IoT-Enabled Predictive Maintenance for Industrial Automation Systems

by

Alisher Yelseitov

Submitted to the Department of Robotics and Mechatronics
in partial fulfillment of the requirements for the degree of

Master of Science in Robotics

at the

NAZARBAYEV UNIVERSITY

Apr 2025

Author .....
Department of Robotics and Mechatronics
Apr, 2025

Certified by .....
Tohid Alizadeh
Assistant Professor
Thesis Supervisor

Accepted by .....
Elizabeth Arkhangelsky
Associate Professor, School of Engineering and Digital Sciences

IoT-Enabled Predictive Maintenance for Industrial Automation Systems

by

Alisher Yelseitov

Submitted to the Department of Robotics and Mechatronics
on Apr, 2025, in partial fulfillment of the
requirements for the degree of
Master of Science in Robotics

Abstract

Under Industry 4.0, the complexity of industrial automation systems is growing, making data-driven, intelligent maintenance methods necessary. In order to detect mechanical failures and pressure anomalies in a modular production line, this thesis investigates the creation of an Internet of Things (IoT)-enabled predictive maintenance (PdM) framework. In order to perform binary classification of mechanical failures and multi-class classification of pneumatic pressure levels, the system incorporates a multi-task neural network that uses sensor-generated time-series data. Node-RED is used for cloud logging and real-time data collection in the experimental setup, and feature extraction is carried out on scenarios that simulate both optimal and problematic operating situations. The model needs more work because, although it performs well in malfunction detection, it has trouble generalizing pressure predictions. Recent developments in PdM, such as the use of machine learning, deep learning, IIoT platforms, and explainable AI, are highlighted in an extensive literature analysis. This work lays the groundwork for future improvements in smart manufacturing systems by offering a scalable and reasonably priced PdM strategy that meets the demands of contemporary industrial automation.

Thesis Supervisor: Tohid Alizadeh
Title: Assistant Professor

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Tohid Alizadeh, for always being patient, open, and helpful during this project. His expert advice, helpful criticism, and thoughtful questions at every milestone have made a big difference in how scientifically sound and useful my work is in real life.

I also want to thank Zhalgas Bolatbayev, my labmate and the only person who stays with us all the time, for all the help he has given me. Everyday work was both useful and fun because Zhalgas was willing to fix Node-RED flows, help set up experiments, and share his technical knowledge. The fact that he worked with others and encouraged them was very important to the success of this study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every day we rely on a wide range of machines. But the truth is that every machine eventually breaks down, unless it's being maintained. Companies follow different maintenance programs to increase operational reliability and reduce costs. Reactive maintenance is one approach, in which the machine is operated to its maximum capacity, and repairs are made only after it breaks down. However, there are some extremely costly components in this intricate system. Since it will be very expensive to restore severely damaged pieces, people really can't risk it breaking down. More significantly, though, it's a safety concern. Because of this, many organizations conduct routine equipment inspections in an effort to stop failure before it happens. Deciding when to perform preventive maintenance is a major difficulty. Researchers must plan conservatively because it cannot predict when a failure will occur, particularly when using safety-critical equipment. However, the losing machine life that is still usable by scheduling maintenance so early, which raises the expenses. Nonetheless, planning maintenance just before a machine breakdown is beneficial if we can accurately forecast when it will happen.

Chapter 2

Literature review

Because of its potential to significantly lower maintenance costs and downtime while improving overall operating efficiency, predictive maintenance, or PdM, has drawn a lot of attention in the context of Industry 4.0 [1]. PdM uses cutting-edge technology like the Industrial Internet of Things (IIoT), machine learning (ML), and artificial intelligence (AI) to avoid and identify industrial breakdowns [16],[3]. PdM has been especially transformed by AI and machine learning, which increase the precision and promptness of maintenance choices. Large amounts of sensor-generated data are efficiently analyzed by machine learning algorithms to identify irregularities and forecast equipment failures. Serradilla et al. emphasized deep learning models' capacity to represent intricate relationships within industrial data, highlighting their great efficacy in predictive maintenance scenarios [15] . The synergy between data-driven insights and operational strategies was also highlighted by Alhuqayl et al., who used IIoT and ML to demonstrate notable increases in maintenance efficiency [3]. Because of its ability to provide real-time monitoring and continuous data collection, the Industrial Internet of Things (IIoT) plays a crucial role in predictive maintenance. TIP4.0, a modular software system that improves sensor data processing and makes efficient real-time condition monitoring possible in industrial settings, was introduced by Resende et al.[14]. The usefulness of edge computing methods in predictive maintenance was further investigated by Lazzaro et al., who showed significant advantages in terms of decreased latency and improved real-time processing capabilities [9]. PdM

implementation still faces significant obstacles in spite of progress. First off, industrial stakeholders' trust is eroded by AI-driven models' incapacity to be explained or interpreted, especially in safety-critical applications [8]. Ucar et al. highlighted ongoing research targeted at improving the usability and trust of AI solutions in PdM [18] and emphasized the importance of model transparency and interpretability. Second, the training of precise predictive models is hampered by the lack of high-quality labeled datasets. Zhang et al. highlighted the importance of synthetic data generation techniques in enhancing model resilience and addressed them as practical ways to address the lack of high-quality datasets for PdM [20]. Furthermore, PdM's broad acceptance is constrained by implementation costs and resource limitations, especially for small and medium-sized businesses (SMEs). This discrepancy was acknowledged in studies by Ahmad and Kamaruddin, who proposed several affordable tactics SMEs could use to gain from predictive maintenance without having to make a significant upfront expenditure [2].

2.1 Importance of the subject

Predictive maintenance (PdM) has emerged as a vital approach within industrial automation due to its profound impact on operational efficiency, cost-effectiveness, and safety. With increasing complexity in manufacturing processes, equipment failures and unexpected downtime can significantly disrupt production schedules, resulting in considerable financial losses and compromised productivity. Traditional maintenance approaches, such as reactive and preventive methods, often lead to either frequent unnecessary inspections or unforeseen failures, thereby incurring high operational costs and resource inefficiencies.

Implementing predictive maintenance strategies addresses these issues by allowing industrial operators to anticipate machinery failures and proactively schedule necessary maintenance activities, significantly enhancing operational reliability and reducing overall maintenance expenditures. Beyond cost savings, predictive maintenance also directly contributes to improved workplace safety by minimizing risks

associated with equipment malfunctions, thus protecting both human personnel and industrial assets.

Moreover, predictive maintenance integrates seamlessly with modern technological advancements such as Industrial Internet of Things (IIoT), advanced analytics, and real-time data processing capabilities, making it a cornerstone of Industry 4.0 initiatives. This integration not only streamlines industrial processes but also drives innovation in operational management and resource optimization. Furthermore, the adoption of predictive maintenance methodologies enables industries to maintain competitive advantages by ensuring high-quality production, reducing downtime, and efficiently managing resources.

Given these factors, developing robust predictive maintenance solutions that incorporate advanced machine learning, IoT technologies, and intuitive visualization tools represents a critical area of both practical and academic interest, making it an essential and timely subject for research and industrial implementation.

2.2 Problem statement

Effective maintenance techniques are required due to the growing complexity of industrial automation systems under Industry 4.0. Conventional approaches, such reactive and preventative maintenance, either underutilize equipment life or run the danger of expensive breakdowns, which results in operational inefficiencies and safety concerns. A viable substitute is provided by Predictive Maintenance (PdM), which makes use of Artificial Intelligence (AI), Machine Learning (ML), and the Industrial Internet of Things (IIoT)[11]. The lack of high-quality labeled data impairs predictive accuracy and reliability; high costs and resource requirements limit small and medium enterprises' (SMEs) ability to implement PdM; and limited explainability in AI models erodes stakeholder trust, especially in safety-critical contexts. By creating cutting-edge PdM frameworks with explainable AI, reliable data processing, and economical deployment techniques, this research tackles these problems.

Chapter 3

Equipment & Dataset

The project was conducted on a modular production line, specifically the FESTO MPS500 (Figure 3-1). The FESTO MPS 500 (Modular Production System) serves as a multifunctional training platform that simulates automation processes akin to actual production lines. This facilitates scenarios concerning the transportation, sorting, processing, and quality control of products, rendering it suitable for the study and development of solutions in Industry 4.0, Predictive Maintenance, digital manufacturing, robotics, and the Internet of Things (IIoT)[6]. This provides a versatile platform for implementing advanced manufacturing concepts and evaluating their efficacy.

The project workflow occurred at the Handling and Processing station. The Handling & Processing station of the FESTO MPS 500 system focuses on the transportation of components, showcasing and practicing standard industrial conveyor activities. Any user-defined operation can be executed on the Processing Unit.:

-quality control (assessing geometric parameters, verifying the presence or absence of holes, color classification utilizing optical sensors);

-mechanical processing (pressing, marking, punching, gluing) utilizing supplementary pneumatic or servo motors;

-automated assembly of minor components (bushing insertion, riveting) with interchangeable tools.

