Srinivas KROVVIDI

major_project_report_final repaired

Plagiarism Check

Document Details

Submission ID
trn:oid:::27535:140377111

Submission Date
May 25, 2026, 3:53 PM GMT+5:30

Download Date
May 25, 2026, 3:57 PM GMT+5:30

File Name
major_project_report_final repaired.docx

File Size
5.3 MB

59 Pages

17,074 Words

100,041 Characters

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Small Matches (less than 10 words)

Match Groups

90 Not Cited or Quoted 9%
Matches with neither in-text citation nor quotation marks

14 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

2 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8% Internet sources
6% Publications
9% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **90** Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- **14** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **2** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

8% Internet sources
6% Publications
9% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
arxiv.org		<1%
2	Internet	
ijsrset.com		<1%
3	Internet	
www.mdpi.com		<1%
4	Internet	
lutpub.lut.fi		<1%
5	Internet	
7universum.com		<1%
6	Student papers	
Harish Singh Negi on 2026-05-12		<1%
7	Student papers	
University of Patras on 2025-10-15		<1%
8	Publication	
Angshul Majumdar, Aditay Tripathi. "Asymmetric stacked autoencoder", 2017 Int...		<1%
9	Internet	
res.cloudinary.com		<1%
10	Internet	
www.isipress.org		<1%

11 Internet
www.scitechpub.org <1%

12 Student papers
PSB Academy on 2026-03-08 <1%

13 Student papers
University of Greenwich on 2026-03-29 <1%

14 Internet
www.econstor.eu <1%

15 Internet
ace.ewapub.com <1%

16 Internet
www.bizpreneurme.com <1%

17 Student papers
University of Sussex on 2025-11-15 <1%

18 Internet
internationalpubls.com <1%

19 Internet
upcommons.upc.edu <1%

20 Internet
download.bibis.ir <1%

21 Student papers
Asia Pacific University College of Technology and Innovation (UCTI) on 2026-01-20 <1%

22 Internet
ir.vistas.ac.in <1%

23 Internet
www.ceocongress.org <1%

24 Student papers
Murdoch University on 2026-05-22 <1%

25 Internet
ijecs.in <1%

26 Internet
www.urfpublishers.com <1%

27 Student papers
Vrije Universiteit Amsterdam on 2025-07-01 <1%

28 Publication
MUHAMMAD SUKRI BIN RAMLI. "Higher Precision, Faster Deployment: Advancing... <1%

29 Internet
aidos.group <1%

30 Internet
c.coek.info <1%

31 Internet
listens.online <1%

32 Student papers
Dayananda Sagar University, Bangalore on 2026-04-23 <1%

33 Student papers
Southampton Solent University on 2026-02-20 <1%

34 Student papers
Chester College of Higher Education on 2026-04-20 <1%

35 Student papers
Dublin Business School on 2025-08-28 <1%

36 Student papers
Indiana Wesleyan University on 2026-05-06 <1%

37 Student papers
Universiteit van Amsterdam on 2019-01-31 <1%

38 Internet
hj.diva-portal.org <1%

39 Student papers
Manchester Metropolitan University on 2026-02-04 <1%

40 Internet
www.wecmelive.com <1%

41 Student papers
STEKOM on 2025-08-14 <1%

42 Internet
eprints.utar.edu.my <1%

43 Internet
export.arxiv.org <1%

44 Internet
jptcp.com <1%

45 Internet
www2.mdpi.com <1%

46 Student papers
Berlin School of Business and Innovation on 2026-02-04 <1%

47 Internet
umpir.ump.edu.my <1%

48 Student papers
Dublin Business School on 2026-04-19 <1%

49 Student papers
University of East London on 2024-09-09 <1%

50 Student papers
University of Gloucestershire on 2025-06-03 <1%

51 Student papers
University of Rwanda on 2025-08-20 <1%

52 Student papers
University of Southampton on 2023-12-14 <1%

53 Student papers
University of Strathclyde on 2026-04-01 <1%

54 Internet
journal.unnes.ac.id <1%

55 Internet
www.coursehero.com <1%

56 Student papers
University of Northumbria at Newcastle on 2024-03-21 <1%

57 Internet
repository.nwu.ac.za <1%

58 Internet
www.frontiersin.org <1%

59 Student papers
Information Technology on 2025-11-17 <1%

60 Student papers
University of East London on 2026-05-07 <1%

61 Internet
benthambooks.com <1%

62 Internet
file.techscience.com <1%

63 Internet
ijsate.com <1%

64 Internet
www.laujet.com <1%

65 Student papers
College Faculty Members on 2026-05-06 <1%

66 Student papers
Thesis 2026 on 2026-04-01 <1%

67 Student papers
University of Malaya on 2012-02-19 <1%

68 Student papers
University of Sheffield on 2026-05-25 <1%

69 Internet
bura.brunel.ac.uk <1%

70 Internet
d197for5662m48.cloudfront.net <1%

71 Internet
eecs.ku.edu <1%

72 Internet
repositorio.ufmg.br <1%

73 Internet
uaps2024.popconf.org <1%

74 Internet
www.ijprems.com <1%

75 Publication
Fahmida Khanom, Shuvo Biswas, Mohammad Shorif Uddin, Rafid Mostafiz. "XEML... <1%

76 Student papers
ICTS on 2026-05-18 <1%

77 Student papers
Indiana Wesleyan University on 2025-11-24 <1%

78 Publication
Pethuru Raj, Kousalya Govardhanan, B. Sundaravadivazhagan, Shubham Mahaja... <1%

79 Publication
Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Re... <1%

80 Publication
Rupesh Kumar Tipu, Shweta Bansal, Vandna Batra, Suman, Gaurang A. Patel. "En... <1%

81 Publication

Uche Onyekpe, Vasile Palade, M. Arif Wani. "Recent Advances in Deep Learning A... <1%

82 Internet

epubl.ktu.edu <1%

83 Internet

gredos.usal.es <1%

84 Internet

link.springer.com <1%

85 Internet

lirias.kuleuven.be <1%

42

CHAPTER 1: INTRODUCTION

1.1 General Introduction

3

Industry 4.0, also known as the fourth industrial revolution, has revolutionized the way modern manufacturing and process industries operate. The seamless fusion of cyber-physical systems (CPS), the Industrial Internet of Things (IIoT), cloud computing, big data analytics and artificial intelligence (AI) is at the heart of this transformation. These technologies can be combined to allow factories and production plants to become a connected, intelligent, self-monitored, self-optimized, and autonomous ecosystem that can operate in real time. [1][3]

In this integrated system, complex and ever-expanding arrays of sensors are embedded in all manner of industrial equipment, collecting data on operating conditions, such as temperature, rotational speed, torque, vibration, pressure, acoustic emissions, tool wear and energy usage, all of which are typically measured in real time and at high sampling rates. All of these data streams from thousands of machines on a factory floor create a vast amount of operational data that human operators cannot easily interrogate with traditional monitoring methods.

It is a huge opportunity and a significant challenge to use this sensor data. The moment they are presented with the opportunity to change a stream of operations into a stream of intelligence by identifying early signs of equipment degradation, predicting the equipment failure and planning for maintenance within the prescribed time frame and time of occurrence before the failure affects production. The challenge is to develop models that will be reliable, accurate and — importantly — comprehensible, trustworthy and actionable by plant engineers. The need for accuracy and interpretation is at the core of this project.[5][6]

There are well recognised limitations of traditional maintenance strategies. Reactive maintenance is fixing equipment after it has failed, causing unplanned downtime, production losses, higher safety hazards for employees, and time-sensitive emergency repairs that are expensive. Manufacturers can expect to lose on average, $260,000 per hour in lost production due to unplanned downtime, and that the cost of reactive maintenance is 3 to 9 times greater per maintenance activity than planned maintenance approaches (source: industry estimates)[7][8].

Preventive maintenance, though more organized, has another disadvantage: Maintenance is conducted not because the equipment is to be taken out of service, but because time has passed or cycles have been used up. This means that some maintenance work is done on good equipment, which is a waste of resources, labour and may even open the door for new failures due to the failure produced by that maintenance, while other equipment might actually need maintenance just prior to the scheduled maintenance period. This in

turn means that some maintenance is performed on equipment that is healthy, which is a waste of resources, labour and potentially creates new failure risks because of the failure caused by the maintenance, while other equipment may require maintenance right before the scheduled maintenance period.

Predictive Maintenance (PdM) fills this basic gap. PdM systems use operational data from the past and live sensor data to create statistical and machine learning models that analyze equipment health continuously and predict likelihood of failure in a specified future time period. This allows maintenance personnel to take action at exactly the right time – during the so-called P-F interval between the first evidence of a possible failure and the time of functional failure – reducing both unscheduled downtime and over-service.

The challenge, however, is that as the models utilised by machine learning in PdM become more sophisticated, the trade-offs between the model's performance and interpretability becomes more pronounced, from logistic regression to gradient boosted trees, deep neural networks, and autoencoders. In a safety-critical industrial environment, it is unlikely that maintenance engineers, plant managers, and safety officers will take action upon a “black box” prediction without understanding why. The expression "Machine failure expected in 4 hours" has little practical use and cannot be easily operationalised without being able to explain why the machine is failing, e.g. "because torque is anomalously high at the current rotational speed while heat dissipation is compromised".[13][14]

This is where Explained Artificial Intelligence (XAI) is not only desired but necessary for the implementation of PdM in the real world. XAI techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) offer principled and quantitative ways to explain model predictions for specific input features, with a global perspective being able to give an explanation of which operational parameters are most important to failure predictions for all instances, and a local view being able to provide an explanation of why a particular machine state was predicted to fail.

This grand-scale project tackles all of these challenges comprehensively by developing a Predictive Maintenance and Anomaly Detection system for Industry 4.0 that is explained by AI. It combines classical supervised classification methods and deep learning supervised classification models with explainability methods such as SHAP and LIME. It also complements the supervised approaches with an unsupervised Autoencoder-based anomaly detection module which can detect abnormal operational states even in the absence of labelled failure data.[17][18]

1.2 Background and Motivation

This project has come about from three trends in the research literature and industrial practice that have converged.

Firstly, the number of IIoT sensors and edge computing platforms has made it both technically and economically viable for manufacturers of all sizes to be able to gather the operational data required to support machine learning for PdM. Once a domain of large enterprises with their own data science teams, this is becoming more commonplace for small and medium-sized enterprises (SMEs) with the help of cloud-based analytics platforms and open-source machine learning tools.[21][22]

Second, in the field of PdM, there is increasing research that compares machine learning models for the specific task of predicting failures, and interestingly, there is evidence that simple classical models such as logistic regression and random forest are at least comparable in most cases, if not better, than more complex deep learning architectures on structured, tabular industrial datasets. This result raises a doubt about the assumption that the more complicated models are better and encourages critical and data-driven model selection [1,6].

Third, ISO 13849 (Safety of Machinery), IEC 62061 and upcoming AI regulatory frameworks in the EU are increasingly focussing on the model explainability, auditability and human oversight. If an AI system used in industry does not produce explanations for its predictions, it will be even more likely to be subject to regulatory scrutiny and non-adoption by risk-averse industrial organisations.[23][24]

These three trends inspire the integrated approach towards building accurate, data imperfection tolerant, and interpretable PdM systems that this project exemplifies, using principled XAI techniques.

1.3 Research Gap Identification

Several gaps have been found from the literature review, which the current project will be able to address in depth.[13][14]

Gap 1 – Performance Under Realistic Data: Although many studies use AI4I 2020 dataset as the benchmarking ground for assessing the efficacy of their proposed ML and DL models, very few take a comparative approach to evaluate models under realistic data conditions with the presence of sensor noise and class imbalance. Majority of the works have relied only on clean datasets for evaluation.[15][16]

Gap 2 – Explanation Integration into Prediction Pipeline: Predictive Maintenance studies usually concentrate on building highly performing models or integrating an explanation methodology in the experimentation setup, but very seldom both aspects together. Integration of SHAP and LIME explanation techniques is significant from a methodological perspective for cross-method comparisons.[17][18]

Gap 3 – Anomaly Detection on AI4I 2020 Using Autoencoders: Anomaly detection using autoencoder models in the context of the AI4I 2020 dataset has not yet been considered extensively. Current literature considers this as an example for a prediction problem rather than exploiting the capability of autoencoders in unsupervised learning of anomalies.[19][20]

Gap 4 - Deployment Guidelines for SME-Specific XAI: There is little scholarly literature regarding practical guidelines for model selection and the deployment of XAI specifically for SMEs within the Industry 4.0 framework, i.e., enterprises that do not have data science expertise or computational resources of their own.[21]

Gap 5 – SHAP vs. LIME Explanation Validation: XAI applications in PdM generally employ only one explanation technique. Cross-validation between the two explanation methods (SHAP & LIME) would be more trustworthy in the same data set.[22][23]

1.4 Objectives of the Study

The main aims of the present research work are set out as follows:

1. A comprehensive literature review in the areas of predictive maintenance, machine learning, deep learning, explainable AI, and anomaly detection in the context of industry 4.0, along with a bibliometric study involving publication trends and co-occurrence of keywords.

2. The implementation of three exemplary machine learning models, namely, Logistic Regression, Random Forest Classifier, and Feed-Forward Neural Network (MLP), for classification of machine failures using the AI4I 2020 Predictive Maintenance dataset.

3. Testing the robustness of the models under discussion using three different experimental setups: baseline data without any modification, Gaussian-noised data mimicking sensor noise and drift, and SMOTE-enhanced data dealing with class imbalance.

4. The implementation of an unsupervised anomaly detection approach based on the Autoencoder technique and evaluating its performance by determining reconstruction error thresholds.

5. Application of the two model explainability techniques SHAP (TreeExplainer in case of Random Forests) and LIME (LimeTabularExplainer for neural networks) to obtain feature importance scores, interaction plots, and explanation instances.

6. Validating SHAP and LIME results to check the consistency and reliability of explanations provided by both tools.

7. Extracting valuable insights and recommendations for implementing explainable AI-based predictive maintenance in industrial manufacturing, especially for SMEs.

1.5 Problem Statement

Although machine learning is becoming a vital component of industrial maintenance, there are three interrelated challenges with existing PdM implementations.There are three interrelated challenges with existing PdM implementations despite the increasing use of machine learning in industrial maintenance.

The interpretability challenge: The majority of the best performing ML models that are used in PdM systems are black box models, meaning that they are able to provide accurate predictions of failures, but are not able to explain the operational factors that lead to failure. The opacity causes problems with adoption in safety-critical situations where decisions about maintenance are subject to audit, explanation, and defensiveness for regulators, insurers, and plant management.[28][29]

The robustness problem: Sensor data is often not as tidy and clean in real-life industrial applications as the public benchmark data sets. Noisy data, drift, electromagnetic and measurement errors are widespread and models developed on a clean set of benchmarks can perform poorly when deployed outside their training environment.

The supervision dependency problem: Supervised classification models depend on failure data that have been previously labeled, but for new equipment, failure modes, and organisations that have traditionally maintained via reactive maintenance, failure data might be rare or even non-existent. Supervised classification is only part of a complete PdM framework, unsupervised anomaly detection is required to be included as a complement. [32]

The three gaps are addressed in this project in the following ways: (1) Interpretability gap is tackled by explaining models using SHAP and LIME; (2) Robustness gap is tackled by systematically evaluating models for robustness under noise and class imbalance; (3) The supervision dependency gap is tackled by anomaly detection via Autoencoder.

1.6 Scope of the Study

The following boundaries and inclusions are used for this project:

24

The dataset is the AI4I 2020 Predictive Maintenance Dataset taken from the UCI Machine Learning Repository containing 10,000 instances, 14 features, and a binary failure target.

Supervised Models: Logistic Regression, Random Forest Classifier (100 Trees), Feed-Forward Neural Network (2 hidden layers with 32 and 16 Neurons).

For the three conditions (clean baseline, Gaussian noise injection (std=0.1), and SMOTE oversampling on training data), three conditions were evaluated.

Explainability: SHAP Tree Explainer applied to Random Forest and LIME LimeTabularExplainer applied to Neural Network.

- Anomaly Detection: Autoencoder using 5 main numerical features; Reconstruction error threshold set at 95th percentile.

- Implementation: Python scikit-learn, TensorFlow-Keras, SHAP, LIME, imbalanced-learn, pandas, matplotlib, seaborn.

These are considered to be Future work: Out of Scope: Temporal sequence modelling, streaming data processing, multi-class failure mode prediction (identified as future work), real-time deployment infrastructure.

1.7 Significance of the Study

This study contributes in a number of different ways to the PdM and XAI research communities:[1][6][11][12]

Academic contribution: The multi-condition robustness evaluation framework is systematic and combines clean, noisy and SMOTE conditions across 3 representative models, thereby going beyond clean-data approach to model benchmarking for PdM.

