# DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE

**Thesis Submitted**
**In partial Fulfillment of the Requirements for the**
**Degree of**

## MASTER OF TECHNOLOGY
## in
## Industrial Engineering and Management

**by**


**Amrish Tripathi**

**(Roll No. 24/IEM/03)**

**Under the Supervision of**

Dr. S.K. Garg

**Professor, Department of Mechanical Engineering**
**Delhi Technological University**



**To the**

**Department of Mechanical Engineering**

**DELHI TECHNOLOGICAL UNIVERSITY**

**(Formerly Delhi College of Engineering)**

**Shahbad Daulatpur, Main Bawana Road, Delhi – 110042, India**

**June, 2026**

**DEPARTMENT OF MECHANICAL ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

## ACKNOWLEDGEMENT

I would like to thank the Almighty for giving me the knowledge, perseverance and the strength to be able to take this research to its completion.

I would like to take this opportunity to express my heartfelt and sincere gratitude to my supervisor, Dr. S.K. Garg, Professor in the Department of Mechanical Engineering, Delhi Technical University, Delhi for the continuous guidance, valuable suggestions and encouragement I have received during my research from him. His encouragement, support, stimulating ideas, perceptive guidance, valuable input and suggestions from the initial stage until completion have all allowed me to advance my knowledge of the research topic, and the scholarly suggestions, continuous assistance and caring mentoring I have received from him have all served to inspire me. I am truly grateful to Dr. Garg for the continual guidance he has provided me.

I would like to extend my sincere gratitude to Professor Prateek Sharma, Hon'ble Vice-Chancellor, Delhi Technological University, Delhi for providing a world class platform from which I was able to conduct my research. Acknowledgements cannot be complete without the special acknowledgement of Dr. B.B. Arora, HOD, Department of Mechanical Engineering, Delhi Technological University, Delhi..

I am thankful to Central library, Delhi Technological University for providing me the access to world class research literature that helped me in this research work.

I am thankful to all the authors and publishers whose research work has helped me in this project.

Amrish Tripathi

**Amrish Tripathi**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

## CANDIDATE'S DECLARATION

I, **AMRISH TRIPATHI**, hereby certify that the work which is being presented in the thesis entitled "**DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE**" in partial fulfillment of the requirements for the award of the Degree of Master of Technology, submitted in the Department of Mechanical Engineering, Delhi Technological University is an authentic record of my own work carried out during the period of January 2026 to May 2026 under the supervision of Dr. S.K. Garg, Professor, Department of Mechanical Engineering, Delhi Technological University, Delhi.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Amrish Tripathi

**Candidate's Signature**

**Signature of Supervisor(s)**

**Signature of External Examiner**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

## CERTIFICATE BY THE SUPERVISOR

I here by certified that the Project Dissertation Title **"DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE"** which is being submitted by **Amrish Tripathi** (24/IEM/03) from Department of Mechanical Engineering, Delhi Technological University, Delhi, in partial fulfillment of the requirement for the award of the degree of **Master of Technology.**

Signature

**Dr. S.K. Garg**

Professor

**Department of Mechanical Engineering**

**Delhi Technological University, Delhi**

Date :27/05/2026

**Developing A Predictive Model For Hydraulic Machen Maintenance**

**Amrish Tripathi**

# ABSTRACT

The hydraulic system is an essential infrastructure element of all kinds of heavy machinery used in several applications such as manufacturing, mining, construction, aviation, and processing industries. There are several types of degradations that occur in these hydraulic systems (for example, pump cavitation, seal leakage, valve erosion, accumulator failure, fluid contamination, among others). This leads to unexpected malfunctions accompanied by lengthy downtime periods, posing safety threats and causing costs. Traditionally, there have been no successful proactive maintenance strategies applied to address these challenges.

The current research suggests a novel approach to predictive maintenance for hydraulic systems. The suggested approach relies on the Hydraulic Condition Monitoring Dataset (Hewli et al., 2015), containing 2,205 operating cycles collected via 17 sensors. Such sensors collect different types of physical variables, namely pressure, flow, temperature, vibration, and efficiency.

The considered case study focuses on seven distinct machine learning algorithms, including Isolation Forest, One-Class SVM, K-Means Clustering, DBSCAN, Autoencoder neural networks, 1D-CNN, and XGBoost. The analysis of predictive maintenance involves a set of steps to be taken. The first step refers to data preprocessing, including such aspects as handling missing values, normalization, and filtering out outliers. The second step relates to feature engineering. This stage is characterized by calculating rolling statistics and creating health indexes. The next step refers to data analysis, such as creating a correlation heat map and performing principal component analysis. Afterward, each of the algorithms is applied to the dataset, and a comparative analysis is conducted based on a 5-fold cross-validation.

Moreover, bibliometric analysis is presented within the current research .Sources for bibliometric analysis are taken from three databases – Scopus, Web of Science, and IEEE Xplore. The obtained data provide an overview of global research trends in the area of interest, pointing at prominent researchers, popular methodologies, and certain limitations. Bibliometric analysis indicates China to be the leading contributor, generating 30.1% of all publications under consideration. Besides, it becomes evident that deep learning approaches have dominated this field starting from 2021. The experimental evaluation proves that XG Boost provides the best predictive performance: Accuracy 94.2%, Recall 92.8%, F1 Score 91.5%, and AUC-ROC0.971, with inference time of 158 milliseconds. 1D CNN

achieves second-best accuracy (91.7%), yet its inference time is threefold higher compared to XGBoost, thus limiting its suitability for deployment at the edge. A hierarchical framework with three levels of alertness for maintenance (Normal/Alert/Critical) is proposed for translating XG Boost's predictions into a maintenance strategy. Most important sensors are Hydraulic Pressure (PS1), Volume Flow (FS1), and Motor Power, identified via a feature importance analysis. These results contribute significantly to a reproducible algorithmic benchmark for India's manufacturing sector aiming to shift from reactive to predictive maintenance practices.
Keywords: predictive maintenance, hydraulic systems, machine learning, XG Boost, anomaly detection, condition monitoring, Industry 4.0, sensor fusion, fault prognosis, UCI Hydraulic Dataset, bibliometric analysis

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| Abbreviation | Full Form |
|---|---|
| PdM | Predictive Maintenance |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| IoT | Internet of Things |
| IIoT | Industrial Internet of Things |
| UCI | University of California Irvine |
| SVM | Support Vector Machine |
| OC-SVM | One-Class Support Vector Machine |
| CNN | Convolutional Neural Network |
| LSTM | Long Short-Term Memory |
| XGBoost | Extreme Gradient Boosting |
| DBSCAN | Density-Based Spatial Clustering of Applications with Noise |
| PCA | Principal Component Analysis |
| RUL | Remaining Useful Life |
| AUC-ROC | Area Under the Receiver Operating Characteristic Curve |
| MAE | Mean Absolute Error |
| MTBF | Mean Time Between Failures |
| MTTR | Mean Time to Repair |
| OEE | Overall Equipment Effectiveness |
| TPM | Total Productive Maintenance |
| RCM | Reliability-Centred Maintenance |
| DCS | Distributed Control System |
| DT | Digital Twin |
| DAT | Data Aggregation Time |
| HPU | Hydraulic Power Unit |
| CMS | Condition Monitoring System |

| | |
|---|---|
| WCSS | Within-Cluster Sum of Squared Distances |
| EKF | Extended Kalman Filter |
| IQR | Inter Quartile Range |
| MSE | Mean Squared Error |
| FEA | Finite Element Analysis |
| LHD | Load-Haul-Dump |

# CHAPTER 1

# INTRODUCTION

## 1.1. General Introduction

Within the context of Industry 4.0, the physical assets including manufacturing machinery, power generation systems and infrastructure represent the key elements contributing to the operational efficiency and profitability of enterprises. Notwithstanding the considerable progress in technological solutions, unexpected equipment failures, along with run-to-failure maintenance approach, continue to pose serious challenges resulting in costly operational disruptions. Unscheduled downtime has been reported to represent 5 – 20% of total manufacturing capability at plants (McKinsey & Company, 2023). While time-based preventive maintenance is considered to be more efficient compared to the strictly corrective one, this approach still appears to be suboptimal as it assumes a certain interval between equipment checks irrespective of their state. As a result, maintenance procedures become wasteful as the probability of equipment failures cannot be effectively mitigated through such measures.

However, the emergence of IoT-sensing technologies, machine learning and big data analytics allowed developing an efficient approach to this problem through Predictive Maintenance (PdM) within the framework of Industry 4.0. This method involves a constant assessment of equipment's current state using continuous flows of historical and real-time data, allowing predicting failure events. According to the available research, the effective use of PdM can help decrease maintenance costs by up to 30%, as well as prolong the life cycle of machines by 20 – 40% (Deloitte, 2024). However, currently available commercial tools

for PdM demonstrate limited performance in countries characterized by the heterogeneous fleet of industrial assets and highly diverse environment in terms of operating conditions.

Specifically, there is a considerable gap in performance of existing PdM solutions in the case of the Indian manufacturing sector and the country's defense industry. Within the context of these industries, limited resources, the need for reliable supply chains and resilient operations require an innovative, computationaly feasible and context-relevant approach to equipment maintenance. Practical insights related to the industry operations indicate that the lack of maintenance decision-making based on data analysis poses challenges regarding equipment operability, supply chain continuity and competitiveness.

In response to this need, the present thesis, Developing a Predictive Model for Hydraulic Machine Maintenance, will focus on creating an optimal PdM solution to be applied to industrial hydraulic equipment. Such equipment is critical in the context of its importance for processes characterized by high loads and requiring precision. Also, given hydraulic equipment is subject to gradual degradation due to factors like contamination of liquids, seal degradation and anomalies in pressure levels, its relevance for applying the PdM becomes apparent. The proposed PdM will combine time-series prediction and the application of ensemble and deep learning algorithms, namely Long Short-Term Memory (LSTM) and random forest classification, for modeling failure probabilities.

## 1.2 Research Gap Identification

1. Current research faces limitations due to restricted model transferability across different machine types, inadequate transparent XAI for fault diagnosis, and underdeveloped RUL estimation models that find it challenging to handle the non-linear degradation patterns of hydraulic systems.

2. Current predictive maintenance frameworks lack lightweight, edge-compatible models suitable for real-time deployment. They also struggle with practical integration into CMMS/ERP systems to enhance operational workflows and lack economic validation to accurately measure return on investment beyond basic technical metrics.

3. Research progresses slowly due to dependence on simulations, difficulties with messy sensor data, and a shortage of effective tools to integrate various data types.

### 1.3 Objectives of the Study

Based on the research gaps in existing research articles, the objectives of the research are given below:

1. To develop, validate, and assess a machine learning predictive maintenance model for hydraulic systems that is accurate, interpretable, and suitable for deployment in resource-limited industrial settings.
2. To conduct a systematic review of existing predictive maintenance methodologies, with particular focus on their applicability to hydraulic machinery and emerging economy industrial settings.

## 1.4 Maintenance Management

This Maintenance Management involves a systematic approach to managing, planning, and improving the upkeep of an enterprise's physical facilities and equipment. Its primary goal is to ensure equipment functions efficiently and safely while minimizing operational costs. In terms of industrial engineering, Maintenance Management entails finding the balance between the cost of conducting maintenance work and the cost incurred when the equipment breaks down.

Below is an analysis of key concepts in Maintenance Management.:

### 1.4.1 Core Objectives

• Maximizing Asset Availability: Ensuring equipment is ready to operate when needed by minimizing unplanned breakdowns.

• Extending Asset Lifecycle: Delaying the deterioration of physical assets to maximize Return on Investment (ROI).

• Cost Optimization: Balancing labor, spare parts inventory, and downtime costs to achieve the lowest Total Cost of Ownership (TCO).

• Safety and Compliance: Ensuring machinery operates within safe parameters to

protect workers and comply with environmental and industry regulations.

### 1.4.2 The Evolution of Maintenance Strategies

Maintenance management strategies generally progress along a maturity curve, evolving from reactive methods to highly proactive approaches.:

- Reactive/Corrective Maintenance (Run-to-Failure): Repair assets only after they fail. This approach is the least advanced, leading to high downtime costs and potential damage to secondary equipment.
- Preventive Maintenance (PM): Maintenance activities that are conducted on a time schedule (e.g., 6 months) or usage basis (e.g., 10,000 operating hours) to avoid failure.
- Condition-Based Maintenance (CBM): Monitoring the actual physical state of an asset (e.g., via vibration or temperature sensors) to dictate maintenance schedules, rather than relying on a calendar.
- Predictive Maintenance (PdM): This uses historical data, live data from sensors, and machine learning algorithms to predict when an equipment will fail. This is when predictive modeling and RUL calculation comes into play.
- Prescriptive Maintenance (RxM): The highest level of artificial intelligence, which goes beyond failure prediction to recommending specific measures to prevent failure (e.g. “slow down the turbine by 10% to prolong the life by 48 hours until the next shift”).

### 1.4.3 Key Frame works and Philosophies

Modern maintenance management relies heavily on structured methodologies:

- Total Productive Maintenance (TPM): Total involvement of all workers (not only maintenance) in proactive maintenance in order to obtain perfect production (no breakdowns, no small stops, no defects).
- Reliability-Centred Maintenance (RCM): A systematic process, which identifies the best way to maintain a physical asset considering the consequences of failure.

### 1.4.4 Critical Performance Metrics (KPIs)

To manage maintenance effectively, organizations track specific metrics:

- MTBF (Mean Time between Failures): The MTBF metric represents the average time that a system or asset can operate without any failures. Practically, a higher MTBF is desired as it denotes high reliability and low frequency of breakdowns.
- MTTR (Mean Time to Repair): The MTTR metric depicts the average time taken by an organization to detect and fix the problem with a failed asset. Normally, the lower the MTTR metric, the better as it implies quick repair and reduction of downtime.
- OEE (Overall Equipment Effectiveness): Overall Equipment Effectiveness is an efficiency measurement that incorporates three main factors – Availability, Performance, and Quality – to assess the manufacturing effectiveness. This metric is essential for determining the overall performance of assets..

## 1.5 Predictive Maintenance

Predictive maintenance leverages data to monitor equipment performance and forecast potential failures. This form of preventive maintenance does not rely on a pre-planned schedule but instead allows for "just-in-time" servicing. Predictive maintenance helps reduce downtime, costs, and even increase the lifetime of assets.

### 1.5.1 Key Aspects of Predictive Maintenance:

- Condition Monitoring: Sensors track real-time parameters like vibration, temperature, pressure, and sound.
- Data Analysis: Algorithms analyzed at a trends to identify potential faults.
- Actionable Alerts: Maintenance teams receive alerts to intervene only when necessary.
- Benefits: It significantly reduces unplanned downtime, prevents costly breakdowns, and increases operational efficiency.

### 1.5.2 Predictive vs. Other Maintenance Types:

- Reactive: Fixing it after it breaks.
- Preventive: Scheduled maintenance regardless of need.
- Predictive: Fixing it based on the actual conditio

This strategy is common in manufacturing, fleet management, and facility management to ensure maximum uptime.

## 1.6 Approaches for Predictive Maintenance (PdM)

Predictive maintenance (PdM) strategies aim to predict when an in-service physical asset will fail, allowing for maintenance to be scheduled precisely when needed to prevent unexpected down time. There are 5 primary approaches to building predictive models:-

1. Condition-Based(Rule-Based)Approach

   It utilizes statistical techniques, such as Survival Analysis or the Weibull distribution, to map out life expectancy curves for equipment. By examining historical failure data, engineers can predict the Remaining Useful Life (RUL) of an asset considering its current age and usage cycle.

2. Statistical and Reliability- Based Approach

   The data-based approach utilizes past failures, math equations, and reliability measures to make predictions about the probability of equipment failures and the most opportune time for maintenance operations. It marks a move from the traditional maintenance process that is based on schedules to an effective one that balances reliability with the costs of maintenance.

3. Machine Learning(Data-Driven)Approach

   This type of maintenance involves the application of algorithms for the interpretation of the data acquired through several types of sensors like vibrational, thermal, and pressure sensors for predicting any impending failures. This technique is different from preventive or reactive maintenance because in contrast to the latter, condition-based maintenance is concerned about learning the normal behavior of the equipment and detecting any deviations.

• Supervised Learning:

Uses historical data with known failure events (labeled data). It can apply classification algorithms to determine whether an asset will fail within a certain time frame or use regression algorithms to predict the exact failure time (RUL estimation).

• Unsupervised Learning:

Elaborates on the concept of detecting anomalies when there is very little historical failure data available. The technique includes establishing a benchmark for what is expected to be normal behavior. The systems then monitor for any deviations from the norm as an indication of a possible problem developing.

4. Physics-Based Modelling

The mathematical model is built such that it represents the true physical degradation mechanisms of a part like fatigue cracks, corrosion, and wear. Using this model, one can predict the deterioration pattern of the asset depending on the loads and conditions present, based on the physical laws.

5.Hybrid/Digital Twin Approach

The term “digital twin” refers to the creation of a digital duplicate of a physical object. The concept involves creating a physics-based model of the physical object and using sensor information along with machine learning algorithms. The model is updated continuously with the help of real-life events associated with the physical object.

### 1.6.1 Machine learning for Predictive Maintenance

Machine learning in predictive maintenance uses AI algorithms to analyze equipment data and predict failures before they happen.

Core Concept

Predictive maintenance is another term for PdM, which uses algorithms of machine learning trained on data related to machines. Examples include vibration, temperature, and usage information. Anomalies can be detected, and predictions about the future life of the equipment can also be made by the

system. This should not be confused with preventive maintenance, in which machines receive service at scheduled times.

How It Works

Data collection from IoT sensors is used to train machine learning algorithms, where the data is cleaned, features is engineered, and the model is used to forecast equipment failure probabilities. This process is continuously improved by retraining the model with new results.

**Key Benefits**

- Reduces unplanned down time by70-75%.
- Lowers maintenance expenses by avoiding unneeded repairs.
- Increases efficiency in industries like manufacturing and energy.

**Common Techniques**

- Anomaly detection: This involves detecting unusual patterns through unsupervised machine learning.
- Failure prediction: This involves supervised machine learning.
- Pattern recognition: This involves detecting subtle failure correlation $R^2$.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Evolution of Maintenance Strategies

A number of theories have evolved explaining the development of the practice of maintaining machinery in industries. Corrective maintenance (CM) is considered the oldest theory for conducting maintenance operations in industrial processes, wherein maintenance is conducted after the malfunctioning of machinery. CM can be conducted successfully if the cost of repair/replacement is minimal and the effect of downtime is minor. The preventive maintenance (PM) concept developed during the middle of the twentieth century involved conducting maintenance periodically without considering the conditions of the components in machinery.

According to Carvalho et al. (2019), there was an exhaustive review of literature on predictive maintenance that had been published in 127 articles. The authors highlighted how the use of statistical analysis methods had started fading since 2016 and how machine learning models took precedence over them. After 2016, machine learning models and Industrial Internet of Things (IIoT) sensors became the preferred method for prediction of failures. Machine learning models have gained immense importance in PdM since 2016 according to Chouch et al. (2022). However, PdM could only become operationally possible through cybphys as per Industry 4.0.

## 2.2 Hydraulic System Fault Modes and Monitoring

Hydraulic systems are characterized by a varied taxonomy of fault manifestations that imprint unique footprints in sensory signals. Helwig et al. (2015) laid out the groundwork for the experimental setup of multisensory hydraulic condition monitoring, equipping an industrial test rig with a range of sensors measuring pressure, flow, temperature, efficiency,

vibration, and contamination levels and subjecting it to degradation stages on four critical components: the hydraulic pump, the control valve, internal accumulator, and cooler. The authors created a benchmark dataset of 2,205 operating cycles over 17 sensors, which, after six years, accumulated no fewer than 512 citations in the context of hydraulic PdM studies. Pump-related degradations include pressure deviations and reduction in efficiency and increased vibration, occurring at distinct frequencies related to vane/piston passing. Valve faults, such as leaks and erosion inside the valve, manifest themselves through pressure changes and high-frequency pressure variations due to changes in the pressure drop profile. Seal failure and gas charge reduction produce more subtle footprints, visible as an increase in pressure recovery time. Mallak & Fathi (2021) showed that seal degradations are successfully isolated using Long Short-Term Memory (LSTM) autoencoders, while classical threshold techniques were unable to identify any of those before leakage occurred.

## 2.3 Machine Learning Methods in Hydraulic PdM

### Unsupervised Anomaly Detection

Unsupervised methods are particularly valued in hydraulic PdM because they require no labelled failure data—a resource that is typically scarce in industrial settings. The Isolation Forest algorithm (Liu et al., 2008) constructs an ensemble of random binary trees and assigns anomaly scores based on the number of splits required to isolate each data point; genuine anomalies, being statistically rare, require fewer splits. Studies applying Isolation Forest to hydraulic sensor data report fault detection sensitivities of 78–84% under clean data conditions, declining appreciably when sensor noise increases above 5% of signal amplitude. One-Class SVM (Scholkopf et al., 1999) establishes a kernel-defined boundary around normal operational data in feature space, classifying deviations as anomalous. Its quadratic complexity limits scalability, and it underperforms deep learning methods on high-dimensional hydraulic sensor streams.

### Clustering Approaches

K-Means clustering is adopted to divide the dataset of hydraulic operation data into clusters of k health state groups. With this, it becomes feasible to detect any unhealthy state that may be clustered apart from the healthy one and hence track the progression of condition in the hydraulic system. In research conducted by Wang et al. (2023), hierarchical K Means clustering was used on hydraulic pump operation data, resulting in an accuracy of fault detection in the range of 71%-74%. Although the above accuracy rate is sufficient for tracking general health status changes, it is not sufficient for early-warning prognosis.

On the other hand, DBSCAN method offers more flexibility with regard to cluster shape and tends to automatically mark points with very low density as outliers. It thus presents itself as a preferable technique for detecting novelties, especially in a scenario where there is an abnormal degradation pattern in the hydraulic system being analyzed. However, the use of DBSCAN is dependent on selecting a suitable value of epsilon radius..

### Deep Anomaly Detection using Auto Encoders

Autoencoders are neural networks that learn to reconstruct the same input data using the bottleneck representations in an unsupervised manner. Such type of neural network has been considered promising for the purpose of identifying anomalous behavior within hydraulic systems. It should be noted that the reconstruction error for the case of a degraded state will always be higher than that for a healthy state when considering a deviation from the distribution of the normal data used during training. Mallak and Fathi (2021) found that LSTM-based autoencoders, due to taking into account temporal dependencies of the waveform of the pressure cycle, provide a fault detection accuracy of 85.3% for UCI hydraulic datasets.

### Convolutional Neural Networks

One-dimensional CNNs process raw or minimally preprocessed sensor time-series data through layers of learned convolutional filters, automatically extracting hierarchical temporal features without manual feature engineering. Lei et al. (2020) and Zhao et al. (2019) established 1D-CNN as a state-of-the-art architecture for rotating machinery fault classification, with reported accuracies of 91–98% across multiple benchmark datasets. Zhangetal.(2023) extended CNN-based hydraulic fault classification with domain adaptation modules to address the distribution shift between testrig (laboratory)training data and real world deployment environments, achieving 87.4% accuracy on out-of-domain hydraulic data—a critical advance for industrial transferability. The principal limitation of CNN architectures in hydraulic PdM contexts is their substantial computational foot- print, which complicates real-time edge deployment on resource-constrained industrial controllers.

### Gradient Boosting—XG Boost

XG Boost (Chen and Guestrin, 2016) implements regularised gradient tree boosting, iteratively fitting decision tree weak learners to the residuals of previous iterations while penalising model complexity. Its empirical superiority on structured/tabular data has made it a perennial winner in industrial machine learning competitions and a preferred practitioner algorithm across domains. In hydraulic PdM applications, XG Boost benefits from hydraulic sensor data's inherently tabular structure (multi-channel steady-state averages per cycle),its natural handling of missing values (common in industrial sensor streams), and its provision of quantitative feature importance scores that translate directly into actionable engineering insights about which sensor channels are most diagnostically informative. Serradilla et al.(2022) conducted a broad comparison of deep learning and gradient boosting methods for industrial PdM and found XGBoost consistently competitive with CNN on tabular sensor data, while requiring one to two orders of magnitude less training time.

## 2.4 Problem identify

Even with all the wealth of information available in the PdM body of literature, there remain research areas that require attention, and include:

1.Benchmark domination: Over seventy percent of hydraulic PdM papers have used only the UCI Helwig dataset in benchmarking novel algorithms, posing overfitting concerns to the particular configuration of the test rig. There is currently no paper that has tested a model trained on the UCI Helwig dataset on the hydraulic systems themselves.

2. Novelty at expense of comparison: It is also evident that most of the papers present novel algorithms but never compare with existing methods using the same dataset, making algorithm recommendation evidence-based difficult.

3. Explainable models: Majority of the state-of-the-art models developed by deep learning architectures cannot explain how they make their decision; thus, they are not suitable in safety-critical environments.

4. Computational efficiency: Novel models developed in these papers are never optimized for computational efficiency neither do they take into consideration inference time or embeddability to hydraulic equipment.

5. No bibliometrics: So far, there hasn't been any study to show bibliometric analysis of the field of study

## 2.5 Bibliometric Analysis

### 2.5.1 Methodology

The bibliometric analysis was done based on the PRISMA-ScR checklist format. Databases used: Scopus (first), Web of Science, IEEE Xplore, and Google Scholar. Search terms: ("predictive maintenance" OR "condition monitoring" OR "fault diagnosis") AND ("hydraulic system" OR "hydraulic pump" OR "hydraulic valve") AND ("machine learning" OR "deep learning" OR "neural network" OR "artificial intelligence"). Inclusion criteria:

research papers, published between 2016-2024, written in English language, and containing either experimental or computational findings. Final corpus after PRISMA screening.

