Predictive Maintenance of Electric Motor Using ML-Based Algorithms for EV Application

By

Tayyaba Aasar
193-FET/ MSEE/F23

Supervisor: Dr. Adnan Omer
Co-Supervisor: Dr. Ayesha Abbasi

Department of Electrical & Computer Engineering
Faculty of Engineering and Technology
INTERNATIONAL ISLAMIC UNIVERSITY
ISLAMABAD

2025

DEDICATED TO

To my lifelong teacher, my father Engr. Aasar Ahmad.

To my supervisor, Dr. Adnan Umar.

To my co-supervisor, Dr. Ayesha Abbasi.

CERTIFICATE OF APPROVAL

Title of Thesis: Predictive Maintenance of Electric Motor Using ML-Based Algorithms for EV Application

Name of Student: Tayyaba Aasar

Registration No: 193-FET/MSEE/F23

Accepted by the Department of Electrical and Computer Engineering, Faculty of Engineering and Technology, International Islamic University, Islamabad, in partial fulfillment of the requirements for the Master of Science degree in Electrical Engineering.

Prof. Dr. Syed Badshah _____

Dean FET IIUI

Dr. Ihsan ul Haq _____

Associate Professor, Chairman

_____ _____

(Internal Examiner)

Prof. _____ _____

(External Examiner - I)

Dr. _____ _____

(External Examiner - II)

Dr. Adnan Omer (Supervisor) _____

Dr. Ayesha Abbasi (Co-Supervisor) _____

25^{th} March, 2025

Abstract

Electric vehicles (EVs) play a crucial role in reducing carbon emissions, with induction motors serving as their core component due to their durability, efficiency, and simplicity. However, induction motors are prone to faults such as stator winding short circuits and bearing failures, which can lead to unexpected breakdowns, increased maintenance costs, and reduced reliability. Traditional maintenance strategies, such as reactive and preventive maintenance, are insufficient for addressing these hidden faults, necessitating the adoption of predictive maintenance (PdM) frameworks. This research focuses on developing a robust PdM framework for induction motors in EVs using machine learning (ML) techniques to enhance fault detection accuracy, reduce downtime, and optimize maintenance schedules.

The study evaluated various ML models, including Artificial Neural Networks (ANNs), Long Short-Term Memory (LSTM), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Trees (DT), Random Forest (RF), Naive Bayes (NB), and Physics-Informed Neural Networks (PINNs), on stator winding and Case Western Reserve University (CWRU) bearing fault datasets. Deep learning models, particularly LSTM and ensemble methods like the voting classifier, outperformed traditional ML models, with LSTM achieving 99.87% accuracy for binary classification and the voting classifier reaching 100% accuracy under certain conditions. PINNs, which incorporate motor dynamics, also performed well, achieving 99.32% accuracy for binary classification. Traditional models like Naive Bayes and KNN struggled with complex fault patterns, emphasizing the need for advanced ML techniques.

This research contributes to the field by integrating machine learning and physics-based modeling to enhance the reliability and efficiency of induction motors in EVs. The findings suggest that AI-driven predictive maintenance frameworks can significantly improve motor health monitoring, reduce maintenance costs, and extend the operational lifespan of EV induction motors. Future work should focus on real-time monitoring, expanding fault coverage, enhancing datasets, and leveraging digital twin technology to further refine and validate the proposed framework.

Keywords: EVs, Induction Motors, Predictive Maintenance, Machine Learning, Fault Detection, LSTM, PINNs, Stator Winding Faults, Bearing Faults.

Acknowledgements

I would like to express my heartfelt gratitude to Allah Almighty for granting me the strength, health, and perseverance to complete this research successfully.

I am deeply indebted to my supervisor, **Dr. Adnan Umar**, whose unwavering guidance, support, and expertise have been instrumental throughout this journey. His constant encouragement and insightful advice have helped shape this work.

I am also grateful to my co-supervisor, **Dr. Ayesha Abbasi**, for her valuable mentorship and constant support. Her expertise and feedback have been crucial in refining my research.

A special and profound thanks to my father, **Engr. Aasar Ahmad**, whose love, sacrifices, and unwavering support have been the driving force behind my success. His encouragement, patience, and belief in my potential have played a significant role in helping me reach this milestone. I owe much of my perseverance and determination to him.

Lastly, I would like to extend my sincere thanks to everyone who has contributed to this research and supported me in various ways.

Tayyaba Aasar

List of Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

AE Autoencoder

ARIMA AutoRegressive Integrated Moving Average

ANOVA Analysis of Variance

BPNN Backpropagation Neural Network

BMS Battery Management System

BDA Big Data Analytics

CNN Convolutional Neural Network

CBM Condition-Based Maintenance

CM Condition Monitoring

CSV Comma-Separated Values

CWRU Case Western Reserve University

DT Decision Tree

DNN Deep Neural Network

DWT Discrete Wavelet Transform

DL Deep Learning

DAQ Data Acquisition

EV Electric Vehicle

EOL End of Life

FFT Fast Fourier Transform

FNN Feedforward Neural Network

FEM Finite Element Method

FL Fuzzy Logic

GAN Generative Adversarial Network

GUI Graphical User Interface

HMM Hidden Markov Model

HMI Human-Machine Interface

IoT Internet of Things

IM Induction Motor

KNN K-Nearest Neighbors

KF Kalman Filter

LSTM Long Short-Term Memory

LR Logistic Regression

ML Machine Learning

MSE Mean Squared Error

MCSA Motor Current Signature Analysis

NB Naive Bayes

NN Neural Network

PdM Predictive Maintenance

PINNs Physics-Informed Neural Networks

PCA Principal Component Analysis

RF Random Forest

RNN Recurrent Neural Network

RUL Remaining Useful Life

SVM Support Vector Machine

SCADA Supervisory Control and Data Acquisition

STFT Short-Time Fourier Transform

TSC Time Series Classification

VFD Variable Frequency Drive

VC Voting Classifier

Contents

List of Tables

Chapter 1

Introduction

1.1 Background

Electric vehicles (EVs) are becoming increasingly important as they help reduce carbon emissions and combat climate change. Unlike traditional vehicles powered by internal combustion engines, electric vehicles are powered by electric motors, which provide numerous advantages such as lower greenhouse gas emissions, quieter operation, and reduced fuel consumption. With growing awareness of environmental issues and the push for cleaner, more energy-efficient transportation, the demand for EVs is rapidly rising. This trend is supported by advancements in technology and improvements in charging infrastructure, as illustrated by the rapid growth in the number of EVs shown in Figure 1.1.