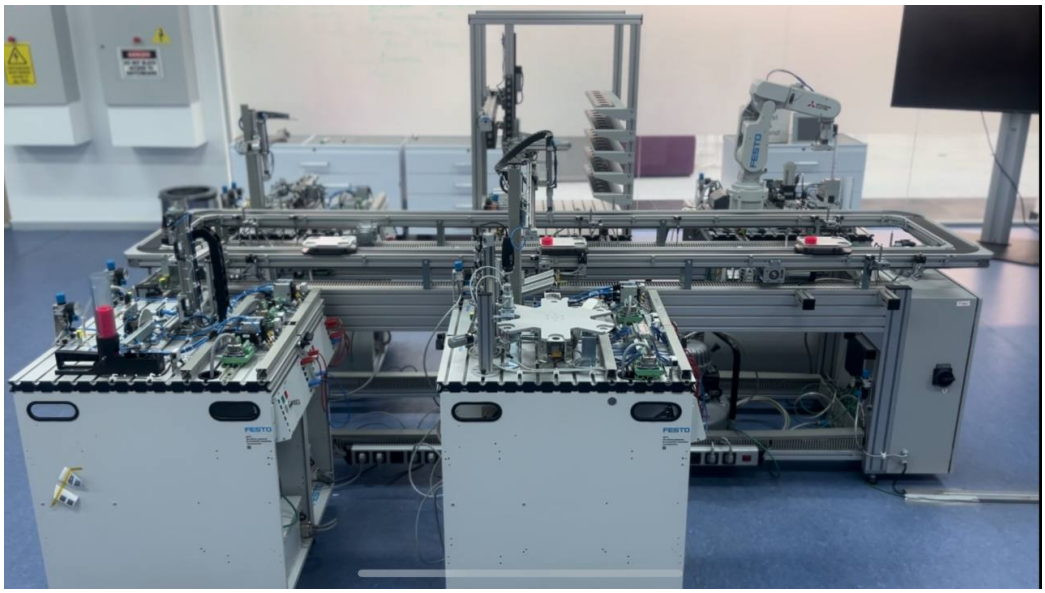


Figure 3-1: Festo MPS 512 modular production system.

The Handling station (Figure 3-2) is equipped with a Pneumatic Linear Drive (Fig. 3-4 (a)) with a movable modular gripper.Beginning with a signal of workpiece arrival, the Handling station's gripper lowers vertically down over the workpiece to pick it up and remove it off the carrier. It then begins traveling horizontally along the Pneumatic Linear Drive from the Downstream Station toward the End Position (Upstream Station); in the middle it is fixed by the intermediate point Sorting Position. The gripper sets the workpiece on the Processing Station and lowers it down at the ending point. Returns up to the Ending point (Upstream Station) and waits for the Processing Station's end signal to return the workpiece to the carrier. The Handling Station is managed by a Siemens S7-1200 PLC (CPU 314C-2 PN/DP) with a SIMATIC I/O-Simulator SM 374, and three positions in pneumatic linear drives are fitted with proximity sensors (Fig. 3-4 (b))

The workflow of the pneumatic linear actuator in the FESTO MPS 500 is realized using three proximity sensors located along the path of the gripper. These points are shown in (Fig. 3-3) and are labeled as follows:

point A (Starting point) corresponds to the Downstream Station variable,

point C (Intermediate point) corresponds to the Sorting Position variable,

point B (Ending point) corresponds to the Upstream Station variable.

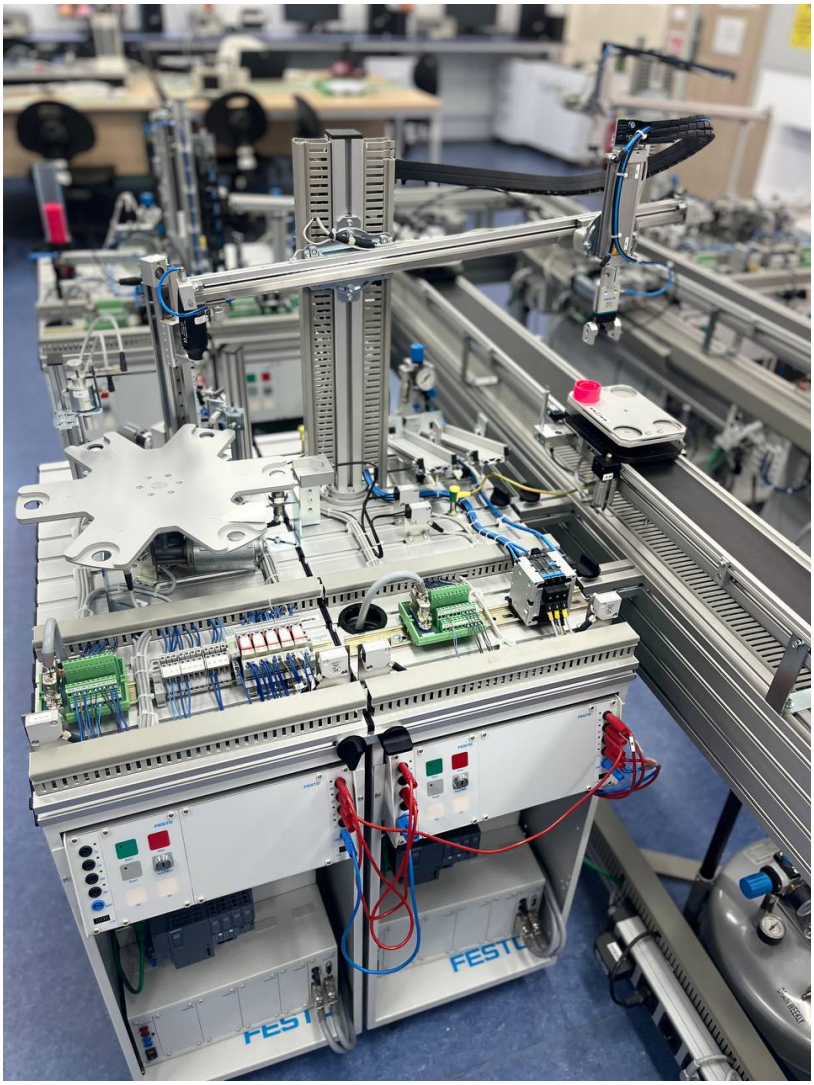


Figure 3-2: Handling station with trolley, controle console and PLC board.

The time stamps are recorded when the gripper crosses these sites in forward and backward directions; from them, the time intervals between nearby points are computed. The training data set's major characteristics are these intervals. For instance,

the intervals A – C and C – B are computed in the forward direction;

the intervals B – C and C – A are computed in the reverse manner.

Since the system is based on a pneumatic actuator, an important parameter is the air supply pressure, which is manually adjusted by the supply valve and monitored by an analog manometer. The normal operating pressure is 5 bar. Any deviation from this value is considered as a potential fault in the system. Reduced pressure results in increased gripper travel time - which may indicate an air leak or weakened force - while excessive pressure can accelerate movement, causing risks to the accuracy and durability of the actuator.

The image furthermore illustrates two critical segments, mech_1 and mech_2, delineating the regions between closely positioned sensors, utilized for ascertaining the existence or nonexistence of mechanical flaws. Under typical conditions, the duration of transit from A to C is around 4 seconds. Nonetheless, if mechanical wear, jamming, or other issues occur, the gripper will traverse the same distance at a much reduced speed. It is crucial to recognize that the lack of a fault in one direction (e.g., forward) does not ensure its nonexistence in the opposite direction (backward); so, predictions are conducted independently in both directions.

Some mechanical faults have been included artificially to create the training dataset, therefore mimicking actual situations. In industrial operation, such problems might be produced, for example, by a damaged gripper sliding mechanism, worn guides, contamination of the linear actuator or pneumatic system deterioration.

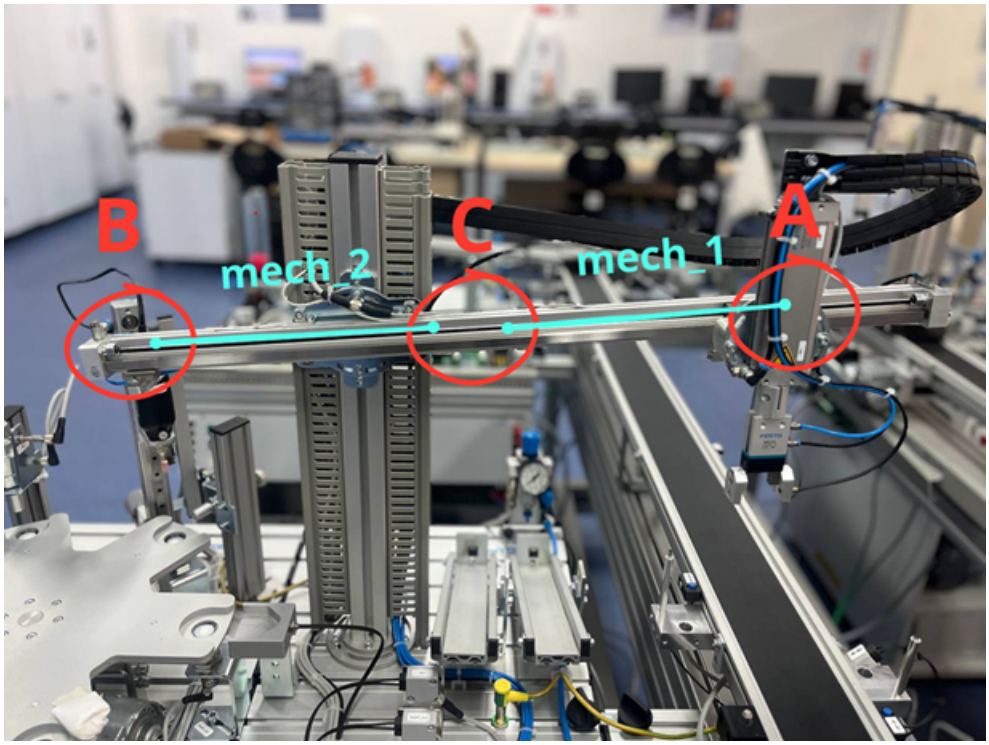
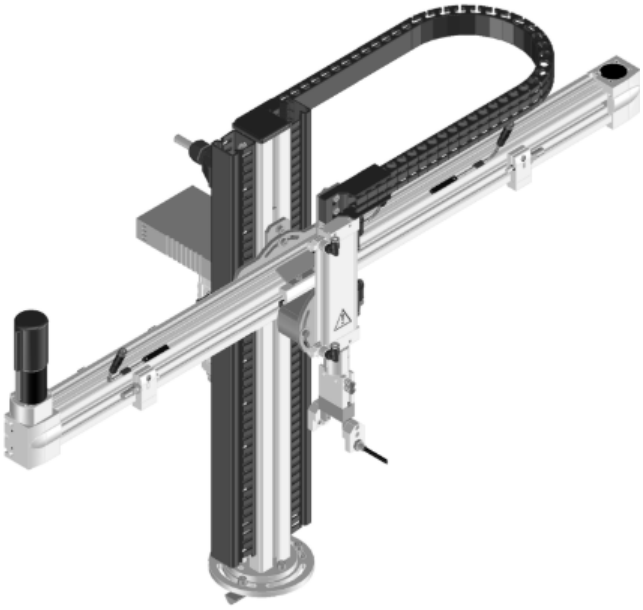