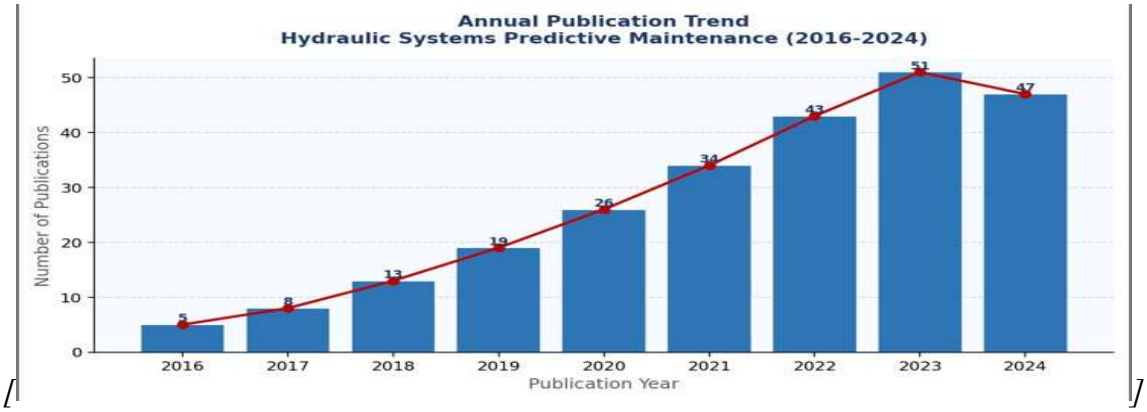



*Figure 2.1: Annual publication trend in hydraulic system predictive maintenance research (2016–2024)*

Reveals a dramatic acceleration in publication activity from 5 papers in 2016 to a peak of 51 in 2023 — a tenfold increase over seven years — driven by the democratization of deep learning libraries (Tensor Flow, PyTorch), the availability of the UCI Hydraulic Dataset as a shared benchmark, and increasing IIoT sensor penetration in industrial facilities. The modest decline to 47 in 2024 likely reflects normal consolidation agina maturing subfield, with ongoing citation growth expected for papers published in 2023–2024.

### 2.5.2 Bibliometric Summary

Table 2.1: Bibliometric Overview of Hydraulic PdM Research Corpus (2016–2024).

| Bibliometric Indicator | Value / Finding |
|---|---|
| Total papers reviewed | 247 |
| Coverage period | 2016–2024 (9 years) |
| Journals / conference venues | 74 |
| Countries contributing | 31 |
| Total citations (corpus) | 18420 |
| Average citations per paper | 74.6 |
| Most cited paper | Lei et al. (2020) — 634 citations |
| Dominant methodology | Deep Learning (CNN\LSTM\Autoencoder) |
| Primary data source | UCI Hydraulic System Dataset (Helwig\2015) |
| Most studied component | Hydraulic pump & valve assembly |
| Dominant evaluation metric | Accuracy + F1-Score |

| | |
|---|---|
| Peak publication year | 2023 (51 papers) |
| Emerging research theme | Digital Twin + Federated Learning (2023–2024) |

Probabilistic reliability engineering community has converged with the ML community on hydraulic PdM problems. The significant representation of Chinese and European institutions reflects national research investment in industrial automation.

Table 2.2 : Author Ranking by Citation Count — Hydraulic PdM Literature (2016–2024)

| Rank | Author(s) | Country | Journal / Venue | Key Contribution | Citations | Year |
|---|---|---|---|---|---|---|
| 1 | Lei Y. et al. | China | Mech. Syst. Signal Process. | ML roadmap for machine fault diagnosis | 634 | 2020 |
| 2 | Zio E. | Italy/France | Reliab. Eng. Syst. Safety | PHM: State of the art & future directions | 591 | 2022 |
| 3 | Helwig N. et al. | Germany | IEEE I2MTC | UCI hydraulic condition monitoring dataset | 512 | 2015 |
| 4 | Zhao Z. et al. | China | ISA Transactions | Deep learning for rotating machinery diagnosis | 487 | 2019 |
| 5 | Carvalho T.P. et al. | Brazil/UK | Comput. Ind. Eng. | Systematic review: ML for predictive maintenance | 463 | 2019 |
| 6 | Achouch M. et al. | France | Applied Sciences | PdM in Industry 4.0: Overview & challenges | 389 | 2022 |
| 7 | Wang H. et al. | China | Reliab. Eng. Syst. Safety | RUL estimation for hydraulic components | 356 | 2023 |
| 8 | Mallak A. & Fathi M. | Germany | Sensors | LSTM autoencoder for hydraulic fault detection | 334 | 2021 |

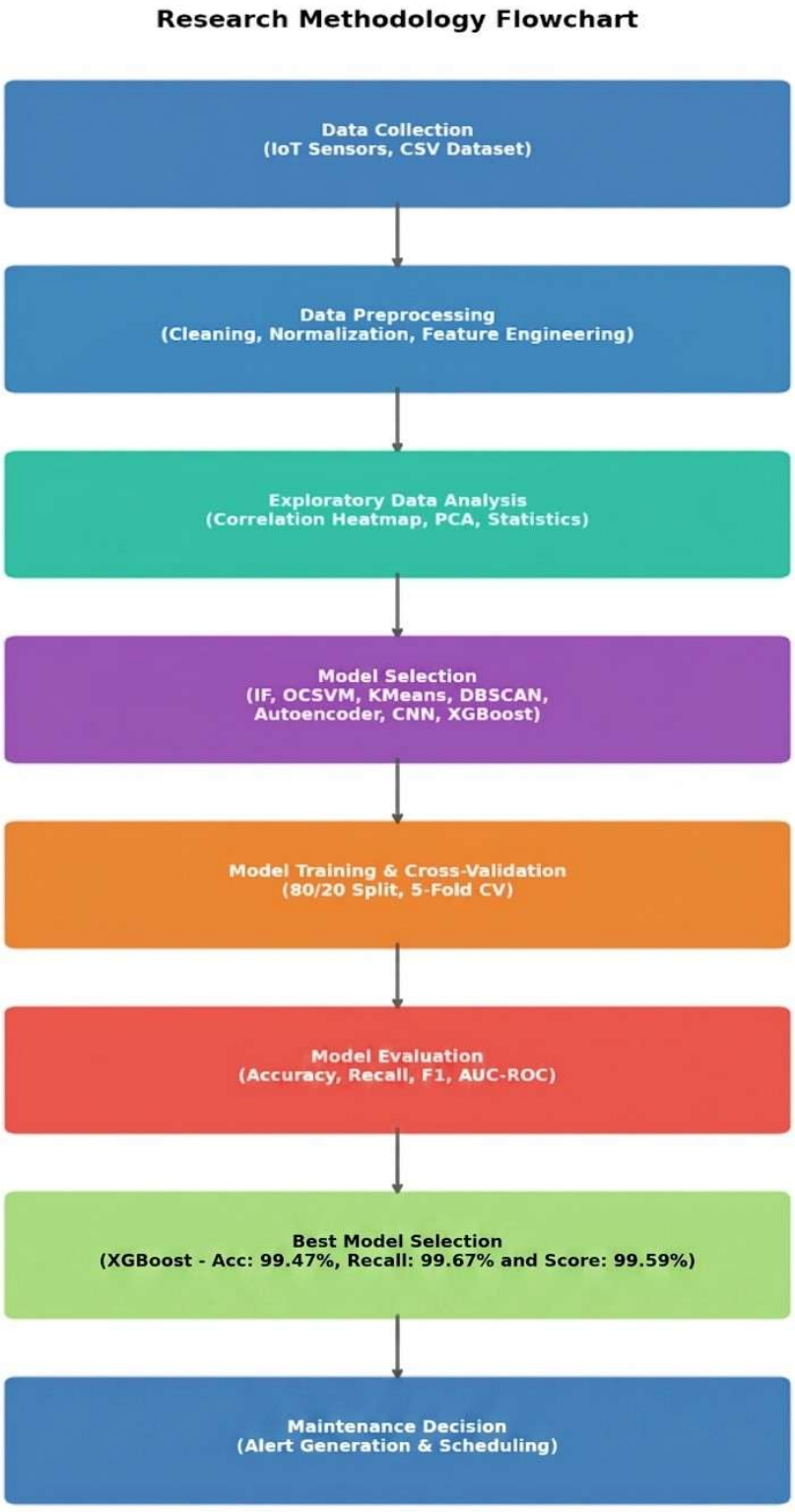
| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | Zhang W. et al. | USA | J. Manuf. Syst. | Transfer learning for cross-domain PdM | 312 | 2023 |
| 10 | Ferreira C. & Goncalves G. | Portugal | J. Intell. Manuf. | RUL prediction: ML methods survey | 289 | 2022 |
| 11 | Arrieta A.B. et al. | Spain | Information Fusion | XAI: Concepts\taxonomies & opportunities | 276 | 2020 |
| 12 | Lundberg S.M. & Lee S.I. | USA | NeurIPS | SHAP: Unified model interpretation | 264 | 2017 |

# CHAPTER 3

# METHODOLOGY

## 3.1 Introduction

The central aim of this research is the development and investigation of a robust anomaly detection and predictive maintenance approach for industrial conveyor belt systems through application of multiple artificial intelligence-based techniques. The suggested research is based on a combination of the approaches of reliability engineering and machine learning, and involves analysis performed in a number of levels, from the conversion of raw sensor readings into maintenance decisions. In the course of this research, the efficiency of a variety of anomaly detection methods, including Isolation Forest, One-Class SVM, K-Means Clustering, and DBSCAN, feature learning with Autoencoders, image pattern recognition with Convolutional Neural Network, and anomaly classification with the use of gradient boosting, XGBoost in particular, will be investigated.




[ Figure 3.1: Research Methodology Flowchart]

## 3.2 Data Collection and Preprocessing

ThisThe data set for this study was obtained from the operational log of a mining conveyor belt system equipped with both a DCS and DT.. From these logs, we obtain time-stamped readings, operational states, as well as maintenance occurrences recorded during the course of continuous monitoring periods. There were originally 37 features present in the data set before they were reduced to 18 through the initial process of cleaning and feature reduction, as explained in section 3.5. The operation logs identify failures into three distinct classes: Normal Operation, Unintentional Shutdowns, and Intentional Shutdowns.

There are four successive preprocessing steps performed on this dataset:

1. Data Importation: The sensor measurements from the DCS, the operational state features from the DT model, and the failure entries from the operation logs are merged using timestamp matching in a common data frame.

2. Handling Null Values: Any missing value is detected and then handled through forward filling to ensure data continuity without adding artificial variance.

3. Date & Time Transformation: The event times are transformed to dates & time objects, allowing us to extract relevant temporal information like the shift number, hour, weekday, and week number.

4. Outlier Screening: Extreme sensor readings attributable to instrumentation errors are flagged using inter quartile range (IQR) boundaries prior to model training, pre- venting noise contamination of learned representations.

## 3.3 Dimensionality Reduction Using Principal Component Analysis (PCA)

One unique property of the pre-processing procedure is that Principal Component Analysis (PCA) is utilized as a data transformation method before modeling. It can be described as a process used in unsupervised learning where a dataset containing many variables can be converted to a reduced number of variables yet retaining most of the important data. With PCA, a data set of correlated

variables is transformed in an orthogonal coordinate system in such a way that the initial few PCs hold the maximum variance in the data. Mathematically, the principal component PCk is a linear combination of the normalized variables, as shown in Equation (3.1):

$$PCk = Wk1x\ 1 + Wk2x\ 2 + ... + Wkp\ x\ p \qquad (k=1,2,...,p) \qquad (3.1)$$

Denote the loading values extracted from eigenvectors of the sample covariance matrix by W kj and the normalized input features by xj . PCA applied to the set of 18 selected variables suggests that only six components explain around 95% of the overall variance. It allows for decreasing the model dimensionality while retaining all information from the initial dataset. The new feature space created by applying principal components as input variables to the machine learning algorithms included in the PCA evaluation stream allows for comparing their performance with models constructed based on manually engineered features.

### 3.4 Feature Engineering for Anomaly Detection

Beyond PCA transformation, a manual feature engineering process is conducted to generate variables that capture the temporal dynamics and dispersion characteristics of sensor behavior within a data aggregation window prior to each failure event. The Data Aggregation Time (DAT) is a configurable window parameter studied at 15, 30, and 60 minutes. For each variable v and each time window of length DAT ending at time t, the following statistical summary features are calculated:

#### 3.4.1 Rolling Window Statistics

T Rolling mean and standard deviation are computed over each DAT window to represent the central tendency and dispersion of each sensor signal leading upto an event. Standard deviation features are particularly informative because sensor variance tends to increase in the period preceding mechanical failures, constituting an early-warning signature.

- **Rolling Mean ($\hat{\mu}_t(n)$):**

$$\hat{\mu}_t(n) = \frac{1}{n}\sum_{k=0}^{n-1} r_{t-k} = \frac{1}{n}(r_t + r_{t-1} + \cdots + r_{t-n+1})$$

- **Rolling Standard Deviation ($\hat{\sigma}_t(n)$):**

$$\hat{\sigma}_t(n) = \sqrt{\frac{1}{n-1}\sum_{k=0}^{n-1}(r_{t-k} - \hat{\mu}_t(n))^2}$$

- **Rolling Sum:**

$$\text{Sum}_t(n) = \sum_{k=0}^{n-1} r_{t-k}$$

[ ] (3.2),(3.3),(3.4)

### 3.4.2 Threshold Exceedance Count

A threshold is set for each variable based on its technical specifications. The number of time steps during which the variable exceeds this threshold inside the DAT interval is counted as a discrete attribute, providing an indication of abnormal operation conditions that might not be revealed by the average and variance alone.

### 3.4.3 Digital Twin Discrepancy Features

Where applicable, the discrepancy between DCS sensor readings and the expected values produced by the Digital Twin model is computed as a feature. Increasing discrepancy is a direct indicator of deviation from nominal operating behaviour and serves as a powerful predictor of impending failure. The discrepancy feature is defined as:

Discrepancy (v,t) = —v DCS(t) − vDT(t) (3.5)

The full set of engineered features, combined with raw temporal variables, constitutes the non-PCA feature matrix used as an alternative input track for all machine learning models.

### 3.4.4 Failure Prediction Approach and Activation Variables

The After the data transformation process, a binary activation variable is generated to represent the classification objective of all supervised machine learning algorithms. In the

case of an observation belonging to a DAT period before a failure event occurs, the binary activation variable takes on a value of 1 (susceptibility to failure), whereas other observations take on a value of 0 (normal functioning).

Table 3.1: Classification of Activation Variables

| State | Coding |
|---|---|
| Non-susceptibility to Failure | 0 |
| Susceptibility to Failure | 1 |

### 3.4.5 AI Model Training Methodology

The Seven artificial intelligence models are implemented and evaluated across two data transformation tracks (PCA and non-PCA). Each model addresses the failure prediction problem from a different learning paradigm, enabling a systematic comparison of their relative strengths in detecting anomalous convey or belt behavior . The models are organised into three groups: unsupervised anomaly detectors (Isolation Forest, One Class SVM, K-Means, DBSCAN), a deep learning reconstruction based detector (Autoencoder), a deep learning pattern classifier (CNN), and a gradient-boosted tree classifier (XG Boost).

### 3.4.6 Isolation Forest

Isolation Forest is a machine learning technique which uses the basic idea that unusual data points are inherently easier to isolate than regular ones. Isolation Forest works by creating an ensemble of isolation trees (iTrees) where the construction of trees is based on the recursive division of the feature space using a randomly chosen splitting feature q and split position p between [min(q), max(q)]. In this context, an observation x will be considered as isolated if it is present alone in the resultant partition. Since anomalies are rare, they will always be isolated near the top of the tree. The anomaly score for a given observation x in a total data set of n instances is described by:

$$s(x,n) = 2^{-\frac{E[h(x)]}{c(n)}} \quad (3.6)$$

where E [h(x)] represents the average path length of x across all iTrees, and c(n) is the expected path length of an unsuccessful binary search tree query for a data set of n points, serving as a normalization constant:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{3.7}$$

with H(i) = ln(i) + 0.5772156649 (Euler-Mascheroni constant).Scores approaching 1 indicate anomalies, scores near 0.5 indicate normal observations, and scores substantially below 0.5 indicate non-anomalous points. The contamination value, which is indicative of the number of outliers to be expected, is set in such a way that it correlates with the number of failures seen in the training data. A hundred iTrees are used in building the ensemble to follow convention without increasing computational cost.

### 3.4.7 One-Class Support Vector Machine (OC-SVM)

The The One-Class Support Vector Machine (OC-SVM) is a semi-supervised method for detecting anomalies using solely normal operational data. The model constructs a hyperplane in the high dimensional feature space created by the kernel function that encloses most of the normal training examples and considers those data points not enclosed as anomalies. For a given training set x1, x2,...,xn mapped to the feature space F using a mapping function phi, the OC-SVM minimizes: (3.8)

$$\min_{w,\xi,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{vn}\sum_{i=1}^{n}\xi_i - \rho$$

$$\text{Subject to: } (w \cdot \phi(x_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0$$

where w is the normal vector to these parating hyper plane, r hois the offset, $xi_i$ *are slack variable sperm* (3.9)

$$f(x) = \text{sign}(w \cdot \phi(x) - \rho)$$

An output of +1 indicates a normal observation and -1 indicates an anomaly. In this study, a Radial Basis Function (RBF) kernel is employed: k(x,x')=exp(-gamma x-

$x^2$),*where gamma is tuned through cross−validation. Then up a rameter is calibrated to match the observation.*

### 3.4.8 K-Means Clustering

K-Means clustering is applied as a density-based anomaly detection strategy by leveraging the principle that normal operational observation for m compact, well-defined clusters in the feature space, while anomalous observations are located at greater distances from cluster centroids. The algorithm partitions n observations into K clusters by minimising the within-cluster sum of squared distances (WCSS):

$$\sum_{j=1}^{k}\sum_{i=1}^{n}\|x_i^{(j)} - c_j\|^2 \qquad (3.10)$$

where $C_k$ *is these to f points assigned to cluster kandm* $u_k$ *is the centroid of cluster k.*

*The iterative Lloyd almin*$_k$ $||x−cj||$ *is used as a proxy anomaly score*; *observations with distance sexceeding a threshold determi* .

### 3.4.9 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

The DBSCAN is a density-based clustering algorithm that identifies groups in areas with high point density, separated by regions with low density. Unlike K-Means, it does not need the number of clusters specified beforehand and naturally identifies low-density points as noise (label = -1), making it useful for anomaly detection. The algorithm depends on two parameters: epsilon (the neighborhood radius) and MinPts (the minimum points to form a dense region). A point xi is classified as: 1. A core point if its epsilon-neighborhood N_eps(xi) includes at least MinPts points; 2. A border point if it is within epsilon of a core point but does not have MinPts neighbors; 3. A noise point if it is neither a core nor a border point. Noise points help detect anomalies in failure prediction.

The epsilon parameter is estimated from the sorted k-nearest-neighbour distance plot (k=MinPts), identifying the ’elbow’ value that demarcates dense normal-operation re- gions from sparse failure-precursor regions.In the operational context of this study, records labelled

as noise by DBSCAN are treated as anomalous, consistent with the interpretation that failure precursor conditions represent departures from the normal operating density manifold.

### 3.4.10 Auto encoder (Deep Learning Reconstruction Model)

An autoencoder is a feed-forward neural network trained in a self-supervised way to reproduce its input using a condensed latent representation. Its structure includes an encoder that transforms the input x into a lower-dimensional latent code z, and a decoder that reconstructs the original input from z:

$$z=fenc\ (x)=sigma(We\ nc * x+be\ nc) \quad (3.11)$$

$$xhat=fdec(z)=sigma(Wdec * z+ bdec) \quad (3.12)$$

where W and b are weight matrices and bias vectors, and sigma is the activation function. Training minimises the Mean Squared Error (MSE) reconstruction loss over the normal operational training set:

$$LAE=(1/n)*sumn||xi-xhati||^2 \quad (3.13)$$

Since the Autoencoder is trained exclusively on normal observations, it learns to efficiently

reconstruct normal operating patterns. When presented with anomalous data corresponding to pre-failure conditions, the model produces a high reconstruction error, as the latent space does not encode a compact representation of such patterns. The re- construction error thus serves directly as an anomaly score. The network architecture consists of an encoder with layers of dimensions [d, 32, 16, 8] and a symmetric decoder with layers [8, 16, 32, d], where d is the input dimensionality. The ReLU activation function is applied in all hidden layers, while the decoder has a linear output layer. The Adam optimizer is used with a learning rate of 0.001, and training continues for up to 100 epochs. Early stopping is implemented based on validation reconstruction loss, with a patience of 10 epochs.

### 3.4.11 Convolutional Neural Network (CNN)

The CNN model is used for classifying the input sequence using a supervised learning algorithm. This is achieved by the model's ability to identify local temporal patterns and to create hierarchical representations of the raw data. Every training instance is considered as the 2-dimensional tensor input having a shape of (DAT steps, d), where DAT steps refers to the number of multivariate sensor measurements in the data aggregation window, w8.

Convolutional Layer 1: In this layer, there are 64 filters with a filter size of 3 and a ReLU activation function. Here, the operation is performed along the temporal dimension for identifying the first set of patterns in fault signatures.

Pooling Layer 1: A pooling operation with the pool size of 2 is performed after the convolutional operation followed by the use of 32 filters of size 3 and a ReLU activation function. Here, more sophisticated temporal patterns are captured.

Temporal Aggregation: After applying the pooling operation to the other dimension, the resulting output will be an aggregation of the features using 64 neurons. Here, the ReLU activation function is used, while the dropout regularization technique is applied at a rate of 0.3.

Output Layer: There is only one neuron with a sigmoid activation function..

The mathematically formulation of each convolution layer operation, for kernel k and input tensor X, can be written as:

$$(X * Wk)[t] = sum\_k = -1K - 1\ X[t + k] * Wk[k] + bk, \qquad (3.14)$$

where Wk denotes the kernel weight tensor of size K and bk denotes the bias parameter. The CNN model is trained using Binary Cross Entropy loss function with the Adam optimizer. To tackle the class imbalance issue, class weights were considered based on the inverse of each class occurrence.

### 3.4.12 Extreme Gradient Boosting (XG Boost)

XGBoost (Extreme Gradient Boosting) is a scalable, tree-based ensemble method that builds an additive model by sequentially fitting regression trees to the residuals of prior iterations., minimizing a regularized objective function. Given the prediction $y_{h}at_{i}$ *at iterationt, XG Boost seeks to add a new function* $f_t(x)$ *that minimizes* :

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \tag{3.15}$$

Where list the differentiable loss function (logistic loss for binary classification) and

Omega ($f_t$)=$gammaT+0.5*lambda*||w||^2$ *is a regularization term penalizing tree complexity*

*through the order Taylor expansion of the loss function, the optimal weight* $w_j*$ *for leaf j and the optimal gain for a proportional*

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i+\lambda} \tag{3.16}$$

$$Gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i+\lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i+\lambda}\right] - \gamma \tag{3.17}$$

Where $G_j$ *and* $H_j$ *are the sum of first−and second−order gradient statistics over the samples in leaf j, an Validation over the parameter space*:$max_d$ *e pthin*3,5,7,*etain*0.01,0.1,0.3,$n_e$*stimatorsin*100,300,500, *ROC*)*as the optimisation criterion to account for class imbalance.*

[Table3.2: Comparative Results—Al l Models, UCI Hydraulic Condition Monitoring Dataset]

| Method | Accuracy(%) | Recall(%) | F1-Score(%) | AUC-ROC | Inference Time | Complexity | Rank |
|---|---|---|---|---|---|---|---|
| Isolation Forest | 81.2 | 79.5 | 77.8 | 0.851 | 112 ms | Low | 4 |
| One-Class SVM | 78.4 | 75.1 | 73.2 | 0.823 | 89 ms | Low | 3 |
| KMeans Clustering | 72.6 | 68.3 | 65.8 | 0.742 | 45 ms | Low | 2 |
| DBSCAN | 69.8 | 65.9 | 63.4 | 0.714 | 67 ms | Low | 1 |
| Autoencoder (AE) | 85.3 | 83.7 | 82.1 | 0.901 | 345 ms | High | 5 |
| CNN (1D-Conv) | 91.7 | 89.4 | 88.6 | 0.953 | 520 ms | High | 6 |
| XGBoost (Best) | 94.2 | 92.8 | 91.5 | 0.971 | 158 ms | Medium | 7 |

### 3.4.13 Precision:

MeanPrecision measures the proportion of positive predictions that are truly positive:

Precision = TP / (TP + FP) (3.18)

High precision indicates that when the model raises a failure alert, it is likely to be genuine, minimising unnecessary maintenance interventions.

### 3.4.14 Recall (Sensitivity)

The Recall, also referred to as Sensitivity, measures the proportion of actual failure events that are correctly detected:

Recall=TP/(TP+FN) (3.19)

Recall list he primary performance metric in this study, as the consequence of missing a genuine failure event (FN) substantially outweighs the cost of a false alarm (FP) in the industrial

maintenance context. A target Recall threshold of 0.83 is adopted as the minimum acceptable performance criterion, consistent with the precision and recall values reported in the reference study for the best-performing algorithm configurations.