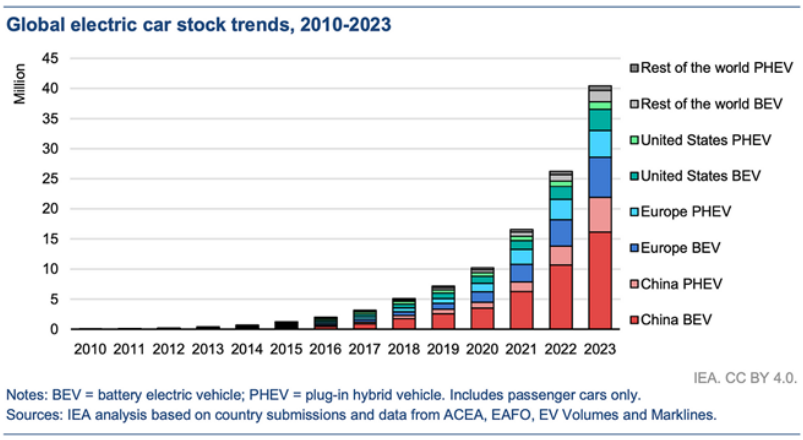


Figure 1.1: Global electric car stock trend 2010-2023 [1].

As the core component of most electric vehicles is the electric motor, which converts electrical energy into mechanical power to move the vehicle. The induction motor is the most

used motor in electric vehicles because it is durable, efficient, and simple in design. Unlike other motors, it does not require brushes or commutators, which makes it more reliable and easier to maintain [2]. However, induction motors can experience various types of faults that impact their performance, including stator winding failures, rotor faults, and bearing issues. These faults can result in poor motor performance, system inefficiency, or even complete motor failure if not detected earlier.

One of the most common issues in induction motors is related to the stator windings, which can experience problems such as insulation breakdown, overheating, or winding short circuits. These faults can significantly reduce the motor's efficiency, leading to power loss, overheating, or even complete failure if not addressed in time. Unlike external components, stator winding faults are often hidden and may not present noticeable symptoms until substantial damage has occurred, making early detection crucial [3].Alongside stator faults, bearing issues are another major concern in induction motors. Bearings support the rotor's rotation, and their failure can result in excessive vibration, noise, and a decline in the motor's performance. Common bearing faults include wear, lubrication failure, and misalignment, all of which can cause severe motor damage if not identified promptly. Advanced monitoring and diagnostic techniques are therefore essential to detect both stator and bearing faults early, ensuring that the motor operates smoothly, efficiently, and reliably over time.

Traditional methods of maintaining motors, such as reactive maintenance, are not ideal for handling these types of faults. Reactive maintenance means that repairs are only made when something breaks down. This approach can be expensive due to unplanned downtime and emergency repairs. It also risks additional damage to other motor components, leading to even higher costs. Another common approach is preventive maintenance, where maintenance is performed regularly based on a set schedule. While this can reduce some issues, it does not address problems that develop between scheduled checks, and it can also lead to unnecessary repairs. Thus, both reactive and preventive maintenance have their limitations, and a better solution is needed to reduce costs and prevent unexpected failures.

One such solution is predictive maintenance (PdM), a strategy that uses data to predict when a motor is likely to fail. Unlike preventive maintenance, which follows a fixed schedule, predictive maintenance focuses on real-time data to identify early signs of wear or damage. Using sensors to monitor things like vibration, and current, predictive maintenance systems can assess the motor's health at any moment. This helps to detect problems before they turn into

major faults, allowing repairs to be scheduled before a breakdown occurs. This early detection can save time and money by preventing unexpected repairs and unplanned downtime [4] with predictive maintenance, the motor's lifespan is extended, and its performance is optimized.

For predictive maintenance to work effectively, machine learning plays a key role. ML algorithms can analyze large amounts of data collected by sensors on the motor, looking for patterns that indicate a potential failure. Algorithms like ANNs, LSTM networks, Random Forest, SVMs, and KNN detect issues such as stator winding faults or rotor failures using historical data. Additionally, PINNs incorporate physical laws and motor dynamics into the learning process, enhancing fault detection accuracy by leveraging domain knowledge. Ensemble methods, such as voting classifiers, further enhance fault detection by combining predictions from multiple models to improve accuracy and reliability. This approach reduces errors, ensures precise monitoring of motor health, and minimizes unexpected failures [5].

The application of machine learning-based predictive maintenance systems offers significant advantages for electric vehicles. These systems enhance fault detection accuracy, enabling early identification and resolution of potential issues. By addressing faults promptly, they help avoid costly repairs, extend the motor's operational lifespan, and minimize unexpected downtime. Unlike preventive maintenance, which involves routine checks and often unnecessary repairs, predictive maintenance relies on real-time data to trigger repairs only when needed. This approach not only optimizes maintenance schedules but also ensures the motor operates efficiently and reliably, aligning with the evolving needs of EV technology.

1.2 Research Problem

EVs depend on induction motors for efficient operation, but these motors can experience faults such as stator winding issues, bearing failures, and rotor problems. Such faults can lead to unexpected breakdowns, increased maintenance costs, and reduced motor reliability. Early and accurate fault detection is crucial for preventing operational disruptions and minimizing maintenance expenses. However, existing research typically addresses specific fault types or motor models and lacks a comprehensive approach to handle the diverse operational conditions in real-world EV systems.

This research aims to develop a robust predictive maintenance framework for induction motors in EVs, leveraging machine learning algorithms to enhance fault detection accuracy

and computational efficiency. By using a comprehensive benchmark dataset and analyzing various operational scenarios, this study will provide a more effective and efficient solution for maintaining induction motors in industrial and real-world EV applications. The goal is to improve motor reliability, reduce downtime, and minimize the total cost of ownership for EVs.

1.3 Research Objectives

General Objective

- To develop a predictive maintenance framework for induction motors in EVs to enhance fault detection, minimize downtime, and improve reliability.

Specific Objectives

- To design a predictive maintenance framework using machine learning for induction motors in EVs.
- To utilize unexplored ML algorithms for accurate and precise fault predictions.
- To propose computationally efficient customized deep architectures with efficient unexploited optimizers.
- To verify the scalability of the proposed method through benchmark datasets.

1.4 Significance of Research

The proposed research is significant as it aims to advance predictive maintenance and fault detection for induction motors in EVs using machine learning. While existing studies have contributed valuable insights into fault detection, they often focus on specific fault types or motor models without providing a comprehensive approach.

This research will address these gaps by developing a robust predictive maintenance framework that integrates various machine learning algorithms, including ANN, LSTM, PINNs, SVM, KNN, Decision Trees, Random Forest, and Naive Bayes, along with ensemble techniques like the voting classifier. By using a comprehensive benchmark dataset and detailed feature engineering, the research aims to enhance fault detection accuracy and reliability, providing practical solutions that will improve the safety, performance, and maintenance efficiency of electric vehicles.

1.5 Thesis Outline

The thesis is organized as follows:

- **Chapter 2** presents the literature review of the proposed study. It highlights the research gaps in the existing literature, citing multiple studies, and includes a table summarizing the limitations and objectives of prior models and techniques used in fault detection and predictive maintenance.
- **Chapter 3** provides the theoretical background of the study. It discusses the fundamentals of 3-phase induction motors, stator winding fault mechanisms, and various techniques, including PINNs and traditional machine learning models, supported by the underlying mathematical and physical concepts.
- **Chapter 4** describes the methodology of the study. It outlines the dataset preprocessing steps, feature engineering techniques, and the implementation of various machine learning models. It also explains the evaluation metrics and experimental setup used to assess the performance of the proposed approach.
- **Chapter 5** presents the results and discussion of the study. It evaluates the performance of all models, analyzes fault classification results, and discusses the significance of the findings for predictive maintenance, focusing on enhancing the reliability and efficiency of 3-phase induction motors.
- **Chapter 6** summarizes the study's key findings, draws conclusions on the effectiveness of the proposed approach, and suggests potential directions for future research in predictive maintenance.

