAS and LSTM hybrid solution

```
        self.position = random.randint(0, 100)

    def move(self):
        self.position += random.choice([-1, 1])  # Move left
            or right

agents = [Agent(i) for i in range(N)]
for _ in range(N):  # Simulate N time steps
    for agent in agents:
        agent.move()
        print(f'Agent {agent.id} is at position {agent.
            position}')

import numpy as np
class ConsensusAgent:
    def __init__(self, id, initial_value):
        self.id = id
        self.value = initial_value

    def update_value(self, neighbors):
```

```
        self.value = np.mean([self.value] + [neighbor.value
            for neighbor in neighbors])

agents = [ConsensusAgent(i, random.uniform(0, N)) for i in
    range(N)]
for _ in range(N):  # Simulate N iterations
    for agent in agents:
        agent.update_value([other for other in agents if
            other != agent])
        print(f'Agent {agent.id} has value {agent.value}')

from uagents import Agent
class MyAgent(Agent):
    def __init__(self, name):
        super().__init__(name)
        self.state = "initial"

    def act(self):
        # Define agent behavior here
        pass

from uagents.protocols import ExchangeProtocol
class MyAgent(Agent):
    def __init__(self, name):
        super().__init__(name)
        self.protocol = ExchangeProtocol()

    def communicate(self, other_agent):
        self.protocol.send_message(other_agent, "Agent
            Decisions!")

from uagents import Simulation
simulation = Simulation()
agent1 = MyAgent("Agent1")
agentN = MyAgent("AgentN")
simulation.add_agent(agent1)
simulation.add_agent(agentN)
simulation.run()
```

A MAS refers to a network of agents collaborating to achieve the same objective (optimize predictive maintenance cost, from generation 1 to generation N). This system comprises numerous individual programs or hardware components (agents) that are simpler to construct and manage. Additionally, these agents can dynamically and swiftly adapt to changes in their environment (energy platforms).

Common data sources for predictive maintenance problems are:

– *Failure history*: The failure history of an energy platform or component within the smart devices and materials.

- *Maintenance history*: The repair history of an energy platform, e.g., error codes, previous maintenance activities or component replacements.
- *Energy platforms conditions and usage*: The operating conditions of a Smart Device, e.g., data collected from sensors.
- *A energy platform features*: The features of an energy platform, e.g., engine size, make and model, location.
- *Operator features*: The features of the operator, e.g., gender, past experience.

The data for this example comes from different sources which are real-time telemetry data collected from an energy platform, error messages, historical maintenance records that include failures and an energy platform information such as type and age.

The performance of the models developed with these algorithms can be measured by computing the difference between the predicted class for a given input versus the actual class of the input. For example, the correct classification will predict data as benign if the input data was benign. To quantify the detection performance of the classifier, the 2×2 confusion matrixes is used (shown in Table 1) as it provides all the possible outcomes of a prediction and has the forms True Positive, True Negative, False Positive, and False Negative of the classifier.

Table 1: Confusion Matrix for Engine Failure Predictions

Class	**Failed Prediction**	**Non-Failed Prediction**
Engine Failed	True Positive (TP)	False Positive (FP)
Engine Not Failed	False Negative (FN)	True Negative (TN)

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{TN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Predictive Maintenance using our proposal solution implementation in Python langage :

- Loading Libraries
  - **Input 1 :**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,
    accuracy_score
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM,
    Activation
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
```

– Loading Dataset

- **Input 2 :**

```
dataset_train = pd.read_csv('../input/PM_train.txt',
    sep=' ', header=None).drop([26, 27], axis=1)
col_names = ['id', 'cycle', 'setting1', 'setting2', '
    setting3', 's1', 's2', 's3', 's4', 's5', 's6', 's7'
    , 's8', 's9', 's10', 's11', 's12', 's13', 's14', '
    s15', 's16', 's17', 's18', 's19', 's20', 's21' ]
dataset_train.columns = col_names
print('Shape of Train dataset: ', dataset_train.shape)
dataset_train.head()

# Output
# Shape of Train dataset:  (20631, 26)
```

- **Output 2:**

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3

s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

- **Input 3:**