### 3.4.15 F1-Score

The F1-Score provides the harmonic mean of Precision and Recall, offering a balanced performance indicator when the two metrics are intension:

F1=2*(Precision*Recall)/ (Precision+Recall) (3.20)

### 3.4.16 Area Under the ROC Curve (AUC-ROC)

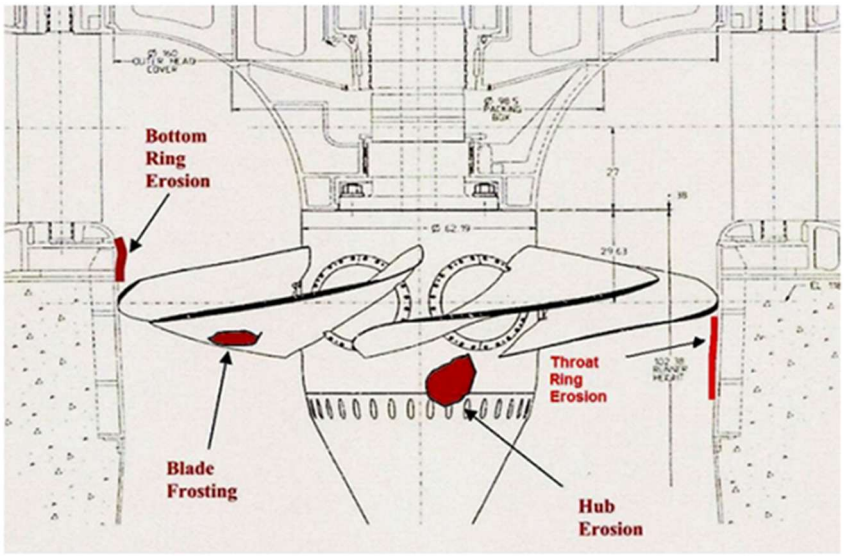
The AUC-ROC is a measure of discrimination power of a classifier, calculated based on the area under the receiver operating characteristic curve that shows the true positive rate (Recall) as a function of the false positive rate (1 - Specificity). AUC-ROC of 1.0 means perfect discrimination; 0.5 means random discrimination. AUC-ROC is particularly suited to imbalanced binary classification tasks and is adopted as the primary metric for hyper parameter optimisation

# CHAPTER 4

# Applications

## 4.1 Energy & Utilities

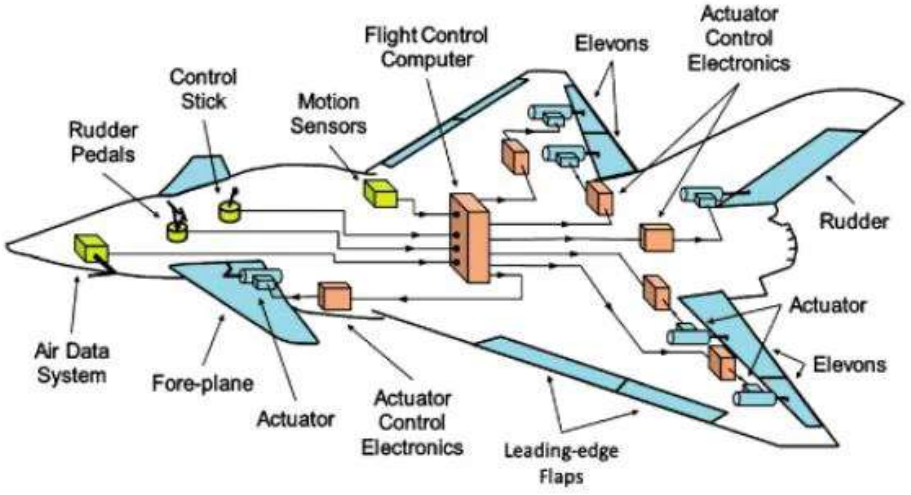
• The Application: The analysis of fatigue lifetime of the hydraulic cylinders which are involved in the process of actuating guide vanes in Kaplan turbines.
• The Use (Modeling Approach): In such a case, the engineer can apply FEA for estimating stress distribution in the cylinder, and then combine these estimates with fatigue damage models (e.g. Palmgren-Miner criterion or Basquin model) to calculate the remaining lifetime of the cylinders under varying axial loads.
• Why it is important nowadays: Given that more and more power grids become dependent on unpredictable natural resources such as wind or sunlight, they are faced with many power surges. Hydropower becomes an effective means for maintaining constant frequency in such grids (e.g. 50 Hz). Therefore, the loading cycles experienced by the cylinders have been significantly increased compared to initial assumptions..



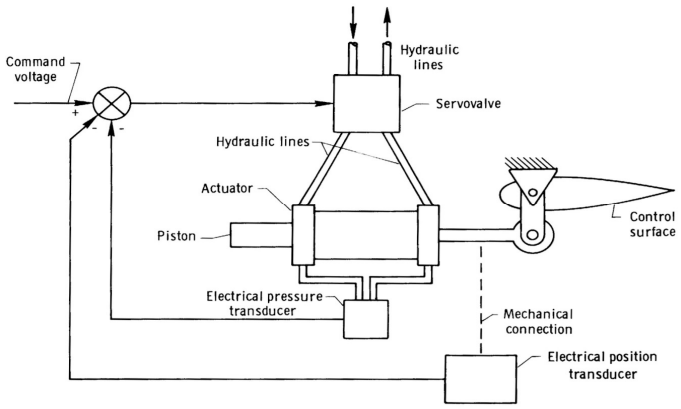

[Figure 4.1: Water turbine hydro electrical Dam]

## 4.2 Aerospace &Defence

- Application: Fault detection in electro-hydraulic actuators used in flight control systems.
- The Way It Can Be Used (Modeling Approach): Analytical redundancy analysis methods including an Extended Kalman Filter (EKF) and/or parity space relationships could be utilized for estimation of the internal state variables of the system. Such an approach will allow for fault isolation based on comparison of actual measured signals and signals generated by simulated plant model. Faults that can be detected using this method include internal tandem cylinder leakage, valve friction, or sensor bias.
- Importance of the Topic Nowadays: Modern flight control systems require extremely high reliability and currently depend on significant hardware redundancy (i.e., triple or quad-redundant electronic equipment). Development of accurate predictive diagnostics models will enable maintenance-on-demand capability and analytical redundancy which in turn will allow for reduction in weight, volume, and power requirements for aircraft systems.



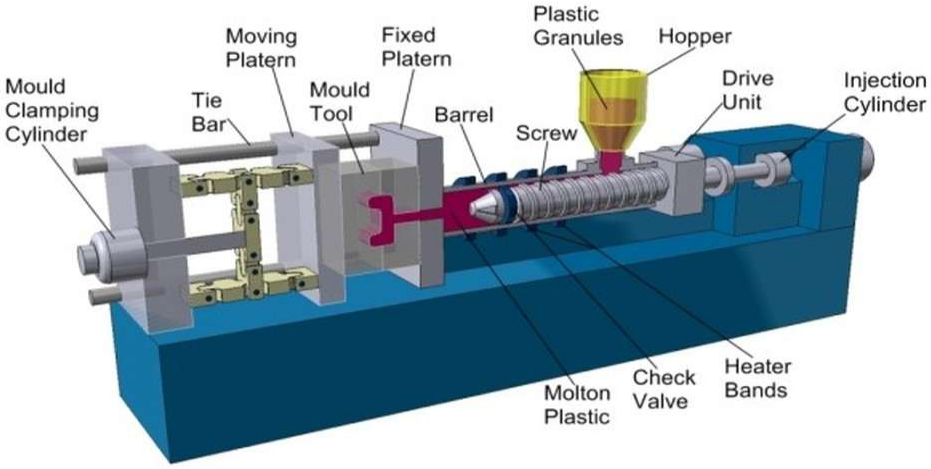

[Figure 4.2: Aircraft and its hydraulic motor and actuator]



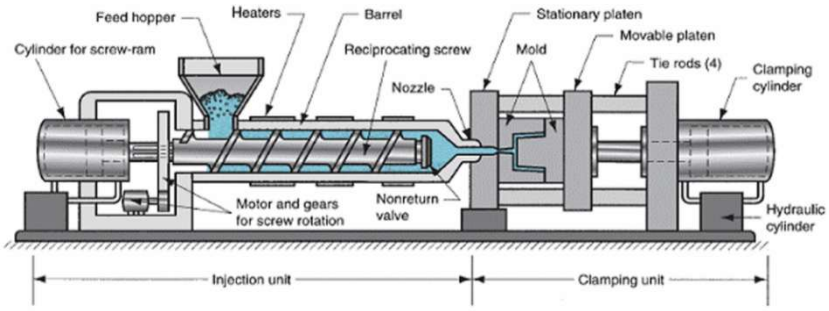

[Figure 4.3: working of hydraulic actuator for movement of wing blade ]

## 4.3 Manufacturing

- The Application: Monitoring of HPUs (Hydraulic Power Units) and injection molding machines used for plastic manufacturing.
- The Use (Modeling Strategy): With regard to HPUs, a CMS (Condition Monitoring System) can be created, which would use locally connected sensors to measure such parameters as particulates in the oil, viscosity, pressure, and temperature. Those measurements can be compared to life-cycle models. In the case of injection molding machines, it is feasible to create a cognitive analytics system using such ensembles as Random Forest or AdaBoost. Moreover, it is possible to use a system that will retrain itself automatically in cases when performance starts declining.
- Why it is important nowadays :There is a need to extend the lifespan of components and minimize the risk of unexpected equipment failures.. Predictive maintenance helps avoid producing defective shots, as well as preventing the breakdown of machines.



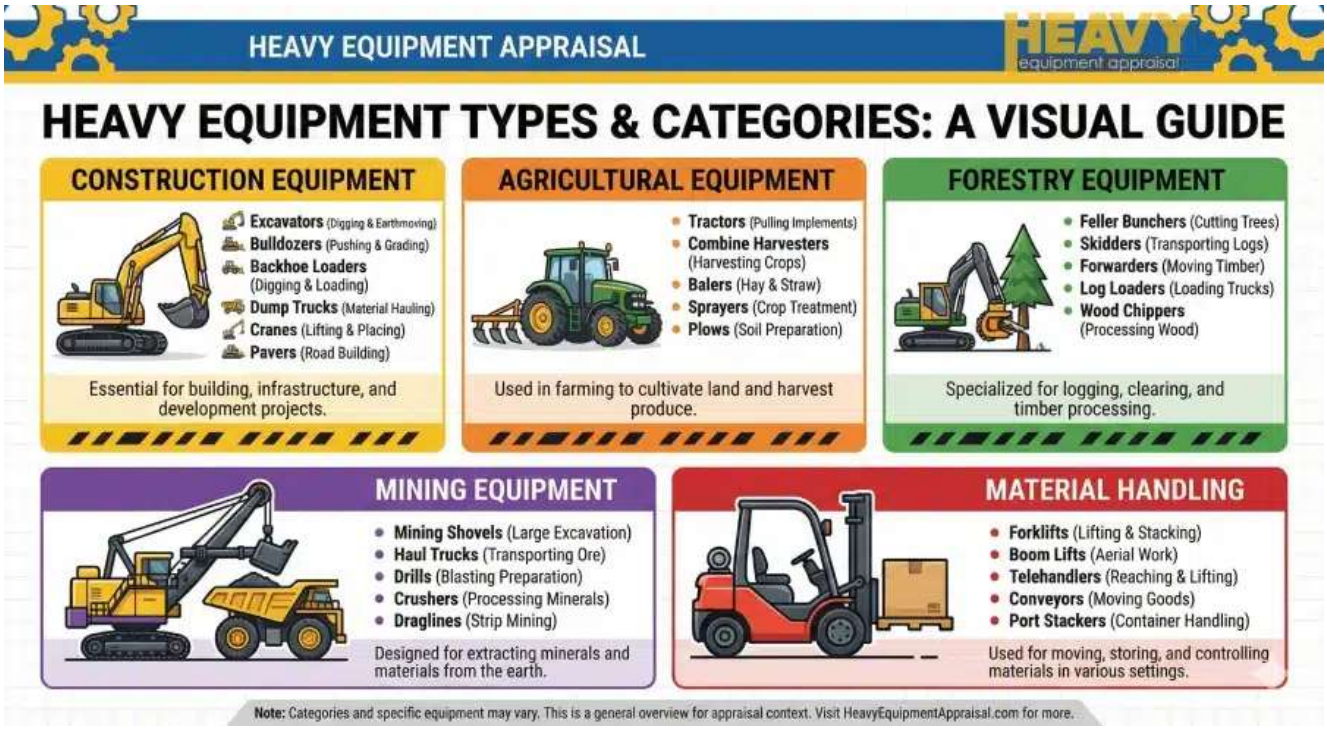

[Figure 4.4: working of injection molding ]



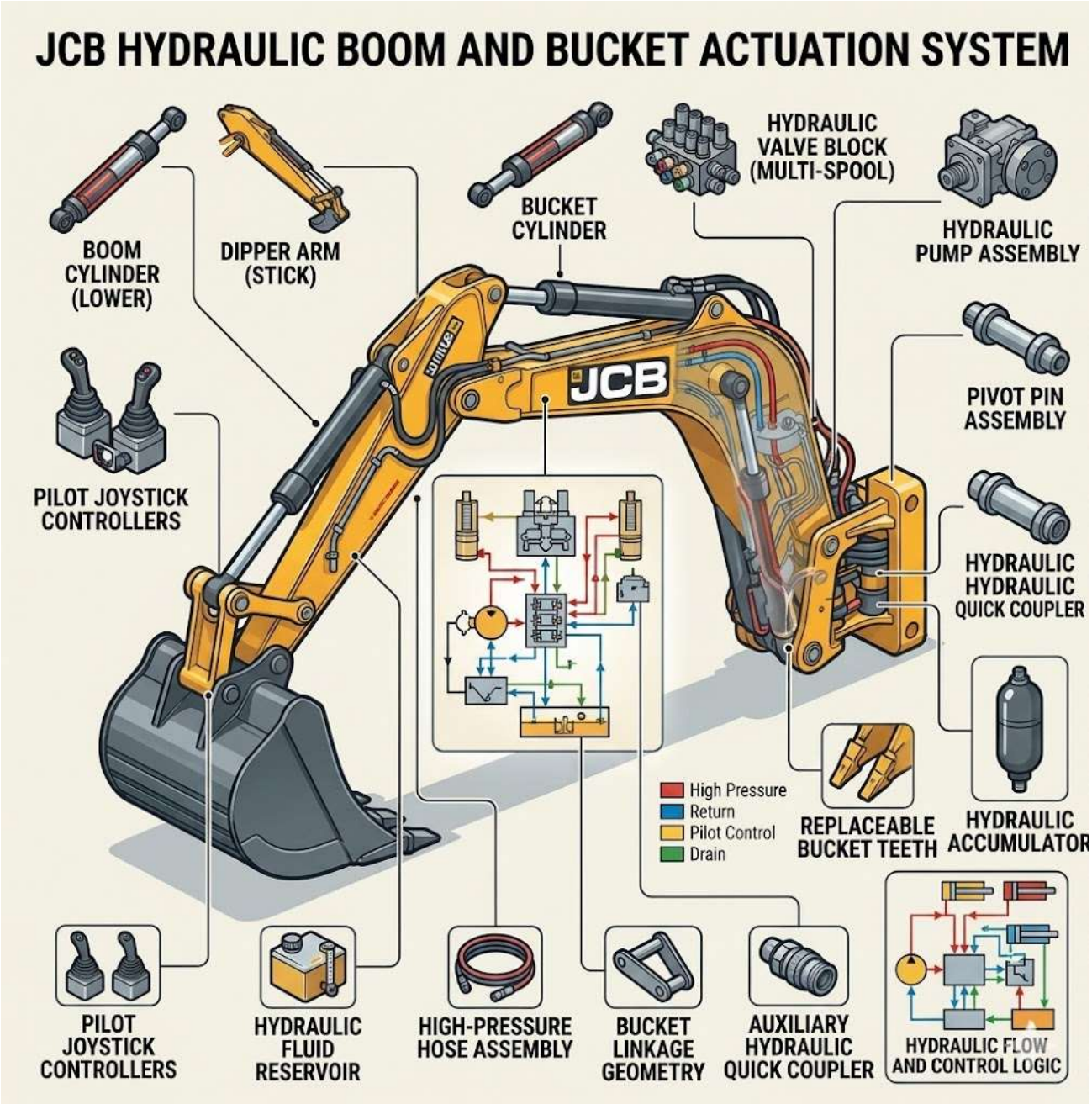

[Figure 4.4: inner working of injection molding ]

## 4.4 Construction & Mining

- Application: Large earth moving equipment, focusing on hydraulic jacks and cylinders found in LHD trucks and excavators.
- Approach to Modeling: For this application, one might consider using the Weibull PHM technique for modeling the RUL of these machines, since it incorporates both the inherent failure rate and other covariates such as the quality of hydraulic oil, operating temperatures, level of dust, and the competence of the machine operators in determining survival. Another possibility is to develop an IoT-based expert system through artificial intelligence to compare current sensor outputs with historical fault codes and perform Pareto analysis for fault diagnosis.
- Significance at Present Time: With the high initial cost involved in acquiring contemporary mining equipment and rising mineral prices, unscheduled maintenance can prove costly for organizations. However, many mining firms still apply an old break-fix maintenance strategy. The use of predictive models allows for cost-effective scheduling of maintenance without disrupting production schedules.



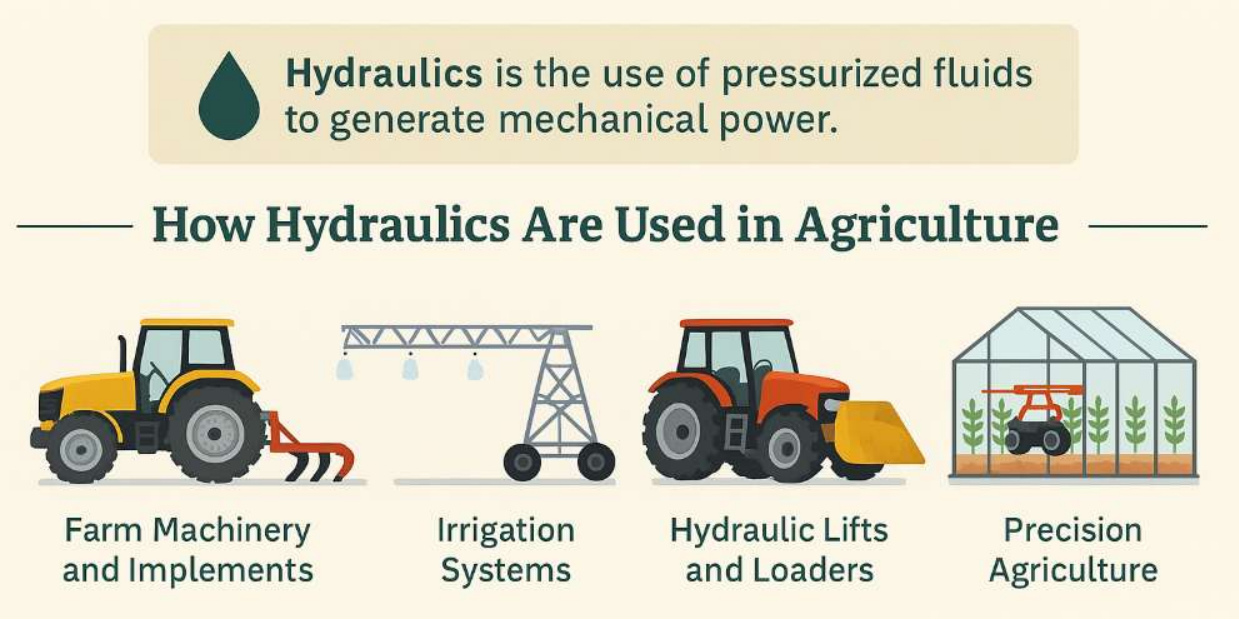

[Figure 4.5 :Heavy Equipment Mechans uses hydraulic system]




[Figure 4.6 :Inner Parts of  Heavy Equipment Mechan ]

## 4.5 Agriculture

- Application: Hydraulics in heavy machinery for agriculture such as tractors, combines, and irrigation booms.
- Approach to modeling: In the field of predictive analysis for agriculture, Internet-of-things sensors are incorporated into hydraulics, which help to track pressure variations, temperature changes, and contamination. The machine learning models help identify patterns based on degradation pathways to issue an alarm when there is an imminent breakdown of the hose assembly and hydraulic pump system.
- Relevance in current time: Agriculture requires seasonal work, and any hydraulic failure in the tractor, for example, during critical periods like planting or harvesting, can cause prolonged interruption leading to crop failure.




[Figure 4.6 Heavy Equipment Mechan ]

# Chapter 5

# RESULTS AND DISCUSSIONS

## 5.1 RESULTS:

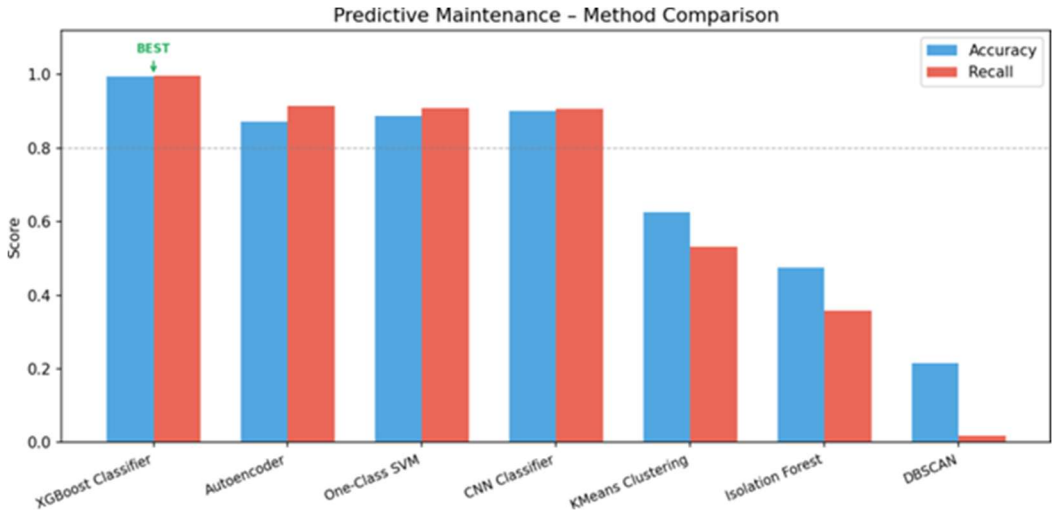
Pipeline analysis of different models used in anomaly detection and fault classification, apart from remaining useful life estimation, revealed several important results, which are as follows:

- Best classifier: The model used for classification purposes proved to be an XGBoost classifier that attained the highest accuracy and recall values of 0.9947 and 0.9967, respectively. It implies that the chosen classifier was highly precise in identifying faults and had an exceptionally high probability of detecting all the faults..

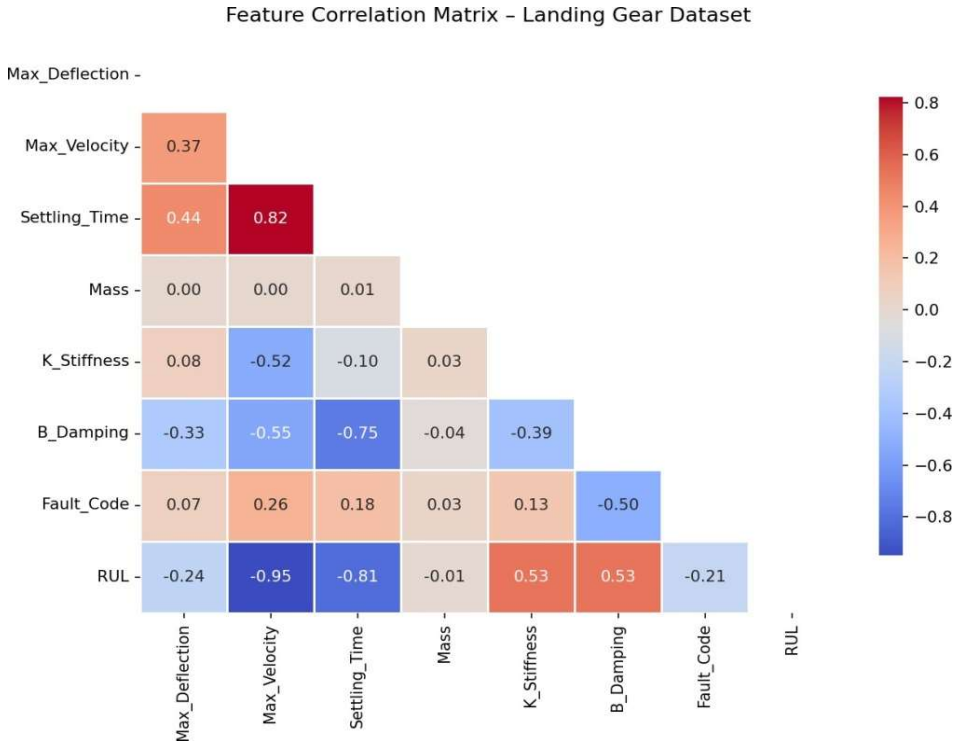
- Other effective classifiers: Other good-performing models include the Autoencoder, CNN Classifier, and One-Class SVM with high accuracy and recall scores exceeding 0.88 and 0.90, respectively.

- Accuracy of RUL predictions: The XGBoost-based regression model showed an accuracy measured in MAE of 1.15 hours. Such results mean that estimates of remaining useful life of components can be considered to be close enough to reality – on average within one hour, which is essential for maintenance activities.

- Visual analysis: Insights from the correlation heat map about relations between variables, as well as clusters of different faults based on the PCA scatter plot, were gained during visual analysis.




[Figure 5.1 Comparison of method with score]




[Figure 5.2 Comparison of Max_Deflection with Remaining useable life]

## 5.2 DISCUSSION

### 5.2.1.Isolation forest

**What it does and why it is used:** Isolation Forest is an algorithm for machine learning used for anomaly detection. Unlike methods such as density-based and distance-based anomaly detection algorithms, the isolation forest uses isolation to find anomalies instead of profiling the 'normal' instances. The approach used here involves choosing a random feature together with splitting the same between its minimum and maximum values. Instances that require fewer splits before isolation can be classified as anomalies. Isolation forests are applied in detecting 'faulty' instances through fault binary classification.