Chapter 2

Literature Review

Ensuring the reliability and safety of EVs is critical, particularly for components like motors, inverters, and battery packs. This literature review synthesizes recent advancements in fault detection and predictive maintenance for these components, highlighting various methodologies and their contributions to enhancing system performance and reliability.

2.1 Introduction to Predictive Maintenance in EVs

Predictive maintenance is becoming increasingly essential in the management of EVs, ensuring the longevity and reliability of their critical components such as batteries, electric motors, and power electronics. As EV adoption continues to grow, there is a pressing need to develop robust maintenance systems to reduce downtime, increase operational efficiency, and enhance vehicle performance. By utilizing data analytics, machine learning, and advanced sensor technologies, predictive maintenance can identify early signs of component degradation, allowing for proactive interventions.

The study by [6] explores the application of machine learning, specifically SVM, in combination with Discrete Wavelet Transform (DWT) for fault diagnosis in Brushless DC (BLDC) motors. This hybrid approach achieved a fault diagnosis accuracy of 98.67%, outperforming other traditional methods such as KNN with DWT (97.49%), showcasing its potential for EV motor fault detection.

Additionally, the research presented by [7] discusses a hybrid fault diagnosis approach using SVM and Naive Bayes classifiers with optimization techniques to detect faults in BLDC motor drives. This method achieved an impressive accuracy of 98.8%, significantly improving

the ability to identify faults such as open and short circuits in EV motors.

As EV technology evolves, monitoring and maintaining battery health has also become a priority. Murgai et al. [8] address the issue of battery degradation using a Scientific Machine Learning (SciML) approach, integrating domain knowledge with neural networks to enhance predictive accuracy. Their model reduces the data requirements for accurate predictions and forecasts of battery degradation, ultimately extending battery life and optimizing long-term energy management.

Other techniques like LSTM networks have also been applied to PdM. Zhang et al. [9] used LSTM networks with vibration monitoring to detect stator faults in BLDC motors, achieving a high accuracy of 97.10%.

In addition to motor and battery issues, temperature prediction for Permanent Magnet Synchronous Motors (PMSMs) has also been studied. The study in Energy Conversion and Management [10] used machine learning methods like random forest and boosting algorithms to predict motor temperature with minimal sensors, offering a cost-effective solution to improve system performance.

Machine learning has also been used for broader condition monitoring of electrical machinery. The research presented by [11] discussed the use of time-series analysis and Principal Component Analysis (PCA) for feature extraction, leading to improved reliability and cost savings in machinery maintenance.

For DC motors, [12] combined machine learning with real-time sensor data to enhance fault detection accuracy and operational efficiency. Similarly, [13] presented a system that combines machine learning, IoT, and predictive modeling to monitor motor parameters and predict failure times, offering better maintenance strategies.

Deep learning approaches have also shown promise. The research in Neural Computing and Applications [14] focused on estimating the Remaining Useful Life (RUL) of BLDC motors using a recurrent neural network (RNN) with attention mechanisms, providing accurate predictions for real-time monitoring.

In the public transport sector, [15] applied machine learning and fuzzy logic to real-time IoT data for predictive maintenance. This approach effectively detected faults, improving safety and reliability in public vehicles.

Industry 4.0 has further advanced predictive maintenance techniques. Author in [16] introduced optimization methods like the Jaya algorithm and Sea Lion Optimization to forecast

maintenance needs, while [17] combined machine learning and data analytics for real-time monitoring and early failure detection.

Finally, [18] explored using ANNs, cloud technology, and IoT platforms for fault detection in BLDC motors. This study improved energy management and early fault detection, enhancing motor performance.

In conclusion, predictive maintenance is revolutionizing the EV industry by ensuring the reliability and performance of critical components. With tools like machine learning, advanced sensors, and innovative diagnostic techniques, PdM is helping EVs become more efficient, reliable, and cost-effective. By moving from reactive to proactive maintenance, EV operators can reduce costs, improve safety, and support a future where electric mobility is more dependable and sustainable.

2.2 Predictive Maintenance for Induction Motors in EV Applications

The application of predictive maintenance in induction motors for EV systems is becoming increasingly important due to the need for high reliability, safety, and cost-effective maintenance. Several studies have focused on using advanced ML techniques to improve the fault detection and diagnosis process, ultimately enhancing the performance of EVs. These studies highlight various methods, including feature extraction, fault classification, and the integration of real-time data, to optimize motor health and minimize downtime in EV applications.

In study [19], Gundewar, Kane, and Andhare developed a novel method to diagnose broken rotor bar (BRB) faults in IMs using time-domain grayscale current signal imaging (TDGCI) combined with a convolutional neural network (CNN). This method eliminates the need for manual feature extraction by automatically extracting features from 2D grayscale images generated from 1D current signals. The approach achieved an impressive classification accuracy of 99.58%, demonstrating its potential for accurate fault detection in EV motor systems.

The work in [20] focuses on designing a three-phase IM for EV applications and employs various ML algorithms to diagnose faults under different load conditions. The study considers fault types like short circuits, high-resistance connections, and open-phase circuits. Algorithms

such as Support Vector Machine (SVM), K-Nearest Neighbors (k-NN), Random Forest (RF), and Deep Learning (DL) were tested, achieving fault detection accuracy levels ranging from 98% to 100%. This work illustrates how ML can be used to enhance the reliability and fault tolerance of motors in EV applications, ultimately contributing to reduced maintenance costs and better system performance.

In [21], researchers explored the application of multiple ML models to diagnose rotor and bearing faults in IMs. The study used vibration data to test various models such as SVM, Multilayer Neural Networks (MNN), Convolutional Neural Networks (CNN), Gradient Boosting Machine (GBM), and XGBoost. The results indicated that SVM and CNN achieved the highest diagnostic accuracy, while XGBoost was the fastest in terms of computation. These findings show how machine learning can facilitate real-time fault diagnosis in EV motor systems, allowing for timely interventions and enhanced system reliability.

In their research [22], Turza et al. tested ML models to detect single-phase faults in IMs under different operational conditions. Their study achieved a high accuracy of 99.9% using the Random Forest algorithm, demonstrating its robustness in fault detection. This highlights the capability of ML techniques to detect faults with high precision, which is crucial for maintaining the operational efficiency of IMs in EVs.

Amit Rai et al. [23] utilized ANN for fault prediction in IMs by analyzing vibration and current signals under varying rotational speeds. The study found that features such as standard deviation played a key role in improving prediction accuracy. This approach shows how ANN can be applied to predict faults and improve the reliability of IMs in EV applications.