Figure 3-3: Pneumatic Linear Drive with labeled points

3.0.1 Use of Synthetic Data and Digital Twin Limitations

Creating synthetic data for training machine learning models would certainly offer notable advantages in strengthening the generalization and resilience of the model. Usually, digital twins—virtual clones of physical systems—are used to generate synthetic data, hence simulating several situations without compromising actual operations. However, developing a comprehensive digital twin model for the current industrial automation system (FESTO MPS 500) was beyond the scope of this thesis due to the complexity involved and the absence of readily available detailed system models for such simulation. Therefore, while acknowledging the value of synthetic data and digital twins for enhancing predictive capabilities, we suggest that establishing a detailed and validated digital twin should be considered as a distinct, future research direction.



(a) Pneumatic Linear Drive with Gripper

(b) Electrical proximity switch Type SME-8

Figure 3-4: a - Pneumatic Linear drive, b - proxomity sensor

3.1 System Architecture & Data Acquisition

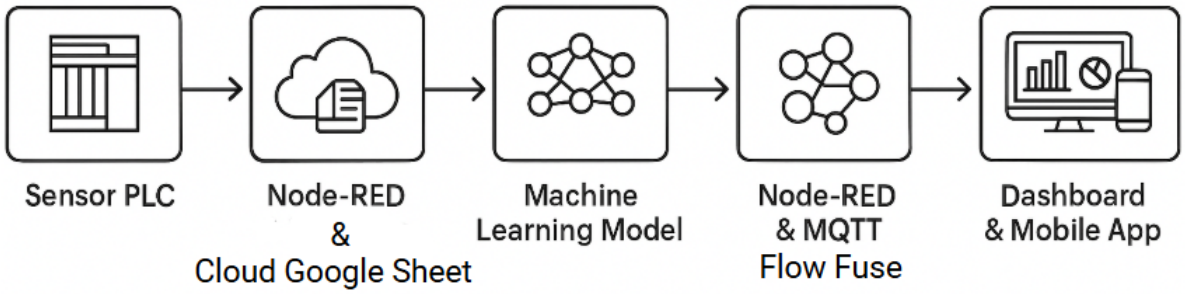


Figure 3-5: System Architecture for IoT-enabled Predictive Maintenance

The end-to-end architecture of the predictive maintenance solution is displayed in Figure 3-5. A Siemens S7-1200 controller first gathers raw signals from the sensors of a FESTO MPS 500 plant, which are then sent to a Node-RED instance running on the FlowFuse platform. In this case, the collected data is pre-filtered, time-stamped, and saved in a Google Sheets table on the cloud, which acts as a central location for the time series. After extracting these historical information, the machine learning model predicts the mechanical defect status and pneumatic pressure and returns the results in JSON format to Node-RED over MQTT. Lastly, the generated predictions are shown in real-time on the Remote-RED smartphone app and the web-based Dashboard, providing operators with immediate access to system status and facilitating prompt equipment maintenance choices.

Specifically for the Internet of Things (IoT) environment, Node-RED is an open-source, flow-based development tool for visual programming of event-driven applications [12]. Node-RED, a browser-based interface built on Node.js and developed by IBM, allows users to connect hardware, APIs, and internet services with a drag-and-drop editor [17]. This low-code platform is perfect for implementing and overseeing predictive maintenance procedures because it facilitates quick prototyping and smooth integration of sensor information with downstream data processing systems. Its usefulness in industrial automation and smart factory settings is further supported by its extensibility, compatibility for MQTT and HTTP protocols, and robust community

ecosystem.

As part of the implementation of the predictive maintenance system, the Node-RED platform was used in conjunction with a Siemens S7-1200 industrial controller. The connection was made through S7 in nodes(Fig. 3-7), which provided access to data from PLC in real time. The TIA Portal development environment was used to configure topics parameters and address signals, providing access to variables stored in the controller. The visual flow diagram Node-RED flow on the FlowFuse platform is shown in Figure 3-6.

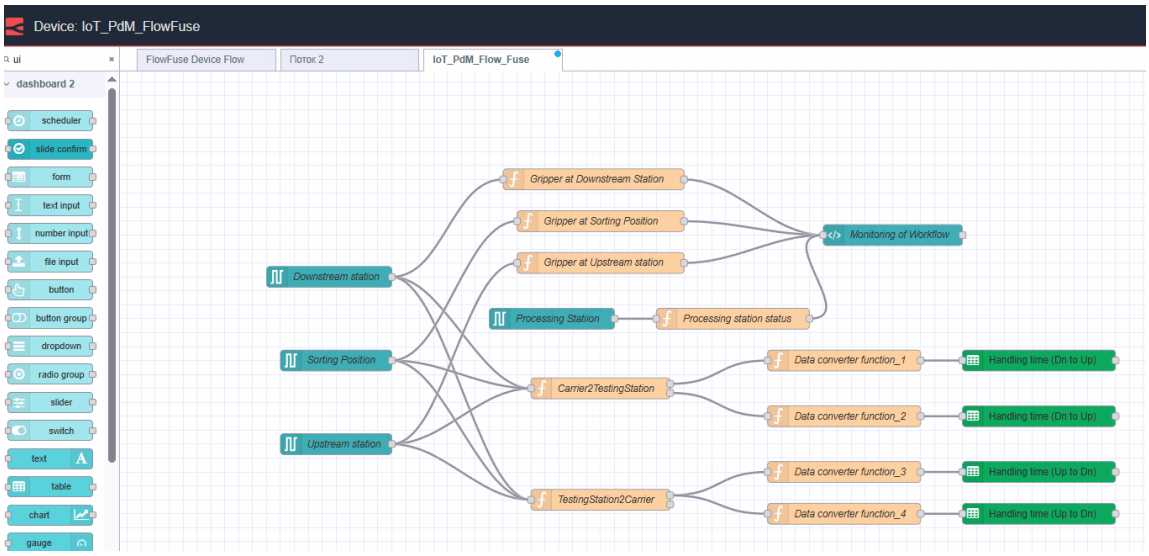


Figure 3-6: Node-RED flow on FlowFuse illustrating data acquisition and preprocessing.

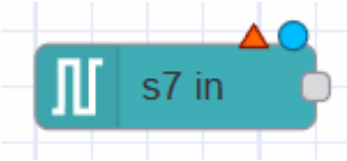


Figure 3-7: The S7 Input nodes Downstream station, Sorting Position and Upstream station read signals from their respective positions on the production line.

The received signals are transmitted to two functional nodes (Fig. 5-1) (Carrier2TestingStation and TestingStation2Carrier), where timestamp processing is performed. Each node implements logic to count the time intervals between key points of the gripper's movement. For example, the Carrier2TestingStation node calculates the difference between the timestamps of Downstream, Sorting, and Upstream stations

to determine how many milliseconds it took the gripper to move from point A to point B. The same logic is implemented in the Carrier2TestingStation node. Similar logic is implemented in the TestingStation2Carrier node, which captures movement in the reverse direction. In this way, a time series is formed, which is the basis for the subsequent construction of the training dataset of the machine learning model.

The Node-RED system was connected with Google Sheets via the GSheet node to facilitate data storage in a time series format (Fig. 3-9). The integration is facilitated via a service account established in the Google Cloud Console, where table access was configured and a corresponding JSON key was generated. Utilizing Google Sheets enables the efficient storage and organization of data from the controller, including computed time intervals, so facilitating later processing, analysis, and training of machine learning models. This approach facilitates the manipulation of cloud data and ensures compatibility with external visualization or processing tools.

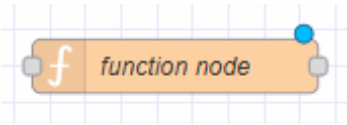


Figure 3-8: Function node allows to write custom JavaScript to transform, filter, or enrich messages and implement logic that built-in nodes alone cannot handle.

Figure 3-9: GSheet node integrates Node-RED with Google Sheets via the Google Sheets API, allowing flows to programmatically read from and write to cloud-based spreadsheets.

3.2 Dataset Description

To train and test the predicted maintenance model, data from a FESTO MPS 500 Handling Station was used to make a custom dataset. The data is supplied in four files: forward_AC.csv and forward_CB.csv (424 records), and backward_BC.csv and backward_CA.csv (424 records), representing the forward and backward movements of the gripper, respectively. Every object in the collection comprises three attributes. The Pressure column indicates the pressure level in the pneumatic system, a crucial indicator for evaluating operational conditions; under standard settings, the pressure is sustained at ±5 bar. The AC_time and CB_time variables in forward.csv, along with the CA_time and BC_time properties in backward.csv, represent the time intervals during which the gripper transitions between the three waypoints (sensors): Downstream Station (A), Sorting Position (C), and Upstream Station (B). The time intervals are measured in milliseconds and represent essential Time Features that indicate the system's dynamics.