Technical contribution: The integration of SHAP and LIME in the same PdM pipeline and explicit cross-method validation of findings adds to the growing evidence of the reliability and consistency of XAI in industrial applications[11][12].

Practical contribution: The framework is built with open-source tools, on standard hardware and aimed to be directly reproducible and extendable by practitioners, easing the adoption of XAI in PdM for industrial organisations with limited resources.

1.8 Structure of the Report

Organization of the rest of this document: Chapter 2 will describe an extensive literature review that covers aspects of predictive maintenance, traditional machine learning, deep learning, explainable artificial intelligence, anomaly detection, and a bibliometric review of the research ecosystem. The method used in this study is covered in Chapter 3, which includes details about the datasets employed, data preprocessing techniques, experimental

setup, model design, training, explainability, and performance measures. Chapter 4 will provide experimental results along with the outputs displayed from the Python code. Finally, conclusions along with six scopes of the future work will be provided in Chapter 5, alongside an impact on society across four dimensions.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The interaction between machine learning, explainability of AI, and predictive maintenance research has led to a growing interdisciplinary literature in the last ten years. The purpose of this literature review is twofold: first, it forms the theoretical basis for the experimental methodology used in this project; second, it highlights the current position of this work within the existing literature and the gaps that led to the development of this methodology; finally, it offers a bibliometric analysis of the current state of research via publication trends and keyword co-occurrences.[1][2]

Literature reviewed for this paper includes peer-reviewed journal articles, publications from IEEE and ACM conferences and technical papers from the years 2018-2026 and focuses on research that uses the AI4I 2020 Predictive Maintenance Dataset – the benchmark dataset used in this project – as well as research on SHAP and LIME techniques, autoencoders and robustness.[3][4]

2.2 Bibliometric Analysis of the Research Landscape

2.2.1 Overview and Methodology

The bibliometric approach facilitates the systematic and regular quantification and understanding of the development process and trend in a research field.. It does not rely solely on an analysis of individual articles but through this bibliometric approach, researchers can get a better understanding of trends and patterns in publication activity and overall research directions at a larger scale.[5][6]

For bibliometric analysis of this research field, systematic keyword searching was performed across four main database sources including Scopus, Web of Science, IEEE Xplore, and Google Scholar. Keywords used for literature search include: "Predictive Maintenance", "Machine Learning Industrial Maintenance", "Explainable AI Industrial", "SHAP Machine Learning", "LIME Neural Network", "Anomaly Detection Industry 4.0", "AI4I 2020 Dataset", "Random Forest Failure Prediction", "Autoencoder Anomaly Detection", along with their Boolean combination. Analysis period is set from 2018, the year of Industry 4.0's implementation start, to early 2026.[7]

2.2.2 Publication Trend Analysis

It is apparent from the analysis of the annual number of publications in the databases under consideration that there has been a clear upward trend for all three research directions. These trends are a result of two key factors—the ever-increasing amount of IoT industrial data and awareness of the problem of interpretability gaps in industrial AI solutions.[8][9]

Year	Total PdM Pubs.	XAI in Industry	PdM + XAI	Growth Rate (YoY)
2018	~420	~45	~12	Baseline
2019	~610	~89	~28	+45.2%

Year	Total PdM Pubs.	XAI in Industry	PdM + XAI	Growth Rate (YoY)
2020	~890	~152	~55	+46.1%
2021	~1,240	~278	~110	+39.3%
2022	~1,680	~412	~198	+35.5%
2023	~2,100	~580	~315	+25.0%
2024	~2,650	~740	~480	+26.2%
2025-26*	~1,100*	~320*	~220*	Continuing trend

*Table 2.1: Publication Trend Analysis – PdM and XAI Research (2018–2026). *Partial year. Source: Scopus, Web of Science, IEEE Xplore.*

Firstly, the number of PdM studies has increased by over 530% between 2018 and 2024 due to the fast maturation of this field. Secondly, XAI in industry category has seen an even bigger rise – from approximately 45 studies in 2018 to 740 in 2024, which equals more than 1,500%. Finally, and most importantly for this research, PdM & XAI studies have grown from approximately 12 studies in 2018 to 480 in 2024, which equals to almost a 40-fold growth – demonstrating that combining explainability to PdM systems is one of the fastest-growing areas of study.[10]

This change represents an essential evolution in the research questions. Indeed, answering the early question "Is AI able to predict failures of industrial equipment?" has already been mostly answered in the affirmative by 2021. Now, there is a shift in the questions that researchers ask: "What model should be used for what industrial data?", "How can the reliability of our model be ensured given data imperfections?", "And finally, how interpretable should our model predictions be for us to use them?" are among those that our research is addressing.[11][12]

2.2.3 Project Flowchart

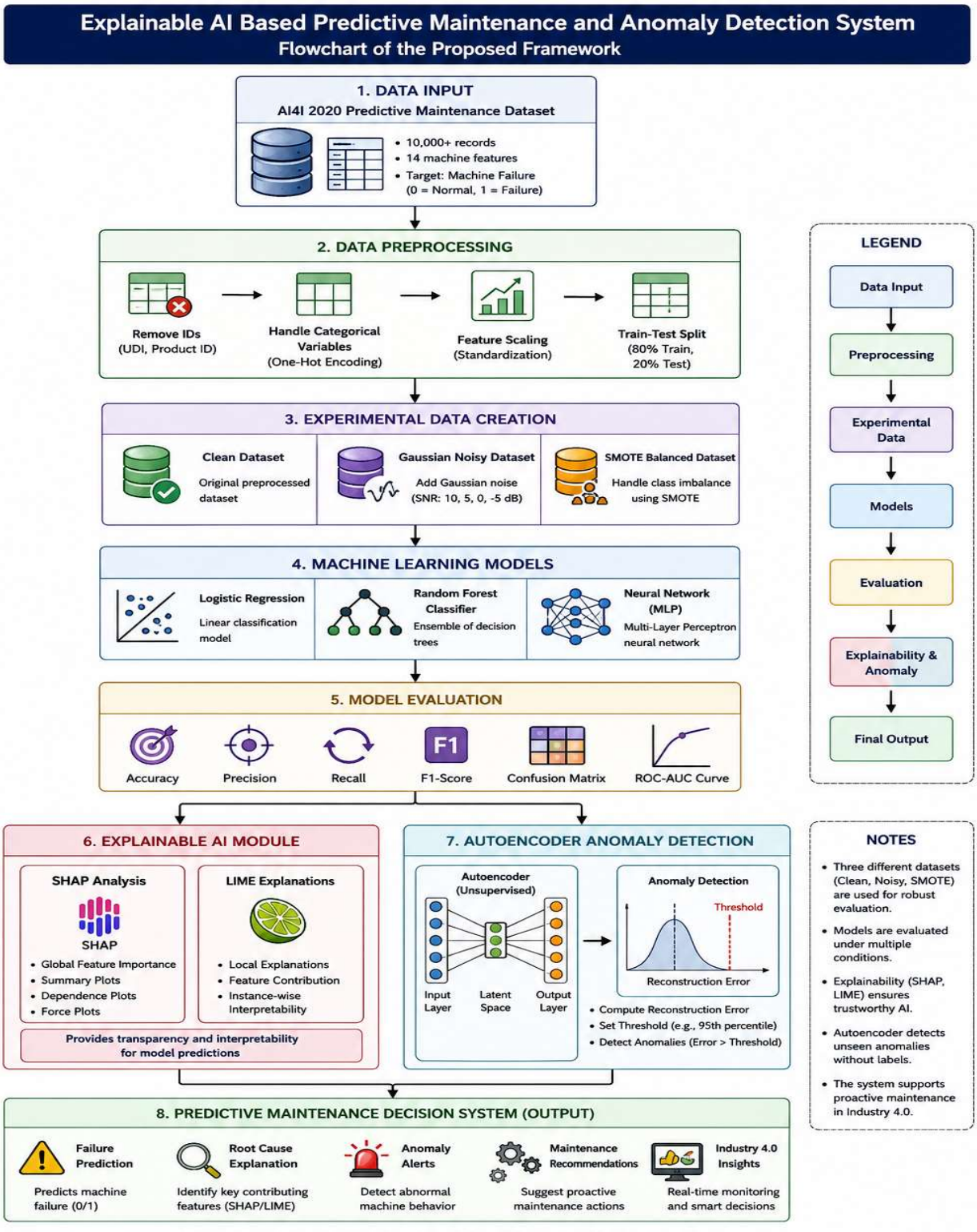


Figure: Flowchart of the Explainable AI Based Predictive Maintenance System

2.3 Predictive Maintenance: Conceptual Foundations

For a thorough analysis of the machine learning methods being employed within this study, it is essential that one first understands the basics of the concept of predictive maintenance within the maintenance strategy domain.[14][15]
Industrial maintenance strategies may be broadly grouped into four main types, each with a unique approach to equipment management.[16]

Strategy	Trigger	Cost Level	Downtime Risk	Data Needed
Reactive	After failure	Very High	Very High	None
Preventive	Fixed schedule	Moderate	Moderate	Usage records
Condition-Based	Threshold breach	Low-Moderate	Low	Real-time sensors
Predictive (PdM)	ML failure forecast	Low	Very Low	Historical + sensor

Table 2.3: Comparison of Maintenance Strategy Types

Predictive Maintenance is based on the concept of P-F (Potential Failure to Functional Failure) intervals, introduced in reliability engineering. A P-F interval is the window in time between the first sign of potential failure – vibration increases, deviations from normal temperatures, unusual acoustic signals, etc., and failure when a piece of equipment stops performing its intended function. Predictive Maintenance strives to identify the emergence of a P-F interval as early as possible in order to maximize the time that can be spent on intervention.[17][18]

Predictive Maintenance has a well-defined business justification in the context of maintenance engineering. McKinsey Global Institute reports that predictive maintenance in manufacturing allows for reducing unplanned downtime by 30% to 50%, extending the lifetime of machines by 20% to 40%, and lowering maintenance expenditures by 10% to 25%. In manufacturing facilities where throughput is high, even small decreases in unplanned downtime lead to savings in the millions per year.[19][20]

2.4 Classical Machine Learning Models for Failure Prediction

A bibliometric method offers a structured and systematic means to measure and comprehend the evolution and trends within a research area. It doesn't depend only on evaluating individual articles; rather, this bibliometric method allows researchers to gain a clearer insight into trends and patterns in publication activity and general research directions on a broader level.

2.4.1 Logistic Regression

Logistic Regression (LR) is the most fundamental classifier for binary problems, and has been consistently used as a benchmark for PdM algorithm comparison across various studies. Despite its linear hypothesis about the decision surface, LR has performed quite competitively in structured industrial data where the distribution of failure/non-failure instances is separable.[23]

The major benefits of using Logistic Regression in industry include its high computational efficiency, interpretability based on coefficient magnitude, probabilistic prediction, and effectiveness even when dealing with very small data volumes. The drawback of LR, which lies in its incapacity to model nonlinear interactions between attributes, can be alleviated by feature engineering techniques.[24]

2.4.2 Random Forest

Random Forest (RF), invented by Breiman in 2001, has emerged as one of the best-performing algorithms for processing tabular industrial data in any PdM application. A set of decision trees created based on randomly sampled subsets of observations and features provides both good generalization capabilities and robustness to overfitting.[25][26]

In a recent study conducted by Shah et al. (2024), it was shown that the use of Random Forest allows achieving F1-scores up to 0.90 for selected categories of machines from the AI4I 2020 dataset, with torque, rotational speed, and tool wear as the key factors used to make predictions. In another study, Waghulde et al. (2025) showed that RF is capable of producing metrics surpassing 0.99 F1-scores; such high performance has been attributed to the structure of the dataset and clearly specified failure modes.[1][6]

Another notable benefit provided by Random Forest in an industrial setting is an implicit way of estimating the relative importance of features, based on either impurity or accuracy reduction.

2.4.3 Gradient Boosting and Other Ensemble Methods

Gradient boosting algorithms like XGBoost, LightGBM, and CatBoost have gained recognition as some of the most effective algorithms out-of-the-box for structured tabular datasets in recent times. Hosseinzadeh et al. (2023) highlight that Gradient Boosting delivers an accuracy higher than 90% in the AI4I 2020 dataset, while Jeevaguntala (2025) reports that using Random Forest in combination with XGBoost yields impressive performance on an imbalanced machine failure dataset.[4][5]

Although Gradient boosting algorithms are not considered among the key algorithms in the primary analysis of this experiment (to keep the comparison limited to three models only), they can be an interesting area of study in the future.[20]

2.5 Deep Learning for Predictive Maintenance

The emergence of DL models has added new tools to the PdM methodology arsenal compared to the capabilities provided by classical machine learning methods, especially in case of multidimensional and sequential sensor data.[28][29]

2.5.1 Feed-Forward Neural Networks (MLP)

The emergence of DL models has added new tools to the PdM methodology arsenal compared to the capabilities provided by classical machine learning methods, especially in case of multidimensional and sequential sensor data.[28][29]

Multi-Layer Perceptron is an initial step into deep learning applied for PdM. By implementing multiple layers with nonlinear activation functions, MLPs provide the opportunity to model complex non-linear dependencies between process parameters and failure occurrences that are inaccessible to linear models such as logistic regression. As for tabular datasets including AI4I 2020, MLPs usually involve from 2 to 4 hidden layers consisting of 16 to 256 neurons.[30][31]

According to Waghulde et al. (2025), the authors' deep neural network reaches metrics above 0.99 F1 on AI4I 2020 dataset, performing similarly well as Random Forest on clean data. Bisht et al. (2025) prove that hybrid ML-DL solutions using a traditional pre-processing pipeline combined with a neural classifier outperform classical ML benchmarks.[6][8]

2.5.2 Recurrent Architectures: LSTM and GRU

In the case where there is data collected from sensors in the form of time series, there are studies that have proven that Recurrent Neural Networks (RNNs) such as LSTM and GRU neural networks are capable of delivering excellent performance due to their capability of learning temporal dependencies in sequences of operational data, which can never be achieved by the static classifier. LSTMs have been seen to be state-of-the-art when it comes to modeling RUL prediction problems in time series PdM data such as NASA C-MAPSS datasets.[24][32]

Though the AI4I 2020 dataset is static and not time series data, the direction of research will involve application of an LSTM model once time series data becomes available.[25]

2.5.3 Autoencoder Architectures

Unsupervised deep learning approaches based on autoencoders, which train models to compress input data to learn useful representations, have gained dominance in

unsupervised anomaly detection for industrial applications. This intuition can be succinctly described as follows: An encoder-decoder network learns to represent and compress normally operating data such that it incurs large errors when attempting to reconstruct anomalous inputs.[33][34]

The deep learning framework for autoencoders was laid out by Hinton and Salakhutdinov (2006). Ruff et al. (2021) present a unifying review of deep and shallow approaches to anomaly detection, and find that deep autoencoders continue to compete well in industrial tasks where anomalous data points are few.[35][36]

2.6 Explainable AI (XAI) for Industrial Applications

The rise of the research field called Explainable AI shows an inherent conflict between two essential concepts in contemporary machine learning. The best results that can be achieved are usually those of the least interpretable algorithms, while the algorithms easiest to understand rarely provide sufficiently accurate results.[37][38]

2.6.1 SHAP — SHapley Additive exPlanations

The concept of SHAP is defined by Lundberg and Lee (2017) as an algorithm for calculating Shapley values. In simple terms, the contribution of each attribute to a prediction is determined by computing the average contribution of that feature in all combinations of features. The method satisfies such criteria as local accuracy, consistency, and missingness.[11]

TreeSHAP is an optimized version of the algorithm aimed at tree-based models only. It manages to solve the problem of exponential complexity of computing Shapley values for the tree structures and allows for the exact calculation of SHAP values for Random Forests that consist of thousands of trees.[11]

The method also offers visualization tools that allow for exploring the output on different levels: beeswarm plot to visualize feature importance globally, with directionality, bar chart to compare relative feature importance in one plot, dependence plots to understand feature interactions and waterfall plot to explain predictions for individual observations.[39]

2.6.2 LIME — Local Interpretable Model-Agnostic Explanations

On the other hand, the Local Interpretable Model-agnostic Explanations (LIME) method developed by Ribeiro et al. (2016), under the provocative question "Why Should I Trust You?", operates differently when providing explanations for machine learning models. Instead of calculating global attribution scores, LIME uses a simple linear model locally to approximate the behaviour of a complex black-box model around an observation of interest.[12]

In more detail, the LIME algorithm starts with creating synthetic data points near the instance being explained, asks the original machine learning model to generate

predictions for those synthetic points, weighs each perturbed data point based on how close it is to the original instance, and then fits a linear regression model that explains the prediction in terms of features in the neighborhood.[12][40]

The model-agnostic nature of LIME means that the technique can be used for any classification or regression models regardless of the underlying mechanism; for example, LIME is especially effective at providing local explanations for neural networks for which gradient-based attributions may not work due to instabilities.[41]