In [24], the authors discuss the use of machine learning techniques for PdM in EV systems. By applying supervised, semi-supervised, and reinforcement learning methods, they were able to classify various faults, thus enhancing the overall system reliability and minimizing the likelihood of breakdowns. This approach further supports the case for using predictive maintenance to extend the lifespan of EV motors and reduce maintenance costs.

Karolina Kudelina et al. [25] presented a comparative analysis of machine learning models for diagnosing broken rotor bars in IMs. Their study, set within the framework of Industry 4.0, integrated IoT with physical systems to improve predictive maintenance strategies. Real-world data from induction machines were used to compare model performance, offering valuable insights into the development of more effective PdM algorithms.

The study in [26] provides an extensive review of fault detection and diagnosis (FDD)

methods in EVs, covering both traditional and emerging data-driven techniques. This review emphasizes the importance of using machine learning to enhance the safety and reliability of EV systems, demonstrating how data-driven PdM approaches can significantly improve fault detection capabilities.

Finally, Mohamed et al. [27] proposed a hybrid ML model for diagnosing faults in IMs through thermal image analysis. Their study used infrared imaging combined with advanced feature selection techniques to identify mechanical faults. The model demonstrated high classification accuracy and sensitivity, showing its potential for broader applications in PdM and fault detection in EV motors.

These studies collectively highlight the growing role of predictive maintenance in induction motors for electric vehicles. The integration of machine learning algorithms, real-time data monitoring, and advanced fault detection methods are proving essential in ensuring motor reliability, reducing maintenance costs, and optimizing the performance of electric vehicles.

2.3 Recent Advances in Bearing Fault Diagnosis Using the Case Western Reserve University Dataset

The diagnosis of bearing faults in rotating machinery has been extensively studied using the Case Western Reserve University (CWRU) dataset, which serves as a benchmark for validating various approaches. Researchers have explored diverse methods to improve the accuracy, efficiency, and robustness of fault detection. Yoo et al. [28] introduced a lightweight CNN model that focuses on dimensionality reduction and low computational costs. By downsampling vibration signals and converting them into spectrograms, their model demonstrated high classification accuracy while maintaining efficiency, setting a strong foundation for further exploration in this domain.

Building upon this foundation, Saghi et al. [29] addressed the limitations of single-scale CNNs by employing a multi-scale CNN architecture combined with a bidirectional gated recurrent unit (GRU). This hybrid approach captured both local and global features of vibration signals, making the model resilient even in noisy conditions.

Further advancing the field, Huang et al. [30] introduced a Wide Deep Convolutional Neural Network (WDCNN) with Squeeze-and-Excitation (SE) mechanisms, which significantly improved feature learning and diagnostic precision, achieving an accuracy of 99% on the

CWRU dataset. Complementary approaches by Bórnea et al. [31] utilized the Hilbert-Huang Transform for feature extraction and applied machine learning algorithms like Random Forest and KNN for classification. Their study emphasized the importance of balanced datasets and effective feature selection for accurate fault detection. Finally, Sawai et al. [32] demonstrated the potential of ensemble learning, combining RF, SVM, and ANN as base models with a gradient boosting classifier. This method achieved a commendable accuracy of 97.7%, showcasing the advantages of leveraging multiple classifiers to enhance diagnostic reliability.

These studies collectively highlight the evolution of fault diagnosis techniques, from lightweight CNN models to advanced hybrid and ensemble approaches. The integration of traditional signal processing methods with deep learning has proven particularly effective, making these techniques highly applicable in industrial scenarios where accurate and robust fault detection is critical.

2.4 Research Based on Inter-Turn Short-Circuit Fault Dataset for Induction Motors

Afriyie [33] explored predictive maintenance for three-phase induction motors, focusing on inter-turn short circuit fault detection and prediction. The study utilized stator current data and MATLAB's predictive maintenance toolbox to develop classification models. SVM and KNN algorithms were compared under varying load conditions (no-load, half-load, and full-load). SVM was identified as more effective in consistently detecting and predicting faults across different load scenarios.

The literature review is summarized in the following table.

Table 2.1: Literature review summary with research gaps

Year	**ML Models Used**	**Dataset**	**Description**	**Research Gap**
2020	LSTM	Vibration monitoring data and fault conditions for BLDC motors	Applies advanced vibration monitoring systems and LSTM networks for stator fault classification in BLDC motors.	Limited studies on integrating LSTM with real-time monitoring systems in industrial applications.
2020	Attention-based Neural Network	Voltage degradation data and fault conditions for electric motors	Utilizes an attention-based neural network to estimate the remaining useful life of electric motors.	Further exploration needed on generalization across different motor types.
2022	KNN, Random Forest	Data for public transportation vehicles	Explores predictive maintenance based on ML for public transportation vehicles.	Lacks application in electric vehicle components specifically.
2022	Hybrid ML Models	Fault diagnosis data for BLDC drives	Investigates hybrid machine learning models for fault diagnosis in BLDC drives.	Few studies have assessed hybrid models' performance in real-time applications.
2023	SVM, k-NN, MLP, RF, DT, GB, XGBoost, DL	Simulation data for healthy and faulty conditions	Focuses on fault diagnosis for a 5HP induction motor in EVs under variable loads, achieving 98-100% accuracy.	Need for comprehensive testing in real-world operational conditions.

Year	ML Models Used	Dataset	Description	Research Gap
2023	KNN, Random Forest	Data for DC motor fault detection	Develops a predictive maintenance algorithm for fault detection and classification in DC motors.	More research is needed on feature selection techniques to improve model accuracy.
2023	Random Forest	Healthy and faulty motor data using d-axis and q-axis conversions	Achieved a 99.9% fault diagnosis accuracy by focusing on operational modes of IMs and statistical features like mean and standard deviation.	Application of these techniques to other types of motors remains underexplored.
2023	Various classifiers	Thermal image data for motor faults	Proposes a machine learning model for diagnosing induction motor defects through thermal image analysis, classifying multiple fault types.	Exploration of multi-sensor data integration for enhanced fault diagnosis is limited.
2023	Back Propagation Neural Network	Magnetic leakage flux data for fault conditions	Analyzes inter-turn short-circuit faults in electric motors, achieving 88.1% accuracy with BPNN.	More robust models are needed for better fault prediction under varying load conditions.

Year	ML Models Used	Dataset	Description	Research Gap
2023	Comparative ML Models	Real data from induction machines	Presents a signal spectrum-based approach for predictive maintenance in induction machines, with model validation on real data.	Need for comparative studies on the efficacy of different models in industrial settings.
2023	CNN	Grayscale current signal images of rotor conditions	Achieves 99.58% accuracy in fault classification of IM using CNN on 2D grayscale images from TDGCI, surpassing traditional methods.	Limited exploration of CNN in diverse operational environments of electric motors.
2023	LSTM	IoT-integrated industrial machine data	Focuses on predictive maintenance of industrial machines using ML and IoT data.	More studies are needed on the effectiveness of LSTM in real-world conditions.
2023	Decision tree, RF, KNN	EV battery maintenance data	Explores ML approaches for battery maintenance prediction in EVs.	Lack of comprehensive studies on integrating battery health with overall vehicle diagnostics.
2024	SVM, XGBoost, Linear & Polynomial Regression	Temperature and motor specifications for EVs	Presents ML models for temperature prediction in EVs, using features like ambient and coolant temperatures.	Limited research on multi-parameter predictive models for EVs under varying conditions.