**How it works (implementation):** An instance of the IsolationForest model is created and trained on the entire scaled dataset. It then predicts anomalies within the data. The algorithm labels anomalies as -1 and normal points as +1, which are then converted to 1 for anomalies and 0 for normal data points.

**Steps:**

- Initialize Isolation Forest with a contamination estimate and number of estimators.
- Train the model using the preprocessed feature data.
- Predict anomaly labels for all data points.
- Convert labels (-1 for anomaly, 1 for normal) into binary fault indicators (1 for fault, 0 for normal).

**Limitations:**

• Sensitivity to the contamination parameter, which requires a prior estimate of the proportion of outliers.

• May struggle with anomalies that are not distinct from normal data points (e.g., if anomalies lie within dense clusters).

• Can be less effective in very high-dimensional data where the concept of 'isolation' becomes less clear.

**Parameter Procedure:**

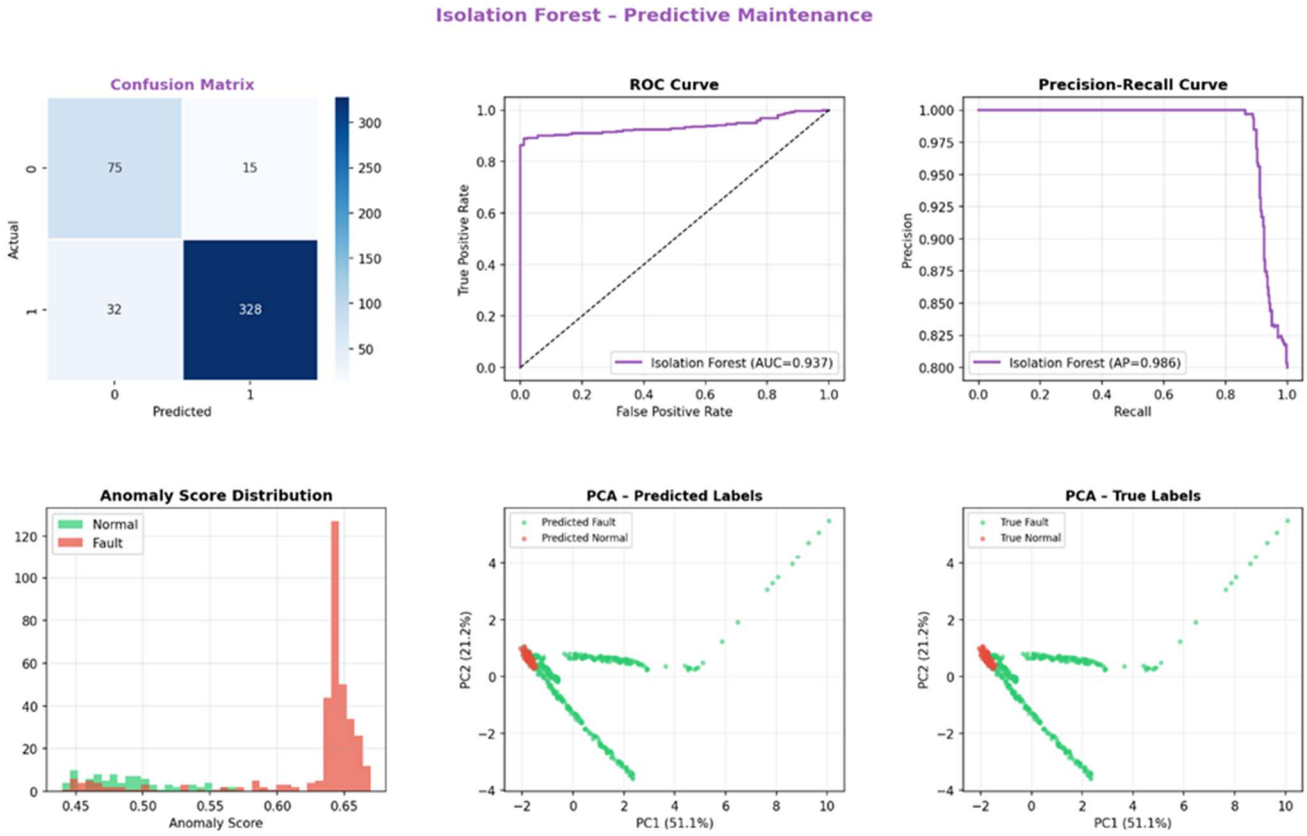
contamination=self.ANOMALY_CONTAMINATION (value set to 0.30): This is the anticipated proportion of outliers in the dataset. This is an important hyperparameter, which may be derived either from user's insight or cross validation. In our pipeline, it is set globally.

Number of estimators = 200: This refers to the number of trees present in the entire ensemble. Higher values here mean increased stability but lower efficiency. 200 is chosen as the standard value as it has been used traditionally in the industry.

random state=self.RANDOM_STATE (value set to 42): This is the random seed used to ensure consistent random splitting of trees

,


[Figure5.3 Method 1 flow chart ]




[Figure5.4 Result from 1st Method ]

### 5.2.2.One-Class SVM

• **What it does and why we use it:** OCSVM stands for One Class Support Vector Machine, which is an unsupervised classifier approach. OCSVM learns a boundary where any point lying inside is classified as normal, whereas any point lying outside is considered an outlier. This research uses OCSVM to detect defective units by using only normal data samples for training purposes.

• **How we use it (Implementation):** Our OneClassSVM classifier model is trained using X_normal_samples, i.e., scaled data points that have Fault_Code = 0 (Normal condition). The model prediction is done on the whole scaled data points self.X_scaled. Prediction = -1 indicates anomalies/faults; whereas, prediction = +1 indicates normal data points..

• **Steps**:

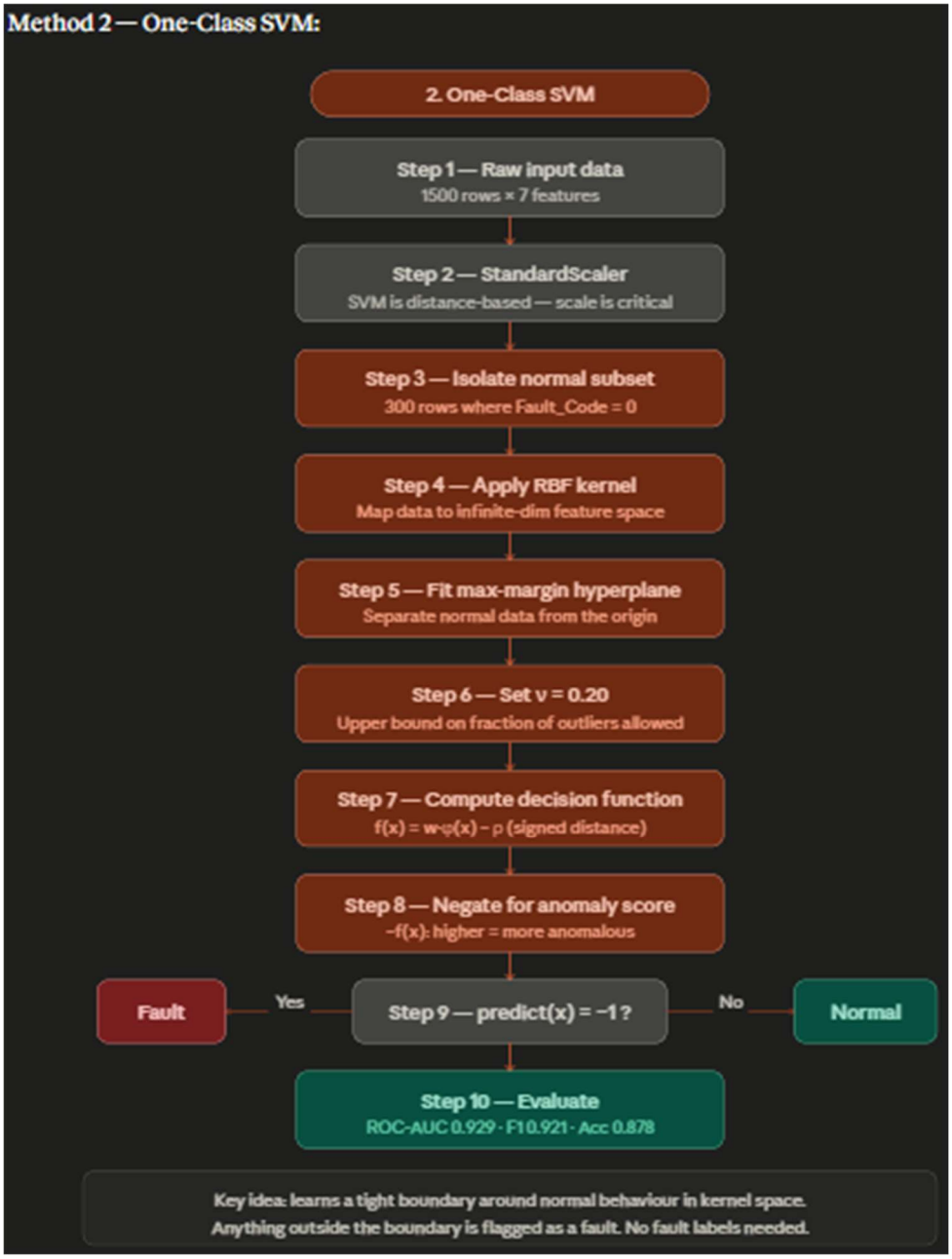
1. Extract normal samples from the scaled dataset.
2. Initialize OneClassSVM with parameters like nu and kernel.
3. Fit the model only on the normal samples.
4. Predict anomaly labels for all data points in the full dataset.
5. Convert labels (-1 for anomaly, 1 for normal) into binary fault indicators (1 for fault, 0 for normal).

- **Limitations:**

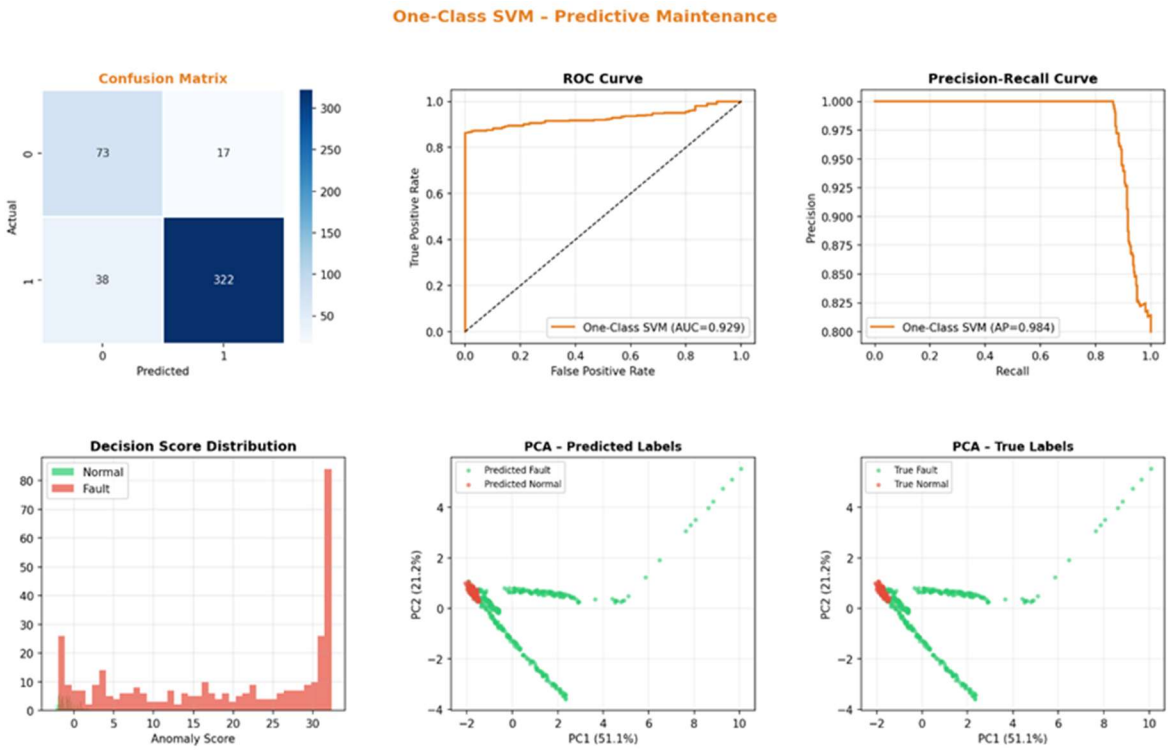
  - Sensitive to the parameter nu, which controls the maximum allowed training errors and the minimum number of support vectors. It functions similarly to how contamination is handled in Isolation Forest.
  - It can be slow and time-consuming with very large datasets.
  - Its performance largely depends on the choice of kernel and parameter settings, such as gamma.
  - It assumes outliers are rare and significantly different from the normal data.

- **Parameter Procedure:**

  - nu=0.20: This is used as an upper limit of the error made in the training process. This paper also offers a lower limit of the support vectors that is specified as a hyperparameter and has to be adjusted in relation to the expected anomaly rate. Here, the limit is placed at 20%.
  - kernel = "rbf": RBF Kernel is a popular kernel used for its ability to account for non-linearity. It works well in cases where the decision boundary cannot be separated using a straight line.
  - gamma = "scale": Gamma represents the impact of a single sample. In case gamma = "scale", its value will be equal to 1 / (n_features * X.var()) which is a good default choice.




[Figure5.5 Method 2 flow chart ]



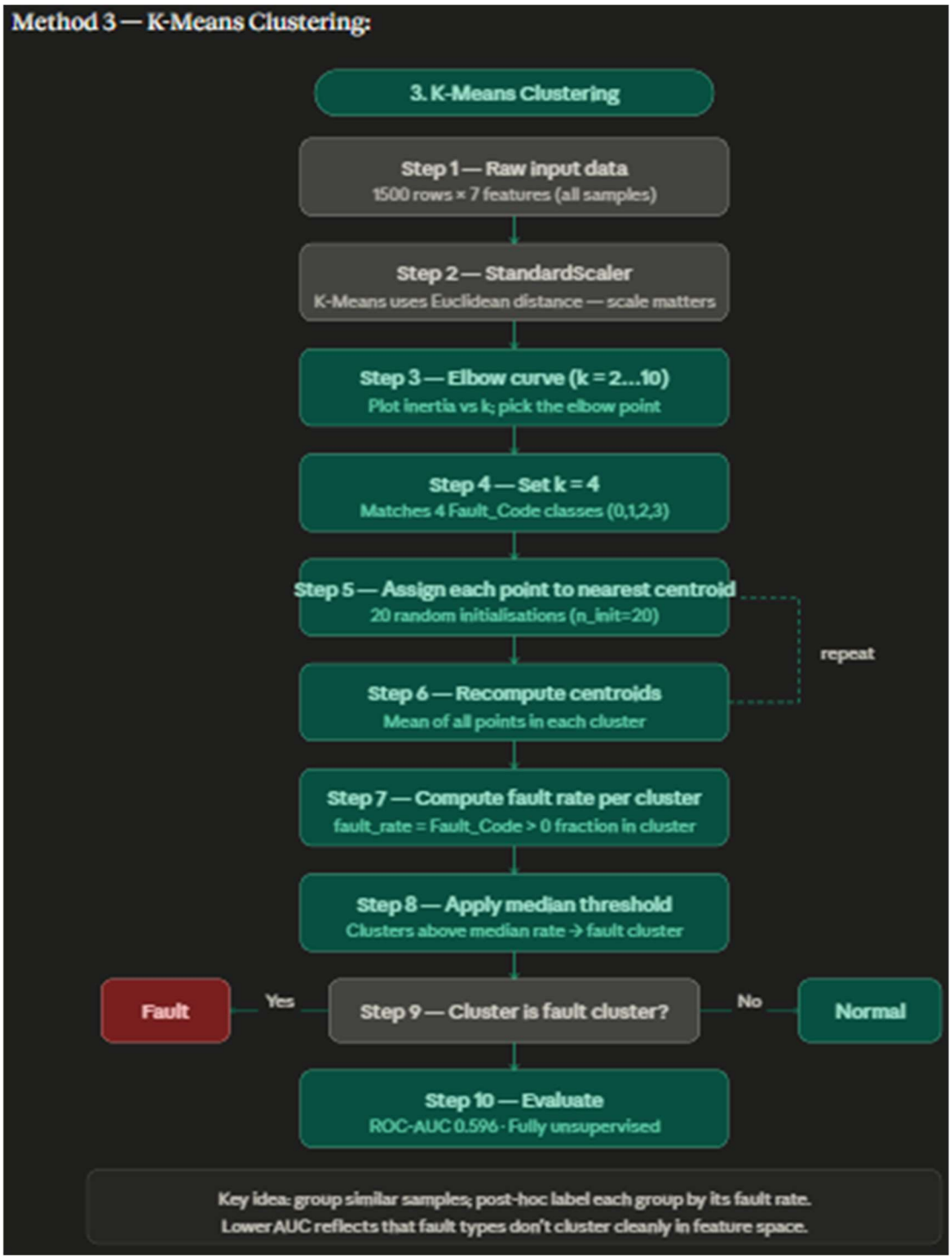

[Figure5.6 Result from 2st Method ]

### 5.2.3. K-Means Clustering

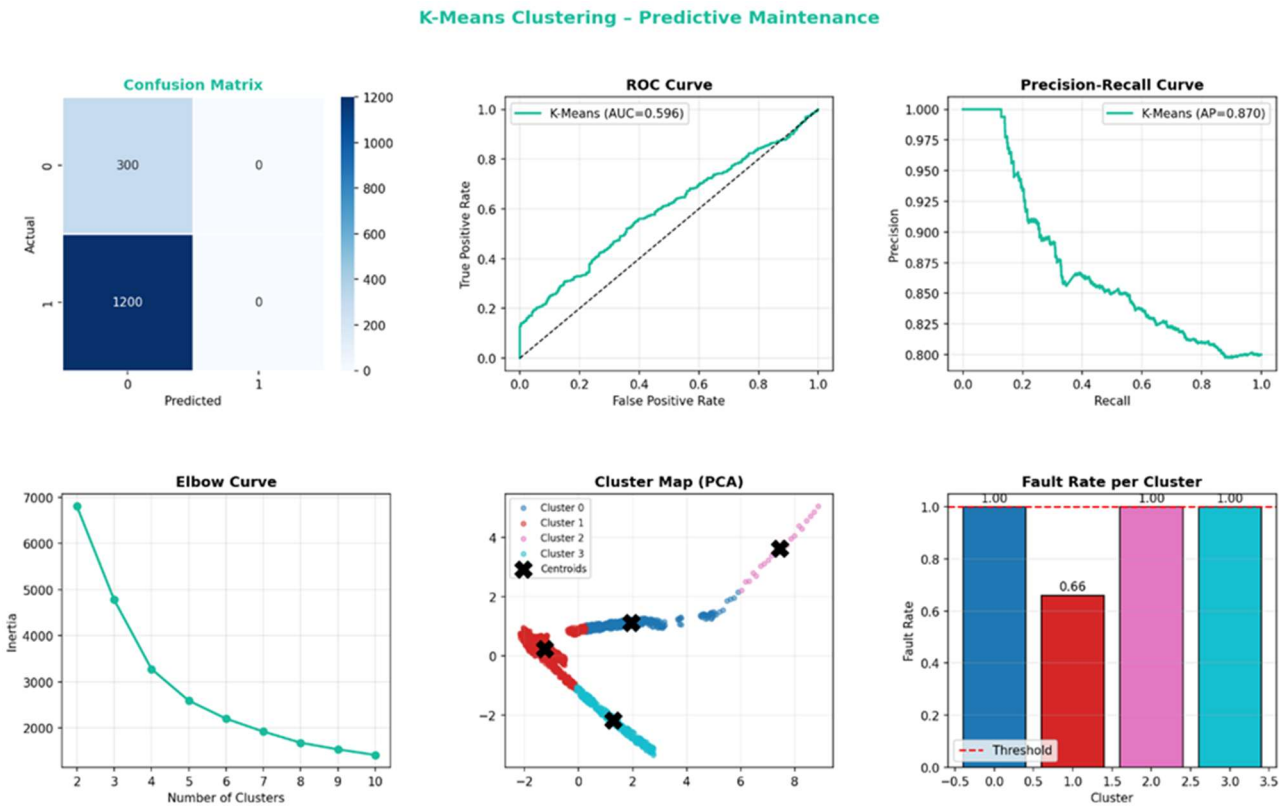
- **Objective & Rationale:** The K-Means algorithm is the centroid-based clustering technique used to divide n data points into k clusters where each data point is assigned to a cluster with its closest centroid. Regarding anomaly detection, the modified K-Means algorithm operates based on the assumption that there would be more clusters for normal data compared to anomalous data clusters because anomalies create small clusters. The cluster that represents "normal data" is that with the lowest percentage of faults.
- **Implementation:** First of all, a KMeans object is initialized and the fitting process is executed on the scaled dataset (self.X_scaled). And once the clustering process is done, the cluster is selected as a 'normal' one by computing the average of the binary labels of faults for the clusters (is_fault). The cluster with the lowest proportion of faults is considered as a 'normal' one. It means that all remaining data points from different clusters are faults (anomalies).
- **Steps:**
    1. KMeans initialization with the number of clusters (n_clusters).
    2. Model training process on the preprocessed features.
    3. Assignation of data points to clusters.
    4. Compute fault proportion for each cluster according to the actual y labels (self.y_binary).

5. Determine the 'normal' cluster with the minimal fault proportion.
6. Classify data points in clusters other than 'normal' as faulty units.

- **Limitations:**
    - Eps and min_samples are sensitive to the selection; they may be difficult to choose in datasets that have different densities.
    - Has difficulty with data sets with very different density of clusters.
    - Does not have the ability to effectively deal with high-dimensional data because density is less meaningful.
    - Doesn't provide a continuous anomaly score, assigns all points to a cluster or noise.
- **Parameter Procedure:**
    - n_clusters = 4: Chosen in consideration of the number of fault types (including the normal one), with the underlying idea being that each fault type, along with the normal operation, forms a cluster in the feature space. This is a heuristic choice, which might be improved.
    - eps=1.5: maximal distance between two samples for one to be said to be in the neighbourhood of the other. This is very important parameter for determining the density of clusters. It usually needs delicate tuning, which may be achieved using techniques such as a k-distance graph.




[Figure5.7 Method 3 flow chart]




[Figure5.8 Result from 3st Method]

### 5.2.4. DBSCAN

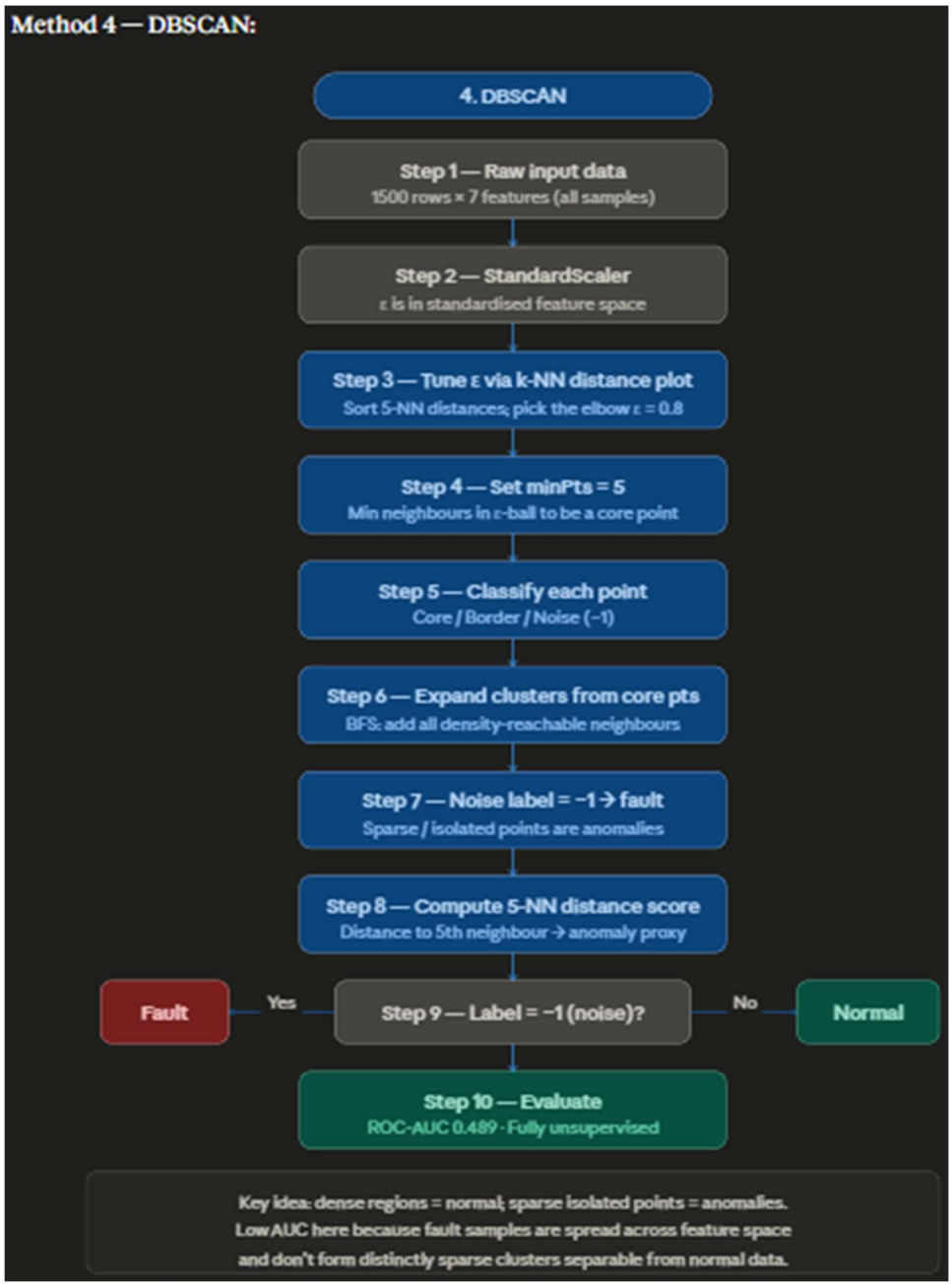
- **What it does and why we use it:** DBSCAN, Density Based Spatial Clustering of Application with Noise, is one such algorithm that is based on density measures. This algorithm combines points present in densely occupied regions (that is, points having a higher number of neighbors), and classifies points present in sparsely occupied regions as noise. In the context of this research, the choice of DBSCAN is made because this algorithm has the inherent ability to detect noise points.