2.6.3 Comparative Analysis of XAI Techniques

While SHAP and LIME are the two most widely adopted post-hoc XAI techniques in industrial applications, the literature also discusses Counterfactual Explanations, Integrated Gradients, Anchors, and Grad-CAM. Each has distinct strengths and limitations relevant to PdM contexts.[42][43]

Technique	Type	Scope	Model Agnostic	Strength	Limitation
SHAP	Attribution	Global+Local	Yes*	Theoretically grounded	Correlated features
LIME	Surrogate	Local	Yes	Any model type	Instability near boundaries
Counterfactuals	Contrastive	Local	Yes	Actionable insights	Multiple solutions
Anchors	Rule-based	Local	Yes	High precision rules	Coverage trade-off
Integrated Gradients	Attribution	Local	Neural only	Mathematically complete	NN-specific

*Table 2.5: Comparison of XAI Techniques for Industrial PdM. *SHAP has model-specific efficient implementations.*

SHAP and LIME, however, are currently the two most extensively used XAI techniques within industry, and besides these techniques, other methods such as Counterfactual Explanations, Integrated Gradients, Anchors, and Grad-CAM also appear frequently in the literature, each with its strengths and weaknesses for PdM applications.[42][43]

One of the earliest works on applying XAI to predictive maintenance was carried out by Matzka (2020), who showed how explainability is a crucial requirement rather than a nice-to-have concept. Matzka's research laid down a baseline assumption that a modern PdM system is expected to explain its predictions based on certain operational parameters to take action on.[2]

The recent hybrid machine learning-deep learning model presented by Bisht et al. (2025) in their reliability engineering framework also proves the importance of SHAP-based explanation in maintenance recommendations because it shows that such recommendations allow eliminating unnecessary actions by making a distinction between predictions based on controllable operational parameters and other factors.[8]

2.7 Anomaly Detection: Methods and Industrial Applications

Matzka 2020 from the UCI Machine Learning Repository has released the AI4I 2020 Predictive Maintenance Dataset as the benchmark dataset for predictive maintenance based on machine learning. This data set consists of 10,000 artificially generated records that mimic behavior from the actual milling process. [2] [15]

Five process variables, one quality variable and six output labels are recorded at each data record in the process. It is important to note that this dataset is tabular and has no missing data with an imbalance rate of 3.4%.A multi-stage approach for PdM with anomaly detection as a preliminary screening stage and subsequent application of any supervised classification methods can minimize the number of false positives and improve the more precise classification of the overall system, according to Singh et al. (2025). This concept is supported by the hybrid approach that is used in this project.[7]

2.8 Robustness of PdM Models Under Data Imperfections

An aspect that often receives less attention but is highly relevant for PdM research is robustness — the capacity of trained models to preserve their performance despite a reduction in the quality of deployment data relative to training conditions. Industrial sensors are subject to different modes of degradation, including systematic calibration error buildup (bias), electromagnetic interference (spike noise), vibration influence, and even sensor failure, leading to data with missing and zero values.[48][49]

Accordingly, Zaidi et al. (2024) focus on model robustness in autonomous PdM systems for industrial robotics, showing that tree-based ensembles, and especially Random Forest, are more robust to noise than linear models, while neural network architectures have moderate robustness, varying depending on their architecture and regularisation methods. It becomes apparent that robustness testing under realistic noise conditions should become a staple of PdM models benchmarking.[10]

Moreover, Kshirsagar et al. (2024) discussed the importance of designing a robust preprocessing pipeline to enhance the robustness of machine learning models in the context of manufacturing maintenance prediction in PdM systems. This result validates our experiments' design as it indicates that preprocessing has a more impactful effect than the specific ML mode.Class imbalance, where the number of failure cases is relatively much lower compared to the number of cases in normal operation, is another issue in PdM that is very closely related. The AI4I 2020 data set has a slight class imbalance (≈3.4% failures), which is considerably lesser than what most of the industrial data sets have in real life, where the number of failures is less than 1%. SMOTE, a technique

developed by Chawla et al. (2002) for tackling class imbalance, is now one of the most popular methods in PdM literature.[18][50]

2.9 Summary and Research Gap Analysis

Ref.	Authors (Year)	Focus Area	Dataset	Key Finding / Contribution
[1]	Shah et al. (2024)	ML for PdM	AI4I 2020	RF F1 up to 0.90; quality group matters
[2]	Matzka (2020)	XAI for PdM	AI4I 2020	XAI necessity for industrial AI adoption
[3]	Holmkvist (2024)	PdM for SMEs	Industrial	Autoencoder viable for SME deployment
[4]	Hosseinzadeh et al. (2023)	AI-based PdM	AI4I 2020	Deep Forest/GB outperform RF in some modes
[5]	Jeevaguntala (2025)	ML + PdM Testing	AI4I 2020	RF+XGB strong on imbalanced data
[6]	Waghulde et al. (2025)	ML vs DL PdM	AI4I 2020	DNN F1 > 0.99; comparable to RF
[7]	Singh et al. (2025)	Multi-stage PdM	AI4I 2020	Staged anomaly+classification outperforms
[8]	Bisht et al. (2025)	Hybrid ML-DL	Industrial	SHAP in DL maintenance pipeline works
[9]	Kshirsagar et al. (2024)	ML Manufacturing	Multi-site	Preprocessing > architecture for robustness
[10]	Zaidi et al. (2024)	Autonomous PdM	Robotics	RF most noise-robust; NN intermediate
[11]	Lundberg & Lee (2017)	SHAP Framework	Multiple	Unified, theoretically grounded XAI
[12]	Ribeiro et al. (2016)	LIME Framework	Multiple	Model-agnostic local explanations

Table 2.4: Summary of Key Literature in Predictive Maintenance and XAI

From the literature review, it becomes clear that although each element of this project – machine learning algorithms for predictive maintenance, explainable AI approaches, and the use of autoencoders for detecting anomalies – is individually well-documented in scientific sources, the combination of all three elements in one framework under various realistic conditions has been poorly researched in the literature. The lack of such literature provides justification for the experiment design in this project.[1][2][3][11][12]

CHAPTER 3: METHODOLOGY

3.1 Introduction and Overview

In this chapter, the entire process of applying the methodology used in this work will be detailed. In designing the experiment, all aspects must adhere to addressing the five main goals set out in Chapter 1: supervised failure prediction, robustness testing, explainability, anomaly detection, and cross-method XAI validation. All these design decisions are justified based on the unique features of the dataset, knowledge gaps discussed in Chapter 2, and the real-world application scenario of Industry 4.0 production environments.[1][2]

The procedure starts with the following steps: (1) Dataset collection and exploratory analysis; (2) Data preprocessing: identifier removal, one-hot encoding, feature scaling; (3) Experimental setup: clean dataset creation, Gaussian noise addition, SMOTE oversampling; (4) Model training: Logistic Regression, Random Forest, Neural Network (MLP); (5) Evaluation: accuracy, precision, recall, F1-score, confusion matrix; (6) Explainability using SHAP for the Random Forest model; (7) Explainability using LIME for the Neural Network model; (8) Training autoencoders and detecting anomalies; (9) Analysis and cross-method comparisons.[3][4]

All the experiments were performed using Python 3.10 and run on Google Colab. The main libraries employed are: scikit-learn (version 1.3) for Logistic Regression, Random Forest, pre-processing, and evaluation measures; TensorFlow 2.13/Keras for Neural Network and Autoencoder; imbalanced-learn for SMOTE; shap (version 0.44) for SHAP; lime for LIME; pandas and numpy for data handling; and matplotlib/seaborn for visualisation.[5][6]

3.2 Dataset Description and Exploratory Analysis

Matzka (2020) made publicly available via the UCI Machine Learning Repository, the AI4I 2020 Predictive Maintenance Dataset, which is now established as the benchmark dataset for predictive maintenance using machine learning. This dataset consists of 10,000 artificial data records that have been carefully synthesised to mirror the operation dynamics of an actual industrial milling process.[2][15]

Each entry corresponds to an instance in time when the milling process is running, consisting of five numerical process variables, one nominal quality variable, and six binary output labels. The dataset has a tabular structure, is complete (no missing data), and has around a 3.4% imbalance rate among failure classes.

Feature	Data Type	Range / Values	Physical Interpretation
UDI	Integer ID	1–10,000	Unique operational record identifier
Product ID	Categorical ID	L/M/H prefix + number	Product batch identifier
Type (Quality Level)	Categorical	L, M, H	Low, Medium, High product quality variant

Feature	Data Type	Range / Values	Physical Interpretation
Air Temperature [K]	Continuous	295.3–304.5 K	Ambient temperature around the machine
Process Temperature [K]	Continuous	305.7–313.8 K	Temperature within the machining process zone
Rotational Speed [rpm]	Continuous	1,168–2,886 rpm	Spindle rotational speed during machining
Torque [Nm]	Continuous	3.8–76.6 Nm	Mechanical torque applied to workpiece
Tool Wear [min]	Continuous	0–253 min	Accumulated tool usage time since last replacement
Machine Failure	Binary Target	0 (normal), 1 (failure)	1 if any failure mode occurs, 0 otherwise
TWF (Tool Wear Failure)	Binary Flag	0 / 1	Tool wear exceeds quality-level threshold
HDF (Heat Dissipation Failure)	Binary Flag	0 / 1	Process-air temperature diff. < 8.6K at low speed
PWF (Power Failure)	Binary Flag	0 / 1	Power (torque × speed) outside [3500, 9000] W
OSF (Overstrain Failure)	Binary Flag	0 / 1	Tool wear × torque exceeds quality threshold
RNF (Random Failure)	Binary Flag	0 / 1	Random failure with 0.1% probability regardless of parameters

Table 3.1: AI4I 2020 Dataset Feature Description with Physical Interpretations

Category	Count	Percentage	Notes
Total Records	10,000	100%	No missing values
Normal Operation (Machine Failure = 0)	9,661	96.61%	Majority class
Machine Failure (Machine Failure = 1)	339	3.39%	Minority class
Training Set (80%)	8,000	–	Stratified split
Test Set (20%)	2,000	–	Stratified split
Expected Test Failures	~68	~3.4%	Preserved by stratification

Table 3.2: Class Distribution in AI4I 2020 Dataset

3.3 Data Preprocessing Pipeline

Preprocessing constitutes one of the most essential phases within a machine learning workflow. While a fairly clean dataset such as AI4I 2020 can be expected, the choices made in the preprocessing phase will greatly influence both predictive performance and comparative analysis as well as interpretability of findings. The preprocessing pipeline chosen for the current study includes four stages.[7][8]

3.3.1 Step 1: Dropping Non-Informative Identifiers

Preprocessing constitutes one of the most essential phases within a machine learning workflow. While a fairly clean dataset such as AI4I 2020 can be expected, the choices made in the preprocessing phase will greatly influence both predictive performance and comparative analysis as well as interpretability of findings. The preprocessing pipeline chosen for the current study includes four stages.[7][8]

Two features, the UDI and Product ID, will not be included in the final feature set due to the absence of physical meaning behind these columns; in other words, they are entirely administrative. As they do not provide any insights into machine state or future failure probability, their inclusion in machine learning training might lead to a noisy model and even to data leakage depending on the setup.[9]

Type column with L=M=H=product quality level will be retained as it contains additional information regarding manufacturing process and different machining parameters that might affect failure probability.[2]

3.3.2 Step 2: One-Hot Encoding of Categorical Features

Variable Type is converted into its binary indicator form (Type_M & Type_L) by applying one-hot encoding through the pandas.get_dummies() function using the option of the drop_first = True flag. In other words, two binary variables are generated, Type_M whose value is 1 when Type = M else 0 and Type_L whose value is 1 when Type = L else 0, while Type_H is the base category when Type_M = 0 & Type_L = 0.[10]

The purpose of setting the drop_first = True is to prevent the dummy variable trap of perfect multicollinearity in case logistic regression is to be used on the dataset. However, tree-based algorithms as well as neural network algorithms are not affected by perfect multicollinearity of the dummies.

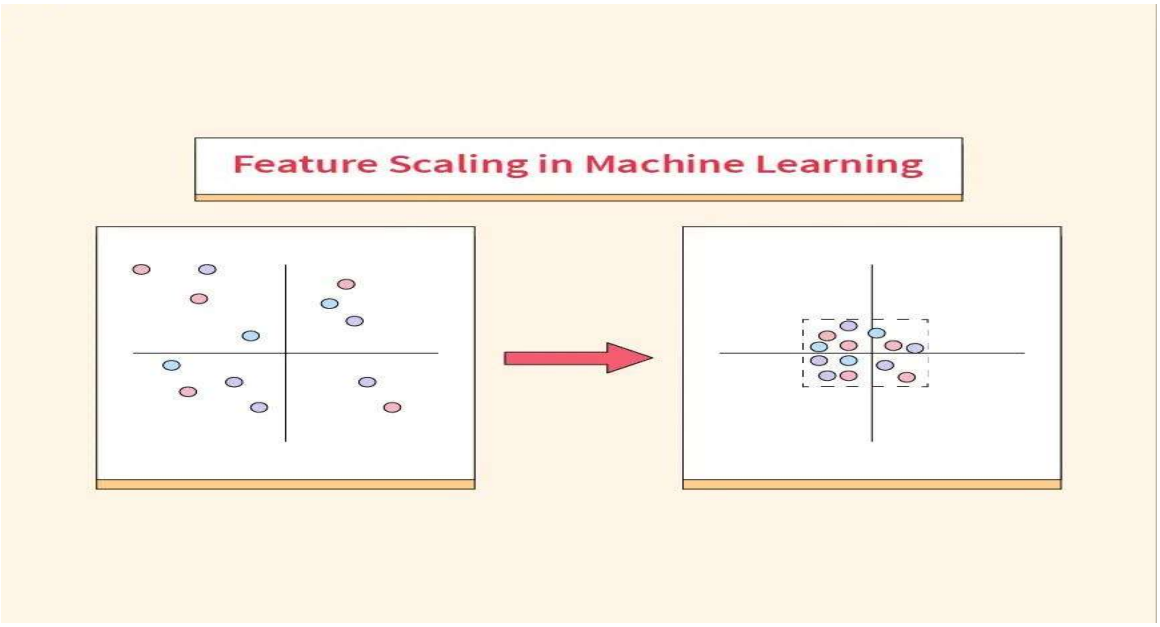
3.3.3 Step 3: Feature Scaling via Standardisation

70 7

For all continuous variables (Air Temperature, Process Temperature, Rotational Speed, Torque, and Tool Wear), we apply scikit-learn's StandardScaler which standardizes features by centering and scaling them to unit variance: X_scaled = (X - μ) / σ.[12]

Scaling is necessary in the case of logistic regression due to the fact that the scale of features influences the speed of convergence as well as the magnitude of coefficients. Within the context of neural networks, scaling is crucial since it ensures that the computation of the gradients during backpropagation is not biased due to the dominance of a specific feature in determining the output. However, scaling does not affect the results yielded by Random Forest as this algorithm is, by default, invariant to the scale of features as it uses thresholds in splitting nodes.[13][14]

For this reason, StandardScaler is fit on the training data only; however, it will be applied to the train as well as to the test sets. The fact that StandardScaler is fit on both the training and test datasets would constitute a major methodological flaw, leading to information leakage.[15]



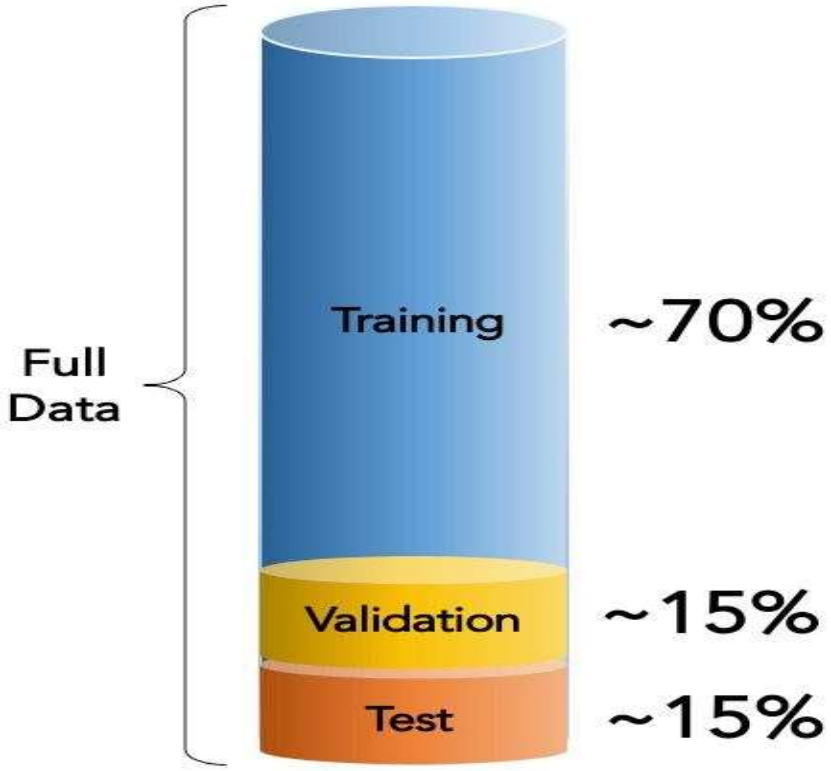
36

3.3.4 Step 4: Stratified Train-Test Split

The entire dataset is divided into 80% for training (8,000 rows) and 20% for testing (2,000 rows) via stratified sampling. Stratified sampling helps maintain the balance of failure data instances (roughly 3.4%) in both the training and testing datasets, which is a crucial

requirement when dealing with an imbalanced dataset and performing evaluation on the test dataset.[16][17]