Year	ML Models Used	Dataset	Description	Research Gap
2024	ANN, SVM, DT, RF	Motor drive and battery system data for EVs	Discusses fault detection and diagnosis in EV motor drives and battery systems using various ML algorithms.	Exploration of combined methodologies for enhanced accuracy in fault detection is required.
2024	RF, SVM, KNN, DT, Naive Bayes, Voting	EV data with multiple fault types and conditions	Focuses on fault detection and classification in EVs using a variety of ML models.	Future work needed on model interpretability and explainability for practitioners.
2024	ANN	Vibration and current signals at multiple rotational speeds	Uses ANN for fault prediction in IM, with high accuracy in using standard deviation as a key statistical feature.	Investigation into real-time implementation of ANN for predictive maintenance in industrial settings is lacking.

2.5 Summary of Identified Gaps and Research Contributions

Despite significant progress in the field of electric motor fault diagnosis, existing literature reveals several limitations. Many studies focus on diagnosing specific fault types, often neglecting a unified approach for comprehensive fault classification. Additionally, the use of advanced or hybrid deep learning models such as LSTM and Physics-Informed Neural Networks (PINNs) remains limited, especially for electromechanical fault prediction in induction motors. Furthermore, most prior works rely solely on either current or vibration signals, reducing diagnostic robustness. A notable gap is the absence of physics-based learning

frameworks that embed motor dynamics directly into model training. Also, many existing approaches are tailored to non-induction motors, making them less applicable to industrial three-phase induction motor systems.

To address these challenges, this study proposes an integrated fault diagnosis framework that combines both vibration and current signals. Covers two types of critical faults, bearing faults and short-circuit faults in the stator winding, under various load conditions. Multiple traditional machine learning models and deep learning models including LSTM and PINNs are employed and compared. Most importantly, the study introduces PINNs into the predictive maintenance domain, offering improved fault classification by incorporating motor physics into the learning process. This comprehensive approach enhances diagnostic accuracy, model robustness, and practical applicability for industrial use cases.

Chapter 3

Theoretical Background

3.1 Introduction

This chapter presents the theoretical foundation for understanding the various concepts central to the development of a predictive maintenance framework for induction motors in EVs. The chapter explores the working principle of induction motors, common fault types, and maintenance strategies. Additionally, it delves into the application of machine learning techniques for fault detection and predictive maintenance, with a focus on machine learning algorithms, feature engineering, and deep learning models [34]. Predictive maintenance, particularly in the context of induction motors used in EVs, has emerged as an effective strategy to optimize motor performance and reduce operational downtime [35].

3.2 Overview of Induction Motors

Induction motors are widely used in EVs due to their robustness, high efficiency, and simple construction. Unlike other types of electric motors, induction motors do not require brushes or commutators, which reduces mechanical complexity and enhances reliability. In an induction motor, alternating current (AC) supplied to the stator generates a rotating magnetic field that induces a current in the rotor. This interaction between the stator and rotor produces mechanical torque, which drives the vehicle's wheels [36].

The motor consists of several key components that work together to ensure its operation. The stator, which is the stationary part of the motor, is responsible for creating the magnetic field that drives the motor. The rotor, on the other hand, is the rotating component that interacts

with the magnetic field generated by the stator, producing the mechanical torque required for operation. Additionally, bearings play a crucial role by supporting the rotor and minimizing friction during its rotation, ensuring smooth and efficient functioning of the motor. The type of induction motor is derived from the type of rotor used. Hence, an induction motor can be either a squirrel-cage or wound type[37]. Figure 3.1 shows the schematic drawing of an induction motor, illustrating these components [38].

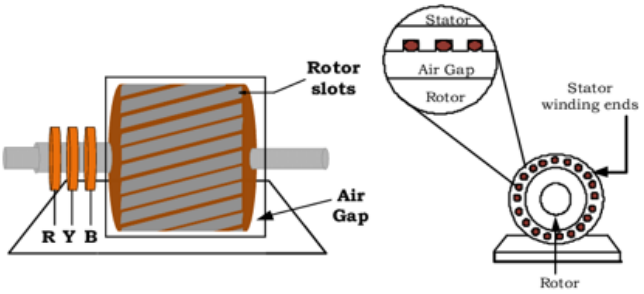


Figure 3.1: A schematic representation of an induction motor

This simple yet efficient design is why induction motors are commonly used in EVs for propulsion [39].

3.3 Faults in Induction Motors

Induction motors are susceptible to severe failures if faults are not identified at an early stage. These faults can be categorized as electrical, mechanical, or environmental and may occur both internally and externally. Some common faults in induction motors include inter-turn short circuits in stator windings, bearing defects, end ring failures, and broken rotor bars[38].

3.3.1 Inter-turn Short Circuit Fault

This fault occurs when the insulation between conductors at different potentials within the same slot is compromised, leading to unintended current flow [40].

3.3.2 Bearing Failures

Bearing failures are categorized into two types:

- **Single Point Fault:** Caused by motor overloading, which leads to a fatigue crack on the bearing surface.
- **Generalized Roughness Fault:** This fault arises due to the deformation of the bearing surface, often resulting from insufficient lubrication or misalignment [41].

3.3.3 Rotor Faults

Rotor faults encompass a range of issues, including:

- **Broken Rotor Bars:** This occurs due to thermal stress, fatigue, or manufacturing defects. Broken rotor bars can lead to reduced torque, excessive heating, and vibrations.
- **End Ring Faults:** These faults are associated with the end rings connecting the rotor bars, caused by thermal or mechanical stress, leading to reduced motor efficiency and abnormal vibrations [42].

Studies by the Electric Power Research Institute (EPRI) reveal that bearing failures are responsible for 42% of induction motor faults. Inter-turn short circuits of the stator windings account for 31%, while rotor-related faults, including end ring failures and broken rotor bars, comprise 9% of reported cases [43].

The classification of these faults is illustrated in Figure 3.2, which provides a visual representation of the various types of faults in an induction motor [38].

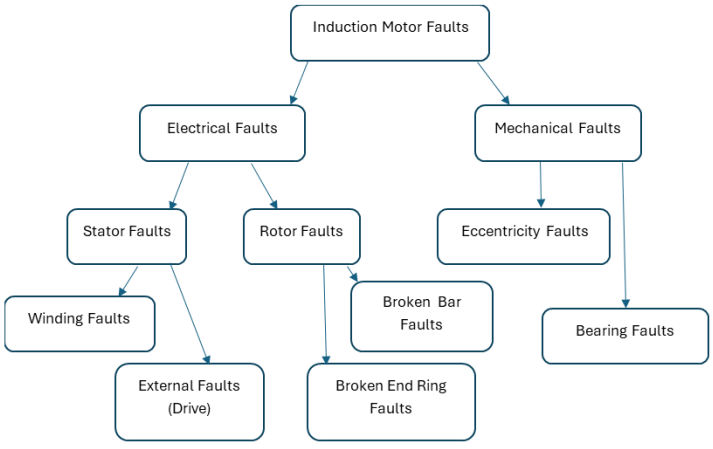


Figure 3.2: Classification of Faults in Induction Motors

Early detection of these faults is crucial for maintaining the motor's efficiency and reliability, as undetected faults can cause progressive damage leading to unplanned downtime

and expensive repairs [44].