The additional binary features mech_1 and mech_2 denote the presence or absence of mechanical faults on individual portions of the gripper motion: a value of 0 indicates no fault and 1 indicates the presence of a mechanical problem. These fields were used in the classification task to build a fault detection model. Pressure in the pneumatic system (bar), which was manually varied in four modes: 3.5 Bar and 4 Bar (leak simulation), 5 Bar (normal mode), 6 Bar (overpressure) The data structure allows analyzing deviations in time intervals under normal and abnormal conditions, as well as taking into account the influence of pressure on the motion pattern. Thus, the obtained dataset provides sufficient coverage of possible operation scenarios and allows solving both regression and binary classification problems.

Pressure (Bar)	**AC_time(ms)**	**mech_1(binary)**	**CB_time(ms)**	**mech_2(binary)**
3.5	3746	0	4022	0
3.5	3718	0	4186	1
3.5	4050	1	4166	0
3.5	3943	1	4106	1
4	3669	0	3918	0
4	3749	0	4072	1
4	4072	1	3813	0
4	4056	1	3887	1
5	3721	0	3759	0
5	3490	0	4124	1
5	3980	1	3606	0
5	3889	1	3979	1
6	1889	0	1956	0
6	1902	0	2166	1
6	2477	1	1740	0
6	2627	1	2118	1

Table 3.1: Dataset scenarios (Forward direction A to B)

The research's dataset consists of time-series sensor readings obtained from a conveyor-based automation system, specifically documenting the behavior of a gripper as it moves between three sensor positions: A, B, and C.The data is categorized into four logs of directional movement:

forward_AC.csv: Gripper movement from Sensor A to C

forward_CB.csv: Gripper movement from Sensor C to B

backward_CA.csv: Gripper movement from Sensor C to A

backward_BC.csv: Gripper movement from Sensor B to C

The division into four files was executed to enhance the precision of the machine learning model. The discrepancy arises from the differing durations spent in one segment while advancing the gripper compared to retracting it. For instance, AC_time is not equivalent to CA_time.

The content of the updated dataset looks like this in the example file "forward_AC.csv":

Pressure	**Time**	**Mechanism**
3.5	3718	0
3.5	4050	1
4	3670	0
4	4070	1
5	3620	0
5	3980	1
6	1900	0
6	2500	1

Table 3.2: The content of "forward_AC.csv"

Pressure: Values representing the air pressure (in bar) applied to the pneumatic gripper system during the movement cycle.

Time: Integer value indicating the duration of the motion segment in milliseconds (e.g., time taken to move from A to B

Mechanism: A binary fault indicator representing the presence (1) or absence (0) of a mechanical fault during that particular movement segment.

Chapter 4

Methodology

4.1 Feature selection

The features selected for training the model were Time and Pressure. The Time variable represents the duration of movement between two sensors (e.g., from A to C) in milliseconds. This metric captures the dynamic response of the mechanical system under operation. The Pressure variable indicates the pneumatic pressure in the system during the movement cycle, recorded in bar. These features were picked because they show the most obvious signs of speed loss or mechanical problems right away. Possible faults, like blocks, wear and tear, or misalignment, are often linked to a delay in the expected movement time or changes in pressure during movement.

4.2 Machine learning Model selection

4.2.1 Random Forest Classifier & Regression

Random Forest Classifier was chosen for the mechanical fault detection task (mech_1, mech_2).[13] This algorithm forms an ensemble of solving trees, each of which is trained on a random subsample of features and objects [5]. Consequently, the mistakes of single trees are averaged, therefore greatly lowering the danger of overtraining and enabling consistent operation even with noisy or related characteristics. Furthermore,

Random Forest lets you estimate feature importance, which helps to identify the most crucial elements (time intervals, pressure) for problem detection in the pneumatic system.

To train the model, the dataset was first split into training and testing subsets [10]. Specifically, 85% of the data was used for training the model, while the remaining 15% was reserved for testing its performance on unseen samples. This approach ensures that the model generalizes well and does not simply memorize the training data.

```
Data shapes:
Original data: 424 samples
Training set: 360 samples (84.9%)
Test set: 64 samples (15.1%)
```

Before training, the features were normalized using the StandardScaler from sklearn. preprocessing.

Normalization was necessary because the features operate on different scales—time is measured in milliseconds while pressure is in bar. Scaling improves the performance of the learning algorithm by ensuring that all features contribute equally to the model training process.

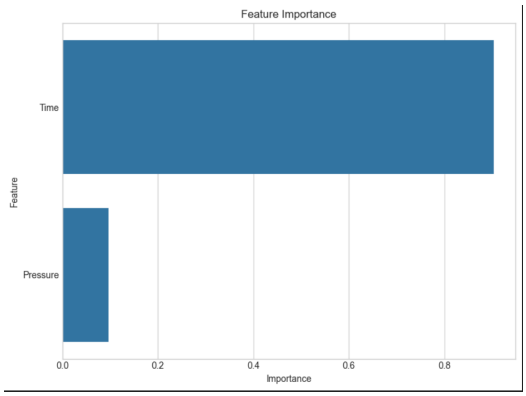
Choosing the best hyperparameters is one of the most important steps in making a machine learning model that works very well. Hyperparameters are model parameters that don't get learned directly during training but have a big effect on the quality in the end. Examples of hyperparameters are the number of trees in a random forest, tree depth, regularization factor, etc. The training process involved hyperparameter optimization using GridSearchCV with 5-fold cross-validation. The grid search explored combinations of hyperparameters such as the number of estimators (n_estimators), maximum tree depth (max_depth), and minimum samples required to split an internal node (min_samples_split). This method allows the model to be validated on different subsets of the training data to avoid overfitting and ensure stable performance. Cross-validation is especially important in predictive maintenance applications where false alarms or missed detections can have significant

operational consequences.

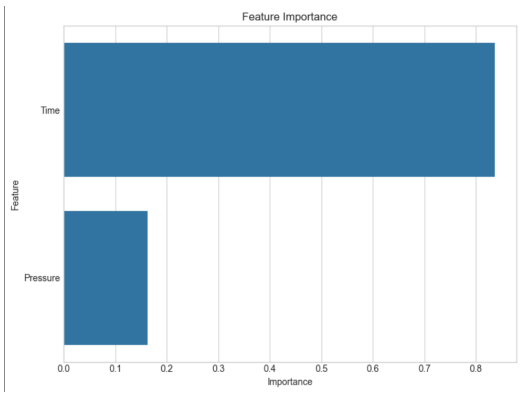
```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
def train_the_model(X_train_scaled, X_test_scaled, y_train,
y_test, scaler, name):
    param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
    }

    # GridSearchCV with 5-fold cross-validation
    grid_search = GridSearchCV(
        RandomForestClassifier(random_state=42),
        param_grid,
        cv=5,
        scoring='accuracy',
        n_jobs=-1
    )
```

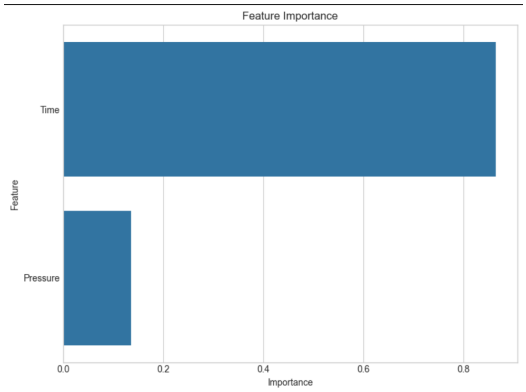
As shown in Figure. 4-1 "Time" was always the most important factor in all of the models. This shows that changes in the length of a movement are more strongly linked to mechanical problems than changes in pressure. In addition to fault classification, a regression model was developed to predict the pneumatic pressure values during the operation of the automated system. This predictive capability enables the system to anticipate potential pressure anomalies, which are often early indicators of performance degradation or failure in pneumatic mechanisms.