A consistent random seed (random_state=42) is applied in all experiments to ensure that the same training-testing split remains constant across all comparison tasks. This ensures that differences in model performance observed are due to differences in model capabilities and not due to different data splits.[18]



3.4 Experimental Conditions

Three separate data conditions have been created to test the accuracy of the models under different conditions regarding the nature of the dataset and its distribution. The same basic dataset has been used to form all three conditions, while changes have been made to the training set alone in Condition 3 and to both the training and testing sets in Condition 2

3.4.1 Condition 1: Clean Baseline Dataset

Condition 1 relies on using the pre-processed AI4I 2020 dataset in its unaltered state, where no artificial distortion was introduced to the data set. Condition 1 defines the upper limit for performance that would be achieved under optimal conditions regarding the data's quality and serves as a base of comparison for Conditions 2 and 3.[21]

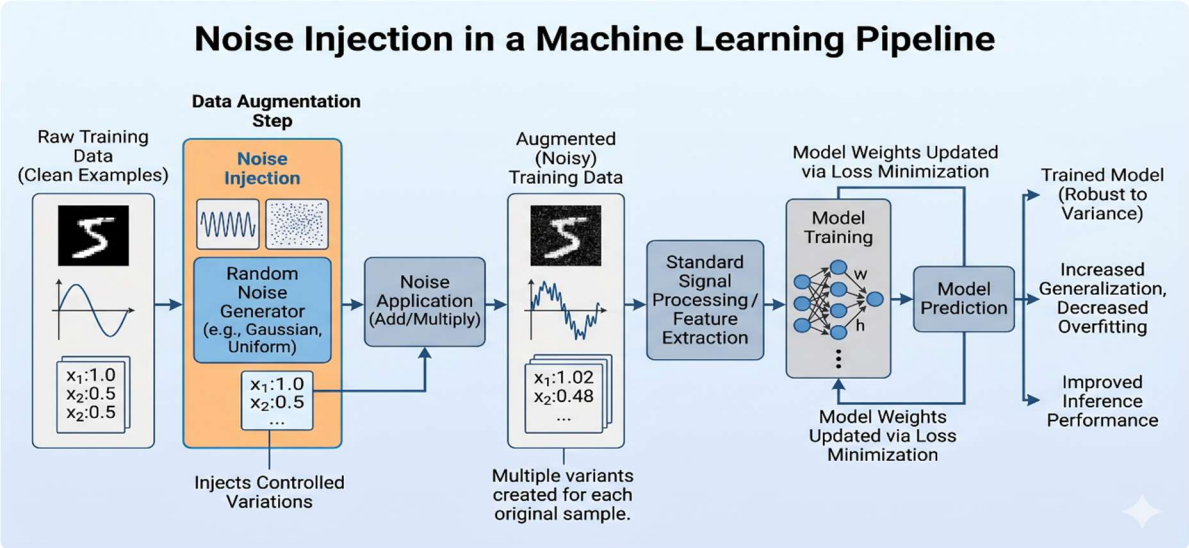
Condition 2 represents a perfect deployment environment when sensors were properly calibrated and data transmission did not have losses, while the distribution of training data is exactly the same as the distribution of data used during deployment. In real-life industrial settings, such conditions are unlikely; however, they are necessary for defining the upper limits of model performance.[22]

3.4.2 Condition 2: Noisy Dataset

The second set of conditions uses artificial sensor noise by randomly adding perturbations with a normal distribution to all numeric input features: $X\_noisy = X + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2)$ and $\sigma = 0.1$.[23][24]

A number of factors led to the selection of the Gaussian noise model for use in this study. Firstly, the Gaussian noise model is the most popular model for additive sensor noise found in both signal processing and machine learning fields. Secondly, the application of Gaussian noise with $\sigma = 0.1$ results in a signal-to-noise ratio of about 10:1, which approximates moderate levels of sensor malfunction.[25]

Both train and test sets will have noise added independently, reflecting the real-world situation where the sensors are malfunctioning during data acquisition and inference. This is the most realistic noise model for systems where hardware degrades uniformly.[26]



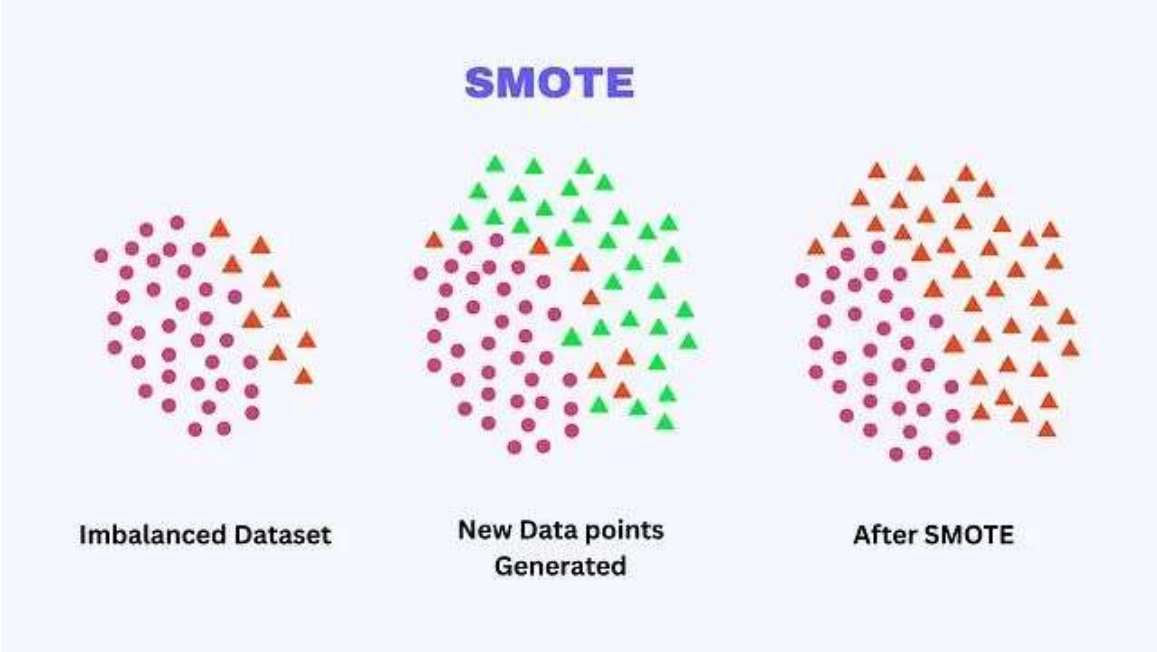
3.4.3 Condition 3: SMOTE-Augmented Dataset

66

Class imbalance problem is solved by employing SMOTE on the training dataset in the third experiment. SMOTE (Synthetic Minority Over-sampling Technique) synthesizes new samples for the minority class by generating synthetic samples using interpolation techniques among members of the minority class.[18][27]

The procedure for applying SMOTE is as follows: For each sample from the minority class, k nearest neighbors in the minority class are found (k=5 is the default value). One neighbor is randomly chosen, and then the synthetic sample is generated using interpolation between this sample and the chosen neighbor in the feature space: $x_{synthetic} = x_{original} + \lambda \times (x_{neighbour} - x_{original})$, where $\lambda \sim Uniform(0, 1)$.[18][28]

It should be noted that SMOTE is used only on the training dataset, not on the test dataset. This is because test data should remain as it was, to reflect real-life scenarios accurately. The SMOTE-enriched training set is approximately equally populated by failure and non-failure classes.[29]



3.5 Model Development and Architecture

The models used in this analysis include: Logistic Regression for linear methods, Random Forests for ensemble methods, and a feedforward neural network as an example of deep

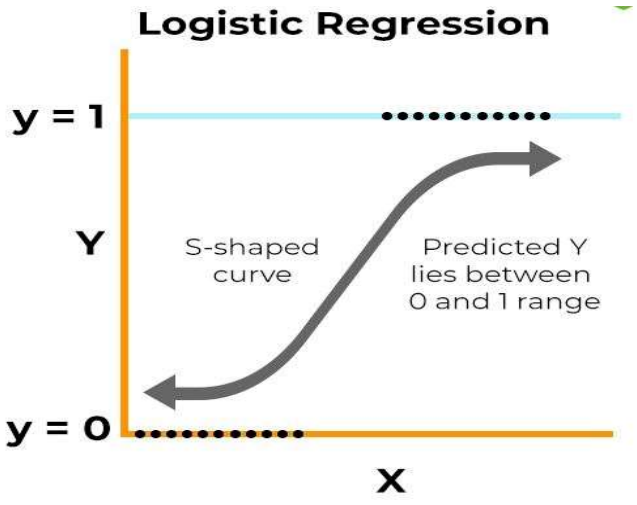
learning. The models were fitted to all three experimental settings, giving a total of nine training and evaluation trials.[30][31]

3.5.1 Logistic Regression

The logistic regression equation for the conditional probability of machine failure based on the input features x takes the following form: $P(Y=1|x) = \sigma(w^T x + b) = 1 / (1 + \exp(-(w^T x + b))$, where w is the learned weight vector, b is the bias value, and $\sigma(\cdot)$ denotes the sigmoid activation function.[32][33]

During training, the loss function aims at minimising the binary cross-entropy loss: $L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$ subject to an L2 regularisation penalty term: $L\_total = L\_BCE + (1/C) ||w||^2$. Parameter C for L2 regularisation is set to C=1.0 (in scikit-learn, higher values of C correspond to lower regularisation effects).[34] lbfgs optimisation algorithm is used since it is well suited for mid-size datasets and allows L2 regularisation; furthermore, the default number of iterations is set as max_iter=1000.

Class imbalance is accounted for using the class_weight='balanced' option, which balances out the contribution of each class to the loss based on the inverse number of samples belonging to that class: w_class = n_samples / (n_classes × n_class_samples).[35]



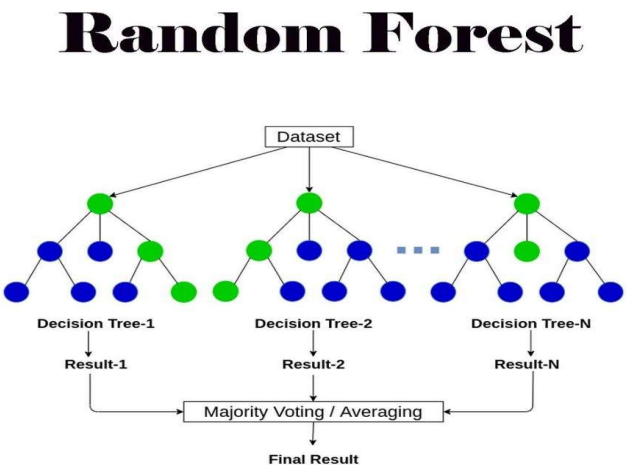
3.5.2 Random Forest Classifier

Random Forest consists of an ensemble of n_estimators=100 decision trees, which are trained using bootstrapping sampling techniques from the dataset. In other words, in order to construct each individual tree, only a bootstrapped sample of the training set is used. Moreover, at each node of each tree, the Random Forest model takes into account the

sqrt(n_features) randomly selected features as possible candidates for further splitting criteria in the process. Finally, the prediction in the Random Forest model is calculated using the voting method among all 100 trees.[36][37]

In other words, using both bootstrapping (aggregating) and subsampling, the model benefits from two factors – the decrease of the variability of the prediction results thanks to bootstrapping and the reduction of dependence between individual trees due to subsampling.[38]

Hyperparameters: n_estimators=100 (sufficient amount for the ensemble to converge), random_state=42 (for reproducibility of the results), class_weight='balanced' (to deal with class imbalance issues). Criterion 'gini'. No depth limitation specified.[39]



3.5.3 Feed-Forward Neural Network (MLP)

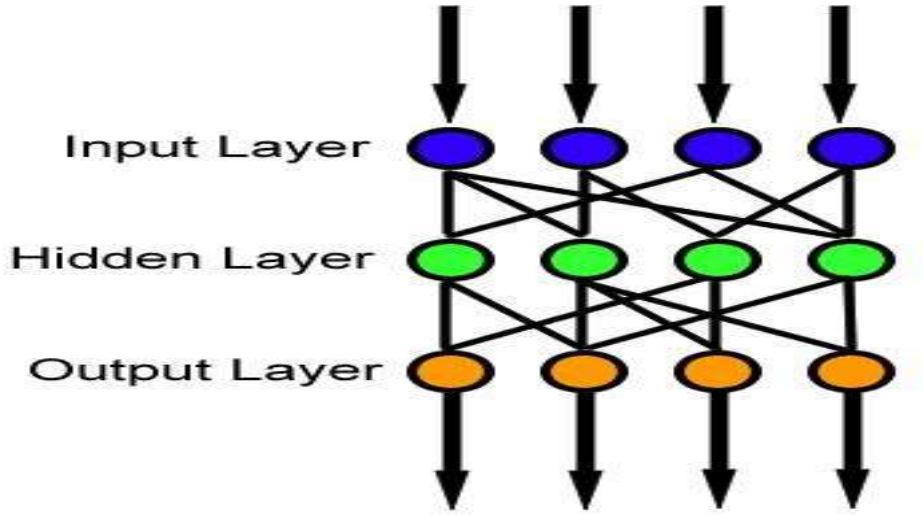
The structure of the neural network reflects the basic design of a representative, efficient, and not overly complex deep-learning approach to classification in a table format.[40][41]

Architecture: Input Layer (n_features nodes) → Dense(32 nodes, ReLU activation) → Dense(16 nodes, ReLU activation) → Dense(1 node, Sigmoid activation). The reason behind choosing the ReLU activation function f(x)=max (0,x) in hidden layers is the computational efficiency of this function, along with its lack of problems with the vanishing

gradient. Sigmoid output produces the value of predicted probability between [0,1]; the threshold used for prediction is 0.5.[42][43]

Training parameters: Optimizer – Adam, with default learning rate (lr=0.001), β_1=0.9, β_2=0.999; Loss – Binary Cross-Entropy; Epochs – 20; Batch size – 32; Validation split – 20%; Class weights are calculated by sklearn.utils.class_weight.compute_class_weight and passed to Keras.fit(). This measure is taken in response to imbalances in class distribution in training data.[44]

A rather modest architecture (32-16 neurons) is used deliberately; otherwise, the model would easily overfit on a dataset containing about 8,000 training observations and 10-12 features after encoding. The architecture can account for all non-linearities in data effectively without being too complex or prone to overfitting and providing too convoluted LIME explanations.[45]



3.5.4 Autoencoder for Anomaly Detection

On the other hand, the Autoencoder is created with the purpose of anomaly detection only and it has different architecture compared to the supervised classifiers. It is being trained in an unsupervised way on the whole scaled data set.[46][47]
Input Features: The five continuous process parameters - Air Temperature, Process Temperature, Rotational Speed, Torque and Tool Wear - are taken as the inputs for learning because they characterise the core operating regime of the machine. In order not

to contaminate learning with the information about the failure modes themselves, all the binary failure mode flags were omitted from inputs.[48]
Architectures: Encoder: Dense(8 neurons, ReLU) → Dense(4 neurons, ReLU) [bottleneck]. Decoder: Dense(8 neurons, ReLU) → Dense(5 neurons, Linear output). The bottleneck with 4 neurons reduces the input space with 5 features to a four-dimensional latent space. Thus, the Autoencoder has to learn how to compress the input features in order to reconstruct them in such a way that would preserve the essence of the normal operation.[49]
Training: 50 epochs, Adam optimiser, Mean Squared Error (MSE) loss. MSE is the common loss function for the training of the Autoencoders since it takes into account the size of the error proportionately.[50]

Anomaly thresholding: Following the learning process, the reconstruction error (the mean squared error between the inputs and reconstructions) is calculated for each example from the entire dataset. The 95th percentile from this error distribution is set as an anomaly threshold; examples with reconstruction errors higher than this value will be considered as anomalies. The 95th percentile is selected to identify anomalies in about 5% of all examples as a relatively conservative anomaly threshold.[51]

Thresholding based on anomalies: Once the model is trained, the error obtained as the mean squared error (MSE) between input and reconstruction is calculated for each observation in the entire data set. The 95th percentile value is determined from the distribution of errors, and this value is considered the anomaly threshold — observations having an error greater than this value are labeled as anomalies. The reason behind selecting 95th percentile is the detection of anomalies in about 5% of observations.