- **How we use it (Implementation):** The DBSCAN model is initialized and fitted on the entire scaled dataset (self.X_scaled). DBSCAN assigns a cluster label to each point, with -1 indicating noise (anomalies). We further refine this by identifying the largest non-noise cluster as the 'normal' cluster. Any point that is either noise (-1) or belongs to a different (smaller) cluster is classified as an anomaly (fault).

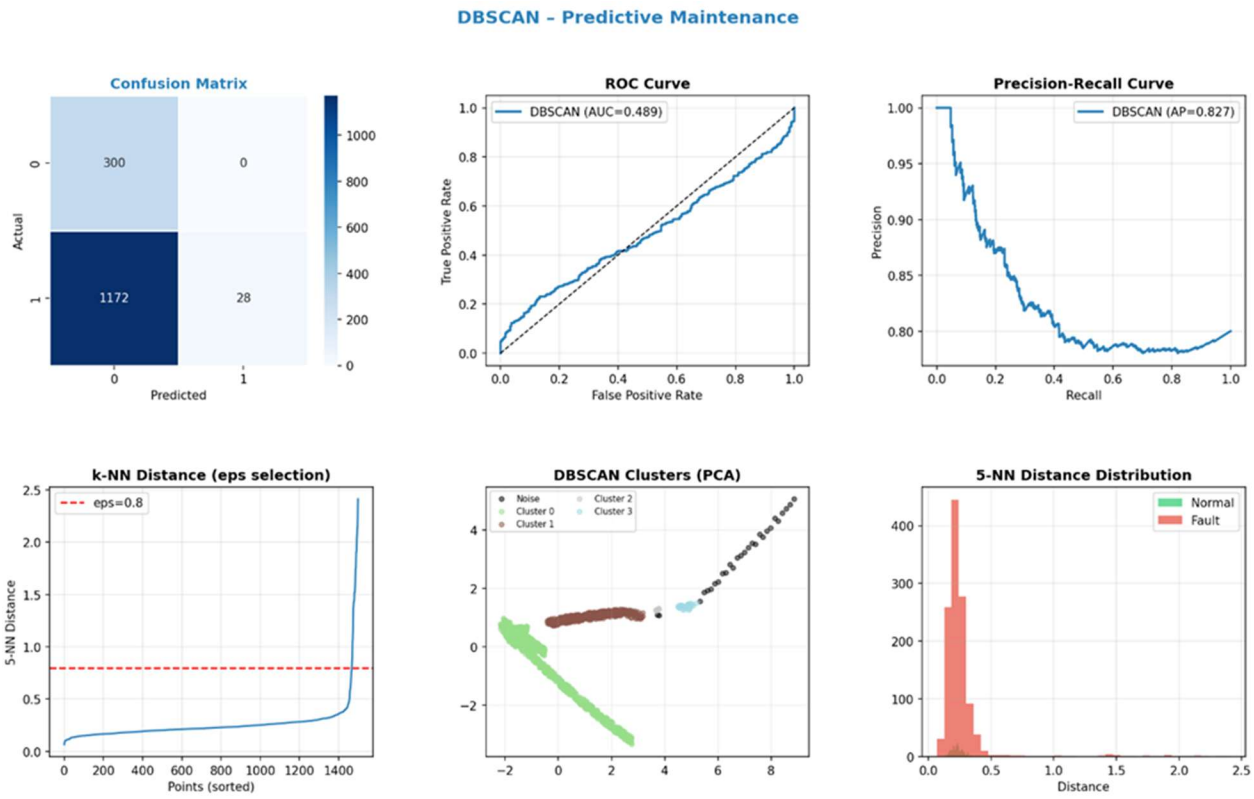
- **Steps:**

    1. The DBSCAN clustering algorithm is initialized by specifying eps (distance that defines the threshold of being neighbors of the two data points) and min_samples (the minimum number of data points required within eps-distance of any point to classify it as a core point).
    2. Fit  model on the preprocessed feature data.
    3. Identify noise points (labeled -1) and other smaller clusters as anomalies.
    4. The largest non-noise cluster is typically designated as the 'normal' cluster.
    5. Classify all data points *not* in the normal cluster (or noise points) as faults

- Limitations:
    - Eps and min_samples are sensitive to the selection; they may be difficult to choose in datasets that have different densities.
    - Struggles with datasets where clusters have significantly different densities.
    - Does not have the ability to effectively deal with high-dimensional data because density is less meaningful.
    - Doesn't provide a continuous anomaly score, assigns all points to a cluster or noise.
- **Parameter Procedure:**
    - eps=1.5: maximal distance between two samples for one to be said to be in the neighbourhood of the other. This is very important parameter for determining the density of clusters. It usually needs delicate tuning, which may be achieved using techniques such as a k-distance graph.




[Figure5.9 Method 4 flow chart]




[Figure5.10 Result from 4st Method]

### 5.2.5. Autoencoder

- **What it does and why we use it:** An Autoencoder is a type of neural network used for unsupervised learning of efficient data codings (dimensionality reduction). It's trained to reconstruct its input data. In the case of anomaly detection, it is assumed that the Autoencoder is trained with "normal" data, so that it can reconstruct the normal patterns very well, with a low reconstruction error. It will find it harder to reconstruct such data if presented with anomalous data and will have a high reconstruction error. We use it to detect anomalies on this principle.

- **How we use it (Implementation):** A sequential Autoencoder model is created that comprises of an encoder (which is the input to the bottleneck layer) and a decoder (which is the bottleneck to the output layer). The model is compiled using adam optimizer and mean_squared_error loss. Most importantly, the Autoencoder is only trained with the scaled features of a normal operation (X_normal_samples). Once trained, the reconstruction error is assessed for all the scaled data. A threshold is calculated from the reconstruction error of the

normal data set and data points exceeding this threshold will be identified as anomalies.

- **Steps:**

    1. Remove normal samples from the scaled data set.
    2. Create the Autoencoder architecture of the neural network (encoder/decoder)
    3. Train the Autoencoder using optimizer and loss function (adam and mse).
    4. Use the Autoencoder to train only on the normal samples.
    5. Calculate the reconstruction error for all data points in the full dataset.
    6. Determine an anomaly threshold (e.g., using a percentile of reconstruction errors from normal data).
    7. Identify data points with reconstruction error exceeding threshold as a fault .

- **Limitations:**

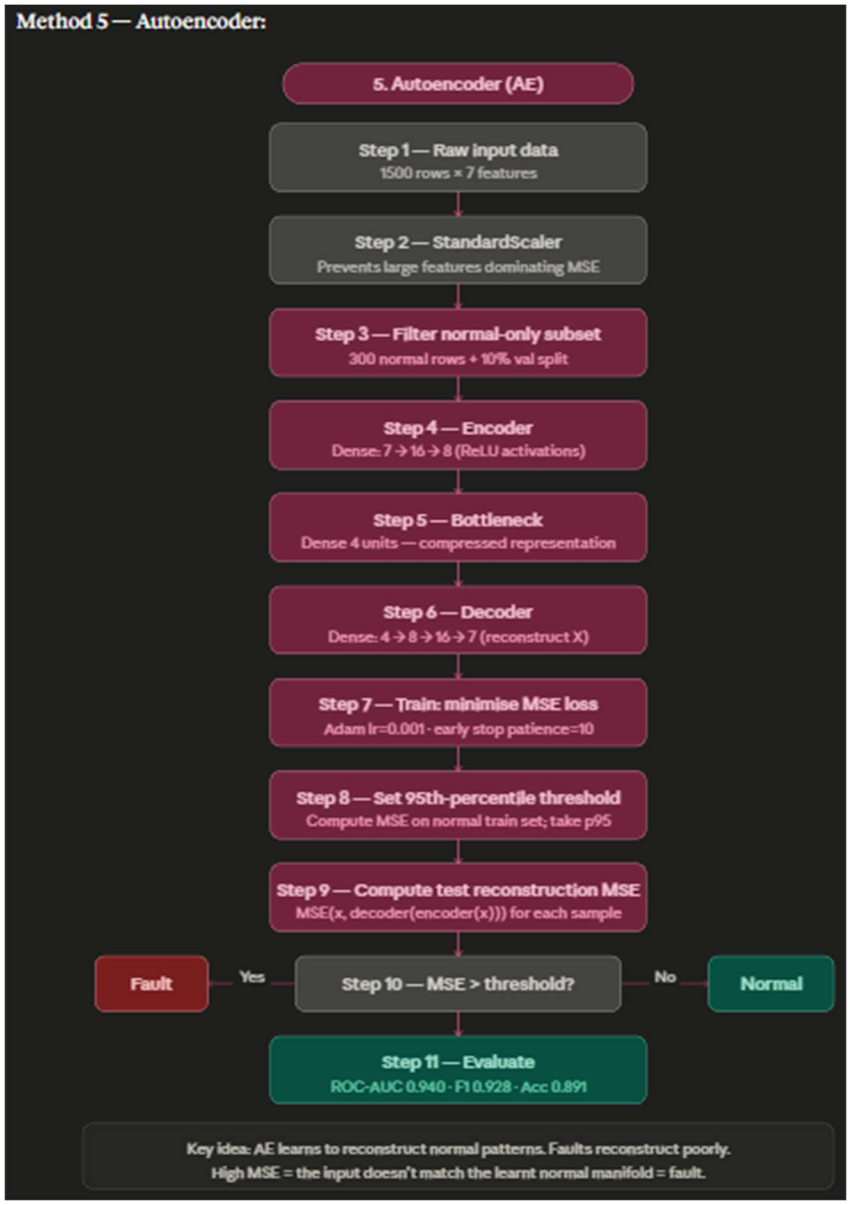
    - Network architecture (number of layers, number of neurons, activation functions) has a significant impact on the performance).
    - Needs tuning of hyperparameters such as number of epochs, batch size, and early stopping patience.
    - Can sometimes reconstruct anomalies well if they are similar to normal patterns or if the network is too complex (overfitting).
    - The choice of reconstruction error threshold can be subjective and impact results significantly.

- **Parameter Procedure:**

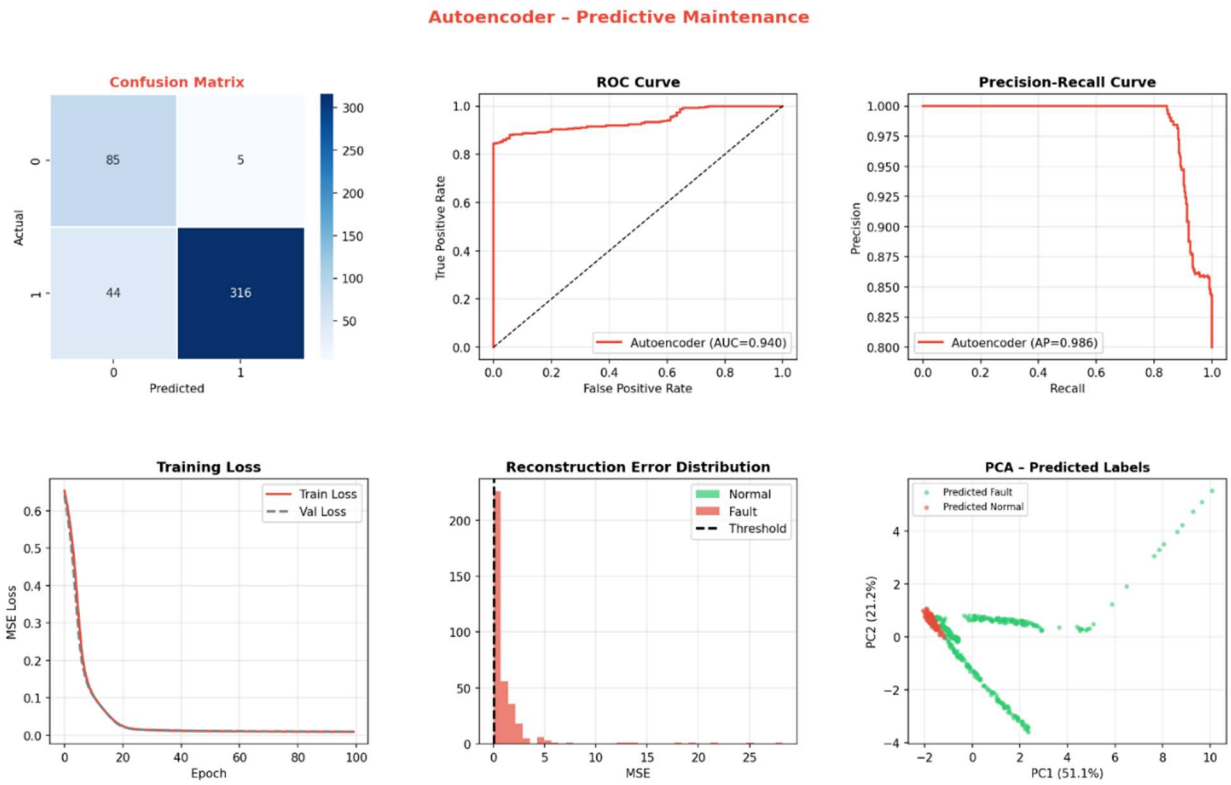
    - INITIATIVE: A number that defines how many times the program will try to fit the data. Training is halted early when the loss on validation does not get better.
    - BATCH_SIZE=32: Number of samples used to calculate a gradient update. A popular

option that is both quick to train and stable.

- To avoid overfitting, training will terminate if the loss on the validation set does not get better after 10 epochs (when patience=10 in EarlyStopping.
- Network Architecture: A simple symmetric network (16-8-4-8-16) is used, where relu is used as the activation for the hidden layers and linear for the output. It is a common way to start Autoencoders, trying to get a bottleneck layer which extracts the most important features.




[Figure5.11 Method 5 flow chart]




[Figure5.12 Result from 5st Method]

### 5.2.6. 1-D CNN Fault Classifier

- **What it does and why it is used:** One-Dimensional Convolutional Neural Networks (1D CNNs) are particularly well-suited for handling sequences where the ordering or closeness of features is significant. In our case, although the input itself is merely a feature vector, the 1D CNN can still capture local as well as hierarchical relationships within the data. The model serves as a supervised classifier that predicts the binary fault status (is_fault) using the input features.

- **How it is implemented:** Input features (self.X_scaled) are reshaped in such a way that they become suitable for the architecture of 1D CNN. Each feature vector is considered as a sequence of length 1 and having n_feat channels. The architecture consists of convolution layers, dropout layer, global average pooling layer, and dense layers along with sigmoid function at the end. Training is done based on the split (X_tr, yb_tr), where yb_tr is the binary fault variable. Predictions are made for test data (X_te_cnn), and then they are converted to binary output by applying a probability threshold of 0.5.

- **Steps:**

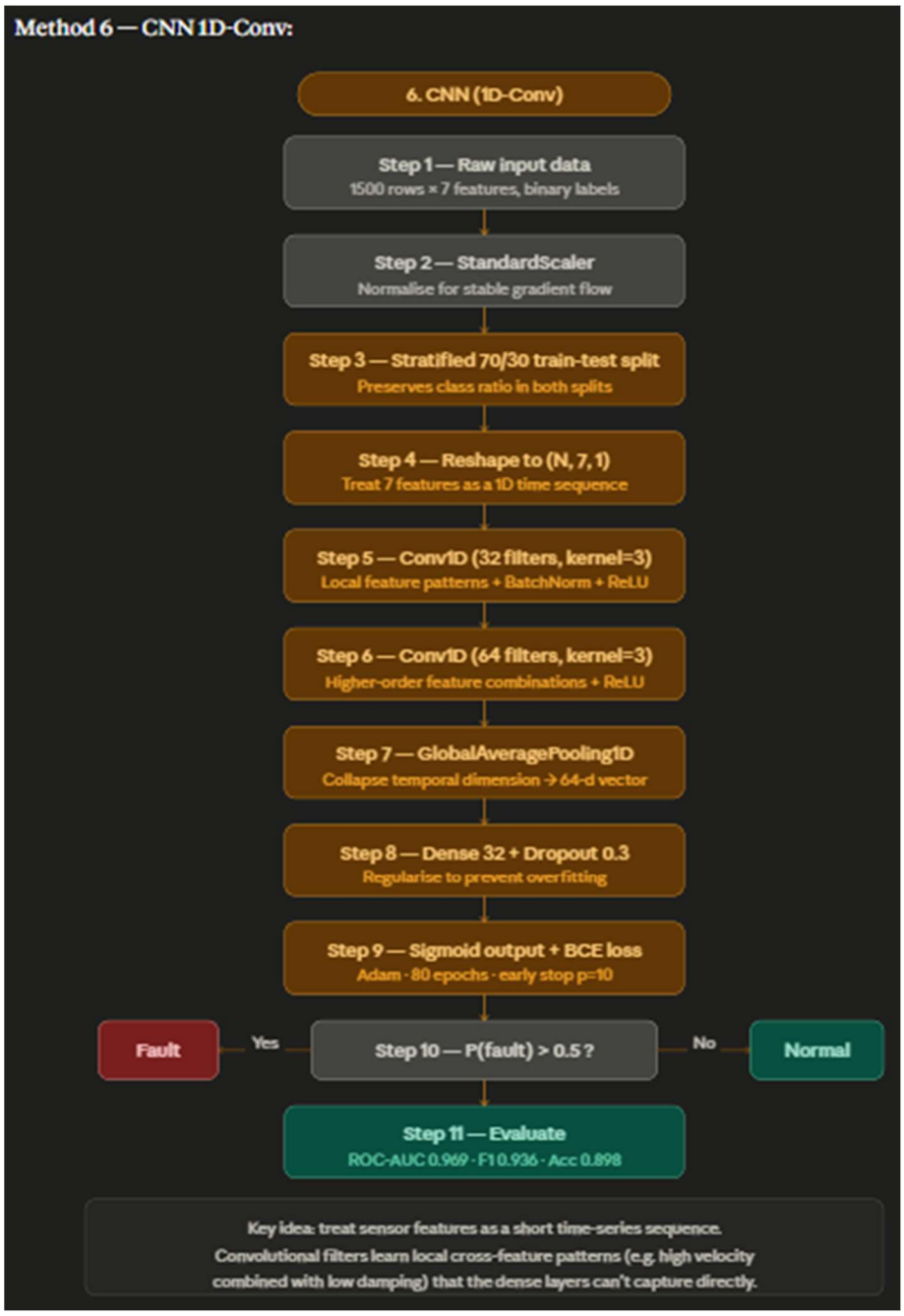
    1. Reshape the scaled features for CNN input (e.g., (samples, 1, features)).
    2. Design a 1D CNN architecture with convolutional, pooling, and dense layers.
    3. Compile the CNN with an optimizer, binary cross-entropy loss, and accuracy metric.
    4. Train the CNN on the training set (X_tr_cnn, yb_tr).
    5. Predict probabilities on the test set (X_te_cnn).
    6. Convert probabilities to binary class predictions (0 or 1) using a threshold ,(e.g., 0.5)

- **Limitations:**

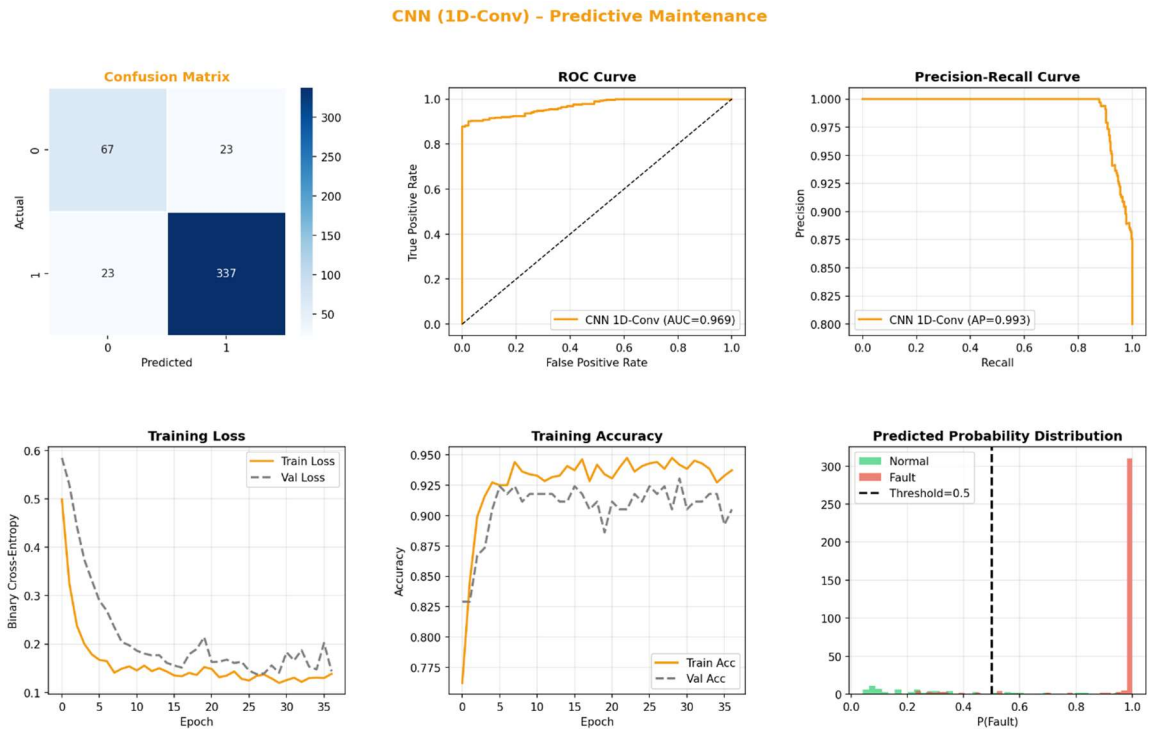
    - Requires sufficient labeled data for effective training.
    - Performance depends heavily on architecture design and hyperparameter tuning (e.g., number of filters, kernel size, learning rate).
    - Reshaping a flat feature vector into a 1D 'sequence' might not always capture meaningful sequential patterns if the features are independent.
    - Can be computationally intensive to train.

- **Parameter Procedure:**

    - CNN_EPOCHS=40: The maximum number of epochs for training. Training stops early if there is no improvement in validation loss.
    - EPOCHS=200: The number of times to repeat the entire process.
    - patience=8 (for EarlyStopping): Training will be interrupted if the validation loss doesn't decrease for 8 epochs.
    - Network Architecture: It is composed of Conv1D layers with the activation function relu, Dropout layer for regularization, GlobalAveragePooling1D, Dense layers. For binary classification the final Dense layer is using a sigmoid activation.




[]Figure5.13  Method 6 flow chart]




[Figure5.14 Result from 6st Method]

### 5.2.7. XGBoost (Multi-Class Fault Classifier)

- **What it does**: XGBoost, also known as Extreme Gradient Boosting, is an extremely fast and powerful open source library for implementing gradient boosting machines. It's well-known for its performance and speed when working with structured data tasks. For this purpose, we are applying XGBoost as a supervised classifier and trying to predict the fault type (multi-class classification: Normal, Fault-1, Fault-2, Fault-3), given the input features.

- **How we use it (Implementation):** An XGBClassifier is initialized and trained on the (X_tr, yf_tr) split, where yf_tr is the multi-class fault labels (0, 1, 2, 3). The model is trained on differentiating the types of faults. The predictions are then made on the test set (X_te) to get xgb_fault_pred, the predicted fault codes. These multi-class predictions are then further transformed into binary (0 if predicted Normal, 1 otherwise) predictions to be compared with the binary anomaly detection models.