3.4 Predictive Maintenance for Induction Motors

Predictive maintenance is a data-driven approach that aims to predict equipment failures before they occur, minimizing downtime and costs. For induction motors, which are vital in industrial operations, PdM uses sensors and analytics to monitor performance and detect early signs of wear or faults. This method bridges the gap between reactive and preventive maintenance, offering a more efficient and cost-effective solution. This section will explore the foundational concepts of maintenance methods and their evolution toward predictive strategies.

3.4.1 Introduction to Maintenance Methods

Maintenance is an essential aspect of ensuring the reliability, performance, and operational efficiency of induction motors. It can be broadly categorized into three methodologies: reactive maintenance, preventive maintenance, and predictive maintenance [45–47].

3.4.1.1 Reactive Maintenance (Run-to-Failure)

This approach involves repairing or replacing equipment after it has failed. Although simple, reactive maintenance is costly due to unplanned downtime and potential damage to surrounding systems. See Figure 3.3 for an illustration of this approach [48].

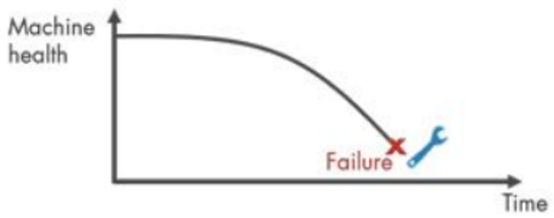


Figure 3.3: Reactive maintenance

3.4.1.2 Preventive Maintenance

This method involves routine inspections and maintenance at scheduled intervals to prevent failure. While preventive maintenance improves reliability compared to reactive methods, it

can result in unnecessary repairs and associated costs . An overview of this approach is shown in Figure 3.4 [48].

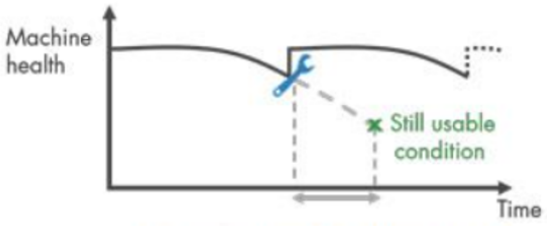


Figure 3.4: Preventive maintenance

3.4.1.3 Predictive Maintenance

Predictive maintenance leverages sensor data and advanced analytical methods to predict when equipment is likely to fail. By identifying potential issues early, PdM optimizes maintenance schedules, minimizes downtime, and reduces repair costs . This concept is illustrated in Figure 3.5 [48].

Figure 3.5: Predictive maintenance

Predictive maintenance, which is the focus of this study, uses a combination of sensor data and machine learning models to identify and classify faults before they escalate into significant failures. A comparison of different maintenance strategies is shown in Figure 3.6 [49].

3.4.2 Data-Driven Fault Diagnosis Methods

Modern predictive maintenance strategies often employ data-driven fault diagnosis methods to achieve high accuracy and reliability. These methods rely on sensor data which is analyzed using machine learning and deep learning models [50].

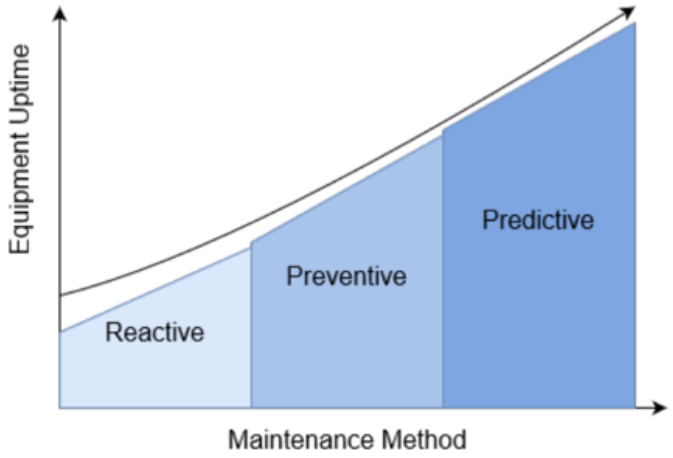


Figure 3.6: Maintenance Strategies

3.5 Machine Learning in Predictive Maintenance

Machine learning is a branch of artificial intelligence that focuses on algorithms capable of automatically identifying patterns in input data. These patterns enable the system to make informed predictions on future data [49, 51]. In contrast to traditional programming, where rules are explicitly defined by the developer, machine learning models learn to identify patterns and relationships from data, deriving rules based on past iterations and outcomes (see Figure 3.7 [49, 52]).

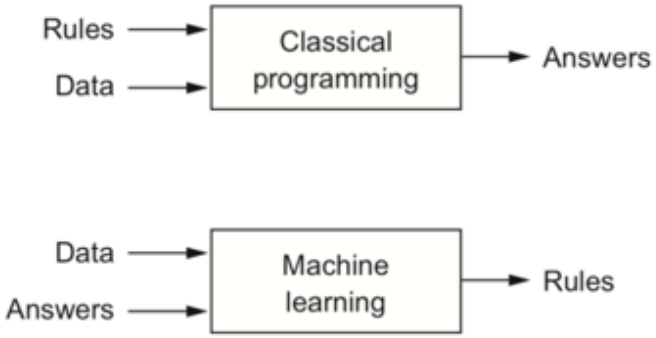


Figure 3.7: Machine Learning Paradigm

3.5.1 Applications of Machine Learning

Machine learning algorithms are especially useful for tasks that are challenging for conventional programming approaches, such as:

- Classification

- Regression
- Machine Translation
- Anomaly Detection

These algorithms are generally categorized into three major types: unsupervised learning, supervised learning, and semi-supervised learning, depending on the nature of the available data [49].

3.5.2 Types of Machine Learning

Machine learning can be broadly categorized into different types based on the nature of data and learning processes. Each type has unique characteristics and is suited for specific applications. The primary types include supervised learning, unsupervised learning, and semi-supervised learning [49].

3.5.2.1 Unsupervised Learning

Unsupervised learning deals with data that is unlabeled, meaning there is no explicit mapping from inputs x to outputs y, with the dataset being represented as $\{x_i\}_{i=1}^{N}$. The primary aim of unsupervised learning is to explore the data's underlying probability distribution and uncover patterns or structures within it [49]. This approach is most used for clustering tasks, where the goal is to group data based on shared characteristics and identify latent factors inherent in the dataset [53].