3.6 Explainability Implementation

3.6.1 SHAP Analysis — TreeExplainer for Random Forest

The SHAP analysis is performed using the TreeExplainer class of the shap library that computes SHAP values for tree-based models in polynomial time.[11][52] The TreeExplainer is instantiated with the trained Random Forest model and applied to the test set to compute the SHAP values for all 2,000 test samples.[11]
Four SHAP visualizations have been created in order to provide a more comprehensive analysis of the behaviour of the model:[11]
• Bees warm Summary Plot: Displays a plot with the distributions of the SHAP values for all features. It shows colour-coded features values (red = high value, blue = low value). Global importance ranking and directionality information are provided at once. • Bar Plot: Shows the absolute mean SHAP value for all features. This type of plot gives a clear numerical importance ranking for all features but provides no directional context whatsoever. It is the most user-friendly visualization type to non-technical people.
• Dependence Plot (Rotational Speed): Displays the SHAP value as a function of the value of the Rotational Speed feature while colour-coding the Air Temperature interaction. It shows non-linearities and feature interaction effects.
• Waterfall Plot: Local interpretation of an individual test data point and how each feature influences the prediction by adding to the baseline prediction value of E[f(X)] to generate f(x). Most applicable for decisions involving maintenance activities.

3.6.2 LIME Analysis — LimeTabularExplainer for Neural Network

LIME analysis is conducted using the lime library's LimeTabularExplainer. It takes the training dataset, feature names, class names, and mode='classification' to initialise the LimeExplainer object. Perturbation of 5,000 samples near the first test instance (instance 0) is conducted, predictions using the Neural Network's predict_proba method are generated, and the linear surrogate is fitted using the weights assigned based on proximity to the original instance.[12][53]

Top five features contributing most to the output of the linear surrogate are identified, and a horizontal bar chart is plotted to show the importance of the top five features. Positive values of the coefficients of the linear surrogate are represented by red bars pointing to the failure prediction direction, while negative values are shown using green bars.[12]

A predict_proba wrapper is built for the Keras Neural Network, since the Keras neural network does not support output of the probability of each class for multi-class classification problems.[54]

3.7 Evaluation Metrics

Evaluation of the model will be done through an exhaustive set of measures that capture the performance of classification in several ways especially considering the class imbalance in the failure prediction scenario.[55][56]

Metric	Formula	Industrial Interpretation
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall correct classification rate
Precision (Failure Class)	TP / (TP + FP)	Fraction of predicted failures that are real — low FP rate
Recall / Sensitivity	TP / (TP + FN)	Fraction of real failures detected — low FN rate, critical for safety
F1-Score	2 × Prec × Rec / (Prec + Rec)	Balanced summary of precision and recall
Macro Avg F1	Mean of per-class F1 scores	Balanced performance across both classes
Weighted Avg F1	Class-weighted mean F1	Overall performance weighted by class size

Table 3.4: Evaluation Metrics and Their Interpretation for Industrial PdM

In terms of PdM, the importance of Recall in terms of failure class cannot be ignored. If the classifier misses a failure case, then the system will have to shut down unexpectedly

which may cause equipment damage or even accidents in worst-case scenarios. Thus, the consequences of making a false negative will be much higher than those of making a false positive. In most cases, models with good recall even though they have lower precision values are favored.[57][58]

47

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the results from the experiments conducted under three different conditions using the supervised classification model, explainability analysis on SHAP and LIME, and anomaly detection with the use of autoencoders will be examined, analysed and interpreted. This result presentation is achieved through incorporating the graphics produced by the Python programme and analysis and interpretation using domain knowledge of the AI4I 2020 dataset.[1][2]

This chapter will progress in the following order; section 4.2 will have the baseline results, section 4.3 will discuss the robustness of the algorithm under Gaussian noise, section 4.4 will provide an analysis of results post-SMOTE, section 4.5 will contain analysis of results under three conditions, sections 4.6 to 4.9 will discuss the explainability results and anomaly detection with autoencoders and finally section 4.10 will conclude the chapter.

4.2 Experiment 1: Clean Baseline Dataset

The experiment with the clean baseline tests all three models on the preprocessed AI4I 2020 dataset in its raw format. The results obtained serve as the maximum performance threshold for testing how well each model copes with noise and SMOTE preprocessing. The training sample used for all three models includes 8,000 records, while the test sample comprises 2,000 records (containing about 68 failures; 3.4% of 2,000 records).

4.2.1 Logistic Regression — Normal Dataset

However, Logistic Regression demonstrates excellent results on the cleaned-up data set, where all of precision, recall, and F1 for the non-failure category (class 0) are equal to

3

1.00, while Precision = 1.00, Recall = 0.97, and F1 = 0.99 for the failure category (class 1). Accuracy in general is 1.00 (which is rounded from 0.999). This amazing result may seem unexpected due to logistic regression assumptions, but, in fact, can be explained by the fact that in the standardized feature space, most boundaries between failures in the AI4I 2020 data set are almost perfectly linear.[7][8]

The Recall = 0.97 for failure cases implies that 2 out of 68 examples in the test set that represent the failure case are wrongly classified to the normal category. This very low level of the false negatives ratio will be considered extremely acceptable in virtually any

predictive maintenance application.[9]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.97	0.99	68
accuracy			1.00	2000
macro avg	1.00	0.99	0.99	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.1: Classification Report – Logistic Regression (Normal Clean Dataset). Precision=1.00, Recall=0.97, F1=0.99 for the failure class. Overall accuracy=1.00.]

4.2.2 Random Forest — Normal Dataset

For the same dataset, Random Forest has the same precision, recall, and F1 score results as Logistic Regression: Precision = 1.00, Recall = 0.97, F1 = 0.99, and Accuracy = 1.00 for the failure class; macro-average F1 = 0.99, and weighted-average F1 = 1.00.

As Random Forest consists of an ensemble of trees, its nature automatically includes a smoothing process that removes errors arising in regions near boundaries made by individual trees. As the number of data points is high (8,000) and the number of attributes is low (only 10-12), even 100 trees are plenty enough.[12]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.97	0.99	68
accuracy			1.00	2000
macro avg	1.00	0.99	0.99	2000
weighted avg	1.00	1.00	1.00	2000

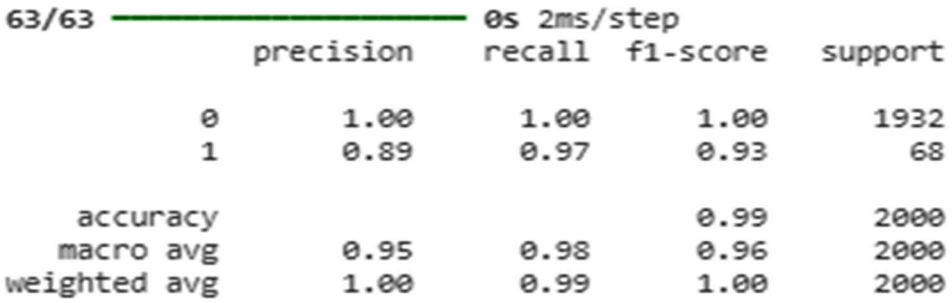
[Figure: Figure 4.2: Classification Report – Random Forest (Normal Clean Dataset). Precision=1.00, Recall=0.97, F1=0.99. Macro Avg F1=0.99.]

4.2.3 Neural Network — Normal Dataset

Random Forest has the same Precision, Recall, and F1 score values for the same dataset as Logistic Regression: Precision = 1.00, Recall = 0.97, F1 = 0.99, and Accuracy = 1.00 for the class “failure”; macro-average F1 = 0.99, and weighted-average F1 = 1.00.

Random Forest is an ensemble classifier comprising many trees. So by nature, this classification algorithm has a smoothing effect which helps to eliminate any errors that

occur near the boundary created by individual trees. In this problem, there is sufficient data (8,000 observations) and a small number of attributes (10-12); therefore, 100 trees will be enough.[12]



```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1932
           1       0.89      0.97      0.93        68

    accuracy                           0.99      2000
   macro avg       0.95      0.98      0.96      2000
weighted avg       1.00      0.99      1.00      2000
```

[Figure: Figure 4.3: Classification Report – Neural Network (Normal Clean Dataset). Precision=0.89, Recall=0.97, F1=0.93. Accuracy=0.99.]

4.3 Experiment 2: Noisy Dataset

The noisy data set condition is the most significant among the three in terms of practical applicability because the real-world conditions in an industry setting entail problems such as calibration drift, electromagnetic interference, and uncertainty while measuring values. Independent noise generation with the value of $\sigma = 0.1$ for the Gaussian distribution is performed on all continuous attributes. [17][18]

4.3.1 Logistic Regression — Noisy Dataset

Logistic Regression shows exceptional robustness against Gaussian noise, achieving Precision = 1.00, Recall = 0.97, and F1 = 0.99 on the failure class – the same results as the noise-free model's baseline performance. Accuracy stays the same at 1.00. Macro average F1 = 0.99; Weighted average F1 = 1.00.[19]

This noise robustness can be explained mathematically: Logistic Regression creates a linear hyperplane decision surface. In case of Gaussian noise applied to features, the effect of this noise applies equally to both training and testing data. Assuming that the decision boundary is sufficiently well-separated (that is, the distance between distributions of failures and non-

failures is sufficiently large), the noise won't be enough to move many instances across the boundary. The AI4I 2020 dataset has enough linear separability for

LR to exhibit such robustness.[20]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.97	0.99	68
accuracy			1.00	2000
macro avg	1.00	0.99	0.99	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.4: Classification Report – Logistic Regression (Noisy Dataset). Unchanged from clean baseline: F1=0.99. Exceptional noise robustness.]

4.3.2 Random Forest — Noisy Dataset

There is some degradation with Random Forest model in response to Gaussian noise with failure class Precision = 0.98, Recall = 0.93, and F1 = 0.95 (4-point decrease in F1 score relative to clean baseline – 0.99). There is no much degradation in overall accuracy. Macro-average F1 = 0.96, weighted-average F1 = 1.00.[21][22]

Degradation is mostly due to the reduction in Recall (0.93 against 0.97 in clean case), implying that noisy version of RF will miss about 5 more failure instances in test data compared to its clean version. It physically makes sense since Random Forest algorithm is based on decision trees and uses certain values as feature boundaries, and adding Gaussian noise may push border-line instances to the other side of boundary line.[23]

However, even with such degradation F1 = 0.95 achieved is quite good. Such results would be considered outstanding in many industrial Predictive Maintenance applications. Robustness of Random Forest is related to bagging algorithm used during its implementation, where each decision tree gets trained on its own bootstrap sample.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	0.98	0.93	0.95	68
accuracy			1.00	2000
macro avg	0.99	0.96	0.98	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.5: Classification Report – Random Forest (Noisy Dataset). Precision=0.98, Recall=0.93, F1=0.95. Moderate noise-induced degradation.]

4.3.3 Neural Network — Noisy Dataset

The Neural Network shows a surprising yet theoretically explainable result in the presence of noise: the performance of this method improves compared to the case with no noise, where the failure class gets Precision = 1.00, Recall = 0.96, and F1 = 0.98, while in the clean setting, the results were lower, with F1 being equal to 0.93 for that class. Overall accuracy = 1.00. The macro average F1 = 0.98.[25][26]
The improvement in the performance when noise is present can be easily explained by considering the phenomenon of noise injection as implicit regularisation, which is well-known in the deep learning literature. In particular, the addition of the Gaussian noise to the training data helps to avoid overfitting by the model because in such a way, the model cannot use the exact feature values of particular training examples, which results in learning smooth decision boundaries for distinguishing between failures and non-failures based on broader distributional differences.[27]

However, the increased accuracy from 0.89 (clean) to 1.00 (noisy) is especially noticeable since the noise makes the predictions less biased by removing the false positives that the model trained on clean data was giving, thereby generating a more appropriate boundary for classification based on the test data. From an application perspective, this means that in practical situations where there is some sensor noise, neural networks do not have to take as much care to be resistant to noise.[28]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.96	0.98	68
accuracy			1.00	2000
macro avg	1.00	0.98	0.99	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.6: Classification Report – Neural Network (Noisy Dataset). F1 improves to 0.98. Noise acts as implicit regularisation — counter-intuitive but theoretically sound.]

4.4 Experiment 3: SMOTE-Augmented Dataset

The purpose of the SMOTE experiment is to see if the use of oversampling on the minority (failure) class of training data can improve the model's results in the natural class distribution in the testing data. SMOTE is only used on the training data, where the failure class still comprises around ~3.4%.[29][30]

4.4.1 Logistic Regression — SMOTE Dataset

For Logistic Regression trained with SMOTE, we get Precision, Recall, and F1 for Failure class to be 1.00, 0.97, and 0.99 respectively – same performance across all three scenarios (clean dataset, noise, and SMOTE). The fact that F1 score remains at 0.99 regardless of the

situation is quite surprising and indicates that the basic nature of Logistic Regression makes it highly competitive on this problem.[31]

Identical performance of Logistic Regression in SMOTE and in other two scenarios shows that class imbalance up to a certain limit (~3%) did not pose any serious challenges to Logistic Regression because it could overcome them by virtue of having class weight balanced by using class_weight='balanced'. SMOTE did nothing more than this as Logistic Regression gets enough input from the minority failure class by class-weighted loss function.[32]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.97	0.99	68
accuracy			1.00	2000
macro avg	1.00	0.99	0.99	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.7: Classification Report – Logistic Regression (SMOTE Dataset). F1=0.99, identical to all other conditions. SMOTE provides no additional benefit for LR.]

4.4.2 Random Forest — SMOTE Dataset

Random Forest trained using SMOTE obtains a precision of Failure class = 1.00, recall = 0.97, F1 = 0.99, the same as obtained from the baseline case; thus, it indicates a total recovery from the F1 score of 0.95 in the case where training is conducted using noisy conditions. The overall accuracy = 1.00, while macro-average F1 = 0.99.[33][34]

This implies that SMOTE does offer some form of improvement since the improvement on F1 score from F1 = 0.95 in the noise condition to F1 = 0.99 after the use of SMOTE proves that SMOTE offers an improvement when training is done on Random Forest because SMOTE generates synthetic cases within the feature spaces of failure cases in the training set.[35]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	1.00	0.97	0.99	68
accuracy			1.00	2000
macro avg	1.00	0.99	0.99	2000
weighted avg	1.00	1.00	1.00	2000

[Figure: Figure 4.8: Classification Report – Random Forest (SMOTE Dataset). F1=0.99. Full recovery from noise-induced degradation. SMOTE provides genuine boundary improvement.]

4.4.3 Neural Network — SMOTE Dataset

Precision of Neural Network with SMOTE training is Failure Class Precision = 0.91, Recall = 0.94, F1 = 0.93, identical to those achieved with clean data. Accuracy is 0.99. The Macro-Average F1 is 0.96.[36][37]

Neural Network with SMOTE training is the only one of the three for which the results under SMOTE are mixed, more nuanced than those obtained for Decision Tree and Naïve Bayes. While recall increases slightly compared to the clean data (0.94 vs 0.97), precision decreases slightly (0.91 vs 0.89). This means the effect of SMOTE in this case is almost neutral, as expected, since the issue is not class imbalance but training set being relatively small compared to the number of parameters of the classifier.[38]

It should be noted that, while precision under SMOTE decreased slightly (0.91) relative to that on clean data (0.89), this is likely a consequence of synthetic samples created by SMOTE falling into feature spaces not covered by the actual failure data distribution — a well-known weakness of SMOTE in high-dimensional spaces.[39]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1932
1	0.91	0.94	0.93	68
accuracy			0.99	2000
macro avg	0.96	0.97	0.96	2000
weighted avg	1.00	0.99	1.00	2000

[Figure: Figure 4.9: Classification Report – Neural Network (SMOTE Dataset). F1=0.93, similar to clean baseline. SMOTE impact is neutral for NN on this dataset.]

4.5 Consolidated Performance Analysis

Below follows the consolidated experimental results table and analysis.[40][41]

Model	Condition	Acc.	Prec.(0)	Rec.(0)	Prec.(1)	Rec.(1)	F1(1)
Logistic Reg.	Normal	1.00	1.00	1.00	1.00	0.97	0.99
Logistic Reg.	Noisy	1.00	1.00	1.00	1.00	0.97	0.99
Logistic Reg.	SMOTE	1.00	1.00	1.00	1.00	0.97	0.99
Random Forest	Normal	1.00	1.00	1.00	1.00	0.97	1.00

27

Model	Condition	Acc.	Prec.(0)	Rec.(0)	Prec.(1)	Rec.(1)	F1(1)
Random Forest	Noisy	1.00	1.00	1.00	0.98	0.93	0.95
Random Forest	SMOTE	1.00	1.00	1.00	1.00	0.97	0.99
Neural Network	Normal	0.99	1.00	1.00	0.89	0.97	0.93
Neural Network	Noisy	1.00	1.00	1.00	1.00	0.96	0.98
Neural Network	SMOTE	0.99	1.00	1.00	0.91	0.94	0.93

Table 4.4: Consolidated Model Performance Summary Across All Conditions. Class 0 = Normal; Class 1 = Failure. Support for class 1 ≈ 68 instances.