```
dataset_test = pd.read_csv('../input/PM_test.txt', sep=
    ' ', header=None).drop([26, 27], axis=1)
```

```
dataset_test.columns = col_names
# dataset_test.head()
print('Shape of Test dataset: ', dataset_train.shape)
dataset_train.head()

# Output
# Shape of Test dataset:  (20631, 26)
```

- **Output 3:**

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3

s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

– Loading Truth table

- **Input 4 :**

```
pm_truth = pd.read_csv('../input/PM_truth.txt', sep=' '
    , header=None).drop([1], axis=1)
pm_truth.columns = ['more']
pm_truth['id'] = pm_truth.index + 1
pm_truth.head()
```

- **Output 4 :**

	more	id
0	112	1
1	98	2
2	69	3
3	82	4
4	91	5

- **Input 5 :**

```
# Generate column max for test data
rul = pd.DataFrame(dataset_test.groupby('id')['cycle'].
    max()).reset_index()
rul.columns = ['id', 'max']
rul.head()
```

- **Output 5 :**

	id	max
0	1	31
1	2	49
2	3	126
3	4	106
4	5	98

- **Input 6 :**

```
# Run to failure
pm_truth['rtf'] = pm_truth['more'] + rul['max']
pm_truth.head()
```

- **Output 6 :**

	more	id	rtf
0	112	1	143
1	98	2	147
2	69	3	195
3	82	4	188
4	91	5	189

The steps of implementing a predictive maintenance model is provided using an example scenario where the goal is to predict failures due to certain components of a smart energy platforms. Typical steps of predictive maintenance such as feature engineering, labelling, training and evaluation are explained using the example data sets. Predictive models are built both using Python packages and Azure SmartDevice Learning Studio.

- **Input 7 :**

```
pm_truth.drop('more', axis=1, inplace=True)
dataset_test = dataset_test.merge(pm_truth, on=['id'],
    how='left')
```

```
dataset_test['ttf'] = dataset_test['rtf'] -
    dataset_test['cycle']
dataset_test.drop('rtf', axis=1, inplace=True)
dataset_test.head()
```

- **Output 7 :**

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	21.61	553.90	2388.04	9050.17	1.3
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	21.61	554.85	2388.01	9054.42	1.3
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	21.61	554.11	2388.05	9056.96	1.3
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	21.61	554.07	2388.03	9045.29	1.3
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	21.61	554.16	2388.01	9044.55	1.3

s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	ttf
47.20	521.72	2388.03	8125.55	8.4052	0.03	392	2388	100.0	38.86	23.3735	142
47.50	522.16	2388.06	8139.62	8.3803	0.03	393	2388	100.0	39.02	23.3916	141
47.50	521.97	2388.03	8130.10	8.4441	0.03	393	2388	100.0	39.08	23.4166	140
47.28	521.38	2388.05	8132.90	8.3917	0.03	391	2388	100.0	39.00	23.3737	139
47.31	522.15	2388.03	8129.54	8.4031	0.03	390	2388	100.0	38.99	23.4130	138

- **Input 8 :**

```
dataset_train['ttf'] = dataset_train.groupby(['id'])['
    cycle'].transform(max) - dataset_train['cycle']
dataset_train.head()
```

- **Output 8 :**

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3

s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	ttf
47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	191
47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	190
47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	189
47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	188
47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	187

- **Input 9 :**

```
df_train = dataset_train.copy()
df_test = dataset_test.copy()
period = 30
df_train['label_bc'] = df_train['ttf'].apply(lambda x:
    1 if x <= period else 0)
df_test['label_bc'] = df_test['ttf'].apply(lambda x: 1
    if x <= period else 0)
df_train.head()
```

- **Output 9 :**

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3

s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	ttf	label_bc
47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	191	0
47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	190	0
47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	189	0
47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	188	0
47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	187	0

- **Input 10: LSTM Network**

```
nb_features = X_train.shape[2]
timestamp = seq_length
model = Sequential()
model.add(LSTM(input_shape=(timestamp, nb_features),
          units=100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='
    adam', metrics=['accuracy'])
model.summary()
```

- **Output 10 :**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50, 100)           50000
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 100)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                30200
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 80,251
Trainable params: 80,251
Non-trainable params: 0
```