3.5.2.2 Supervised Learning

In contrast, supervised learning relies on labeled data, where each input x is associated with a corresponding output y (also referred to as the target), as represented by $\{(x_i, y_i)\}_{i=1}^{N}$. The goal here is for the model to learn how to predict the output y from unseen inputs x in the future [54]. Supervised learning problems are further divided into two categories:

- **Classification:** The model learns to map inputs x to discrete output classes $y \in \{1, 2, \ldots, C\}$ where C represents the number of classes. If there are only two possible classes, it's termed binary classification, while multi-class classification applies when there are more than two classes[49].

- **Regression:** The model maps inputs x to continuous, real-valued outputs $y \in \mathbb{R}$.

3.5.2.3 Semi-Supervised Learning

Semi-supervised learning strikes a balance between supervised and unsupervised learning by using a mix of labeled and unlabeled data. This method is particularly useful when only a small portion of the data is labeled, and the rest remains unlabeled. Semi-supervised models use both types of data to predict y from x, effectively combining the strengths of both unsupervised and supervised approaches [49, 54].

3.5.3 Model Generalization

The primary goal of machine learning models is to generalize effectively to new, unseen data, a capability referred to as the model's capacity. If a model performs poorly on both training and unseen data, it is considered to be underfitting. Conversely, if it excels on training data but struggles with new data, it is said to be overfitting. Achieving an optimal balance between underfitting and overfitting is crucial for ensuring both accurate predictions and strong generalization. This concept is visually summarized in Figure 3.8 [49], which illustrates the trade-off between underfitting and overfitting in model generalization.

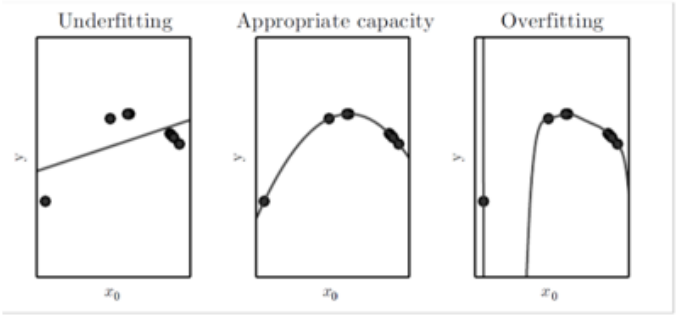


Figure 3.8: Model Generalization.

3.5.4 Training the Model

For a machine learning model to generalize well, it must undergo a training process. During training, the model's predictions $\hat{y}$ are compared with the actual values y, and the prediction error is calculated. The model then learns and improves by minimizing this error. The error is

typically quantified using a loss function, and various loss functions are available, each focusing on different aspects of model performance[49].

3.5.5 Optimization of the Model

The process of adjusting the model to minimize the prediction error is known as the optimization algorithm . The choice of loss function and optimization method, along with their respective internal parameters, directly influences the model's performance [49].

In predictive maintenance, machine learning algorithms are applied to analyze sensor data and detect faults early, preventing failures and reducing downtime. Various machine learning techniques have shown promise in the field of predictive maintenance. These techniques include ANNs, LSTM networks, KNN, and SVM. Below, we explore these techniques.

3.5.6 Artificial Neural Networks

ANNs are computational models inspired by the structure and function of the human brain. They consist of interconnected processing units, known as neurons, which apply nonlinear transformations (activation functions) to input data, referred to as features. Each feature is assigned a weighted value that reflects its significance in the learning process.

Deep Learning, a specialized branch of Machine Learning, extends ANNs by incorporating multiple hidden layers. This allows the model to recognize complex patterns and extract meaningful insights from data. It leverages hierarchical representation learning, where deeper layers progressively capture more abstract features, enhancing the model's ability to generalize and make accurate predictions [55].

3.5.6.1 Activation Functions

Activation functions are essential in determining how input data is transformed and the corresponding output is generated. Different types of activation functions are used based on the specific application [56]. While hidden layers in a neural network typically utilize a single activation function, the output layer often employs a distinct function suited to the given task or prediction [57].Hidden layers, positioned between the input and output layers, pass their processed outputs to subsequent layers [49]. While Artificial Neural Networks (ANNs) can have zero or more hidden layers, deep neural networks generally consist of at least three hidden

layers [58]. Below are some commonly used activation functions:

- **Linear Function:** The simplest form of activation function, scaling the input x by a constant c (Equation 3.1) [59].

$$f(x) = c \cdot x \tag{3.1}$$

- **ReLU (Rectified Linear Unit):** This function outputs 0 for negative values and retains positive values unchanged, without any upper bound equation 3.2. ReLU is widely used due to its efficiency across various applications [60].

$$f(x) = \max(0, x) \tag{3.2}$$

- **Softmax Function:** Produces probability values for target labels, ensuring that all probabilities sum to 1 equation 3.3 [61].

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{3.3}$$

- **Sigmoid Function:** A nonlinear function that maps inputs to a range between 0 and 1, making it effective for classification tasks equation 3.4 [62].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

- **Tanh Function:** Similar to the sigmoid function but maps inputs to a range between -1 and 1 equation 3.5 [63].

$$f(x) = \tanh(x) \tag{3.5}$$

3.5.6.2 Network Training

Deep learning models improve their predictions through a process known as gradient descent. This iterative optimization method adjusts model parameters to minimize a predefined loss function . The initial weights are typically set randomly, and gradient descent computes the derivative of the loss function with respect to the weights in each layer [64]. These weights are then updated iteratively to reduce the error, following the process of backpropagation Figure 3.9 [49].

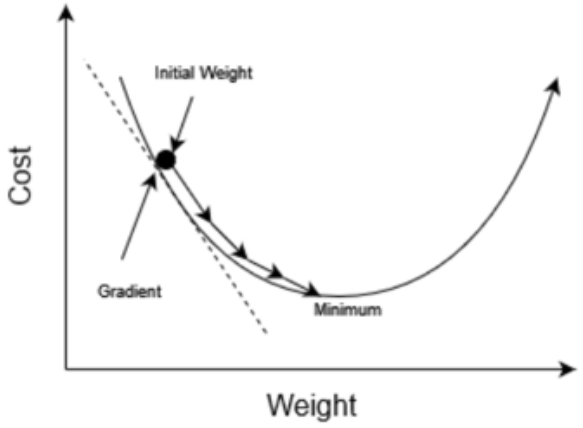


Figure 3.9: Gradient Descent

The success of the training process depends on several hyperparameters, such as the learning rate and batch size. The learning rate determines the step size for weight updates. A small learning rate slows convergence, while a large learning rate risks overshooting the minimum and failing to converge. The batch size refers to the number of training samples used for updating weights. Small batch sizes improve accuracy but increase computational cost, whereas large batches are less computationally intensive but may sacrifice accuracy [65].

Training and evaluating a model typically involves splitting the dataset into subsets for training, validation, and testing. The training set is used for learning, the validation set assists in hyperparameter tuning, and the test set evaluates final model performance. K-fold cross-validation is a popular method where the dataset is divided into k subsets, and the model is trained k times, each time using a different subset for validation [49, 66].