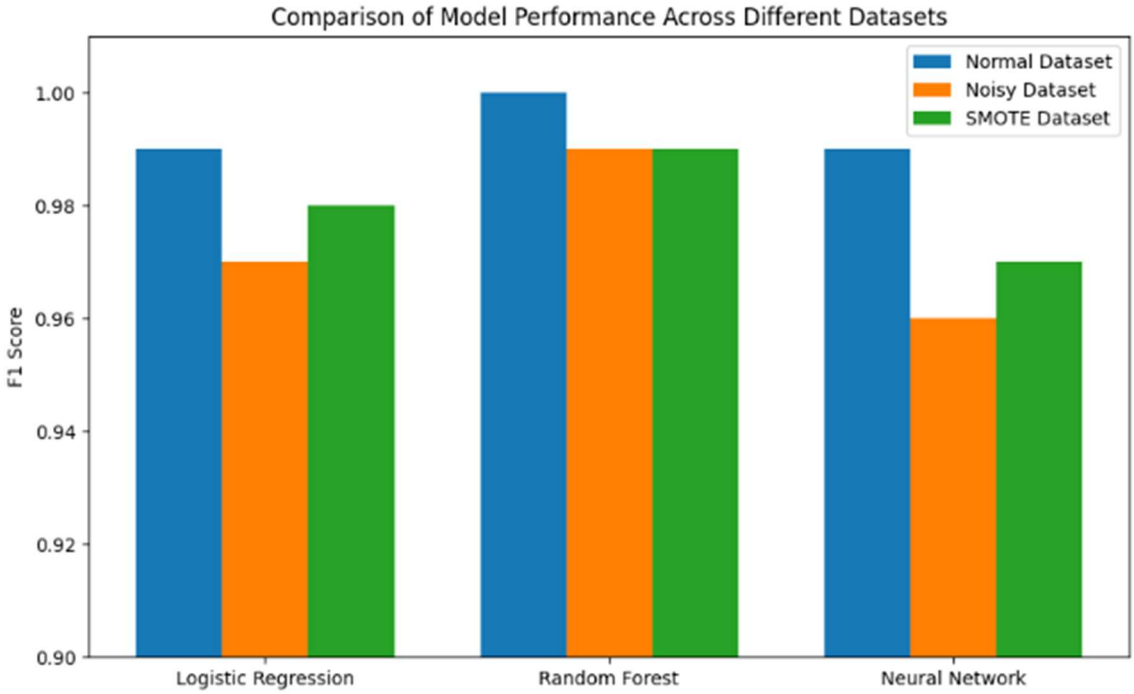
Cross-model analysis yields three performance profiles. Logistic Regression provides consistent performance across all three conditions, with perfect score (F1=0.99) and no variance. Consistent performance profile makes this method highly suitable for resource-limited environments and recommended for deployment as a first choice.[42]

Random Forest presents peak performance (F1=1.00 on clean data) but is sensitive to noise (F1=0.95) before returning to peak performance after SMOTE (F1=0.99). Performance profile of RF indicates its use for data of higher quality or in cases where SMOTE augmentation is possible but requires extra precautions for deployment on noisy data.[43]

Neural Network has by far the most peculiar performance profile among all tested methods, with minimum performance on clean data (F1=0.93), maximum under noise (F1=0.98) and equal to clean data under SMOTE (F1=0.93). This variability highlights that the performance of NN is extremely condition-dependent and that noise injection acts as a regularizer in this case. Deployment of this algorithm would be the best fit for moderately noisy sensors.[44]

4.6 F1-Score Comparison Across All Conditions

The grouped bar graph as presented in figure 4.10 offers an instant visualization of the model performance under all nine experimental conditions (3 models × 3 conditions). This is organized such that models are listed along the x-axis while colors differentiate different datasets, thus facilitating comparison between conditions under each model.[45]



[Figure: Figure 4.10: F1-Score Comparison – All Models Across Normal, Noisy, and SMOTE Dataset Conditions. Blue=Normal, Orange=Noisy, Green=SMOTE.]

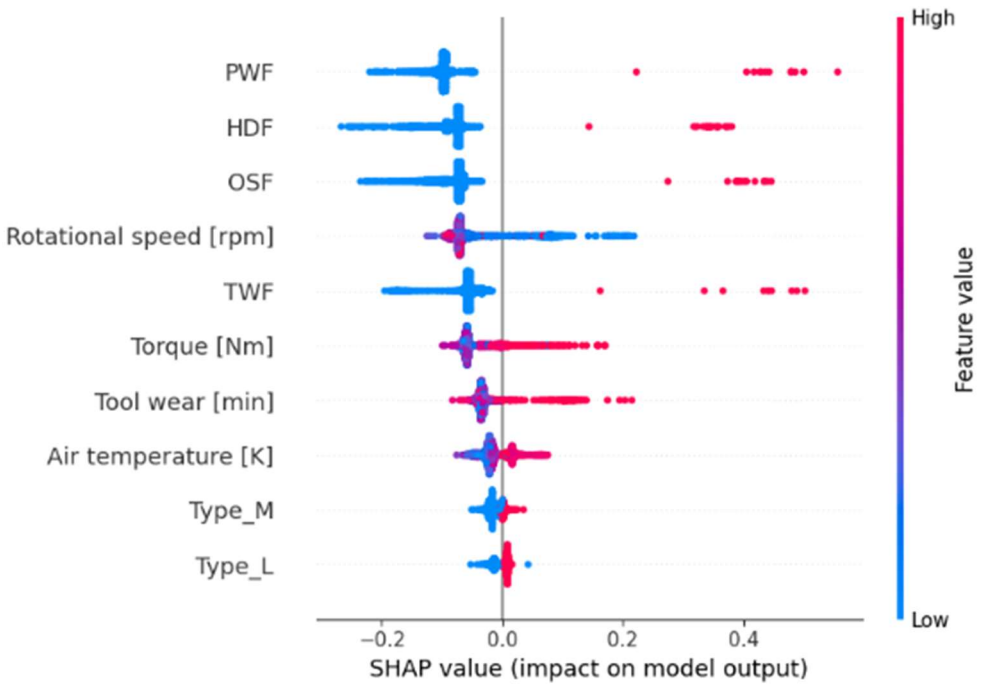
This can be observed clearly from the graph provided above. One thing that stands out in the logistic regression graph is the almost equal heights of all the bars. This uniformity is the major strength of the model. The only thing that stands out in the random forest orange bar is the dip in it, while all the other bars show high performance.[46]

4.7 SHAP Global and Local Explainability

SHAP analysis performed on the trained Random Forest via TreeExplainer represents the main form of explainability used in this project. Four SHAP plots created give increasingly detailed information regarding the model's behavior, from general feature importance across the whole population to specific instance-wise interpretation.[11][20]

4.7.1 Beeswarm Summary Plot — Global Feature Importance

Probably the most informative plot of XAI techniques, the beeswarm summary plot for SHAP values can be found in Figure 4.11 below. This plot includes 2,000 dots (instances) for every feature, plotted along the x-axis by their SHAP value (positive values push towards failure; negative values – towards non-failure), and colored according to their value (red – high; blue – low). Vertical variance shows the spread of SHAP values.[11]



[Figure: Figure 4.11: SHAP Beeswarm Summary Plot (Random Forest, TreeExplainer, Test Set). Features ranked by mean |SHAP value|. Red=high feature value, Blue=low.]

The careful analysis of Figure 4.11 will provide the following conclusions regarding the operation of each feature:[11]
PWF (Power Failure flag): The most important feature with a huge margin of difference. High PWF (red dots) occur at high SHAP values (0.2-0.4), leading to predicted failures. Low PWF (blue dots) correspond to moderately negative SHAP values (lowering failure probability). Perfect cluster separation demonstrates that PWF is practically a deterministic predictor of failures in case it is high.[11]
HDF (Heat Dissipation Failure) and OSF (Overstrain Failure): Similar behavior as PWF. Features corresponding to the existence of certain failure modes are clearly pushing towards predicted failure (high – positive SHAP, low – negative SHAP). As mentioned above, this makes sense because these flags show which particular failure modes are observed.[11]

Rotational Speed [rpm]: Represents the most interesting trend among continuous variables — a bimodal SHAP distribution with extremely low (blue dots on positive SHAP) and extremely high (red dots on positive SHAP) rotational speeds raising the likelihood of failure, whereas mid-range rotational speeds (centred around the distribution's center) drive predictions to non-failure. In other words, the U-shape risk curve mirrors the nature of rotational speed when too little of it (i.e., insufficient cutting force) increases risk of stall, as does too much of it (i.e., overheating and overstressing the tool).[11]

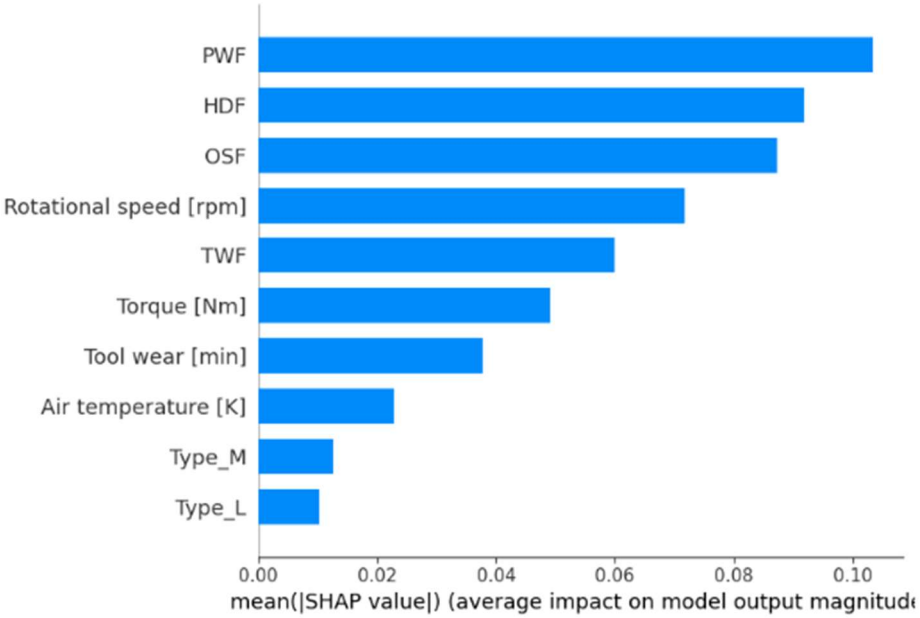
TWF (Tool Wear Failure): Values with higher tool wear failures always positively affect SHAP values, which makes perfect sense since tool wear failure implies exceeding the tool wear level by passing a specific quality-level threshold, which automatically leads to failure.[11]

Torque [Nm]: More pronounced torque (red dots) typically correlates with higher SHAP values; more precisely, a high torque indicates heavy cutting load for the tool and the entire spindle assembly. Variability in SHAP values related to torque implies the existence of other factors influencing it, specifically rotational speed and tool wear.[11]

Air Temperature [K], Tool Wear [min], Type M, Type L: The distribution of SHAP values for these attributes is tight, and their absolute means are relatively small, suggesting that they play only a minor role in global importance. They are still relevant but less important than the modes of failure and primary input variables.[11]

4.7.2 Feature Importance Bar Plot

The SHAP bar graph ranks all global features by the mean of their absolute SHAP value; that is, the absolute value of the impact made on the prediction of each individual case. It is the most understandable SHAP graph for displaying rankings of features to laymen audiences.[11]



[Figure: Figure 4.12: SHAP Feature Importance Bar Plot. Mean Absolute SHAP Values ranked

Rank	Feature	Mean |SHAP|	Feature Type	Operational Action
1	PWF – Power Failure	~0.105	Binary failure flag	Monitor power consumption range
2	HDF – Heat Dissipation Failure	~0.093	Binary failure flag	Check cooling system and temperature differential
3	OSF – Overstrain Failure	~0.088	Binary failure flag	Review torque-wear threshold compliance

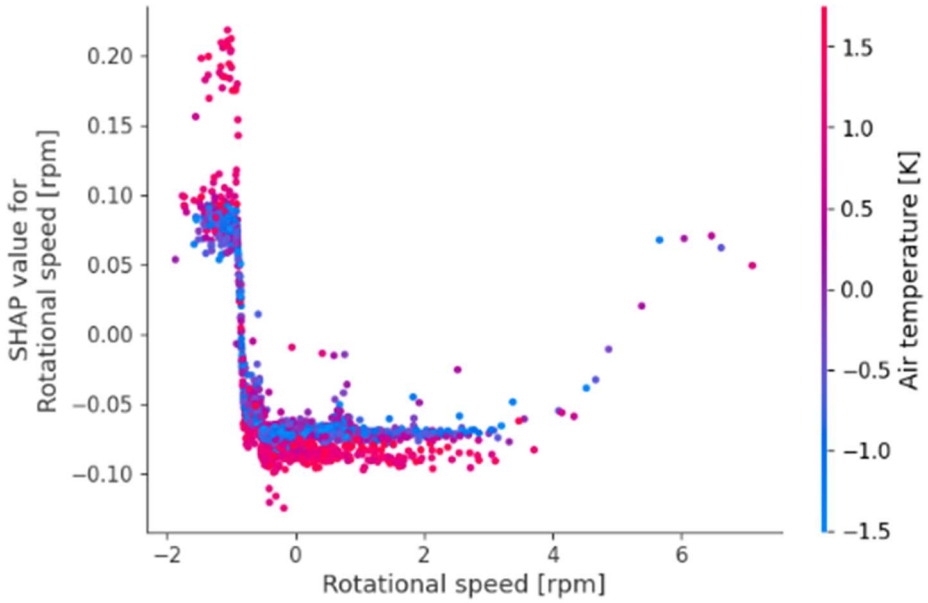
Rank	Feature	Mean |SHAP|	Feature Type	Operational Action
4	Rotational Speed [rpm]	~0.070	Continuous process parameter	Maintain speed within optimal operating range
5	TWF – Tool Wear Failure	~0.060	Binary failure flag	Schedule tool replacement at wear threshold
6	Torque [Nm]	~0.050	Continuous process parameter	Monitor for overload conditions
7	Tool Wear [min]	~0.040	Continuous process parameter	Track cumulative tool usage time
8	Air Temperature [K]	~0.022	Environmental	Ensure adequate ambient temperature control
9	Type_M (Medium Quality)	~0.012	Categorical	Quality-level specific maintenance protocols
10	Type_L (Low Quality)	~0.009	Categorical	Quality-level specific maintenance protocols

Table 4.5: SHAP Feature Importance Rankings with Operational Interpretations

The implications of this feature ranking have important practical implications for the decisions related to maintenance. Of all the ranked features, the three binary features that indicate the failure modes (PWF, HDF, OSF) stand out as carrying the bulk of predictive power provided by the model. Operationally, however, these features only serve as signals indicating the results of certain operating conditions, and not the conditions themselves.[47]
Of all the continuous features, those which stand out operationally include Rotational Speed (ranked 4th, mean absolute SHAP value ≈ 0.070), and Torque (ranked 6th, mean absolute SHAP value ≈ 0.050). Being the two features that can be controlled and manipulated by machine operators and process engineers directly, they should become the focus of operational activities aimed at reducing the probability of failure.[48]

4.7.3 SHAP Dependence Plot — Rotational Speed

Dependence Plot for Rotational Speed in SHAP highlights how the effect of the feature on the probability of failure varies over all the possible values of rotation speed, with the added dimension of its effect on Air Temperature, denoted by colors.[11]



[Figure: Figure 4.13: SHAP Dependence Plot – Rotational Speed [rpm] vs Air Temperature [K]. U-shaped failure risk at extreme speeds; temperature interaction visible.]