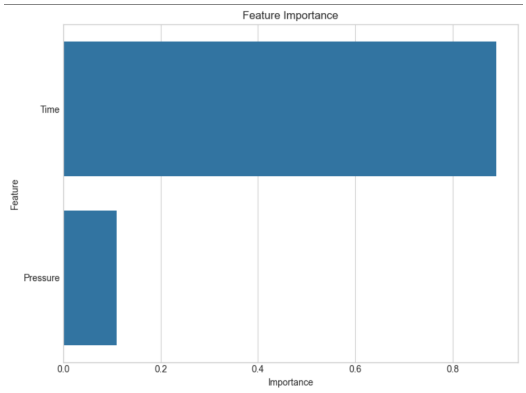
(a) forward_AC



(b) forward_CB



(c) backward_CA

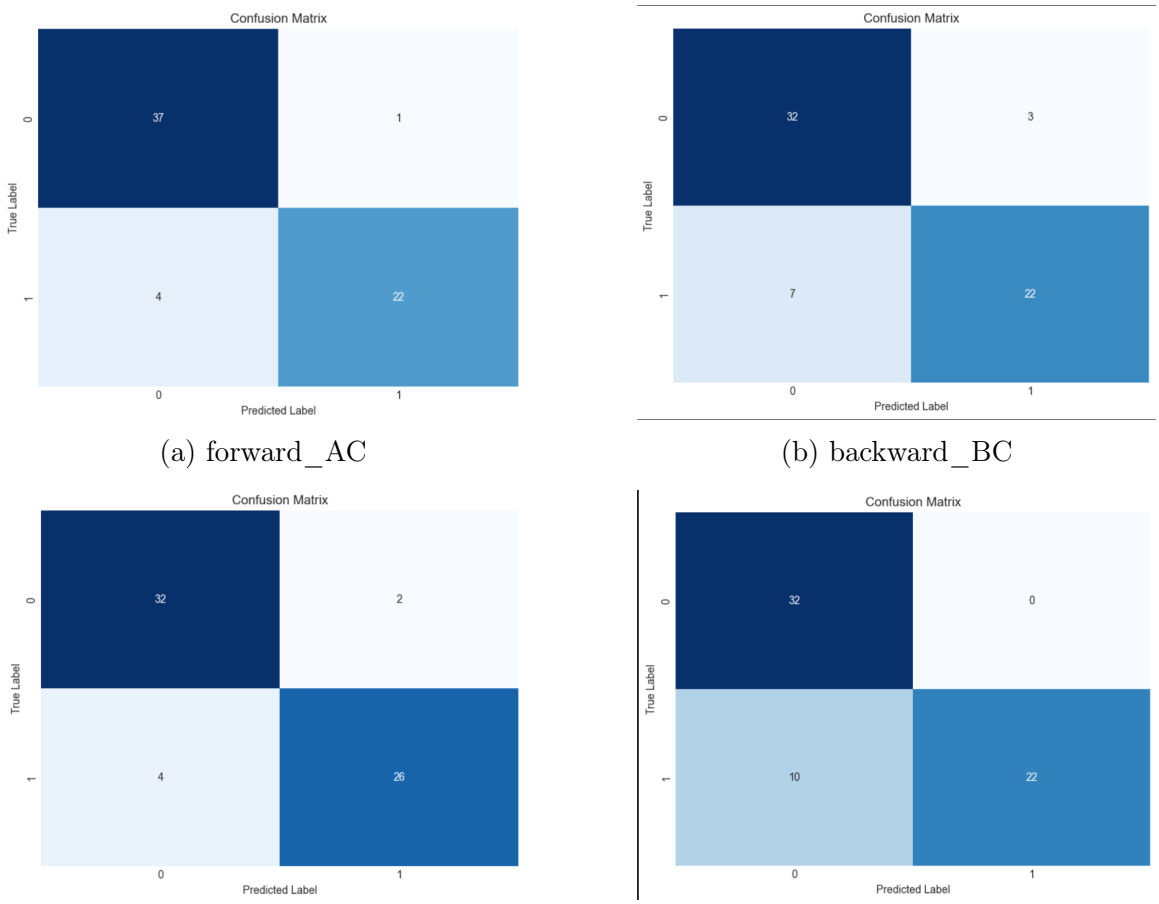


(d) backward_BC

Figure 4-1: Feature Importance Bar Chart illustrating the relative influence of each feature (AC_time, CB_time, CA_time, BC_time). Higher values indicate greater significance of the feature in predicting mechanical faults and pressure anomalies in the predictive maintenance system.

4.2.2 Evaluation Metrics

The model underwent evaluation through various metrics post-training. The Confusion Matrix (Fig. 4-2) serves to illustrate the quantities of true positives (accurately identified faults), true negatives (accurately identified normal operations), false positives (incorrect alarms), and false negatives (overlooked faults). This matrix provides insights into the model's accuracy and the balance of its predictions across the two classes.



(a) forward_AC

(b) backward_BC

(c) backward_CA

(d) forward_CB

Figure 4-2: Confusion Matrix, trained with parameters n_estimators=100, max_depth=None, min_samples_split=10

forward_AC The model showed the best balance with only 1 false positive and 4 false negatives out of 64 checks. The high accuracy and completeness make this set a benchmark for evaluation.

forward_CB. There are completely no false positives (FP = 0), however the model missed 9 failures (FN = 9). This reduces the completeness of fault detection and requires a threshold adjustment or class weighting.

backward_CA. Both 2 false positives and 9 missed failures were recorded. This heterogeneous error indicates that the backward data is more "noisy" or requires feature refinement.

backward_BC. The model performed fairly balanced with 3 false positives and 4 false negatives. The performance is close to ideal, although the small number of FPs

	precision	recall	f1-score	support
0	0.90	0.97	0.94	38
1	0.96	0.85	0.90	26
accuracy			0.92	64
macro avg	0.93	0.91	0.92	64
weighted avg	0.92	0.92	0.92	64

Figure 4-3: Classification report of forward_AC

indicates possible oversensitivity.

forward_AC (Fig. 4-3). For this dataset, the model showed high robustness to overfitting and excellent generalizability: the cross-validation accuracy was 94.72 % and 92.19 % on the delayed test sample. The gap of 2.53 % indicates a good stabilization of the results when n_estimators=100 and min_samples_split=10, without limiting the depth of the trees (max_depth=None).

forward_CB (Fig. 4-4). On the forward C→B data, the accuracy on cross-validation was 90.56%, whereas on the test sample it dropped to 84.38%, reflecting a gap of 6.18%. The model with 50 trees and the same partitioning threshold (min_sample_split=10) does not sufficiently adapt to the more variable temporal profiles of this site. The complete absence of false positives (FP=0) indicates high specificity, but the significant number of false negatives (FN=9) requires increased regularization and possibly more trees.

backward_CA (Fig. 4-5). For backward C→A movement, the algorithm achieved an outstanding 95.51% on cross-validation, but demonstrated only 84.38% on the test - a gap of more than 11%. Symptom of overtraining

backward_BC (Fig. 4-6). The model on backward B→C showed balanced quality: cross-validation - 92.99 %, test - 90.62% (gap 2.37%) The small number of false positive (FP=3) and false negative (FN=4) errors indicates that the model copes well with the noise and variability of the data at this site.

Clear explanation of each metric in the table Precision: The proportion of correctly identified positive cases among all predicted positives.

Recall: The proportion of correctly identified positive cases out of all actual pos-

	precision	recall	f1-score	support
0	0.76	1.00	0.86	32
1	1.00	0.69	0.81	32
accuracy			0.84	64
macro avg	0.88	0.84	0.84	64
weighted avg	0.88	0.84	0.84	64

Figure 4-4: Classification report of forward_CB

itives.

F1-score: A balanced harmonic mean of precision and recall, suitable for evaluating model performance, especially with imbalanced data.

Support: Number of actual occurrences of each class in the dataset.

Accuracy: Overall proportion of correctly classified instances across all classes.

Macro avg: Average of precision, recall, and F1-score calculated separately for each class, and then averaged equally. Useful for balanced insights across classes.

Weighted avg: Similar to macro average but weighted by the number of instances (support) in each class, providing a class-size-sensitive evaluation.

	precision	recall	f1-score	support
0	0.76	1.00	0.86	32
1	1.00	0.69	0.81	32
accuracy			0.84	64
macro avg	0.88	0.84	0.84	64
weighted avg	0.88	0.84	0.84	64

Figure 4-5: Classification report of backward_CA

	precision	recall	f1-score	support
0	0.89	0.94	0.91	34
1	0.93	0.87	0.90	30
accuracy			0.91	64
macro avg	0.91	0.90	0.91	64
weighted avg	0.91	0.91	0.91	64

Figure 4-6: Classification report of backward_BC

On the forward_CB dataset, the classifier achieved an accuracy of 85.94%. The model performed exceptionally well on normal samples (class 0) with a recall of 1.00, while fault samples (class 1) had a lower recall of 0.72. Despite this, the model maintained a balanced macro F1-score of 0.86, showing a solid tradeoff between sensitivity and specificity.

The regressor had a test R^2 score of 0.5193 and MSE of 0.1202, which, while lower than forward_AC, still indicates moderate predictive capability for pressure estimation

Table 4.1: Performance Overview Across All scenarios

Dataset	CV Accuracy	Test Accuracy	n_estimators		min_samples_split
forward_AC	94.72%	92.19%	100	—	10
forward_CB	90.56%	84.38%	50	—	10
backward_CA	95.51%	84.38%	50	—	5
backward_BC	92.99%	90.62%	100	—	5

4.2.3 Regression Model for Pressure values

Random Forest Regressor was used for the continuous pressure prediction task. Hyperparameters were selected using GridSearchCV and 5-fold cross validation on the training sample. The grid was permuted with:

n_estimators (number of trees) = [50, 100, 200]

max_depth (maximum depth) = [None, 10, 20]

min_sample_split (minimum points for splitting) = [2, 5, 10]

For each of the four directions of motion (forward_AC, forward_CB, backward_CA, backward_BC), different optimal parameters were found to provide the best combination of accuracy and generalization ability.

Analysis of the results

forward_AC: The best regression in quality — the model explains about 76% of the pressure variance and allows for a very low MSE (0.0565), which indicates adequate and stable predictions.

forward_CB: a decrease in R^2 to 0.52 and an increase in MSE to 0.12 indicates more noisy or less informative features in this area; the model still provides practical benefits, but requires refinement of the feature space.

backward_CA: despite the high CV R^2 (0.81), the quality drops to R^2=0.49 on the test, a sign of overfitting; it is worth introducing a max_depth limit or increasing min_samples_split.

backward_BC: a balanced result (R^2 =0.65) with a moderate MSE (0.0865) shows that the model successfully captures the dependencies in the opposite direction of this section.

Table 4.2: Results of Random Forest Regressor model

Dataset	Test MSE	Test R^2
forward_AC	0.1732	0.8168
forward_CB	0.2035	0.7848
backward_CA	0.0772	0.9205
backward_BC	0.2170	0.7766

4.2.4 Gradient Boost Regression and Classifier

After testing the system with Random Forest models for both classification and regression, more tests were done with Gradient Boosting Classifier (GBC) and Gradient Boosting Regressor (GBR) to see how well they worked and how well they could generalize. Gradient Boosting was chosen as a strong option because it works well with structured data and can handle overfitting as long as it is tuned correctly. It is known for reducing bias through sequential learning.