- **Input 11: LSTM Network**

```
# fit the network
model.fit(X_train, y_train, epochs=10, batch_size=200,
    validation_split=0.05, verbose=1,
          callbacks=[EarlyStopping(monitor='val_loss',
              min_delta=0, patience=0, verbose=0, mode=
              'auto')])
```

- **Output 11 :**

```
<keras.callbacks.History at 0x7f2a963ef1d0>
```

- **Input 12:**

```
# training metrics
scores = model.evaluate(X_train, y_train, verbose=1,
    batch_size=200)
print('Accuracy: {}'.format(scores[1]))
```

- **Output 12 :**

```
20531/20531 [==============] - 8s 392us/step
Accuracy: 0.9751108148252323
```

- **Input 13 :**

```
y_pred = model.predict_classes(X_test)
print('Accuracy of model on test data: ',
    accuracy_score(y_test, y_pred))
print('Confusion Matrix: \n', confusion_matrix(y_test,
    y_pred))
```

- **Output 13 :**

```
Accuracy of model on test data:  0.9913050169282857
Confusion Matrix:
 [[12633    31]
 [ 82   250]]
```

- **Input 14 :**

```
def prob_failure(SmartDevice_id):
    SmartDevice_df = df_test[df_test.id ==
        SmartDevice_id]
    SmartDevice_test = gen_sequence(SmartDevice_df,
        seq_length, seq_cols)
    m_pred = model.predict(SmartDevice_test)
    failure_prob = list(m_pred[-1] * 100)[0]
    return failure_prob

SmartDevice_id = 16
print('Probability that SmartDevice will fail within 30
    days: ', prob_failure(SmartDevice_id))
```

- **Output 14 :**

```
Probability that SmartDevice will fail within 30 days:
    0.004578595
```

5 Conclusion

AI-powered predictive analytics is a powerful method to help ensure the smooth and efficient operation of all elements within the energy infrastructure. It can help anticipate and prevent failures and identify the patterns that can make a valuable information source for future improvements.

By adopting this approach in SONELGAZ energy efforts, we can enable more sustainable operations by scheduling repairs and maintenance targeted at the weak points and failure modes indicated by AI algorithms and MAS. It will help energy factories act proactively and save time and money on resources by avoiding unnecessary repairs and procedures or unexpected downtimes caused by sudden malfunctions.

Smart Predictive Maintenance uses Big Data and AI algorithms to anticipate equipment issues before they occur, revolutionizing traditional maintenance procedures across energy industries. By being proactive, this method reduces downtime, improves operational efficiency, and ensures service reliability and safety.

As future work, current algorithms and models still need to be improved. Now, they are based on the deviation from the reference baseline, and it is necessary to monitor existing energy platforms to obtain new and additional data. As a new data about the failures of the energy industry platforms becomes available from the intelligent sensors and mobility robots, the threshold for this condition of failure will be assessed. And as long as this value will be significantly different from the energy platforms incident data, it will be used by the smart optimized algorithms into the energy engineering process to improve its generalization, robustness, and accuracy.

But to establish levels of degradation for the solar energy platforms, it is essential to know the usage of the solar panels and the ‘culture of maintenance’

of the energy platforms, and it is not an easy task. This means that probably we will have little information about the state of deterioration or degradation of the solar panels and production central electricity platforms at the very beginning. And even though the smart optimized algorithms will show some alarms, the component will be working until failure.

This paper serves as a valuable resource for energy industry professionals looking to harness the power of AI and autonomy robotics to drive innovation and excellence in renewable energy platforms infrastructure smart predictive maintenance.

Implementing AI in maintenance offers several benefits for companies as SONELGAZ Algeria, such as, lower costs, process improvement, extended equipment lifecycle.