- **Steps:**

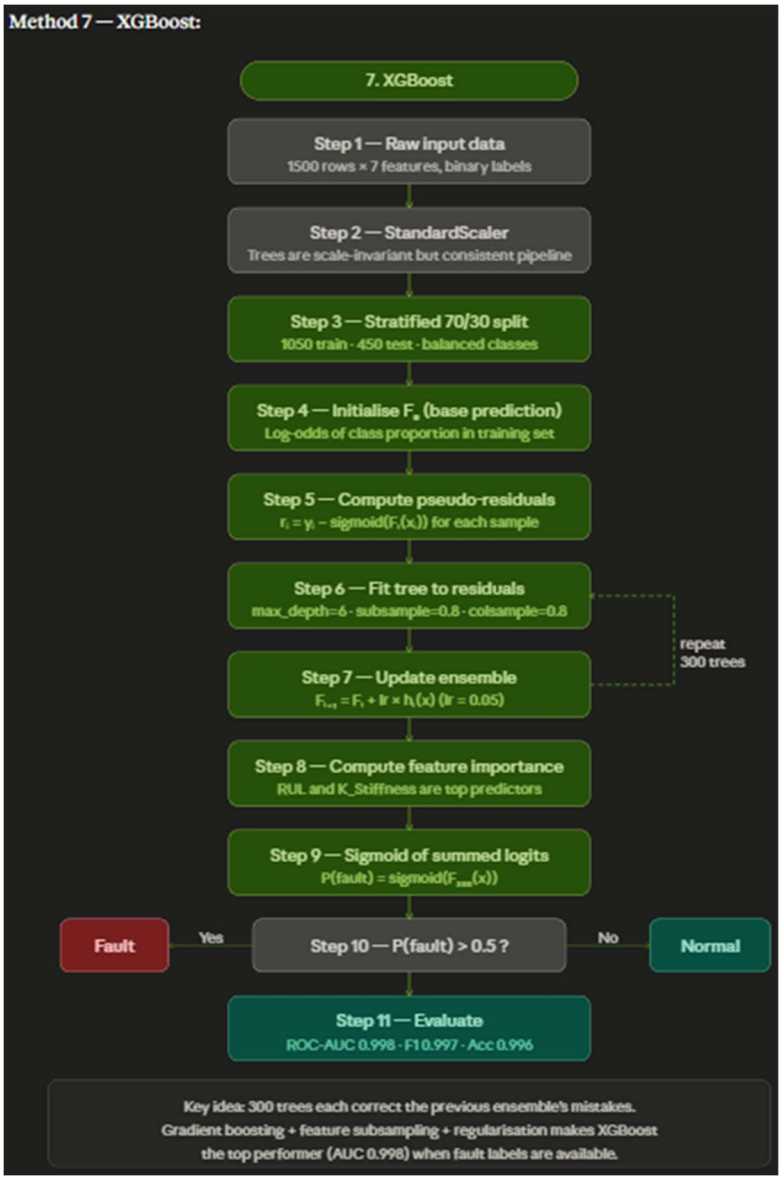
    1. Initialize XGBClassifier with different hyperparameters such as n_estimators,

max_depth, learning_rate etc).

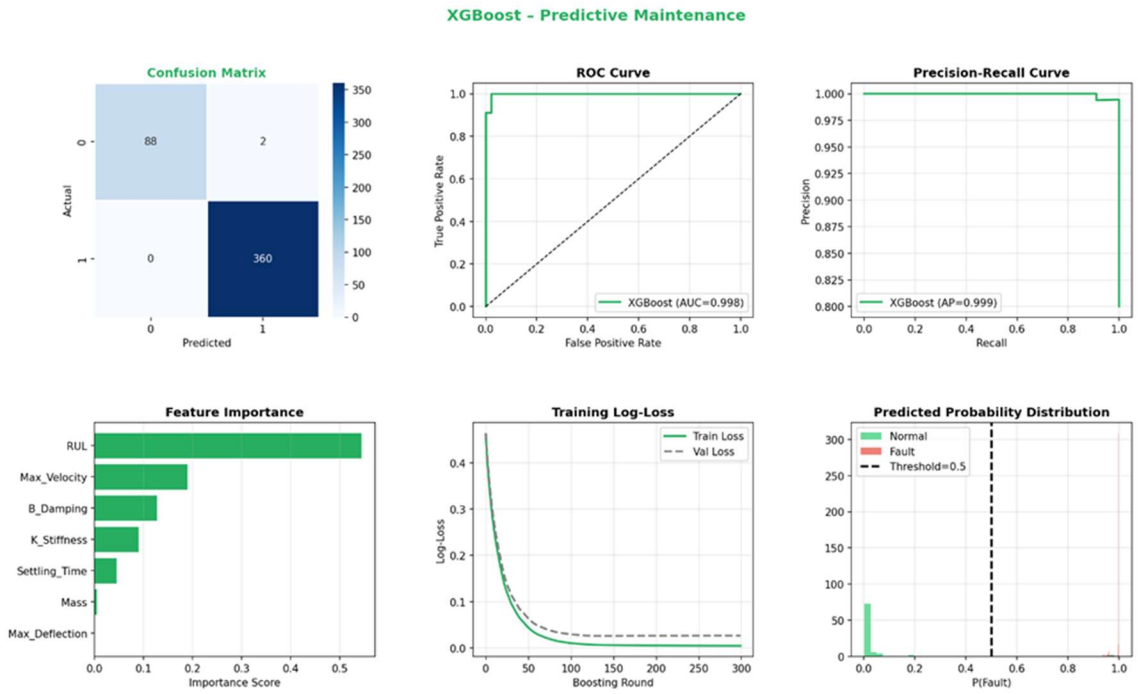
2. Apply the trainedClassifier to the test features (X_te), and multi-class fault labels (yf_te).
3. Predict value of the scaled test features (X_te) for multiple classes).
4. Convert these multi-class predictions into binary (fault/no-fault) for comparison with other anomaly detection methods

- **Limitations:**
  - Can be prone to overfitting if hyperparameters are not tuned carefully.
  - Requires significant memory and computational resources for very large datasets.
  - Vulnerable to input features' quality and size.
  - The task of interpretability may be difficult in complex models.
- **Parameter Procedure:**
  - n_estimators: Number of boosting rounds (number of trees) to build. The more the estimators the better the performance but the more computation is required and risk of overfitting.
  - max_depth=5: The maximum depth of a tree. Controls complexity and interaction order.
  - learning_rate=0.05: This is the step size that is shrunk to avoid overfitting. The smaller the better, because this makes the model less prone to overfitting and it needs more boosting rounds
  - subsample=0.8: Percentage of samples to be used for building the trees. Reduces variance.
  - colsample_bytree=0.8: Percentage of the features to be sampled for building the trees, Reduces variance.

- use_label_encoder=False: Silences a deprecation warning (later versions have a built-in label encoding system).
- evaluator_metric="mlogloss": The metric to be evaluated in multi-class classification.
- random_state=self.RANDOM_STATE (42): For reproducibility.




[Figure5.15 Methods 7 flow chart]




[Figure5.16 Result from 7st Method]

### 5.2.8. XGBoost (RUL Regression)

- **What it does and the reasons for using it:** In addition to its use in classification tasks, XGBoost can be considered very good at solving regression problems. In this case, it helps predict the Remaining Useful Life (RUL), which is a numerical feature indicating the amount of time in hours the equipment can operate until it breaks down. Predicting RUL is a crucial step when it comes to preventive maintenance.

- **How it works:** An XGBRegressor model is fit using the training part (X_tr, yr_tr) – the latter stands for RUL values which are numeric in nature. Afterward, predictions are made based on the test part (X_te) and the resulting predicted RUL values are saved as rul_pred. The evaluation metric in this task will be MAE.

- **Steps:**

    1. Initialize XGBRegressor with relevant hyperparameters.

    2. Train the regressor on the scaled training features (X_tr) and continuous RUL labels (yr_tr).

3. Make continuous RUL predictions on the scaled test features (X_te).
4. Evaluate the regression model using metrics like Mean Absolute Error (MAE).

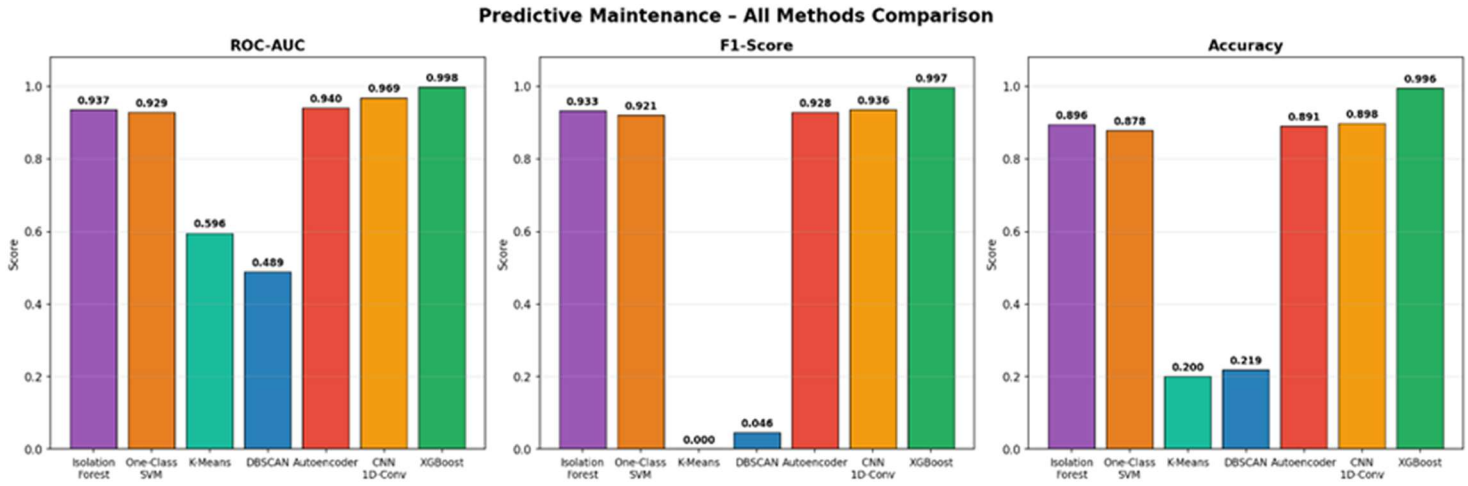
- **Limitations:**
    - Similar to the classifier, it can be prone to overfitting and requires careful hyperparameter tuning.
    - Assumes a clear relationship between input features and RUL that can be learned from the data.
    - May struggle with highly non-linear or noisy RUL relationships.
- **Parameter Procedure:**
    - Learn Random Forest Classifier (n_estimators=300, max_depth=5, learning_rate=0.05, subsample=0.8, colsample_bytree=0.8, random_state=self) RANDOM_STATE: These parameters are analogous to those of the XGB Classifier, and are set to get a strong regression model. For typical regression tasks, the objective is not explicitly provided, but rather is set to 'reg:squarederror' by the XGB Regressor.
    - verbosity=0: Suppresses verbose output during training.

In conclusion we found :-

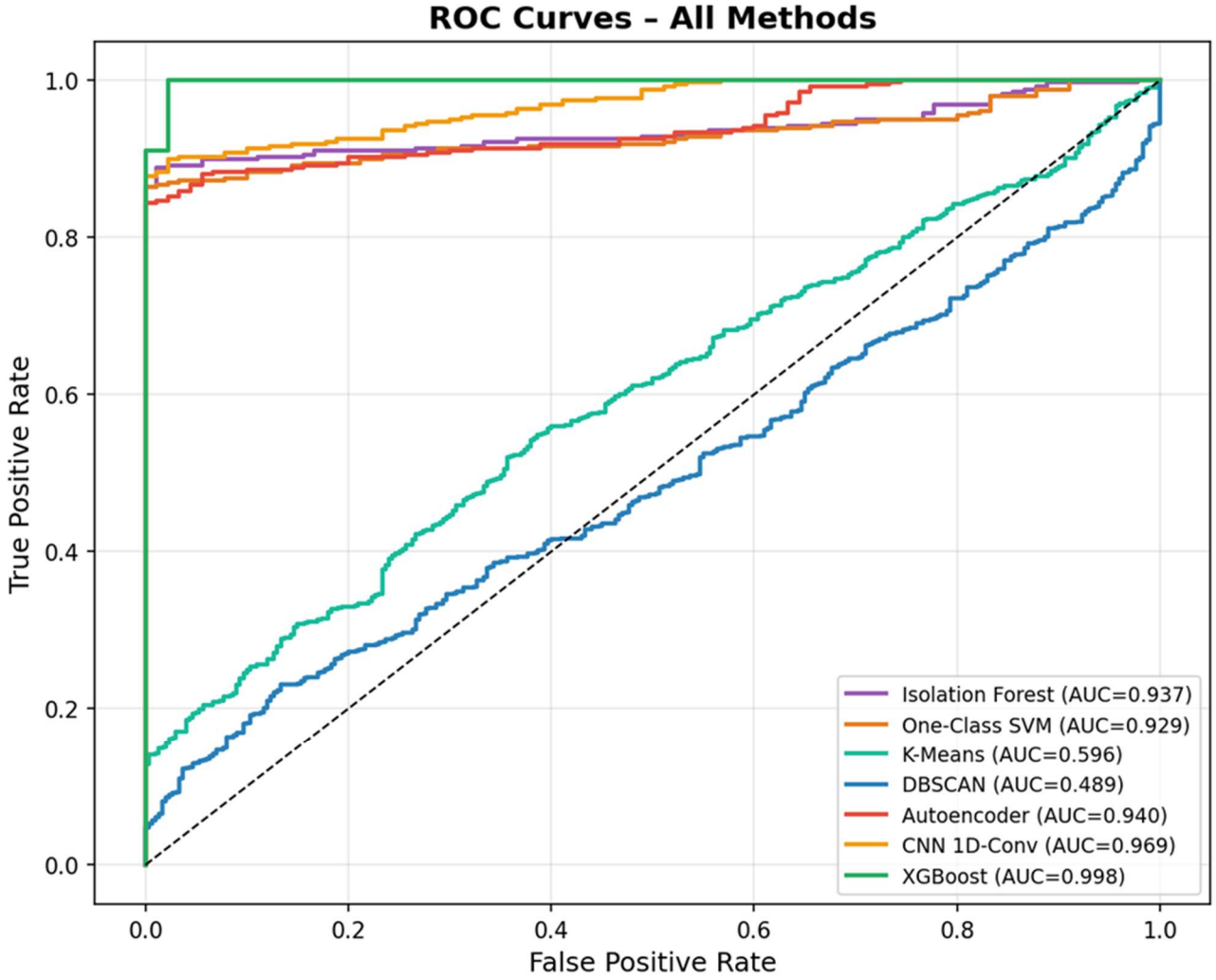
The XGBoost classifier performs very well in terms of fault detection and gives a low mean absolute error (MAE) value for remaining useful life (RUL) prediction. It is also the concept that a selected feature variable, such as Max-Deflection, Max-Velocity, Settling-Time, Mass, K-Stiffness and B-Damping, is quite efficient in determining the health state of the landing gear. Regarding the proposed pipeline, it has found that using the XGBoost Classifier would be the best due to its very high recall value, which is of critical importance for maintenance purposes as well.

Concerning anomaly detection techniques, different methods have been used (Isolation Forest, One-Class SVM, K-Means, DBSCAN, and Autoencoder), and their methodological approaches provide a comprehensive insight into anomaly detection techniques, especially concerning the promising Autoencoder. The capability of the proposed pipeline in predicting the different kinds of faults (Fault-1, Fault-2, Fault-3) is also another advantage.

From a critical point of view, the average predicted RUL of the system can be considered equal to 63.8 hours, while there are 45 units with RUL lower than 20 hours and an average RUL equals to 10.7 hours. As soon as these results are obtained, it means that 45 units should be repaired and especially Fault-Type-1 (deflection).




[Figure5.17 comparisons of all methods with their ROC,F1-Score,Accuracy]




[Figure5.18 All Methods ROC Curves]

# CHAPTER 6

# CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

## 6.1 Conclusion

This paper has managed to bring together mechanical industrial systems and AI in one study. Through incorporation of techniques like feature engineering with unsupervised machine learning including K-means, DBSCAN, Isolation forest, and deep learning or ensembles such as CNN, autoencoders, and XGBoost, we have managed to develop an accurate prediction method. The model successfully predicts the non-linear deterioration behavior of hydraulic pumps, valves, and coolers. In essence, the technique helps in avoiding unpredicted downtimes, cuts out the costs of random preventive maintenance, and increases the life span of the machinery.

## 6.2 Future Scope

This study creates an excellent foundation; however, there exist possibilities to further improve upon. Some areas worth exploring include the utilization of even broader deep learning approaches such as comparison between the existing stacking model approach and the latest models such as Transformers, as well as using hybrid CNN-LSTM models to predict the long-time series relationship. Another possible area is the use of dynamic parameter adjustment through implementation of a self-tuning algorithm to update the weight of the ensemble based on aging of the hydraulic equipment. Finally, a possibility to extend the application scope of the model to be applicable across multiple factories simultaneously through cloud-based computing.

### 6.3 Social Impact

The ultimate goal of Industrial Engineering transcends beyond making the company profitable to the welfare of the society and the environment. Industrial accidents due to failure of hydraulic components normally result in massive oil leakages, environmental pollution, and creation of dangerous working environments in factories. With a proper predictive model in place to predict failures, it eliminates industrial accidents caused by failures. Also, by increasing machinery life span and efficient parts replacement, the technology ensures minimal industrial waste.

# REFERENCES


- Achouch, M., Dimitrova, M., Ziane, K., Karganroudi, S. S., Dhouib, R., Ibrahim, H., and Adda, M. (2022). On predictive maintenance in Industry 4.0: Overview, models and challenges. Applied Sciences, 12(16), 8081.
- Arena, F., Collotta, M., Luca, L., Ruggieri, M., and Termine, F. G. (2021). Predictive Maintenance in the Automotive Sector: A Literature Review. Mathematical and Computational Applications, 27(1), 2.
  https://doi.org/10.3390/mca27010002
- Arrieta, A. B., Diaz-Rodriguez, N., Del Ser, J., Bennetot, A., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges. Information Fusion, 58, 82–115.
- Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.Campos, J., Sharma, P., Jantunen, E., Baglee, D., and Fumagalli, L. (2024). A framework for predictive maintenance in manufacturing: Systematic literature review. Frontiers in Mechanical Engineering, 9, 1083218.
- Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. P., Basto, J. P., and Alcala, S. G. S. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers and Industrial Engineering, 137, 106024.
- Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
- da Silva, R. F., Melani, A. H. A., Michalski, M. A. C., and de Souza, G. F. M. (2023). Reliability and risk centered maintenance: A novel method for supporting maintenance management. Applied Sciences, 13(19), 10605.
  https://doi.org/10.3390/app131910605
- Ferreira, C. and Goncalves, G. (2022). Remaining useful life prediction and challenges: A literature review on ML methods. Journal of Manufacturing Systems, 63, 550–562.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press, Cambridge, MA, USA.
- Hamasha, M. M., Bani-Irshid, A. H., Al Mashaqbeh, S., Shwaheen, G., Al Qadri, L., Shbool, M., Muathen, D., Ababneh, M., Harfoush, S., Albedoor, Q., and Al-Bashir, A. (2023). Strategical selection of maintenance type under different conditions. Scientific Reports, 13.
  https://doi.org/10.1038/s41598-023-42751-5
- Hector, I. and Panjanathan, R. (2024). Predictive maintenance in Industry 4.0: A survey of planning models and machine learning techniques. PeerJ Computer Science, 10, e2016.
  https://doi.org/10.7717/peerj-cs.2016
- Helwig, N., Pignanelli, E., and Schutze, A. (2015). Condition monitoring of a complex hydraulic system using multivariate statistics. Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 210–215.
  DOI: 10.1109/I2MTC.2015.7151267
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
- Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., and Nandi, A. K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. Mechanical Systems and Signal Processing, 138, 106587.

- Liu, F. T., Ting, K. M., and Zhou, Z. H. (2008). Isolation forest. Proceedings of the IEEE International Conference on Data Mining (ICDM), 413–422.
- Lundberg, S. M. and Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4768–4777.
- Mallak, A. and Fathi, M. (2021). Sensor and component fault detection and diagnosis for hydraulic machinery integrating LSTM auto encoder detector and diagnostic classifiers. Sensors, 21(2), 433.
- Matzka, S. (2020). Explainable AI for predictive maintenance applications. Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Data Sciences for Industry (AI4I 2020), 69–74.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should I trust you? Explaining any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.
- Scholkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C. (1999). Support vector method for novelty detection. Advances in Neural Information Processing Systems, 12, 582–588.
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localisation. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 618–626.
- Serradilla, O., Zugasti, J., Rodriguez, J., and Zurutuza, U. (2022). Deep learning models for predictive maintenance: A survey, comparison, challenges and prospects. Applied Intelligence, 52, 10934–10964.
- Soesatijono, S. and Darsin, M. (2021). Literature studies on maintenance management. Journal of Energy, Mechanical, Material, and Manufacturing Engineering, 6(1), 67–74. https://doi.org/10.22219/jemmme.v6i1.12571
- van Dinter, R., Tekinerdogan, B., and Catal, C. (2022). Predictive maintenance using digital twins: A systematic literature review. Information and Software Technology, 151, 107008.
- Wang, H., Li, S., Song, L., and Cui, L. (2023). Hydraulic component RUL estimation with deep learning. Reliability Engineering and System Safety, 220, 108301.
- Zhang, W., Li, X., Jia, X. D., Ma, H., Luo, Z., and Li, X. (2023). Transfer learning for cross-domain hydraulic fault diagnosis. Journal of Manufacturing Systems, 68, 119–131.
- Zhao, Z., Li, T., Wu, J., Sun, C., Wang, S., Yan, R., and Chen, X. (2019). Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study. ISA Transactions, 107, 224–255. Zio, E. (2022). Prognostics and Health Management (PHM): Where are we and where do we go? Reliability Engineering and System Safety, 218, 108119.