The Gradient Boosting models were used on the same four datasets as the Random Forest models: forward_AC, forward_CB, backward_CA, and backward_BC. GridSearchCV was used to find the best hyperparameters, and cross-validation scores, test accuracy (for GBC), and regression error metrics (for GBR) were used to measure performance.

Table 4.3: Classification report for the `forward_AC` dataset

	Precision	Recall	F1-score	Support
0	0.90	0.95	0.92	38
1	0.92	0.85	0.88	26
Accuracy			0.91	64
Macro average	0.91	0.90	0.90	64
Weighted average	0.91	0.91	0.91	64

For the forward_AC dataset, the Gradient Boosting Classifier demonstrated balanced predictive performance across both classes. The model achieved a precision of 0.92 and a recall of 0.85 for fault detection (class 1), while maintaining a F1-score of 0.91 overall. The confusion matrix confirmed reliable detection of faults with

Table 4.4: Regression metrics for the `forward_AC` dataset

Metric	Value
Best Regressor Params	learning_rate=0.05, max_depth=3, n_estimators=100
CV R^2	0.8337
Test MSE	0.0565
Test R^2	0.7657

only a few misclassifications. The model's test accuracy reached 91%, with macro and weighted averages also maintaining high alignment, indicating consistency across class imbalances. The corresponding regression model (GBR) for pressure prediction achieved a cross-validation R^2 score of 0.8337, and a test R^2 of 0.7657, indicating strong generalization. The Mean Squared Error was low at 0.0565, reflecting close agreement between actual and predicted pressure values

Table 4.5: Classification report for the `forward_CB` dataset

	Precision	Recall	F1-score	Support
0	0.78	1.00	0.88	32
1	1.00	0.72	0.84	26
Accuracy			0.91	64
Macro average	0.91	0.90	0.90	64
Weighted average	0.91	0.91	0.91	64

Table 4.6: Regression metrics for the `forward_CB` dataset

Metric	Value
Best Regressor Params	learning_rate=0.05, max_depth=3, n_estimators=100
CV R^2	0.6395
Test MSE	0.1202
Test R^2	0.5193

On the forward_CB dataset, the classifier achieved an accuracy of 85.94%. The model performed exceptionally well on normal samples (class 0) with a recall of 1.00, while fault samples (class 1) had a lower recall of 0.72. Despite this, the model maintained a balanced macro F1-score of 0.86, showing a solid tradeoff between sensitivity and specificity.

The regressor had a test R^2 score of 0.5193 and MSE of 0.1202, which, while lower than forward_AC, still indicates moderate predictive capability for pressure estimation

Table 4.7: Classification report for the `backward_CA` dataset

	Precision	Recall	F1-score	Support
0	0.79	0.94	0.86	35
1	0.91	0.69	0.78	29
Accuracy			0.83	64
Macro average	0.85	0.82	0.82	64
Weighted average	0.84	0.83	0.82	64

Table 4.8: Regression metrics for the `backward_CA` dataset

Metric	Value
Best Regressor Params	learning_rate=0.05, max_depth=3, n_estimators=100
CV R^2	0.6395
Test MSE	0.1202
Test R^2	0.5193

For the backward_CA dataset, GBC achieved a cross-validation accuracy of 95.51%, though test accuracy decreased to 82.81%. This drop may suggest slight overfitting. The classifier achieved a precision of 0.91 for fault detection, with a recall of 0.69, leading to a weighted average F1-score of 0.82.

GBR got a test R^2 of 0.4895 and an MSE of 0.1265 on the regression job. These numbers are a little lower, which means that the pressure predictions are less accurate for data that hasn't been seen yet.

The backward_BC dataset produced consistently strong results. The classifier achieved 89.06% test accuracy with balanced precision and recall (approximately 0.89 for both classes). This led to an F1-score of 0.89 across all evaluation metrics.

The GBR model performed well with a test R^2 of 0.6526 and a MSE of 0.0865, suggesting the model is capable of capturing meaningful patterns in pressure fluctuation.

Table 4.9: Classification report for the `backward_BC` dataset

	Precision	Recall	F1-score	Support
0	0.89	0.91	0.90	34
1	0.90	0.87	0.88	30
Accuracy			0.89	64
Macro average	0.89	0.89	0.89	64
Weighted average	0.89	0.89	0.89	64

Table 4.10: Regression metrics for the `backward_BC` dataset

Metric	Value
Best Regressor Params	learning_rate=0.05, max_depth=3, n_estimators=100
CV R^2	0.5826
Test MSE	0.0865
Test R^2	0.6526

Best Regressor Params: Displays the optimal hyperparameters found through cross-validation for the regression model (Gradient Boosting Regressor).

CV R^2 (Cross-Validation R^2): Average coefficient of determination from cross-validation, representing the proportion of variance explained by the model across multiple training subsets.

Test MSE (Mean Squared Error): Indicates average squared differences between predicted and actual values. Lower values indicate better predictive accuracy.

Test R^2: Represents the coefficient of determination calculated on the test dataset, showing how well future predictions might align with actual observations.

Table 4.11: Performance Overview Across 4 Datasets: Gradient Boosting Classifier

Dataset	Max Depth	#Estimators	CV Accuracy	Test Accuracy
forward_AC	3	100	94.17%	90.62%
forward_CB	3	100	90.00%	85.94%
backward_CA	3	100	95.51%	82.81%
backward_BC	3	100	92.71%	89.06%

Table 4.12: Performance Overview Across 4 Datasets: Gradient Boosting Regressor

Dataset	Max Depth	#Estimators	CV R^2	Test MSE	Test R^2
forward_AC	3	100	0.8337	0.0565	0.7657
forward_CB	3	100	0.6395	0.1202	0.5193
backward_CA	3	100	0.8138	0.1265	0.4895
backward_BC	3	100	0.5826	0.0865	0.6526

4.3 Comparative Evaluation: Random Forest vs. Gradient Boosting

Table 4.13: Random Forest vs. Gradient Boosting

Dataset	Random Forest (RFC)		Gradient Boosting (GBC)	
	CV Accuracy	Test Accuracy	CV Accuracy	Test Accuracy
forward_AC	94.72%	92.19%	94.17%	90.62%
forward_CB	90.56%	84.38%	90.00%	85.94%
backward_CA	95.51%	84.38%	95.51%	82.81%
backward_BC	92.99%	90.62%	92.71%	89.06%

Based on the experimental results obtained from four real-world datasets, both Random Forest (RFR/RFC) and Gradient Boosting (GBR/GBC) models demonstrated strong performance in the context of predictive maintenance. But a close look at the two shows that each method has clear benefits in certain situations. Overall, the Random Forest models were more stable, easier to tune, and worked better on both classification and regression tasks. They did especially well in fault assessment, which needs to be very sensitive (recall). As an example, RFC models sent:

- High test accuracies (up to 92.2%) with balanced precision and recall

- Stable F1-scores across both fault and non-fault classes
- Superior pressure prediction accuracy with R^2 scores exceeding 0.81 in some datasets

In contrast, the Gradient Boosting models (GBR/GBC) were more sensitive to hyperparameter tuning but still achieved competitive results. They often reached higher cross-validation scores and provided more granular control over model performance through the learning rate. Notably, GBC models showed:

- Strong precision, especially in detecting fault cases (up to 0.92 in forward_AC)
- High test accuracy (e.g., 91% on forward_AC)
- Pressure predictions that were reasonably accurate (R^2 up to 0.76), though with slightly more variance than RFR

Even with these benefits, Gradient Boosting models sometimes overfitted, resulting in high cross-validation accuracy but lower performance on test sets (for example, the backward_CA dataset). This means that boosting might work better in perfect situations, but it needs to be carefully tuned and tested to keep its generalizability.

In conclusion, while both Random Forest and Gradient Boosting models demonstrated strong capabilities for predictive maintenance, the Random Forest approach emerged as the more practical and reliable choice for this specific application. Its performance was consistently stable across all datasets, showing high accuracy in fault classification and superior generalization in pressure regression. The Random Forest Classifier achieved a balanced tradeoff between precision and recall, which is particularly important in maintenance scenarios where missing a fault can lead to operational failures. Additionally, the Random Forest Regressor produced higher R^2 scores and lower mean squared errors, indicating better alignment between predicted and actual pressure values.

Chapter 5

Integration & Visualization

5.1 Publishing Predictions via MQTT

5.1.1 MQTT Node

The system sends real-time predictions to Node-RED using Message Queuing Telemetry Transport (MQTT) protocol after getting the results from the machine learning model. A lightweight network message protocol geared toward high latency and bandwidth-limited settings, MQTT Active use is in Internet of Things (IoT) systems, which need quick and dependable transmission of tiny messages between devices and a server.

The data transmission is done through an MQTT Broker, which is a server responsible for routing messages between the sender (publisher) and the receiver (subscriber). For this project, the public broker test.mosquitto.org on port 1883 was used. Node-RED acts as a subscriber to a specific topic (mps512/feedback0202) where it receives messages from the model. The model, implemented in Python, generates predictions in the form of a dict dictionary containing values on the following keys:

```
{
  "mech_1": 0,
  "mech_2": 0,
  "pressure_prediction": 5
}
```

Here mech_1 and mech_2 take values of 0 or 1 and indicate the presence or absence of a mechanical fault in the respective sections, and pressure_prediction displays the predicted pressure in the system. Before being sent, this dictionary is serialized into JSON (JavaScript Object Notation) format, which is a common way of structured data transfer between applications.

Transmission is implemented via the paho-mqtt library as follows:

```
mqtt_client.publish(MQTT_TOPIC, payload)
```

The data is then instantly available in Node-RED via the mqtt in-node, where it can be visualized on the Dashboard, logged, or used in decision logic.