References

1. Thyago P Carvalho, Fabrízzio AAMN Soares, Roberto Vita, Roberto da P Francisco, João P Basto, and Symone GS Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.
2. Marwa Assim, Qasem Obeidat, and Mustafa Hammad. Software defects prediction using machine learning algorithms. In *2020 International conference on data analytics for business and industry: way towards a sustainable economy (ICDABI)*, pages 1–6. IEEE, 2020.
3. Clayton Miller, Pandarasamy Arjunan, Anjukan Kathirgamanathan, Chun Fu, Jonathan Roth, June Young Park, Chris Balbach, Krishnan Gowri, Zoltan Nagy, Anthony D Fontanini, et al. The ashrae great energy predictor iii competition: Overview and results. *Science and Technology for the Built Environment*, 26(10):1427–1447, 2020.
4. Jay Lee, Edzel Lapira, Shanhu Yang, and Ann Kao. Predictive manufacturing system-trends of next-generation production systems. *Ifac proceedings volumes*, 46(7):150–156, 2013.
5. Jose-Raul Ruiz-Sarmiento, Javier Monroy, Francisco-Angel Moreno, Cipriano Galindo, Jose-Maria Bonelo, and Javier Gonzalez-Jimenez. A predictive model for the maintenance of industrial machinery in the context of industry 4.0. *Engineering Applications of Artificial Intelligence*, 87:103289, 2020.
6. Jinjiang Wang, Peilun Fu, Laibin Zhang, Robert X Gao, and Rui Zhao. Multilevel information fusion for induction motor fault diagnosis. *IEEE/ASME Transactions on Mechatronics*, 24(5):2139–2150, 2019.
7. Maria Drakaki, Yannis L Karnavas, Panagiotis Tzionas, and Ioannis D Chasiotis. Recent developments towards industry 4.0 oriented predictive maintenance in induction motors. *Procedia computer science*, 180:943–949, 2021.
8. Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE systems journal*, 13(3):2213–2227, 2019.
9. Sam Heim, Jason Clemens, James E Steck, Christopher Basic, David Timmons, and Kourtney Zwiener. Predictive maintenance on aircraft and applications with digital twin. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4122–4127. IEEE, 2020.

10. Murat Yildirim, Nagi Z Gebraeel, and Xu Andy Sun. Integrated predictive analytics and optimization for opportunistic maintenance and operations in wind farms. *IEEE Transactions on power systems*, 32(6):4319–4328, 2017.
11. Pooja Malik, Anita Gehlot, Rajesh Singh, Lovi Raj Gupta, and Amit Kumar Thakur. A review on ann based model for solar radiation and wind speed prediction with real-time data. *Archives of Computational Methods in Engineering*, 29(5):3183–3201, 2022.
12. F-y Shen, W Li, D-n Jiang, and H-j Mao. Autonomous predictive maintenance of quadrotor uav with multi-actuator degradation. *The Aeronautical Journal*, pages 1–25, 2024.
13. Tran Minh Tuan and Pham Thi Hoa. Emerging technologies in secure data sharing, predictive maintenance, and autonomous systems in 5g networks. *Journal of Humanities and Applied Science Research*, 7(9):1–15, 2024.
14. Diego Galar and Uday Kumar. Robotics and artificial intelligence (ai) for maintenance. *Monitoring and Protection of Critical Infrastructure by Unmanned Systems*, pages 206–223, 2023.
15. Kathrin Kirchner, Grzegorz Leszczyński, Marek Zieliński, and Bartosz Jędrzejczak. Predictive maintenance as an artificial intelligence service: a study of value creation. In *Handbook of Services and Artificial Intelligence*, pages 31–52. Edward Elgar Publishing, 2024.
16. Maria G Ioannides, Anastasios P Stamelos, Stylianos A Papazis, Erofili E Stamataki, and Michael E Stamatakis. Internet of things-based control of induction machines: Specifics of electric drives and wind energy conversion systems. *Energies*, 17(3):645, 2024.
17. Madhu Puttegowda and Sharath Ballupete Nagaraju. Artificial intelligence and machine learning in mechanical engineering: Current trends and future prospects. *Engineering Applications of Artificial Intelligence*, 142:109910, 2025.
18. John Eriksson. Machine learning for predictive maintenance on wind turbines: Using scada data and the apache hadoop ecosystem, 2020.