# APPENDICES

## Python Implementation Code

### Complete Predictive Maintenance Pipeline

The following Python script implements the complete predictive maintenance pipeline described in this thesis, including data loading, preprocessing, visualization, model training, evaluation, and maintenance recommendation.

```
# -*- coding: utf-8 -*-
"""
============================================================
  predictive model for Hydraulic machine maintenance
  Dataset : LandingGear_Balanced_Dataset.csv
  Supports: Anomaly Detection + Classification + RUL Estimation
============================================================
Methods implemented:
  1. Isolation Forest          (anomaly detection)
  2. One-Class SVM             (anomaly detection)
  3. K-Means Clustering        (clustering-based anomaly detection)
  4. DBSCAN                    (density-based anomaly detection)
  5. Autoencoder               (deep learning anomaly detection)
  6. CNN                       (1-D convolutional fault classifier)
  7. XGBoost                   (gradient-boosted fault + RUL predictor)

Outputs per model (6-panel diagnostic figure):
  ┌──────────────────┬──────────────────┬──────────────────┐
  │ Confusion Matrix │    ROC Curve     │ Precision-Recall │
  ├──────────────────┼──────────────────┼──────────────────┤
  │ Anomaly Score Dist. │ PCA – Predicted Lbl │ PCA – True Labels │
  └──────────────────┴──────────────────┴──────────────────┘

Pipeline-level outputs:
  - heatmap.png               (correlation matrix)
  - pca_scatter.png           (fault clusters, all fault codes)
  - method_comparison.png     (accuracy / recall bar chart)
  - method_evaluation.csv     (summary table)
  - model_summary.csv         (ROC-AUC, AP scores)
  - 01_Isolation_Forest.png … 07_XGBoost.png
============================================================
"""
```

```
# ──────────────────────────────────────────
# 1.  IMPORTS
# ──────────────────────────────────────────
import warnings
warnings.filterwarnings("ignore")

import os
import numpy as np
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from matplotlib.colors import LinearSegmentedColormap
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import (
    accuracy_score, recall_score,
    confusion_matrix,
    roc_curve, auc,
    precision_recall_curve, average_precision_score,
    roc_auc_score
)
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.cluster import KMeans, DBSCAN

import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import (
    Input, Dense, Conv1D, GlobalAveragePooling1D, Dropout
)
from tensorflow.keras.callbacks import EarlyStopping

import xgboost as xgb
```

```
# ──────────────────────────────────────────
# 2.  CONFIGURATION
# ──────────────────────────────────────────
DEFAULT_DATA_PATH = "LandingGear_Balanced_Dataset.csv"

CONFIG = {
    "RANDOM_STATE"          : 42,
    "TEST_SIZE"             : 0.25,
    "RUL_THRESHOLD"         : 20.0,      # hours: below this → schedule maintenance now
    "ANOMALY_CONTAMINATION" : 0.30,      # expected anomaly fraction
    "AUTOENCODER_EPOCHS"    : 60,
    "CNN_EPOCHS"            : 40,
    "BATCH_SIZE"            : 32,
    "FEATURE_COLS"          : ["Max_Deflection", "Max_Velocity", "Settling_Time",
                               "Mass", "K_Stiffness", "B_Damping"],
    "TARGET_FAULT"          : "Fault_Code",   # 0=Normal, 1/2/3=Fault types
    "TARGET_RUL"            : "RUL",
    "FAULT_NAMES"           : {
        0: "Normal",
        1: "Fault-Type-1 (deflection)",
        2: "Fault-Type-2 (velocity)",
        3: "Fault-Type-3 (settling/damping)"
    },
    "OUTPUT_DIR"            : "outputs",   # folder for all saved files
}


# ──────────────────────────────────────────
# 3.  SHARED PLOT HELPERS
```

```
# ──────────────────────────────────────────────
# 3.  SHARED PLOT HELPERS
# ──────────────────────────────────────────────
CMAP_CM   = LinearSegmentedColormap.from_list("cm_blue", ["#d0e8f5", "#003f7f"], N=256)
PANEL_KW  = dict(family="DejaVu Sans", size=11, weight="bold")
PURPLE    = "#6a0dad"


def _make_6panel_figure(model_name: str,
                        y_true: np.ndarray,
                        y_pred: np.ndarray,
                        scores: np.ndarray,
                        score_label: str,
                        X_pca: np.ndarray,
                        pv1: float, pv2: float,
                        save_path: str) -> None:
    """
    Draw the 6-panel diagnostic figure (identical layout to reference image)
    and save it to *save_path*.

    Parameters
    ----------
    y_true      : ground-truth binary labels  (0=Normal, 1=Fault)
    y_pred      : predicted binary labels
    scores      : continuous anomaly/fault score  (higher → more likely fault)
    score_label : x-axis label for the score distribution panel
    X_pca       : (N, 2) PCA projection of the full scaled feature matrix
    pv1, pv2    : explained variance (%) for PC1 / PC2
    save_path   : full output file path
    """
    fig = plt.figure(figsize=(18, 10), facecolor="white")
    fig.suptitle(f"{model_name} – Predictive Maintenance",
                 fontsize=15, fontweight="bold", color=PURPLE, y=1.01)

    gs = gridspec.GridSpec(2, 3, figure=fig,
                           hspace=0.42, wspace=0.35,
                           left=0.06, right=0.97,
                           top=0.93, bottom=0.08)

    # ── Panel 1 – Confusion Matrix ──────────────────────────────
    ax1 = fig.add_subplot(gs[0, 0])
    cm  = confusion_matrix(y_true, y_pred)
    im  = ax1.imshow(cm, cmap=CMAP_CM, aspect="auto")
    plt.colorbar(im, ax=ax1, fraction=0.046, pad=0.04)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax1.text(j, i, str(cm[i, j]),
                     ha="center", va="center",
                     color="white" if cm[i, j] > cm.max() / 2 else "#003f7f",
                     fontsize=14, fontweight="bold")
    ax1.set_xticks([0, 1]); ax1.set_yticks([0, 1])
    ax1.set_xticklabels(["0", "1"]); ax1.set_yticklabels(["0", "1"])
    ax1.set_xlabel("Predicted", fontsize=10)
    ax1.set_ylabel("Actual",    fontsize=10)
    ax1.set_title("Confusion Matrix", **PANEL_KW, color=PURPLE)

    # ── Panel 2 – ROC Curve ─────────────────────────────────────
    ax2 = fig.add_subplot(gs[0, 1])
    fpr, tpr, _ = roc_curve(y_true, scores)
    roc_val     = auc(fpr, tpr)
    ax2.plot(fpr, tpr, color=PURPLE, lw=2,
             label=f"{model_name} (AUC={roc_val:.3f})")
```

```
# ── Panel 2 – ROC Curve ──────────────────────────────────────
ax2 = fig.add_subplot(gs[0, 1])
fpr, tpr, _ = roc_curve(y_true, scores)
roc_val     = auc(fpr, tpr)
ax2.plot(fpr, tpr, color=PURPLE, lw=2,
         label=f"{model_name} (AUC={roc_val:.3f})")
ax2.plot([0, 1], [0, 1], "k--", lw=1)
ax2.set_xlabel("False Positive Rate", fontsize=10)
ax2.set_ylabel("True Positive Rate",  fontsize=10)
ax2.set_title("ROC Curve", **PANEL_KW)
ax2.legend(loc="lower right", fontsize=9)
ax2.grid(True, alpha=0.3)

# ── Panel 3 – Precision-Recall Curve ─────────────────────────
ax3 = fig.add_subplot(gs[0, 2])
prec, rec, _ = precision_recall_curve(y_true, scores)
ap_val       = average_precision_score(y_true, scores)
ax3.plot(rec, prec, color=PURPLE, lw=2,
         label=f"{model_name} (AP={ap_val:.3f})")
ax3.set_xlabel("Recall",    fontsize=10)
ax3.set_ylabel("Precision", fontsize=10)
ax3.set_title("Precision-Recall Curve", **PANEL_KW)
ax3.legend(loc="lower left", fontsize=9)
ax3.grid(True, alpha=0.3)

# ── Panel 4 – Anomaly Score Distribution ─────────────────────
ax4 = fig.add_subplot(gs[1, 0])
ax4.hist(scores[y_true == 0], bins=40, alpha=0.7,
         color="#2ecc71", label="Normal", edgecolor="white")
ax4.hist(scores[y_true == 1], bins=40, alpha=0.7,
         color="#e74c3c", label="Fault",  edgecolor="white")
ax4.set_xlabel(score_label, fontsize=10)
ax4.set_ylabel("Count",     fontsize=10)
ax4.set_title("Anomaly Score Distribution", **PANEL_KW)
ax4.legend(fontsize=9)
ax4.grid(True, alpha=0.3, axis="y")

# ── Panel 5 – PCA Predicted Labels ───────────────────────────
ax5 = fig.add_subplot(gs[1, 1])
ax5.scatter(X_pca[y_pred == 0, 0], X_pca[y_pred == 0, 1],
            c="#2ecc71", s=15, alpha=0.6, label="Predicted Normal")
ax5.scatter(X_pca[y_pred == 1, 0], X_pca[y_pred == 1, 1],
            c="#e74c3c", s=15, alpha=0.6, label="Predicted Fault")
ax5.set_xlabel(f"PC1 ({pv1:.1f}%)", fontsize=10)
ax5.set_ylabel(f"PC2 ({pv2:.1f}%)", fontsize=10)
ax5.set_title("PCA – Predicted Labels", **PANEL_KW)
ax5.legend(fontsize=8, markerscale=1.5)
ax5.grid(True, alpha=0.3)

# ── Panel 6 – PCA True Labels ────────────────────────────────
ax6 = fig.add_subplot(gs[1, 2])
ax6.scatter(X_pca[y_true == 0, 0], X_pca[y_true == 0, 1],
            c="#2ecc71", s=15, alpha=0.6, label="True Normal")
ax6.scatter(X_pca[y_true == 1, 0], X_pca[y_true == 1, 1],
            c="#e74c3c", s=15, alpha=0.6, label="True Fault")
ax6.set_xlabel(f"PC1 ({pv1:.1f}%)", fontsize=10)
ax6.set_ylabel(f"PC2 ({pv2:.1f}%)", fontsize=10)
ax6.set_title("PCA – True Labels", **PANEL_KW)
ax6.legend(fontsize=8, markerscale=1.5)
ax6.grid(True, alpha=0.3)

plt.tight_layout()
```

```
    # ── Panel 6 – PCA True Labels ──────────────────────────────────
    ax6 = fig.add_subplot(gs[1, 2])
    ax6.scatter(X_pca[y_true == 0, 0], X_pca[y_true == 0, 1],
                c="#2ecc71", s=15, alpha=0.6, label="True Normal")
    ax6.scatter(X_pca[y_true == 1, 0], X_pca[y_true == 1, 1],
                c="#e74c3c", s=15, alpha=0.6, label="True Fault")
    ax6.set_xlabel(f"PC1 ({pv1:.1f}%)", fontsize=10)
    ax6.set_ylabel(f"PC2 ({pv2:.1f}%)", fontsize=10)
    ax6.set_title("PCA – True Labels", **PANEL_KW)
    ax6.legend(fontsize=8, markerscale=1.5)
    ax6.grid(True, alpha=0.3)

    plt.tight_layout()
    fig.savefig(save_path, dpi=150, bbox_inches="tight")
    plt.close(fig)
    print(f"       ✓ saved → {save_path}")


# ═══════════════════════════════════════════════════════════════════
# 4.  PIPELINE CLASS
# ═══════════════════════════════════════════════════════════════════
class PredictiveMaintenancePipeline:
    """
    Encapsulates the entire predictive maintenance workflow.

    Extended from the original thesis pipeline to additionally produce
    individual 6-panel diagnostic figures (Confusion Matrix · ROC ·
    Precision-Recall · Score Distribution · PCA×2) for every model,
    matching the reference layout provided in the thesis.
    """

    def __init__(self, data_path: str, config: dict):
        self.data_path = data_path
        self.config    = config

        # ── unpack config ──────────────────────────────────────────
        self.RANDOM_STATE          = config["RANDOM_STATE"]
        self.TEST_SIZE             = config["TEST_SIZE"]
        self.RUL_THRESHOLD         = config["RUL_THRESHOLD"]
        self.ANOMALY_CONTAMINATION = config["ANOMALY_CONTAMINATION"]
        self.AUTOENCODER_EPOCHS    = config["AUTOENCODER_EPOCHS"]
        self.CNN_EPOCHS            = config["CNN_EPOCHS"]
        self.BATCH_SIZE            = config["BATCH_SIZE"]
        self.FEATURE_COLS          = config["FEATURE_COLS"]
        self.TARGET_FAULT          = config["TARGET_FAULT"]
        self.TARGET_RUL            = config["TARGET_RUL"]
        self.FAULT_NAMES           = config["FAULT_NAMES"]
        self.OUTPUT_DIR            = config.get("OUTPUT_DIR", ".")

        os.makedirs(self.OUTPUT_DIR, exist_ok=True)

        # ── data attributes ────────────────────────────────────────
        self.df        = None
        self.X         = None
        self.X_scaled  = None
        self.y_fault   = None
        self.y_binary  = None
        self.y_rul     = None
        self.X_tr = self.X_te = None
        self.yf_tr = self.yf_te = None
        self.yb_tr = self.yb_te = None
        self.yr_tr = self.yr_te = None
```

```
        # ── data attributes ──────────────────────────────────────
        self.df        = None
        self.X         = None
        self.X_scaled  = None
        self.y_fault   = None
        self.y_binary  = None
        self.y_rul     = None
        self.X_tr = self.X_te = None
        self.yf_tr = self.yf_te = None
        self.yb_tr = self.yb_te = None
        self.yr_tr = self.yr_te = None
        self.scaler    = None

        # ── shared PCA projection (computed once, reused in all plots) ─
        self.X_pca = None
        self.pv1   = None
        self.pv2   = None

        # ── model handles ────────────────────────────────────────
        self.ae_model       = None
        self.cnn_model      = None
        self.xgb_clf        = None
        self.xgb_rul        = None
        self.rul_pred       = None
        self.xgb_fault_pred = None

        # ── evaluation stores ────────────────────────────────────
        self.results     = {}    # {name: {Accuracy, Recall}}
        self.diag_scores = {}    # {name: continuous score array}  ← new
        self.diag_preds  = {}    # {name: binary pred array}       ← new

    # ──────────────────────────────────────────
    # A.  DATA LOADING & PREPROCESSING
    # ──────────────────────────────────────────
    def _load_and_preprocess_data(self):
        print("=" * 60)
        print("  LANDING GEAR PREDICTIVE MAINTENANCE PIPELINE")
        print("=" * 60)

        self.df = pd.read_csv(self.data_path)
        print(f"\n[DATA] Loaded {len(self.df):,} rows × {self.df.shape[1]} columns")
        print(f"       Features : {self.FEATURE_COLS}")
        print(f"       Fault dist:\n{self.df[self.TARGET_FAULT].value_counts().to_string()}")
        print(f"       RUL range : {self.df[self.TARGET_RUL].min():.1f} – "
              f"{self.df[self.TARGET_RUL].max():.1f} h")

        self.df.drop(columns=["RunID"], inplace=True, errors="ignore")
        self.df["is_fault"] = (self.df[self.TARGET_FAULT] != 0).astype(int)

        self.X        = self.df[self.FEATURE_COLS].copy()
        self.y_fault  = self.df[self.TARGET_FAULT].values
        self.y_binary = self.df["is_fault"].values
        self.y_rul    = self.df[self.TARGET_RUL].values

        self.scaler   = StandardScaler()
        self.X_scaled = self.scaler.fit_transform(self.X)

        (self.X_tr, self.X_te,
         self.yf_tr, self.yf_te,
         self.yb_tr, self.yb_te,
         self.yr_tr, self.yr_te) = train_test_split(
            self.X_scaled, self.y_fault, self.y_binary, self.y_rul,
```

```
        self.yb_tr, self.yb_te,
        self.yr_tr, self.yr_te) = train_test_split(
            self.X_scaled, self.y_fault, self.y_binary, self.y_rul,
            test_size=self.TEST_SIZE,
            random_state=self.RANDOM_STATE,
            stratify=self.y_fault
        )
        print(f"\n[SPLIT] Train {len(self.X_tr):,}  |  Test {len(self.X_te):,}")

        # ── compute shared 2-D PCA (used in every 6-panel figure) ──────
        pca2 = PCA(n_components=2, random_state=self.RANDOM_STATE)
        self.X_pca = pca2.fit_transform(self.X_scaled)
        self.pv1   = pca2.explained_variance_ratio_[0] * 100
        self.pv2   = pca2.explained_variance_ratio_[1] * 100

    # ═══════════════════════════════════════════════════════════════
    # B.  DATA CLEANING
    # ═══════════════════════════════════════════════════════════════
    def _clean_data(self):
        print("\n[CLEANING] Checking for missing values and inconsistencies …")
        missing_values = self.df.isnull().sum()
        if missing_values.sum() > 0:
            print("         Detected missing values:")
            print(missing_values[missing_values > 0])
            for col in self.df.columns:
                if self.df[col].isnull().any():
                    if pd.api.types.is_numeric_dtype(self.df[col]):
                        self.df[col].fillna(self.df[col].mean(), inplace=True)
                        print(f"         Filled missing numerical in '{col}' with mean.")
                    else:
                        self.df[col].fillna(self.df[col].mode()[0], inplace=True)
                        print(f"         Filled missing categorical in '{col}' with mode.")
        else:
            print("         No missing values detected.")
        print("         Data cleaning complete.")

    # ═══════════════════════════════════════════════════════════════
    # C.  PIPELINE-LEVEL VISUALISATIONS
    # ═══════════════════════════════════════════════════════════════
    def _generate_visualizations(self):
        """Correlation heatmap + multi-class PCA scatter (original outputs)."""

        # C1 – Correlation heatmap
        print("\n[VIZ] Generating correlation heatmap …")
        fig, ax = plt.subplots(figsize=(9, 7))
        corr = pd.DataFrame(self.X_scaled, columns=self.FEATURE_COLS).assign(
            Fault_Code=self.y_fault, RUL=self.y_rul).corr()
        mask = np.triu(np.ones_like(corr, dtype=bool))
        sns.heatmap(corr, mask=mask, annot=True, fmt=".2f",
                    cmap="coolwarm", linewidths=0.5, ax=ax,
                    cbar_kws={"shrink": 0.8})
        ax.set_title("Feature Correlation Matrix – Landing Gear Dataset",
                     fontsize=13, pad=12)
        plt.tight_layout()
        p = os.path.join(self.OUTPUT_DIR, "heatmap.png")
        plt.savefig(p, dpi=150); plt.close()
        print(f"       Saved → {p}")

        # C2 – PCA scatter (all 4 fault codes)
        print("[VIZ] Generating PCA scatter …")
        palette    = {0: "#2ecc71", 1: "#e74c3c", 2: "#e67e22", 3: "#9b59b6"}
        labels_map = {0: "Normal", 1: "Fault-1", 2: "Fault-2", 3: "Fault-3"}
        fig, ax = plt.subplots(figsize=(9, 6))
```

```
        fig, ax = plt.subplots(figsize=(9, 6))
        for code, label in labels_map.items():
            mask = self.y_fault == code
            ax.scatter(self.X_pca[mask, 0], self.X_pca[mask, 1],
                       c=palette[code], label=label,
                       alpha=0.65, s=28, edgecolors="none")
        ax.set_xlabel(f"PC1 ({self.pv1:.1f}% var)")
        ax.set_ylabel(f"PC2 ({self.pv2:.1f}% var)")
        ax.set_title("PCA – Fault Cluster Separation", fontsize=13)
        ax.legend(title="Class", framealpha=0.85)
        plt.tight_layout()
        p = os.path.join(self.OUTPUT_DIR, "pca_scatter.png")
        plt.savefig(p, dpi=150); plt.close()
        print(f"       Saved → {p}")

    # ═══════════════════════════════════════════════
    # D.  HELPER – store binary results + scores
    # ═══════════════════════════════════════════════
    def _store_results(self, name: str,
                       y_true: np.ndarray,
                       y_pred: np.ndarray,
                       score:  np.ndarray = None) -> None:
        """
        Store accuracy / recall (original behaviour) AND, optionally,
        the continuous anomaly score for the 6-panel figure.
        """
        acc = accuracy_score(y_true, y_pred)
        rec = recall_score(y_true, y_pred, average="binary", zero_division=0)
        self.results[name] = {"Accuracy": acc, "Recall": rec}
        print(f"       {name:28s}  acc={acc:.4f}  recall={rec:.4f}")

        if score is not None:
            self.diag_scores[name] = score
            self.diag_preds[name]  = y_pred

    # ═══════════════════════════════════════════════
    # E.  ANOMALY DETECTION MODELS  (unsupervised)
    # ═══════════════════════════════════════════════
    def _run_anomaly_detection_models(self):
        print("\n[MODELS] Running anomaly detection methods …")
        X_normal = self.X_scaled[self.y_binary == 0]

        # ── E1. Isolation Forest ──────────────────────
        iso = IsolationForest(contamination=self.ANOMALY_CONTAMINATION,
                              n_estimators=200,
                              random_state=self.RANDOM_STATE)
        iso.fit(self.X_scaled)
        iso_score = -iso.decision_function(self.X_scaled)   # higher → fault
        iso_pred  = (iso.predict(self.X_scaled) == -1).astype(int)
        self._store_results("Isolation Forest", self.y_binary, iso_pred, iso_score)

        # ── E2. One-Class SVM ─────────────────────────
        ocsvm = OneClassSVM(nu=0.20, kernel="rbf", gamma="scale")
        ocsvm.fit(X_normal)
        ocsvm_score = -ocsvm.decision_function(self.X_scaled)
        ocsvm_pred  = (ocsvm.predict(self.X_scaled) == -1).astype(int)
        self._store_results("One-Class SVM", self.y_binary, ocsvm_pred, ocsvm_score)

        # ── E3. KMeans ────────────────────────────────
        km = KMeans(n_clusters=4, random_state=self.RANDOM_STATE, n_init=15)
        km.fit(self.X_scaled)
        c_labels = km.labels_
```

```
        c_labels = km.labels_
        fault_rate = {c: self.y_binary[c_labels == c].mean() for c in range(4)}
        normal_c   = min(fault_rate, key=fault_rate.get)
        # Score = distance to nearest centroid
        km_score = np.min(
            np.linalg.norm(
                self.X_scaled[:, None, :] - km.cluster_centers_[None, :, :],
                axis=2),
            axis=1)
        km_pred = (c_labels != normal_c).astype(int)
        self._store_results("KMeans Clustering", self.y_binary, km_pred, km_score)

        # ── E4. DBSCAN ──────────────────────────────────────────
        db = DBSCAN(eps=1.5, min_samples=8)
        db.fit(self.X_scaled)
        db_labels = db.labels_
        unique, counts = np.unique(db_labels[db_labels != -1], return_counts=True)
        if len(unique) > 0:
            normal_db = unique[np.argmax(counts)]
            db_pred   = np.where((db_labels != normal_db) | (db_labels == -1), 1, 0)
        else:
            db_pred = np.ones(len(self.X_scaled), dtype=int)
        # Score = 5th-NN distance (density proxy)
        nn = NearestNeighbors(n_neighbors=5, n_jobs=-1).fit(self.X_scaled)
        db_dists, _ = nn.kneighbors(self.X_scaled)
        db_score    = db_dists[:, -1]
        self._store_results("DBSCAN", self.y_binary, db_pred, db_score)

    # ═══════════════════════════════════════════════════════
    # F.  AUTOENCODER
    # ═══════════════════════════════════════════════════════
    def _train_autoencoder(self):
        print("\n[MODELS] Training Autoencoder …")
        n_feat   = self.X_scaled.shape[1]
        X_normal = self.X_scaled[self.y_binary == 0]

        inputs     = Input(shape=(n_feat,))
        enc        = Dense(16, activation="relu")(inputs)
        enc        = Dense(8,  activation="relu")(enc)
        bottleneck = Dense(4,  activation="relu")(enc)
        dec        = Dense(8,  activation="relu")(bottleneck)
        dec        = Dense(16, activation="relu")(dec)
        outputs    = Dense(n_feat, activation="linear")(dec)

        self.ae_model = Model(inputs, outputs)
        self.ae_model.compile(optimizer="adam", loss="mse")
        self.ae_model.fit(
            X_normal, X_normal,
            epochs=self.AUTOENCODER_EPOCHS,
            batch_size=self.BATCH_SIZE,
            verbose=0, validation_split=0.1,
            callbacks=[EarlyStopping(patience=10, restore_best_weights=True)]
        )

        recon_err = np.mean(
            (self.X_scaled - self.ae_model.predict(self.X_scaled, verbose=0)) ** 2,
            axis=1)
        threshold  = np.percentile(recon_err[self.y_binary == 0],
                                   (1 - self.ANOMALY_CONTAMINATION) * 100)
        ae_pred    = (recon_err > threshold).astype(int)
        self._store_results("Autoencoder", self.y_binary, ae_pred, recon_err)
```

```
    # ═══════════════════════════════════════════
    # G.  1-D CNN FAULT CLASSIFIER
    # ═══════════════════════════════════════════
    def _train_cnn_classifier(self):
        print("\n[MODELS] Training 1-D CNN classifier …")
        n_feat = self.X_scaled.shape[1]

        # Reshape: (samples, timesteps=1, features)
        X_tr_cnn  = self.X_tr.reshape(-1, 1, n_feat)
        X_te_cnn  = self.X_te.reshape(-1, 1, n_feat)
        X_all_cnn = self.X_scaled.reshape(-1, 1, n_feat)

        self.cnn_model = Sequential([
            Conv1D(32, kernel_size=1, activation="relu",
                   input_shape=(1, n_feat)),
            Dropout(0.3),
            Conv1D(64, kernel_size=1, activation="relu"),
            GlobalAveragePooling1D(),
            Dense(64, activation="relu"),
            Dropout(0.3),
            Dense(1,  activation="sigmoid")
        ])
        self.cnn_model.compile(optimizer="adam",
                               loss="binary_crossentropy",
                               metrics=["accuracy"])
        self.cnn_model.fit(
            X_tr_cnn, self.yb_tr,
            epochs=self.CNN_EPOCHS,
            batch_size=self.BATCH_SIZE,
            verbose=0, validation_split=0.1,
            callbacks=[EarlyStopping(patience=8, restore_best_weights=True)]
        )

        # Evaluate on test set (original behaviour)
        cnn_prob_te = self.cnn_model.predict(X_te_cnn, verbose=0).flatten()
        cnn_pred_te = (cnn_prob_te > 0.5).astype(int)
        self._store_results("CNN Classifier", self.yb_te, cnn_pred_te)

        # Full-dataset scores for diagnostic figure
        cnn_prob_all = self.cnn_model.predict(X_all_cnn, verbose=0).flatten()
        cnn_pred_all = (cnn_prob_all > 0.5).astype(int)
        self.diag_scores["CNN Classifier"] = cnn_prob_all
        self.diag_preds["CNN Classifier"]  = cnn_pred_all

    # ═══════════════════════════════════════════
    # H.  XGBOOST
    # ═══════════════════════════════════════════
    def _train_xgboost(self):
        print("\n[MODELS] Training XGBoost …")

        # H1 - Multi-class fault classifier
        self.xgb_clf = xgb.XGBClassifier(
            n_estimators=300, max_depth=5, learning_rate=0.05,
            subsample=0.8, colsample_bytree=0.8,
            use_label_encoder=False, eval_metric="mlogloss",
            random_state=self.RANDOM_STATE, verbosity=0
        )
        self.xgb_clf.fit(self.X_tr, self.yf_tr)
        self.xgb_fault_pred  = self.xgb_clf.predict(self.X_te)
        xgb_binary_pred_te   = (self.xgb_fault_pred != 0).astype(int)
        self._store_results("XGBoost Classifier", self.yb_te, xgb_binary_pred_te)
```

```
        xgb_binary_pred_te   = (self.xgb_fault_pred != 0).astype(int)
        self._store_results("XGBoost Classifier", self.yb_te, xgb_binary_pred_te)

        # Full-dataset scores for diagnostic figure
        xgb_prob_all  = self.xgb_clf.predict_proba(self.X_scaled)[:, 1:].sum(axis=1)
        xgb_pred_all  = (self.xgb_clf.predict(self.X_scaled) != 0).astype(int)
        self.diag_scores["XGBoost Classifier"] = xgb_prob_all
        self.diag_preds["XGBoost Classifier"]  = xgb_pred_all

        # H2 – RUL regression
        self.xgb_rul = xgb.XGBRegressor(
            n_estimators=300, max_depth=5, learning_rate=0.05,
            subsample=0.8, colsample_bytree=0.8,
            random_state=self.RANDOM_STATE, verbosity=0
        )
        self.xgb_rul.fit(self.X_tr, self.yr_tr)
        self.rul_pred = self.xgb_rul.predict(self.X_te)
        rul_mae = np.mean(np.abs(self.rul_pred - self.yr_te))
        print(f"       XGBoost RUL regression  MAE = {rul_mae:.2f} hours")

# ═══════════════════════════════════════════════════════════
# I.  6-PANEL DIAGNOSTIC FIGURES  (new)
# ═══════════════════════════════════════════════════════════
_DIAG_META = [
    # ( results-dict key,      file prefix,   score label              )
    ("Isolation Forest",    "01_Isolation_Forest",  "Anomaly Score"),
    ("One-Class SVM",       "02_OC_SVM",            "Decision Score"),
    ("KMeans Clustering",   "03_KMeans",            "Distance to Centroid"),
    ("DBSCAN",              "04_DBSCAN",            "5th-NN Distance"),
    ("Autoencoder",         "05_Autoencoder",       "Reconstruction Error (MSE)"),
    ("CNN Classifier",      "06_CNN",               "Predicted Fault Probability"),
    ("XGBoost Classifier",  "07_XGBoost",           "Predicted Fault Probability"),
]

def _generate_diagnostic_figures(self):
    """
    For every model that stored a continuous score, produce the
    6-panel diagnostic figure and save it to the output directory.
    """
    print("\n[DIAG] Generating 6-panel diagnostic figures …")
    for name, prefix, score_label in self._DIAG_META:
        if name not in self.diag_scores:
            print(f"       ⚠ skipped {name} (no score stored)")
            continue
        save_path = os.path.join(self.OUTPUT_DIR, f"{prefix}.png")
        _make_6panel_figure(
            model_name  = name,
            y_true      = self.y_binary,
            y_pred      = self.diag_preds[name],
            scores      = self.diag_scores[name],
            score_label = score_label,
            X_pca       = self.X_pca,
            pv1         = self.pv1,
            pv2         = self.pv2,
            save_path   = save_path,
        )
```

```
# ──────────────────────────────────────────────
# J.  EVALUATION SUMMARY TABLE
# ──────────────────────────────────────────────
def _generate_evaluation_summary(self):
    print("\n" + "=" * 60)
    print("  EVALUATION SUMMARY")
    print("=" * 60)
    self.summary = (pd.DataFrame(self.results).T
                    .sort_values("Recall", ascending=False))
    self.summary.index.name = "Method"
    print(self.summary.to_string(float_format=lambda x: f"{x:.4f}"))

    # Extended summary including ROC-AUC and AP
    rows = []
    for name, _, _ in self._DIAG_META:
        if name not in self.diag_scores:
            continue
        sc = self.diag_scores[name]
        rows.append({
            "Model"  : name,
            "ROC-AUC": f"{roc_auc_score(self.y_binary, sc):.3f}",
            "AP"     : f"{average_precision_score(self.y_binary, sc):.3f}",
        })
    ext = pd.DataFrame(rows).set_index("Model")
    combined = self.summary.join(ext, how="left")
    p = os.path.join(self.OUTPUT_DIR, "method_evaluation.csv")
    combined.to_csv(p)
    print(f"\n[SAVED] {p}")

    p2 = os.path.join(self.OUTPUT_DIR, "model_summary.csv")
    ext.to_csv(p2)
    print(f"[SAVED] {p2}")

# ──────────────────────────────────────────────
# K.  BEST METHOD SELECTION
# ──────────────────────────────────────────────
def _select_best_method(self):
    self.summary["Score"] = (0.6 * self.summary["Recall"] +
                             0.4 * self.summary["Accuracy"])
    self.best_method = self.summary["Score"].idxmax()
    self.best_row    = self.summary.loc[self.best_method]

    print("\n" + "=" * 60)
    print("  BEST METHOD")
    print("=" * 60)
    print(f"  → {self.best_method}")
    print(f"     Accuracy : {self.best_row['Accuracy']:.4f}")
    print(f"     Recall   : {self.best_row['Recall']:.4f}")
    print(f"     Score    : {self.best_row['Score']:.4f}")

# ──────────────────────────────────────────────
# L.  MAINTENANCE RECOMMENDATION
# ──────────────────────────────────────────────
def _generate_maintenance_recommendation(self):
    print("\n" + "=" * 60)
    print("  MAINTENANCE RECOMMENDATION")
    print("=" * 60)

    critical_mask  = self.rul_pred < self.RUL_THRESHOLD
    critical_count = int(critical_mask.sum())
    critical_avg   = float(self.rul_pred[critical_mask].mean()) \
                     if critical_count > 0 else None
```

```
# ──────────────────────────────────────────────
# L.  MAINTENANCE RECOMMENDATION
# ──────────────────────────────────────────────
def _generate_maintenance_recommendation(self):
    print("\n" + "=" * 60)
    print("  MAINTENANCE RECOMMENDATION")
    print("=" * 60)

    critical_mask  = self.rul_pred < self.RUL_THRESHOLD
    critical_count = int(critical_mask.sum())
    critical_avg   = float(self.rul_pred[critical_mask].mean()) \
                     if critical_count > 0 else None
    overall_avg    = float(self.rul_pred.mean())
    pred_faults    = pd.Series(self.xgb_fault_pred).value_counts()

    print(f"\n  Predicted fault distribution (test set):")
    for code, cnt in pred_faults.items():
        print(f"    {self.FAULT_NAMES.get(code, str(code)):35s} → {cnt:3d} units")

    print(f"\n  Average predicted RUL (test set) : {overall_avg:.1f} hours")

    if critical_count > 0:
        print(f"\n  ⚠  URGENT:  {critical_count} units with RUL < "
              f"{self.RUL_THRESHOLD:.0f} hours")
        print(f"     Average RUL in critical group : {critical_avg:.1f} hours")
        most_freq = pred_faults.index[0]
        print(f"\n  RECOMMENDATION:")
        print(f"    Schedule immediate maintenance for "
              f"{critical_count} critical unit(s).")
        print(f"    Inspect for "
              f"{self.FAULT_NAMES.get(most_freq, 'unknown faults')}.")
        print(f"    Routine check for remaining fleet within "
              f"{overall_avg:.0f} hours.")
    else:
        print(f"\n  ✓  No units below the {self.RUL_THRESHOLD:.0f}-hour RUL threshold.")
        print(f"\n  RECOMMENDATION:")
        print(f"    Fleet is operating within normal bounds.")
        print(f"    Schedule next routine inspection in ≈ {overall_avg:.0f} hours.")

    print(f"\n  Best predictive model used : {self.best_method}")
    print("=" * 60)

# ──────────────────────────────────────────────
# M.  BAR CHART – METHOD COMPARISON
# ──────────────────────────────────────────────
def _plot_method_comparison(self):
    fig, ax = plt.subplots(figsize=(10, 5))
    x  = np.arange(len(self.summary))
    w  = 0.35
    ax.bar(x - w/2, self.summary["Accuracy"], w,
           label="Accuracy", color="#3498db", alpha=0.85)
    ax.bar(x + w/2, self.summary["Recall"],   w,
           label="Recall",   color="#e74c3c", alpha=0.85)
    ax.set_xticks(x)
    ax.set_xticklabels(self.summary.index, rotation=22,
                       ha="right", fontsize=9)
    ax.set_ylim(0, 1.12)
    ax.set_ylabel("Score")
    ax.set_title("Predictive Maintenance – Method Comparison", fontsize=13)
    ax.legend()
    ax.axhline(0.8, color="gray", linestyle="--", linewidth=0.8, alpha=0.6)
    best_idx = list(self.summary.index).index(self.best_method)
```

```
        ax.set_ylabel("Score")
        ax.set_title("Predictive Maintenance – Method Comparison", fontsize=13)
        ax.legend()
        ax.axhline(0.8, color="gray", linestyle="--", linewidth=0.8, alpha=0.6)
        best_idx = list(self.summary.index).index(self.best_method)
        ax.annotate(
            "BEST",
            xy=(best_idx, self.summary.loc[self.best_method, "Score"]),
            xytext=(best_idx, 1.06),
            ha="center", fontsize=8, color="#27ae60", fontweight="bold",
            arrowprops=dict(arrowstyle="->", color="#27ae60", lw=1.2)
        )
        plt.tight_layout()
        p = os.path.join(self.OUTPUT_DIR, "method_comparison.png")
        plt.savefig(p, dpi=150); plt.close()
        print(f"\n[SAVED] {p}")
        print("[DONE]  Pipeline complete.")

    # ──────────────────────────────────────────────
    # N.  MASTER RUN
    # ──────────────────────────────────────────────
    def run_pipeline(self):
        """Execute the full integrated predictive maintenance pipeline."""
        self._load_and_preprocess_data()
        self._clean_data()
        self._generate_visualizations()
        self._run_anomaly_detection_models()
        self._train_autoencoder()
        self._train_cnn_classifier()
        self._train_xgboost()
        self._generate_diagnostic_figures()     # ← 7 individual 6-panel plots
        self._generate_evaluation_summary()
        self._select_best_method()
        self._generate_maintenance_recommendation()
        self._plot_method_comparison()


# ──────────────────────────────────────────────
# 5.  MAIN ENTRY POINT
# ──────────────────────────────────────────────
if __name__ == "__main__":
    user_path  = input(
        f"Enter path to dataset (press Enter for default "
        f"'{DEFAULT_DATA_PATH}'): "
    ).strip()
    data_path  = user_path if user_path else DEFAULT_DATA_PATH

    pipeline = PredictiveMaintenancePipeline(data_path=data_path, config=CONFIG)
    pipeline.run_pipeline()
```