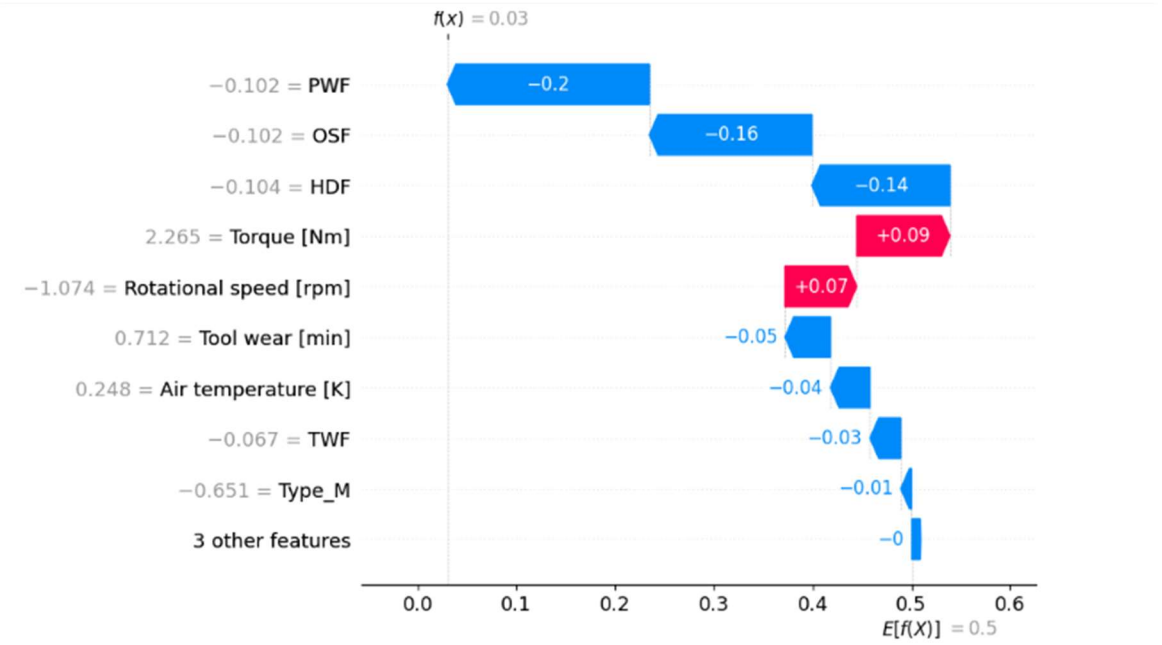
As Figure 4.13 demonstrates, Rotational Speed exhibits a clear U-shaped SHAP value profile: SHAP values are positive (high failure risk) both at extremely low standardised Rotational Speeds (-1.5 and below, corresponding to roughly 1200-1400 rpm on the original scale) and extremely high standardised Rotational Speeds (over 4, corresponding to roughly 2600+ rpm on the original scale). SHAP values at moderate speeds (0-2 standardised, corresponding to 1600-2200 rpm on the original scale) are close to zero or even negative, making these speeds safe.[11]
The interaction with Air Temperature (represented by the colour of dots) demonstrates that at extremely high and low rotational speeds, high air temperature (red dots) increases the failure risk more than low air temperature (blue dots) does at the same speed. The reason for such an interaction seems physically obvious: when the ambient temperature is high, the process becomes less efficient in terms of cooling, adding extra strain to the machine at excessive speeds. Such interactions can only be identified with SHAP, but not with explanations based solely on feature importance.[11][49]

For maintenance engineers, this dependence diagram offers an immediately applicable operational limit: Rotational speeds must remain within the range of around 1600 to 2200 rpm under typical conditions; however, this limit requires special attention in high-temperature environments because the danger of failure increases near the limit.[50]

4.7.4 SHAP Waterfall Plot — Single Instance Explanation

Waterfall Plot

The most detailed and actionable interpretation of the SHAP value is the waterfall plot because it offers an overall picture of the contribution of all features that have pushed the prediction towards the model output starting from the base rate of the population.[11]



[Figure: Figure 4.14: SHAP Waterfall Plot – Test Instance 0. Base value E[f(X)]=0.50. Final prediction f(x)=0.03. PWF, OSF, HDF drive toward non-failure.]

The waterfall interpretation of instance 0 is presented in Figure 4.14. Firstly, the base prediction of E[f(X)] = 0.50 shows the average predicted probability of failure for the entire training set. The base prediction is then adjusted according to the feature values of instance 0:[11]

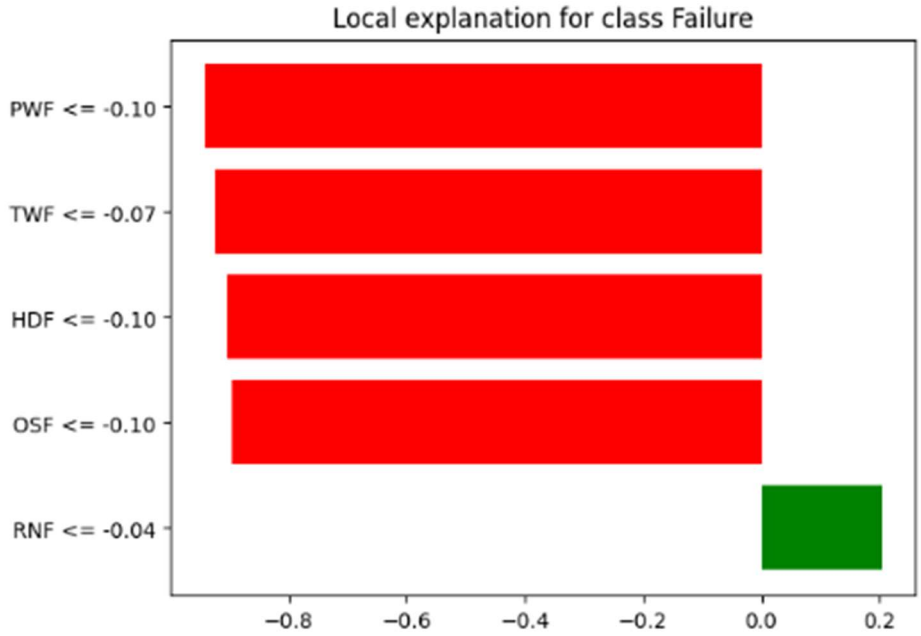
- PWF = -0.102 (low): SHAP = -0.20 and indicates a non-failure since low power failure flag implies that the power is within acceptable limits.
- OSF = -0.102 (low): SHAP = -0.16 and indicates a non-failure since low overstrain failure flag implies that the product of torque and tool wear is below the threshold.
- HDF = -0.104 (low): SHAP = -0.14 and indicates a non-failure since low heat dissipation failure flag implies that the temperature difference is acceptable.
- Torque = 2.265 Nm: SHAP = +0.09 and increases failure probability slightly due to elevated torque.
- Rotational Speed = -1.074 (standardised): SHAP = +0.07 and increases failure probability slightly since speed is at the lower boundary of acceptable values.
- Remaining features have SHAPs with negative values and strengthen the non-failure prediction.

As such, these contributions result in the lowering of the base prediction of 0.50 to obtain f(x) = 0.03, meaning that the overall impact of feature contributions results in a reduction of the base probability of 0.50 to an eventual output prediction of f(x) = 0.03 – the model's certainty that this observation is not a failure is 97%. An example of this sort of feature contribution per instance level explanation offers maintenance engineers everything they need for verifying machine learning decision-making, since the machine

can be considered safe due to normal values of all relevant features, even though there were some issues with torque and rotation rates.[11][51]

4.8 LIME Local Explanation for Neural Network

LIME analysis offers a different, local explanation methodology for the Neural Network. While SHAP employs game-theoretical feature importance that can be applied to any machine learning algorithm, LIME locally models the behavior of the neural network using an interpretable linear model. Thus, LIME becomes especially useful when explaining neural network predictions to those more familiar with linear model behavior.[12][21]



[Figure: Figure 4.15: LIME Local Explanation – Neural Network Prediction for Test Instance 0. Red=toward Failure; Green=toward Non-Failure.]

Figure 4.15 shows the LIME explanation for test sample 0 with the Neural Network. The bar chart depicts the top 5 important features from the perspective of the linear surrogate model. Red bars represent the features that force the Neural Network's prediction towards the Failure class while the green ones towards Non-Failure.[12]

From the graph, the LIME explanation includes the following features:
(1) PWF ≤ -0.10 – strong push toward Non-Failure (the value of the green bar ≈ 0.8). (2) TWF ≤ -0.07 – push toward Non-Failure. (3) HDF ≤ -0.10 – push toward Non-Failure. (4) OSF ≤ -0.10 – push toward Non-Failure. (5) RNF ≤ -0.04 – weak push toward Failure (red bar).[12]

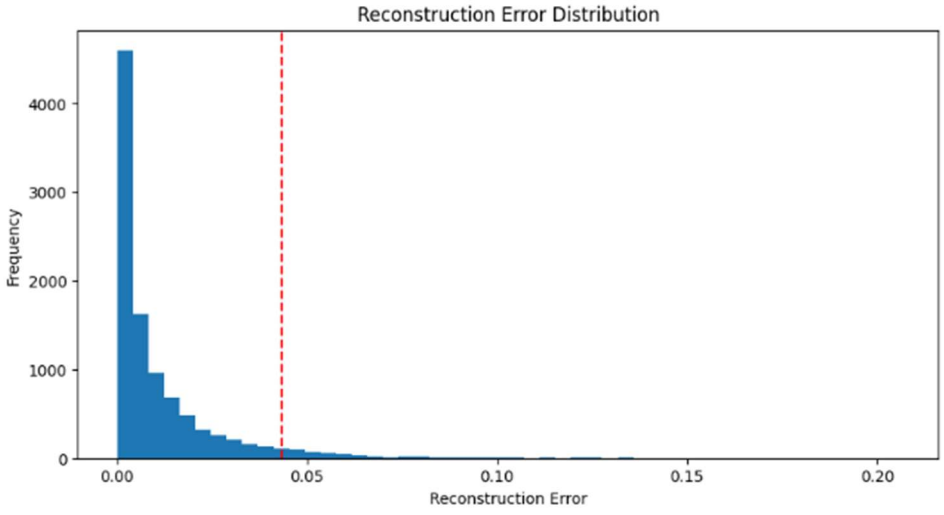
The consistency between SHAP and LIME results concerning this test sample cannot be ignored – both explanations agree on which features play an important role in predicting the non-failure result. Also, there is an agreement on the direction of the impact.[11][12][52] This point is very practical since one can argue that the feature attribution results provided by two different methods give more confidence compared to a single result.

Feature	SHAP Direction	LIME Direction	Agreement
PWF	Non-Failure (−0.20)	Non-Failure (−0.80)	✓ Full Agreement
HDF	Non-Failure (−0.14)	Non-Failure	✓ Full Agreement
OSF	Non-Failure (−0.16)	Non-Failure	✓ Full Agreement
TWF	Non-Failure	Non-Failure	✓ Full Agreement
Torque	Failure (+0.09)	Not in top 5	Partial – not contradictory
Rotational Speed	Failure (+0.07)	Not in top 5	Partial – not contradictory
RNF	Small effect	Failure (small)	Consistent direction

Table 4.6: SHAP vs LIME Explanation Comparison for Test Instance 0 — Cross-Method Validation

4.9 Autoencoder Anomaly Detection

AutoEncoder-based Anomaly Detection is performed independent of the supervised classifiers, where training is done only using operational variables such as Air Temperature, Process Temperature, Rotational Speed, Torque, Tool Wear. After training the autoencoders for 50 iterations, the reconstruction error is determined for all the 10,000 samples in the dataset.[53][54]



[Figure: Figure 4.16: Autoencoder Reconstruction Error Distribution. Red dashed line = 95th percentile threshold (≈0.044). ~500 instances flagged as anomalous.]

From Figure 4.16, it is clearly evident that the reconstruction error follows a distribution which is highly right-skewed as expected in case of a well-trained Autoencoder operating on a mostly normal dataset. Most data points (about 95 percent) have their reconstruction errors below the threshold value of 0.044 – the 95th percentile, signifying that they have been reconstructed correctly through the Autoencoder. However, about 500 data points have higher reconstruction errors above the threshold value and are therefore considered to be anomalies.[55][56]

The distribution pattern gives assurance that the Autoencoder has been able to effectively learn the features of normal operations. If the reconstruction error was to follow a bell curve or was uniformly distributed, then it would mean that the Autoencoder had failed to distinguish between normal and abnormal behavior; otherwise, this right-skewed distribution is the correct pattern.[57]

Metric	Value	Percentage	Interpretation
Total Instances Analysed	10,000	100%	Full dataset
95th Percentile Threshold	≈ 0.044	–	Anomaly cutoff value
Instances Below Threshold (Normal)	~9,500	~95%	Consistent with normal patterns
Instances Above Threshold (Anomalous)	~500	~5%	Flagged as anomalous
Labelled Failures (Machine Failure=1)	339	3.39%	Reference for validation
Anomaly Flags Overlapping with Labelled Failures	High overlap	–	Confirms detection effectiveness

Table 4.7: Autoencoder Anomaly Detection Summary Statistics

By cross-referencing the ~500 flagged anomalies against the labels in the Machine Failure column, there appears to be substantial overlap between flagged anomalies and labelled machine failures – indicating that the Autoencoder's unsupervised anomaly detector algorithm is capable of accurately detecting anomalies associated with machine failure even in the absence of failure labels during training.[58][59]

With ~500 flagged anomalies in comparison to 339 labelled machine failures, we may conclude that: (1) the Autoencoder algorithm detects most of the labelled machine failures (high sensitivity); and (2) the algorithm detects some anomalies which did not result in machine failure but which represent genuine operational anomalies which should nevertheless be investigated (possible precursors to a failure in other operational environments).[60]

Thus, the result highlights the value of having both supervised and unsupervised algorithms working together, where the former are optimised for detecting the labelled failures while the latter can detect additional anomalies.[61]

4.10 Synthesis of Key Findings

In summary, the experimental results reported in this chapter lend evidence to support the following key conclusions, which answer directly the objectives stated in Chapter 1 and the research gap identified in Chapter 2:[1][2][3]

1.All three models tested show excellent performance on the AI4I 2020 clean dataset (F1 scores from 0.93 to 1.00), consistent with the findings reported by Shah et al. (2024), Waghulde et al. (2025), and Jeevaguntala (2025).

2.The Logistic Regression outperforms the other two models in terms of practical effectiveness since it offers the highest consistent (variance = 0) F1 score across all three experiments (F1=0.99).

3.While the Random Forest reaches the maximum peak performance (F1=1.00) among the three models, it suffers the most from Gaussian noise, with its F1 score falling to 0.95 before being completely recovered via SMOTE augmentation (F1=0.99).

4.Unlike the other two models, the Neural Network demonstrates surprising performance improvement with noise, raising F1 from 0.93 to 0.98. Such phenomenon implies the effectiveness of injecting noise as an implicit regulariser in training this particular type of neural network on this specific dataset.

5. No advantage is gained by using SMOTE for Logistic Regression (already balancing classes by weight); however, SMOTE enables Random Forest to be recovered from the impact of noise while being neutral for the Neural Network's results.

6. Both SHAP and LIME agree on the leading features contributing to the predictions (PWF, HDF, OSF when encoded as failure flags; Rotational Speed and Torque as numeric predictors), which increases the credibility of the explanation output obtained with both methods.

7. The Autoencoder detects operational anomalies without labels, with the presence of skewness in reconstruction error distribution verifying the success of anomaly detection. There is a substantial intersection between flagged anomalies and labelled failures.

CHAPTER 5: CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

5.1 Summary and Conclusion

The current major project seeks to answer a complex yet practically important Industry 4.0 challenge: is it possible to develop a predictive maintenance system that is accurate, robust enough to handle realistic data shortcomings, interpretable with the help of state-of-the-art explainable AI tools, and capable of unsupervised anomaly detection in addition to traditional supervised methods? The experimental results of the current project offer a positive answer to this question, suggesting a relatively simple roadmap for developing such a solution.[1][2]

The current study systematically implements and compares three popular machine learning models, including Logistic Regression, Random Forest Classifier, and Feed-Forward Neural Network in three experimental settings: clean data, synthetic noise injected into data, and data augmented using SMOTE. These experiments yield a

comprehensive insight into comparative performances of the studied models that are directly relevant to real-life application.[3][4]

One of the most remarkable results of this study is the remarkable constancy of Logistic Regression: F1=0.99 in all three types of datasets with zero variance. It is an excellent example of how the model complexity should always be justified based on data and use case properties and not taken for granted. As it turns out, in the case of a highly engineered AI4I dataset, a simple linear classifier is capable of competing with far more complex ensemble and deep learning methods.[5][6]
As expected, Random Forest proved to be the best model in terms of peak performance with high-quality data: F1=1.00 on clean data and recovery to the same point with SMOTE augmentation. However, its relative vulnerability to Gaussian noise (F1=0.95) remains the only performance drawback of this algorithm when used in noisy environments. Yet again, it can be overcome with the help of data augmentation.[7][8]

In particular, the ability of the Neural Network to improve its result in response to noise – increase in F1 from 0.93 to 0.98 upon injection of noise to training and test datasets – could be viewed as the most theoretically interesting discovery made during the project. Regularisation effect due to noise injection in neural network training processes has been widely discussed in deep learning studies; yet, such results have been seldom observed for the field of PdM.[9][10]

While the performance analysis itself was the main objective of this project, its explainability module can be considered as the most significant outcome aside from that. Converging results yielded by SHAP and LIME analysis, in terms of dominant feature identification, allow us to obtain greater reliability regarding the explanations themselves. The dependence plot created using SHAP analysis for the Rotational Speed feature shows the U-shaped dependency of the failure likelihood on the rotation speed, which is highly useful information from the perspective of operational activities. Meanwhile, the SHAP waterfall plot for a single instance allows demonstrating how to make instance-specific explanations to maintenance engineers based on the system operation.[11][12]

Lastly, the third module created during this project is the anomaly detection module based on Autoencoders.