Figure 5-1: MQTT node - the configuration node for connecting to the MQTT Broker

5.1.2 FlowFuse & Dashboard

After the machine learning model generates its prediction, the results are sent to the Node-RED FlowFuse platform via the MQTT protocol and the corresponding MQTT in (MQTT_CONNECTION) node. The received data is in JSON format, from where it is sent to several function nodes for processing and conversion

The Pressure Output, Mech_1 Output, and Mech_2 Output from the ML Model

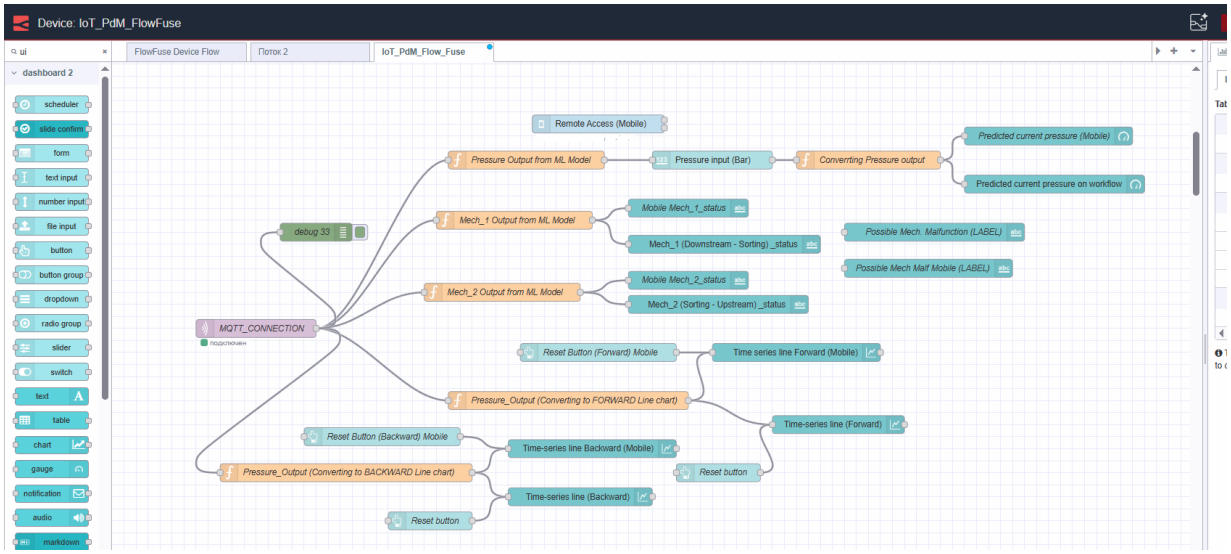


Figure 5-2: Node-RED Flow in communication with Dashboard

nodes extract and format the relevant parts of the JSON response. The data is then sent to the UI nodes, which visualize it in the Dashboard.(Fig. 5-2)

Thus, FlowFuse and Node-RED make it easy to organize and see predictive analytics data from a machine learning model. This allows to promptly react to changes and make decisions on preventive maintenance of equipment in real time.

To display the current pressure level in the system, a gauge node is used - a pressure gauge (Pressure managing), which visualizes the pressure in bars. The data is first passed through a numeric input node (Pressure input Bar) as well as an additional functional node (Converting Pressure output) that corrects or scales the values before displaying them. To indicate mechanical faults, a text node (Possible Mechanical Malfunction LABEL) is used to receive the states mech_1 and mech_2 from the function nodes. These nodes display the fault status in real time as simple text messages: “Yes“/”No”.

Description of the Dashboard(Figure. 5-3)

The Dashboard, implemented on the Node-RED FlowFuse platform, is the main user interface for controlling, monitoring and analyzing the Handling Station status of the FESTO MPS 500 system. The upper left part of the panel (Control Unit) contains the control elements in the form of station control buttons (RESET, START,

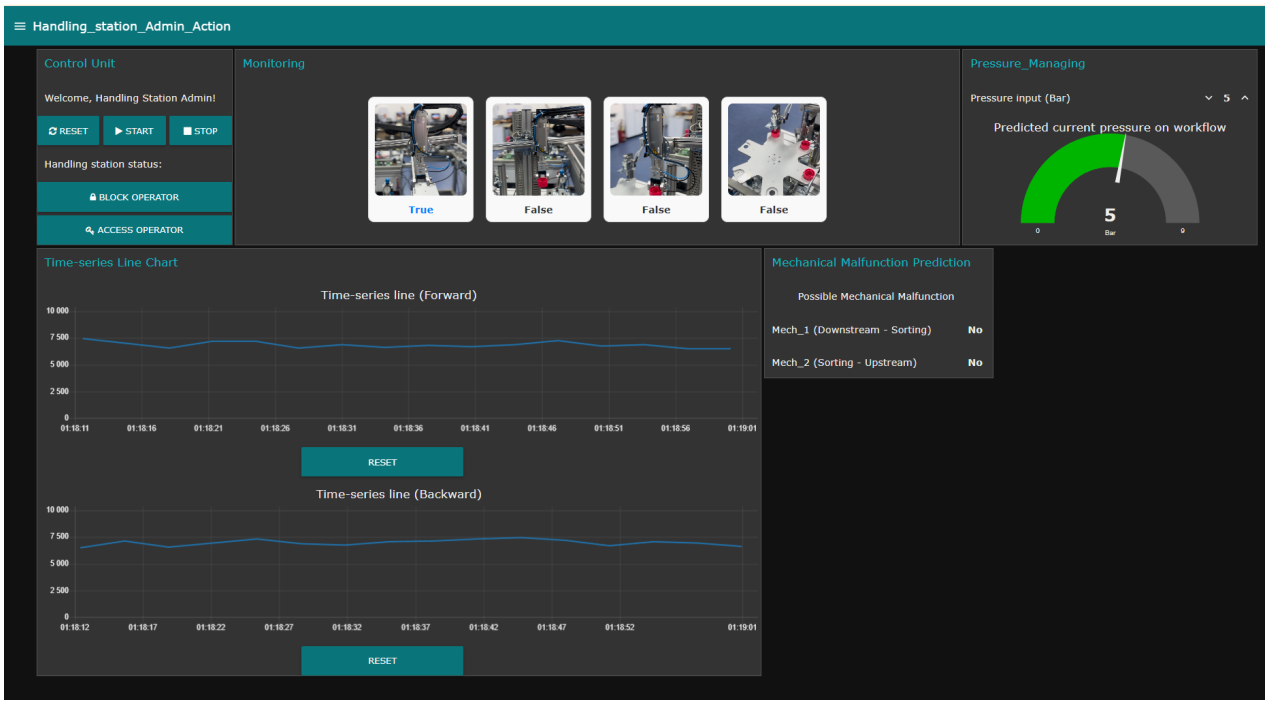


Figure 5-3: Dashboard interface

STOP), as well as additional BLOCK OPERATOR and ACCESS OPERATOR buttons for restricting and granting operator access. The center top portion of the panel (Monitoring) demonstrates the status of station operation through a series of sensor and camera images showing the current state of movement of the gripper and parts in the station. Below each image is a status (True/False) indicating the current status of the corresponding stage.

In the upper right part (Pressure Managing) in (Fig. 5-3) there is a gauge widget that visualizes the current air pressure in the system. It also displays the numerical value of pressure in bars, which allows you to quickly track its deviation from normal values (5 bar) and timely identify possible malfunctions.

In the lower part of the Dashboard there are two Time-series Line Charts, which display in real time the time intervals spent by the gripper to move between the sensors in Forward and Backward directions. Both graphs have separate RESET buttons that allow the operator to clear the history and restart data collection. To the right of the graphs is a block (Mechanical Malfunction Prediction) showing the status of possible mechanical faults on the Mech_1 (Downstream - Sorting) and Mech_2 (Sorting -

Upstream) sections. It also contains "Yes/No" text labels that clearly indicate the current status of the sections, which greatly facilitates quick diagnosis of the system status

Note: The Dashboard that has been presented is a prototype that is designed to demonstrate the monitoring system's capabilities and data visualization. The data displayed on the Dashboard is artificially generated and is intended to illustrate the interface's functionality.

5.1.3 Mobile Dashboard

Remote access (Fig. 5-4) allows you to connect to your dashboard in Node-RED from your mobile device via the Remote-RED app available for iOS and Android platforms.

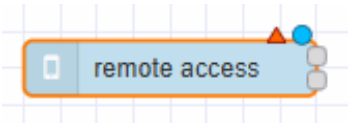


Figure 5-4: Remote access node to connect Node-RED with Mobile via Remote RED Application

This feature is implemented through a special remote-access configuration node where you set the connection parameters: protocol (usually HTTP), port number (1880), base URL (/ui) and server location (e.g. Europe, Germany). Once the parameters are properly configured, Node-RED generates an unique QR code. Once scanned, the mobile device will automatically contact the Node-RED server.

The application interface replicates key elements of the main Dashboard on PC and is adapted for convenient use on mobile devices. (Fig. 5-5)

This provides operational control of the predictive maintenance system, allowing remote monitoring of the pneumatic actuator, current pressure levels and mechanical failures

At the top of the mobile Dashboard is a Pressure Managing widget, similar to the main Dashboard, showing the current predicted air pressure in the pneumatic system. The pressure value is clearly presented in the form of a pressure gauge scale, which allows you to quickly assess the current situation and identify possible deviations from the standard value (5 bar).

Below is a block (Possible Mechanical Malfunction Status), which provides information on the presence or absence of mechanical faults in the sections:

Mech_1 (Downstream – Sorting)

Mech_2 (Sorting – Upstream)

Status is displayed by simple text labels “Yes” or “No”, providing convenient and quick diagnostics of the mechanical status of the system directly from the mobile device.