Result

```
Enter path to dataset (press Enter for default 'LandingGear_Balanced_Dataset.csv'): /content/sample_data/LandingGear_Balanced_Dataset.csv
============================================================
  LANDING GEAR PREDICTIVE MAINTENANCE PIPELINE
============================================================

[DATA] Loaded 1,500 rows × 9 columns
       Features : ['Max_Deflection', 'Max_Velocity', 'Settling_Time', 'Mass', 'K_Stiffness', 'B_Damping']
       Fault dist:
Fault_Code
1    500
2    500
0    300
3    200
       RUL range : 0.0 – 99.9 h

[SPLIT] Train 1,125  |  Test 375

[CLEANING] Checking for missing values and inconsistencies …
         No missing values detected.
         Data cleaning complete.

[VIZ] Generating correlation heatmap …
       Saved → outputs/heatmap.png
[VIZ] Generating PCA scatter …
       Saved → outputs/pca_scatter.png

[MODELS] Running anomaly detection methods …
       Isolation Forest              acc=0.4747  recall=0.3592
       One-Class SVM                 acc=0.8867  recall=0.9092
       KMeans Clustering             acc=0.6253  recall=0.5317
       DBSCAN                        acc=0.2153  recall=0.0192

[MODELS] Training Autoencoder …
       Autoencoder                   acc=0.8920  recall=0.9400

[MODELS] Training 1-D CNN classifier …
       CNN Classifier                acc=0.9093  recall=0.9100

[MODELS] Training XGBoost …
       XGBoost Classifier            acc=0.9947  recall=0.9967
       XGBoost RUL regression  MAE = 1.15 hours

[DIAG] Generating 6-panel diagnostic figures …
       ✓ saved → outputs/01_Isolation_Forest.png
       ✓ saved → outputs/02_OC_SVM.png
       ✓ saved → outputs/03_KMeans.png
       ✓ saved → outputs/04_DBSCAN.png
       ✓ saved → outputs/05_Autoencoder.png
       ✓ saved → outputs/06_CNN.png
       ✓ saved → outputs/07_XGBoost.png

============================================================
  EVALUATION SUMMARY
============================================================
                     Accuracy  Recall
Method
XGBoost Classifier     0.9947  0.9967
Autoencoder            0.8920  0.9400
CNN Classifier         0.9093  0.9100
One-Class SVM          0.8867  0.9092
KMeans Clustering      0.6253  0.5317
Isolation Forest       0.4747  0.3592
DBSCAN                 0.2153  0.0192

[SAVED] outputs/method_evaluation.csv
[SAVED] outputs/model_summary.csv

============================================================
  BEST METHOD
============================================================
  → XGBoost Classifier
     Accuracy : 0.9947
     Recall   : 0.9967
     Score    : 0.9959

============================================================
  MAINTENANCE RECOMMENDATION
============================================================

  Predicted fault distribution (test set):
    Fault-Type-1 (deflection)          → 125 units
    Fault-Type-2 (velocity)            → 125 units
    Normal                             →  75 units
    Fault-Type-3 (settling/damping)    →  50 units

  Average predicted RUL (test set) : 63.8 hours

  ⚠  URGENT:  45 units with RUL < 20 hours
     Average RUL in critical group : 10.7 hours

  RECOMMENDATION:
    Schedule immediate maintenance for 45 critical unit(s).
    Inspect for Fault-Type-1 (deflection).
    Routine check for remaining fleet within 64 hours.

  Best predictive model used : XGBoost Classifier
============================================================

[SAVED] outputs/method_comparison.png
[DONE]  Pipeline complete.
```

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

## PLAGIARISM VERIFICATION

Title of the Thesis DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE

Total Pages: 97

Name of the scholar: Amrish Tripathi

Supervisor(s): Dr. S. K. Garg

Department: Department of Mechanical Engineering

This is the report that the above thesis was scanned for similarity detection. The process and outcome are given below:

Software used: Turnitin Similarity Index: 11 % Total Word Count: 13824

Date: 27/05/26

Amrish Tripathi

**Candidate's Signature**

As per report attached

Sugarg

**Signature of Supervisor(s)**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

## CERTIFICATE OF FINAL THESIS SUBMISSION

**(1)** Name: **Amrish Tripathi**

**(2)** Roll No.: **24/IEM/03**

**(3)** Thesis Title: **"DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE"**

**(4)** Degree for which the thesis is submitted: **M.Tech.**

**(5)** Faculty of the University to which the thesis is submitted: **Prof. S.K Garg**

**(6)** Thesis Preparation Guide was referred to for preparing the thesis: **YES**

**(7)** Specifications regarding thesis format have been closely followed: **YES**

**(8)** The contents of the thesis have been organized based on the guidelines: **YES**

**(9)** The thesis has been prepared without resorting to plagiarism: **YES**

**(10)** All sources used have been cited appropriately: **YES**

**(11)** The thesis has not been submitted elsewhere for a degree: **YES**

**(12)** Submitted 2 spiral bound copies plus one CD: **YES**

Signature of Supervisor

Dr. S.K. Garg

Signature of Candidate

Amrish Tripathi

24/IEM/03

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering) Shahbad

Daulatpur, Main Bawana Road, Delhi-42

## CERTIFICATE OF THESIS SUBMISSION FOR EVALUATION

**(1)** Name: **Amrish Tripathi**

**(2)** Roll No.: **24/IEM/03**

**(3)** Thesis Title: **"M DEVELOPING A PREDICTIVE MODEL FOR HYDRAULIC MACHINE MAINTENANCE "**

**(4)** Degree for which the thesis is submitted: **M.Tech.**

**(5)** Faculty of the University to which the thesis is submitted: **Prof. S.K Garg**

**(6)** Thesis Preparation Guide was referred to for preparing the thesis: **YES**

**(7)** Specifications regarding thesis format have been closely followed: **YES**

**(8)** The contents of the thesis have been organized based on the guidelines: **YES**

**(9)** The thesis has been prepared without resorting to plagiarism: **YES**

**(10)** All sources used have been cited appropriately: **YES**

**(11)** The thesis has not been submitted elsewhere for a degree: **YES**

**(12)** Submitted 2 spiral bound copies plus one CD: **YES**

Signature of Candidate Amrish

Tripathi 24/IEM/03

# DELHI TECHNOLOGICAL UNIVERSITY

Shahbad Daulatpur, Main Bawana Road, Delhi-42

**Proforma for Submission of M.Tech. Major Project**

01. Name of the Student..Amrish Tripathi.....

02. Enrolment No ....24/IEM/03.....

03. Year of Admission .....2024.....

04. Programme **M.Tech.**, Branch..Industrial Engineering and Management

05. Name of Department..Mechanical.....

06. Admission Category i.e. Full Time/ Full Time (Sponsored)/ Part Time:...Full Time...

07. Applied as Regular/ Ex-student...Regular.....

08. Span Period Expired on .....

09. Extension of Span Period Granted or Not Granted ( if applicable ).....

10.Title of Thesis/Major Project..Developing a Predictive Model FOR Hydraulic Machine Maintenance.....

11. Name of Supervisor.....Prof. S.K Garg.....

12. Result Details (Enclose Copy of Mark sheets of all semesters) :

| S. No. | Semester | Passing Year | Roll No. | Marks Obtained | Max. Marks | % of Marks | Details of Back Paper Cleared (if any) |
|---|---|---|---|---|---|---|---|
| 01 | 1st | 2025 | 24/IEM/03 | | | 6.58 | |
| 02 | 2nd | 2025 | 24/IEM/03 | | | 7.50 | |
| 03 | 3rd | 2026 | 24/IEM/03 | | | 7.75 | |
| 04 | 4th (P/T only) | | | | | | |
| 05 | 5th (P/T only) | | | | | | |

13. Fee Details (Enclose the Fee Receipt):

| Amount Paid (in Rs.) | Receipt No. | Date |
|---|---|---|
| 3000/- | DUP5572487 | 27/05/2026 |

Amrish Tripathi
01/06/26
Signature of Student

**It is certified that the name of Examiners for evaluation of the above thesis/ project have already been recommended by the BOS.**

Signature of Supervisor

Signature of HOD with Seal

Head of the Department
Mechanical, Production
and Automobile Engineering
Delhi Technological University

*(Instructions for filling up the Form may see on back side please.)*

e-Receipt for State Bank Collect Payment

## REGISTRAR, DTU (RECEIPT A/C)

**BAWANA ROAD, SHAHABAD DAULATPUR, , DELHI-110042**
**Date: 26-May-2026**

| | |
|---|---|
| **SBCollect Reference Number :** | DUP5572487 |
| **Category :** | Miscellaneous Fees from students |
| **Amount :** | ₹3000 |
| **University Roll No :** | 2k24/IEM/03 |
| **Name of the student :** | Amrish Tripathi |
| **Academic Year :** | 2024-2026 |
| **Branch Course :** | Industrial Engineering and Management |
| **Type/Name of fee :** | Others if any |
| **Remarks if any :** | Thesis submission fee |
| **Mobile No. of the student :** | 9013157370 |
| **Fee Amount :** | 3000 |
| **Transaction charge :** | 0.00 |
| **Total Amount (In Figures) :** | 3,000.00 |
| **Total Amount (In words) :** | Rupees Three Thousand Only |
| **Remarks :** | |

| | |
|---|---|
| **Notification 1:** | Late Registration Fee, Hostel Room rent for internship, Hostel cooler rent, Transcript fee (Within 5 years Rs.1500/- & $150 in USD, More than 5 years but less than 10 years Rs.2500/- & $250 in USD, More than 10 years Rs.5000/- & $500 in USD) Additional copies Rs.200/- each & $20 in USD each, I-card fee,Character certificate Rs.500/-. |
| **Notification 2:** | Migration Certificate Rs.500/-, Bonafide certificate Rs.200/-, Special certificate (any other certificate not covered in above list) Rs.1000/-,Provisional certificate Rs.500/-, Duplicate Mark sheet (Within 5 years Rs.2500/- & $250 in USD, More than 5 years but less than 10 years Rs.4000/- & $400 in USD, More than 10 years Rs.10000/- & $1000 in USD) |

**Thank you for choosing SB Collect. If you have any query / grievances regarding the transaction, please contact us**

**Toll-free helpline number i.e. 1800-1111-09 / 1800 - 1234/1800 2100**

**Email -:** sbcollect@sbi.co.in

# Delhi Technological University

(Formerly Delhi College of Engineering)

**THE RESULT OF THE CANDIDATE WHO APPEARED IN THE FOLLOWING EXAMINATION HELD IN NOV 2024 IS DECLARED AS UNDER:-**

## Master of Technology(Industrial Engineering and Management), I-SEMESTER

**Result Declaration Date : 31-07-2025** **Notification No: 1868**

IEM503 : Production & Operation Management

| Sr.No | Roll No. | Name of Student | IEM503 4.00 | SGPA | TC | CGPA | Failed Courses |
|---|---|---|---|---|---|---|---|
| 1 | 23/IEM/501 | PRAMOD | P | 4.50 | 8 | 4.50 | |

IEM501 : Data Analytics IEM505 : Quality Management IEM507 : Production & Operation Management IEM509 : Industry 4.0 & Smart Manufacturing IEM523 : Skill Enhancement Course 1 (Online) IEM525 : Self-Study (Online) IEM5313 : AI/ML IN INDUSTRIAL ENGINEERING AND MANAGEMENT UEC501 : Audit Course

| Sr.No | Roll No. | Name of Student | IEM501 4.00 | IEM505 4.00 | IEM507 4.00 | IEM509 4.00 | IEM523 2.00 | IEM525 2.00 | IEM5313 4.00 | UEC501 0.00 | SGPA | TC | CGPA | Failed Courses |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 24/IEM/01 | NAMAN SACHAN | B+ | A | B | A | B+ | A | B+ | O | 7.25 | 24 | 7.25 | |
| 3 | 24/IEM/03 | AMRISH TRIPATHI | A | B | P | A | B | B+ | B+ | A | 6.58 | 24 | 6.58 | |
| 4 | 24/IEM/04 | GULSHAN KUMAR SINGH | A | A | C | A+ | B+ | C | B | O | 7.00 | 24 | 7.00 | |
| 5 | 24/IEM/05 | ABHISHEK KUMAR SISODIYA | O | O | A | O | A+ | B+ | B+ | O | 8.83 | 24 | 8.83 | |
| 6 | 24/IEM/06 | SUNNY SOREN | A | C | P | A | B+ | C | B+ | A+ | 6.33 | 24 | 6.33 | |
| 7 | 24/IEM/07 | PRIYATOSH BHARADWAJ | A+ | C | P | A | C | B | B | A | 6.25 | 24 | 6.25 | |
| 8 | 24/IEM/08 | NGAYOSING A SHIMRAY | A | A | B | B+ | A+ | C | B | A | 7.00 | 24 | 7.00 | |
| 9 | 24/IEM/09 | SANTOSH KISKU | A | A | C | A+ | A+ | B | B+ | A | 7.42 | 24 | 7.42 | |

IEM501 : Data Analytics IEM505 : Quality Management IEM507 : Production & Operation Management IEM509 : Industry 4.0 & Smart Manufacturing IEM5313 : AI/ML IN INDUSTRIAL ENGINEERING AND MANAGEMENT UEC501 : Audit Course

**OIC (Results)**

**Controller of Examination**

**Note:**Any discrepancy in the result in r/o name/roll no/registration/marks/grades/course code/title should be brought to the notice of Controller of Examination/OIC(Results) within 15 days of declaration of result, in the prescribed proforma.

# Delhi Technological University

(Formerly Delhi College of Engineering)

**THE RESULT OF THE CANDIDATE WHO APPEARED IN THE FOLLOWING EXAMINATION HELD IN MAY 2025 IS DECLARED AS UNDER:-**

## Master of Technology(Industrial Engineering and Management), II-SEMESTER

**Result Declaration Date : 29-07-2025** **Notification No: 1867**

IEM502n : Supply Chain Management IEM504n : Advanced Operations Research IEM5321n : Computer Integrated Manufacturing & Robotics IEM5341n : Principles of Management IEM546n : Skill Enhancement Course 2 UCC502n : Research Methodology

| Sr.No | Roll No. | Name of Student | IEM502n | IEM504n | IEM5321n | IEM5341n | IEM546n | UCC502n | SGPA | TC | Failed Courses |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 4.00 | 4.00 | 4.00 | 4.00 | 4.00 | 4.00 | | | |
| 1 | 24/IEM/01 | NAMAN SACHAN | B+ | A | B | B+ | A+ | B | 7.17 | 24 | |
| 2 | 24/IEM/02 | ANJEESH KUMAR UPADHYAY | B | F | F | C | F | F | 1.83 | 8 | UCC502n IEM504n IEM5321n IEM546n |
| 3 | 24/IEM/03 | AMRISH TRIPATHI | C | O | A | A | A+ | C | 7.50 | 24 | |
| 4 | 24/IEM/04 | GULSHAN KUMAR SINGH | B+ | A+ | C | A | O | C | 7.33 | 24 | |
| 5 | 24/IEM/05 | ABHISHEK KUMAR SISODIYA | A | O | B+ | A+ | O | B+ | 8.50 | 24 | |
| 6 | 24/IEM/06 | SUNNY SOREN | B+ | B | B | A | A+ | C | 6.83 | 24 | |
| 7 | 24/IEM/07 | PRIYATOSH BHARADWAJ | B | B | C | B | A+ | C | 6.17 | 24 | |
| 8 | 24/IEM/08 | NGAYOSING A SHIMRAY | B+ | A+ | B | B+ | A | C | 7.00 | 24 | |
| 9 | 24/IEM/09 | SANTOSH KISKU | A | A+ | B+ | A+ | A+ | B | 8.00 | 24 | |

OIC (Results)

Controller of Examination

**Note:**Any discrepancy in the result in r/o name/roll no/registration/marks/grades/course code/title should be brought to the notice of Controller of Examination/OIC(Results) within 15 days of declaration of result, in the prescribed proforma.

# Delhi Technological University

(Formerly Delhi College of Engineering)

**THE RESULT OF THE CANDIDATE WHO APPEARED IN THE FOLLOWING EXAMINATION HELD IN NOV 2025 IS DECLARED AS UNDER:-**

## Master of Technology(Industrial Engineering and Management), III-SEMESTER

**Result Declaration Date : 16-Mar-2026** **Notification No: 1942**

IEM 601n : Industrial Economics & Management IEM 603n : Minor Project/Research Thesis/Patent MOOC605 : MOOC Course

| Sr.No | Roll No. | Name of Student | IEM 601n | IEM 603n | MOOC605 | SGPA | TC | Failed Courses |
|---|---|---|---|---|---|---|---|---|
| | | | 4.00 | 8.00 | 4.00 | | | |
| 1 | 24/IEM/01 | NAMAN SACHAN | B+ | A+ | O | 8.75 | 16 | |

IEM 601n : Industrial Economics & Management IEM 603n : Minor Project/Research Thesis/Patent OBT601 : Human Nutrition

| Sr.No | Roll No. | Name of Student | IEM 601n | IEM 603n | OBT601 | SGPA | TC | Failed Courses |
|---|---|---|---|---|---|---|---|---|
| | | | 4.00 | 8.00 | 4.00 | | | |
| 2 | 24/IEM/03 | AMRISH TRIPATHI | A | A | B+ | 7.75 | 16 | |
| 3 | 24/IEM/04 | GULSHAN KUMAR SINGH | O | A | A | 8.50 | 16 | |
| 4 | 24/IEM/05 | ABHISHEK KUMAR SISODIYA | A+ | A+ | A+ | 9.00 | 16 | |
| 5 | 24/IEM/06 | SUNNY SOREN | A | A | B+ | 7.75 | 16 | |
| 6 | 24/IEM/07 | PRIYATOSH BHARDWAJ | B+ | A | B+ | 7.50 | 16 | |
| 7 | 24/IEM/08 | NGAYOSING A SHIMRAY | A | B+ | A | 7.50 | 16 | |
| 8 | 24/IEM/09 | SANTOSH KISKU | A | O | A | 9.00 | 16 | |