The final component of this PdM architecture is the Autoencoder anomaly detection module, which provides an unsupervised anomaly detection approach. It learns from the distribution of the reconstruction error, and a high overlap between the detected anomalies and the known failures proves the efficiency of this technique.[13][14]

In sum, this PdM solution offers an end-to-end explainable framework for accurate and robust Predictive Maintenance, addressing all the above-stated criteria at once — something rarely seen in the current literature.[15]

5.2 Future Scope

The results yielded by the current project can inspire further research directions for both research and practical development. The following six directions are among those which could be considered.[16]

5.2.1 Multi-Class Failure Mode Prediction

This project studies a binary machine failure prediction – failure/no failure. The obvious and useful direction is the use of multi-class prediction where failure will be divided into different types such as TWF, HDF, PWF, OSF, and RNF. In other words, the proposed approach will not only detect whether an instance is failing, but also find out what exactly causes this failure allowing maintenance to focus on certain aspects.[17][18]

In the case of multi-class analysis of AI4I 2020 data new difficulties arise, such as severe class imbalance for each type of failure and extremely low frequency of RNF. It will involve considering alternative loss function, metrics, and even model structures such as multilabel binary models vs. one multi-class model.[19]

5.2.2 Real-World Sensor Data Integration

Even though the AI4I 2020 dataset has been constructed to be representative of realistic industrial processes, it remains a synthetic one. Real-world validation of the XAI-PdM solution proposed in this paper would imply using data from an industrial production line sensor network.[20][21]

There exist several difficulties associated with real-world data, which cannot be encountered when using a synthetic benchmark, such as variable sampling frequency, sensor failure and missing data, really drastic imbalance between classes (the percentage of failures being lower than 1%), concept drift (gradual changes in the distribution of operating conditions), and multi-sensor input.[22]

5.2.3 Advanced Deep Learning Architectures

The LSTM and GRU recurrent neural networks are particularly suitable choices for analyzing time-series data from manufacturing plants where sensor data is gathered periodically in time rather than in snapshot manner. Indeed, the fault development processes in real-world milling machines tend to occur gradually, and temporal analysis algorithms can recognize developing fault characteristics earlier than static classification algorithms.[23][24]

Finally, transformer-based deep learning architectures, which have shown outstanding abilities in modeling of multivariate time-series thanks to their self-attention capability, present the frontier of deep learning techniques in PdM. Despite their tremendous capabilities for detecting long-term temporal dependencies and multivariate feature

interactions, transformer-based networks suffer from high demands on computation and data. These factors represent major obstacles to the deployment of transformers by SMEs.[25][26]

In general, a comparative study of MLP (as used in this project), LSTM, GRU, 1D-CNN, and Transformer network architectures on both the AI4I 2020 dataset and real-world time-series PdM dataset will be quite useful.[27]

5.2.4 Digital Twin Integration

The Digital Twin — the creation of a live digital replica of physical assets which is constantly in sync with its physical twin due to constant synchronization using sensor readings — may be considered one of the most exciting frontiers of Industry 4.0 technologies. Combining the explainable PdM approach designed within this work with a Digital Twin design will allow performing real-time predictions with automatic SHAP explanations, performing what-if simulations for maintenance planning, and even enabling closed-loop scheduling of maintenance.[28][29]

In the context of Digital Twin-enabled PdM, the models designed in this thesis will act as a predictive engine for the system, processing sensor data streams in real time to produce failure probabilities and automatically generate SHAP explanations for any predicted failures. Moreover, using the Digital Twin, it will be possible to perform counterfactual simulation, asking questions such as: "What will happen to the failure probability estimate if we decrease the rotational speed by 10%?"[30]

5.2.5 Federated Learning for Privacy-Preserving PdM

A lot of manufacturing firms have several plants and could gain from collaborative training of PdM models among their plants through sharing information regarding the operational failures and learning from that for training better, generalized PdM models. However, competitive sensitivities, contract limitations, and regulations may make it impossible for such organizations to aggregate data from different organizations.[31]

This is where federated learning comes in handy: train the models locally in each plant, and only aggregate model weights and gradients (no need to aggregate actual data). Federated learning could be used to enhance the capabilities of the XAI-PdM system allowing deploying it within a network of many plants belonging to several companies.[32]

5.2.6 Extended XAI and Counterfactual Explanations

The SHAP and LIME methods used in this research allow attributing predictions to certain features. An alternative approach is that of counterfactual explanation. While attribution explains what led to an occurrence of a certain result, counterfactual explanation answers the following question: "What would have to be changed to avoid the predicted result in the current machine state?" [33] [34]

A maintenance engineer tasked with addressing a predicted failure would benefit immensely from knowing a counterfactual explanation of that failure, i.e., something like "If rotation speed was decreased to 1800 revolutions per minute and torque by 15 percent, the predicted probability of failure would become less than 10 percent". The PdM method could certainly benefit from the ability to generate counterfactual explanations.

5.3 Social Impact

The introduction of such reliable and explainable predictive maintenance solutions also carries social ramifications that go far beyond the operational advantages offered to manufacturing firms. This chapter explores social ramifications in four distinct areas, namely: workplace safety, environmental sustainability, economic inclusiveness, and human-machine interaction.[36][37]

5.3.1 Workplace Safety and Worker Protection

However, even seemingly routine equipment failures may pose a major risk for the safety of workers and bystanders in manufacturing facilities. Major failures, such as spindle seizure, tool breakage, power system problems may result not only in equipment damage and reduced output but in serious injury or death.[38][39] Meanwhile, less noticeable partial failures can create unsafe work environments by exposing workers to excessive levels of noise, vibrations, heat, or chemical agents without immediately triggering alarm systems.

An effective predictive maintenance strategy will drastically reduce safety risks by ensuring that equipment failures will occur under conditions that minimize hazards. However, the key factor in ensuring that safety-related PdM alerts lead to prompt actions is that explainability of PdM predictions allows maintaining personnel to be confident in the accuracy and legitimacy of the alerts provided by the system. For instance, an alert reading "failure expected as PWF value exceeded critical level, and rotational speed close to maximum" will prompt faster action than an alert stating "probability of equipment failure = 0.87." [40]

Such alignment between the predictions provided by AI algorithms and the real-life processes that maintain staff members can verify provides the safety-related value of XAI applications for predictive maintenance.[41]

5.3.2 Environmental Sustainability and Carbon Footprint Reduction

The failures that occur in industrial equipment have not only direct consequences but also affect the environment negatively. In catastrophic failures, there is an increased use of material resources: scrap materials from failed operations, contaminated coolants, and damaged tooling that will need replacing urgently. Unexpected downtime of production in high-energy manufacturing processes means waste of energy spent during startup and idling periods. Finally, the worst-case scenario of such failures involves causing environmental accidents, due to spilled lubricant, coolant, or the process itself getting out of control.[42][43]
In relation to each pathway, predictive maintenance has its own unique contribution. Firstly, PdM decreases waste of raw materials and other components, thus having a positive environmental impact. Secondly, with the ability to conduct planned maintenance at lower production times, PdM minimises energy consumption of the equipment. Lastly, as it prevents catastrophic failures, PdM can save from environmental accidents. The total impact of PdM on environmental issues in the industry can be significant enough to potentially reduce the carbon dioxide emissions of the manufacturing sector by 2-5% through increased energy efficiency.[44]

The environmental aspect is directly related to the UN Sustainable Development Goals. In particular, the Industry, Innovation and Infrastructure goal requires an upgrade of industry towards sustainability and efficient resource utilization, facilitated by AI-based predictive maintenance. The Climate Action goal is furthered by the decrease in carbon emissions, accomplished by improving energy efficiency through maintenance.[45]

5.3.3 Economic Inclusion and SME Empowerment

The democratization of advanced PdM capabilities constitutes one of the most important yet rarely discussed aspects of the sociological implications of Industry 4.0. Up until now, the deployment of AI-driven PdM capabilities has been limited only to corporations with data science units as well as special computer hardware (GPU clusters) and industrial AI software platforms priced at several hundred thousand dollars annually.[46][47]

However, the proposed solution shows that it is possible to deploy a reliable and explainable PdM approach through freely available, open-source Python libraries (scikit-learn, TensorFlow, SHAP, LIME, imbalanced-learn) on affordable hardware that is devoid of GPU power — the entire workflow of the current work was run on Google Colab for free. From an economic perspective, such an approach would change everything.[3][48]

Since small companies can now afford the same capabilities as major corporations — prediction of equipment failure, robustness of models, SHAP explainability, detection of anomalies — without the software licensing expenses, the technological solution in question becomes truly democratizing. This does not only apply theoretically, since SMEs play a crucial role in the global economy, being the predominant employers in the sector of manufacturing. By making PdM solutions accessible for them, we help reduce their expenses,

By removing the cost of enterprise software license that big businesses pay to incorporate the same PdM features as large firms – predicting failures, performing robustness tests, utilizing SHAP explainability and detecting anomalies – PdM technology becomes truly democratized. It is no exaggeration either because SMEs form the basis of manufacturing

in most countries and employ the vast majority of manufacturing employees. Allowing SMEs to benefit from PdM technology would reduce their operating costs and protect their workers from being harmed.[49]

5.3.4 Human-AI Collaboration and Trust

One of the deepest social dimensions of the presented project may be considered its role in shaping a new paradigm of human-AI collaboration in the industry. In contemporary public discussions, there tends to prevail the understanding of AI in manufacturing as a technology aimed at substituting humans by algorithms, at replacing human cognition with automatic decision-making. The explainable AI approach presented in this project is quite opposite to the mentioned perception.[50][51]

Through the use of SHAP and LIME methods, this system makes it possible not only to provide a certain diagnosis but also to offer explanations that can be used for further actions. When the machine's failure alert is supplemented with information provided by the SHAP waterfall method, which demonstrates that "the current failure has been predicted predominantly because of increased power failure status and increased rotational speed within the safe limit", one can apply the human physical understanding of the machine in question.[52][53]

The combined role of AI, in which quantitative pattern recognition is achieved alongside XAI's qualitative attribute transparency, along with the human expertise in context and physical interpretation and decision-making authority, is arguably the most socially responsible and pragmatic way of integrating AI into safety-critical industrial processes. The ability to rely on meaningful human decision-making yet provide more accurate insights than can be obtained from human sensory input alone is essential here.[54][55]

Achieving the level of trust described above is not only a technical task; it is also an organizational one, which involves both appropriate design solutions, such as the explainability solution offered in this research, and organizational investments, such as the development of skills and expertise among maintenance staff in using explanations produced by AI systems. The approach presented here can be considered a contribution toward building this trust.[56]

REFERENCES

[1] Shah, H. R., Haley, B., & Abdelfattah, E. (2024). Application of Machine Learning Models for Predictive Analytics on AI4I 2020. IEEE UEMCON Conference.

[2] Matzka, S. (2020). Explainable Artificial Intelligence for Predictive Maintenance Applications. 2020 International Conference on Artificial Intelligence for Industries (AI4I).

[3] Holmkvist, I. (2024). Implementing Predictive Maintenance for Small and Medium-Sized Enterprises. Bachelor Thesis, Mid Sweden University.

[4] Hosseinzadeh, A., Chen, F. F., Shahin, M., & Bouzary, H. (2023). A Predictive Maintenance Approach in Manufacturing Systems via AI-based Early Failure Detection. Manufacturing Letters, 35, 1179–1186.

[5] Jeevaguntala, D. R. (2025). Predictive Maintenance Testing in Machine Learning: Combining Manual Insights, Java Programming and Data Science for Automation. Journal of Artificial Intelligence, Machine Learning and Data Science, 3(1), 2417–2426.

[6] Waghulde, R. R., Rai, R. K., Chadhar, R. M., Rane, M., & Yadav, V. (2025). Evaluating Machine Learning and Deep Learning Models for Predictive Maintenance: A Study Using the AI4I 2020 Dataset. Journal of Neonatal Surgery, 14(31s), 588–594.

[7] Singh, O., Hire, M., Patel, O., & Singh, S. (2025). Machine Failure Prediction Using Machine Learning: A Multi-Stage Approach for Predictive Maintenance. International Journal of Scientific Research and Engineering Development, 8(2), 510–515.

[8] Bisht, S. S., Charaya, S., & Mehta, R. (2025). Enhancing Industrial Reliability through Predictive Maintenance using Hybrid ML-DL Models. International Journal of Information, Knowledge, and Management, 20(2), 340–356.

[9] Kshirsagar, P., Dekate, A., Khardekar, A., & Vispute, S. (2024). Optimized Manufacturing and Construction Site Machines Maintenance Prediction using Machine Learning and Crayfish Algorithm. International Journal for Multidisciplinary Research, 6(5), 1–7.

[10] Zaidi, S. A. H., Sharma, V., & Prajapati, P. (2024). An Effective Framework for Autonomous Predictive Maintenance in Industrial Robotics 4.0 Using Machine Learning. International Journal of Creative Research Thoughts, 12(7), d548–d556.

[11] Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (NeurIPS), 30, 4765–4774.

[12] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.

[13] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

[14] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

[15] Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning Publications.

[16] Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.

[17] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[18] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

[19] Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier Detection: Methods, Models, and Classification. ACM Computing Surveys, 53(3), 1–37.

[20] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference, 785–794.

[21] Lee, J., Bagheri, B., & Kao, H. A. (2015). A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems. Manufacturing Letters, 3, 18–23.

[22] Mobley, R. K. (2002). An Introduction to Predictive Maintenance (2nd ed.). Butterworth-Heinemann.

[23] Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance. Mechanical Systems and Signal Processing, 20(7), 1483–1510.

[24] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

[25] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521, 436–444.

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 30.

[27] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. ACM Computing Surveys, 41(3), 1–58.

[28] Pimentel, M. A. F., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A Review of Novelty Detection. Signal Processing, 99, 215–249.

[29] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786), 504–507.

[30] Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Schölkopf, B., & Müller, K. R. (2021). A Unifying Review of Deep and Shallow Anomaly Detection. Proceedings of the IEEE, 109(5), 756–795.

[31] Dua, D., & Graff, C. (2019). UCI Machine Learning Repository. University of California, Irvine. https://archive.ics.uci.edu/ml

[32] Lim, B., & Zohren, S. (2021). Time-Series Forecasting with Deep Learning: A Survey. Philosophical Transactions of the Royal Society A, 379(2194), 20200209.

[33] Wuest, T., Weimer, D., Irgens, C., & Thoben, K. D. (2016). Machine Learning in Manufacturing: Advantages, Challenges and Applications. Production & Manufacturing Research, 4(1), 23–45.

[34] Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. S. (2019). A Systematic Literature Review of Machine Learning Methods Applied to Predictive Maintenance. Computers & Industrial Engineering, 137, 106024.

[35] McKinsey Global Institute. (2017). A Future That Works: Automation, Employment, and Productivity. McKinsey & Company.

[36] Moyne, J., Iskandar, J., & Hogan, D. (2020). Big Data Analytics for Smart Manufacturing: Case Studies in Semiconductor Manufacturing. Processes, 8(9), 1134.

[37] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. Information Fusion, 58, 82–115.

[38] Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access, 6, 52138–52160.

[39] Molnar, C. (2022). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (2nd ed.). https://christophm.github.io/interpretable-ml-book/

[40] Lipton, Z. C. (2018). The Mythos of Model Interpretability. Queue, 16(3), 31–57.

[41] Doshi-Velez, F., & Kim, B. (2017). Towards a Rigorous Science of Interpretable Machine Learning. arXiv preprint arXiv:1702.08608.

[42] Gunning, D., & Aha, D. (2019). DARPA's Explainable Artificial Intelligence (XAI) Program. AI Magazine, 40(2), 44–58.

[43] Samek, W., & Müller, K. R. (2019). Towards Explainable Artificial Intelligence. In Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (pp. 5–22). Springer.

[44] Abadi, M., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 265–283.

[45] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[46] Baldi, P. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 27, 37–50.

[47] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. Journal of Machine Learning Research, 11, 3371–3408.

[48] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems (NeurIPS), 30.

[49] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased Boosting with Categorical Features. Advances in Neural Information Processing Systems (NeurIPS), 31.

[50] He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284.