At the bottom of the interface there are two line graphs (Time series line (Forward) and Time-series line (Backward)) showing real-time dynamics of time intervals of gripper movement between sensors in both directions. A RESET button is available below each graph, allowing you to reset the data and restart monitoring if necessary

Thus, the mobile Dashboard fully duplicates the most important functions of the main interface, allowing the operator to quickly monitor the status of the equipment regardless of its physical location, which significantly increases the ease.

(a) Mobile Dashboard 1st screen

(b) Mobile Dashboard 2nd screen

Figure 5-5: Mobile Dashboard interface

Chapter 6

Future Directions

6.1 Technological advancements

To make predictions even more accurate, the first thing that needs to be done is to **increase the size and range of the dataset**. The model has only been trained on a few cycles of gripper movement at a set pressure, so it can't yet think about all the different ways it could work or the worst things that could happen. In the next study, it would be smart to collect more information about a lot of different pressures (for example, from 3 to 7 bar), response times, and the mechanism's level of wear. To do this, real readings can be taken over a long period of time, and specific tests can be done within factory limits, such as testing at low and high pressures and creating leaks or wedges on purpose.

In addition, it's worth "including additional scenarios" in the dataset that show normal and unusual situations. For example, starting with an empty tray, capturing parts that aren't the normal size or weight, alternating between forward and backward cycles without breaks, and going over the same route more than once to show how wear builds up over time are all possibilities. With this much flexibility, the model can be taught to react more strongly to rare but important events, lowering the number of false positives and real faults that are missed.

When it comes to the Dashboard interface, most of the indicators are set artificially today (they are static numbers made up to show how they work). For real work, you

will need to set up the Dashboard to send pressure, mech_1, mech_2 states, and time series data that comes from the model. It is important to make sure that Node-RED's 'mqtt out' nodes post data quickly and without any errors. On the Dashboard side ('mqtt in', 'chart', 'gauge'), JSON deserialization, timestamp binding, and automatic graph updates happen. So that the panel shows a correct picture of the system's state, it is also important to skip processing (when messages haven't arrived) and smooth out "jumps" (using moving average or filtering).

Lastly, signs of the "health" of communication and "monitoring the integrity of the transmission channel" should be considered for operational simplicity and dependability. For instance, the MQTT configuration node can communicate with the Dashboard using heartbeat messages. In the event that many alarms go unnoticed in a row, it will activate emergency mode, which includes a red indicator and the storage of the most recent accurate data. By taking this route, we can build a control panel that is more suited to production than demo, and we can be sure that issues with the model and the visualization channel will be identified quickly.

6.2 Deploy models on Edge devices

Moving trained models to Edge devices like the Raspberry Pi, the NVIDIA Jetson Nano, or specialty microcontrollers that support TinyML (ESP32) can cut down on prediction delays and network load by a large amount. The gadget doesn't send the "raw" data to the cloud; instead, it does inference locally and sends only the results, like alarms. This means that the system can still work even if it can't talk to the central computer. In Banbury et al., examples of real-world applications are given.Wang 2024Edge and Bai 2018 Empirical are two sources.[19], [4].

6.3 Predicting the remaining useful life (RUL)

Unlike the binary classification of "serviceable/broken", the Remaining Useful Life (RUL) prediction allows you to estimate how many cycles or time are left before

equipment failure. This approach provides more accurate recommendations for routine maintenance and minimizes unplanned downtime. Regression methods (Weibull models, gradient boosting) or recurrent neural networks (LSTM) are usually used for RUL.[3]

6.4 Deep neural networks for time series analysis

Artificial neural networks trained on time series data To represent long-term relationships and uncover subtle nonlinear patterns in the dynamics of time intervals and pressure, we can use recurrent neural networks (LST, GRU) and temporal convolutional networks (TCN). They are pre-trained to represent characteristics in "raw" sequences, unlike decision trees[7].

Chapter 7

Conslusion

This study proposes and implements an integrated IoT-enabled predictive maintenance system for the Handling Station module on the FESTO MPS 500 platform. The acquisition of data from the Siemens S7-1200 controller via S7 nodes in Node-RED (FlowFuse) and its subsequent storage in Google Sheets facilitated the creation of a comprehensive time series of variables (AC_time, CB_time, CA_time, BC_time, Pressure, mech_1, mech_2). The ensemble models of Random Forest exhibited remarkable efficiency: the classifier attained an accuracy of 92.2% and an F1 score of almost 0.90 in identifying mechanical failures, while the regressor accounted for over 76% of the pressure variance ($R^2 = 0.7657$) with a minimal MSE. An alternative verification via Gradient Boosting validated the robustness of the ensemble methods.

An affordable and easily deployed platform for real-time monitoring has been made available through the use of MQTT prediction transmission and visualization in the FlowFuse Dashboard and the Remote-RED mobile app. Further, we checked that the panel shows real model results and not just demo data.

Current scenarios only cover a restricted pressure range (approximately 5 bar) and intentionally produced failures are the main limits of the dataset. Another issue is the variety and amount of the dataset. Extensive data collection is necessary at various pressures (3-7 bar), loads, and wear conditions to improve accuracy and reliability.

Four directions are highlighted as prospects:

- Assembling and adding to a dataset means gathering real work cycles that happen in a lot of different conditions and settings.
- Remaining Resource Prediction (RUL): Moving from binary classification to time-to-failure estimation to optimize maintenance schedules.
- Using deep networks for time series (LSTM, GRU, TCN): identification of complex nonlinear and long-term dependencies.
- Deploying models on Edge hardware (Raspberry Pi, Jetson, TinyML): local inference with minimal delays and autonomy in case of loss of communication.

The execution of these measures will establish a genuinely industrial solution that minimizes downtime, optimizes maintenance expenses, and delivers timely, precise predictions of equipment status.

Bibliography

[1] F. M. Abdelillah, H. Nora, S. Ouchani, and S. M. Benslimane. Predictive maintenance approaches in industry 4.0: A systematic literature review. *IEEE Transactions on Industrial Informatics*, 18(1):15–25, 2024.

[2] R. Ahmad and S. Kamaruddin. Data-driven predictive maintenance: A comprehensive review on ml algorithms and data quality issues. *Reliability Engineering & System Safety*, 219, 2022.

[3] S. O. Alhuqayl, A. T. Alenazi, H. A. Alabduljabbar, and M. A. Haq. Improving predictive maintenance in industrial environments via IIoT and machine learning. *International Journal of Advanced Computer Science and Applications*, 15(4):627–640, 2024.

[4] Shaojie Bai, Zico Kolter J. and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. In *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[5] C. Cutler, D. Cutler, and J. R. Stein. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

[6] R. Doe and J. Smith. IoT-Based Predictive Maintenance for Electrical Machines. *International Research Journal of Modern Engineering and Technology*, 5(12):50–58, December 2024.

[7] Md Hassan et al. A deep learning framework for time series forecasting. *Procedia Computer Science*, 147:672–677, 2019.

[8] R. Kanagavelu, K. Jayaraman, R. Vengatesan, B. R. Sridhar, and R. Buyya. Federated learning for advanced manufacturing based on industrial iot data analytics. In Sunil Sharma and Pethuru Raj, editors, *Implementing Industry 4.0*, pages 143–162. Springer, 2021.

[9] A. Lazzaro, D. M. D'Addona, and M. Merenda. A detailed study on algorithms for predictive maintenance in smart manufacturing: Chip form classification using edge machine learning. *IEEE Open Journal of Industrial Electronics Society*, 5:1190–1205, 2024.

[10] B. Lu, D. B. Durocher, and P. Stemper. Predictive maintenance techniques. *IEEE Industry Applications Magazine*, 15(6):–, 2019.

[11] J. N. A. Malaiyappan, G. Krishnamoorthy, and S. Jangoan. Predictive maintenance using machine learning in industrial iot. *International Journal of Innovative Science and Research Technology (IJISRT)*, pages 1909–1915, 2024.

[12] J. Paleyes, K. Brecht, A. López, and et al. An empirical methodology for evaluating flow-based programming in iot systems. In *Proc. 6th International Conference on IoT Architectures*, 2022.

[13] Sasmita Pani, Omkar Pattnaik, and Binod Kumar Pattanayak. Predictive maintenance in industrial iot using machine learning approach. *International Journal of Intelligent Systems and Applications in Engineering*, 12(14s):521–534, 2024.

[14] C. Resende, D. Folgado, J. Oliveira, B. Franco, W. Moreira, A. Oliveira-Jr., A. Cavaleiro, and R. Carvalho. Tip4.0: Industrial internet of things platform for predictive maintenance. *Sensors*, 21(14):4676, 2021.

[15] O. Serradilla, E. Zugasti, and U. Zurutuza. Deep learning models for predictive maintenance: A survey, comparison, challenges, and prospect. *ACM Transactions on Cyber-Physical Systems*, 5(1):1–34, 2020.

[16] A. M. Al Shahrani, M. A. Alomar, K. N. Alqahtani, M. S. Basingab, B. Sharma, and A. Rizwan. Machine learning-enabled smart industrial automation systems using internet of things. *Sensors*, 23(1):324, 2023.

[17] M. Smith, A. Kumar, and R. Patel. Design and implementation of a node-red–based open-source scada system. *Energy*, 15(4):234–245, 2023.

[18] A. Ucar, M. Karakose, and N. Kırımça. Artificial intelligence for predictive maintenance applications: Key components, trustworthiness, and future trends. *Applied Sciences*, 14(2):898, 2024.

[19] J. Wang, L. Zhang, and M. Chen. Edge computing-based proactive control method for industrial systems. *Scientific Reports*, 14:12345, 2024.

[20] J. Zhang, X. Wang, and Z. Wu. Synthetic data generation approaches for predictive maintenance: A review. *Journal of Manufacturing Systems*, 68:475–487, 2023.