3.5.6.3 Loss Functions

Loss functions quantify how well a model performs by measuring the error between predicted and actual values. The goal of training is to minimize the loss function, as lower errors indicate better model performance [49, 67]. Common loss functions include:

- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted values $\hat{y}_i$ and actual values y_i, treating deviations equally (Equation 3.6). MAE is not sensitive to outliers [68].

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.6}$$

- **Mean Squared Error (MSE):** Calculates the mean of squared differences between predicted and actual values. It penalizes larger deviations more heavily due to squaring (Equation 3.7) [69].

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.7}$$

- **Categorical Cross-Entropy:** Categorical Cross-Entropy is a loss function commonly used for multi-class classification tasks, where the model assigns a single label from multiple possible classes. It measures the discrepancy between the predicted probability distribution and the actual class labels, guiding the model to improve its predictions (Equation 3.8) [70].

$$\text{CCE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.8}$$

- **Binary Cross-Entropy:** Applicable to binary or multi-label classification tasks, selecting one label out of two options for each attribute (Equation 3.9) [71].

$$\text{BCE} = -\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.9}$$

Selecting the appropriate activation function for the output layer is critical to ensure compatibility with the loss function. Table 3.1 outlines common combinations for different problem types [49].

Problem Type	Activation Function
Regression	Linear
Binary Classification	Sigmoid
Multi-Class Classification	Softmax

Table 3.1: Common Activation Functions for Different Problem Types

3.5.6.4 Optimization Methods

Optimizing model weights involves using algorithms that minimize the loss function. These algorithms vary in how they update weights, incorporating techniques like momentum (considering previous updates) and learning rate decay (reducing the learning rate over time). Common optimization methods include:

- **Stochastic Gradient Descent (SGD):** Updates model weights using either a single data sample per iteration (true SGD) or a small subset of samples (mini-batch SGD). True

SGD introduces more noise but can help escape local minima, while mini-batch SGD balances computational efficiency and convergence stability [72].

- **Adagrad:** Adapts the learning rate based on gradient information for each parameter, allowing better handling of sparse data [63].
- **RMSProp:** Maintains a moving average of squared gradients to normalize parameter updates, stabilizing training on noisy datasets [64].

These optimizers, along with appropriate hyperparameters, play a key role in effective model training[37].

3.6 Long Short-Term Memory

LSTM is a specialized type of Recurrent Neural Network (RNN) developed to overcome limitations of traditional RNNs, particularly the vanishing gradient problem. LSTMs are designed to retain long-term dependencies in sequential data, making them well-suited for applications such as time series analysis, speech recognition, and language modeling. Unlike feed-forward neural networks, LSTMs incorporate feedback loops, enabling them to process sequential information effectively. Their architecture includes memory cells and gating mechanisms that regulate information flow, determining what should be retained or discarded at each time step. This selective memory capability allows LSTMs to focus on relevant data while mitigating the challenges faced by standard RNNs.

LSTMs are extensively used in various domains, including time series forecasting, speech recognition, natural language processing, video analysis, and healthcare. In predictive maintenance, they analyze sensor data over time to detect anomalies and predict potential faults in industrial machinery, such as pumps, motors, and turbines. This early fault detection capability helps prevent failures, reduces maintenance costs, and minimizes downtime, ultimately improving operational efficiency [73].

3.6.1 Types of Gates in LSTM

An LSTM contains three primary gates:input gate, forget gate, and output gate that control the flow of information into and out of the memory cell.

The **input gate** regulates the incorporation of new information into the memory cell. By analyzing both the current input and the previous hidden state, it determines which data should be retained for future processing, ensuring that only relevant information is stored. The **forget gate** decides what information should be discarded, removing unnecessary data from the memory to maintain efficiency. The **output gate** manages the flow of information from the memory cell to the output of the LSTM, selecting which parts of the stored information should be passed on to the next layer or time step.

All three gates are activated using sigmoid functions, which output values between 0 and 1. These gates are trained through backpropagation, allowing the network to learn when to open or close them based on the input and the hidden state [73].

3.6.2 Structure of LSTM

An LSTM network is composed of multiple LSTM cells, each equipped with three key gates: the input gate, forget gate, and output gate. These gates regulate the flow of information, allowing the network to selectively retain or discard data at each time step, which helps in capturing long-term dependencies in sequential data.

At the core of each LSTM cell is a memory cell that stores information from previous time steps, influencing the current output. This memory mechanism enables LSTMs to effectively process sequential information over extended periods. The output from each LSTM cell is passed to the next, facilitating continuous learning and analysis across multiple time steps [73].

Due to their ability to remember and selectively forget information, LSTMs are widely used in tasks such as time series forecasting, language modeling, and sequential data prediction [73].

3.6.3 LSTM for Fault Detection in Induction Motors

3.6.3.1 Sequential Data Analysis

Motor data, such as vibration signals and current readings, is often collected over time. This makes the problem inherently sequential, where the state of the motor at a given time depends on its previous states. LSTM is specifically designed to capture long-term dependencies in sequential data, which makes it well-suited for time series data from motors.

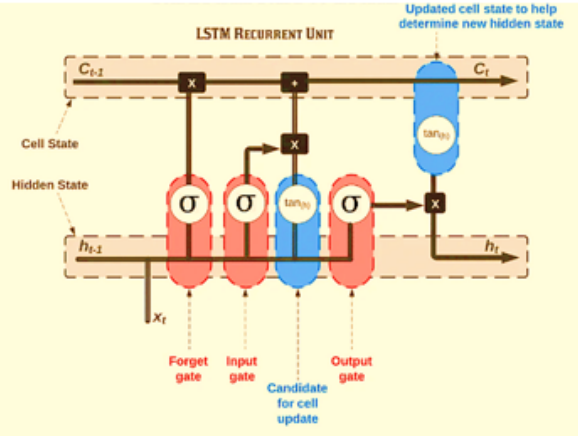


Figure 3.10: Structure of LSTM

3.6.3.2 Handling Temporal Dependencies

Faults in induction motors may not manifest immediately but develop over time. For example, a winding short circuit fault could cause gradual changes in the motor's current or vibration pattern. LSTM can learn these temporal dependencies and identify patterns that indicate the onset of a fault, even if the fault is subtle at first.

3.7 K-Nearest Neighbors

KNN is a simple, instance-based learning algorithm used for both classification and regression tasks. It operates on the principle of proximity or similarity in the feature space. The basic idea behind KNN is that a data point is classified based on the majority class of its neighbors in the feature space [74].

3.7.1 Algorithm

- **Choose the number of neighbors (K):** Select the number of nearest neighbors (K) you want to use for making the prediction.
- **Distance metric:** Calculate the distance between the input data point and the points in the training set. Common distance metrics are Euclidean, Manhattan, or Minkowski distance.
- **Vote:** For classification, the data point is assigned to the class that most of its K nearest

neighbors belong to. For regression, the average of the K nearest neighbors' values is taken.

3.7.2 Advantages

- Simple to implement.
- Non-parametric: does not make any assumptions about the underlying data distribution.
- Effective for smaller datasets with fewer dimensions.

3.7.3 Disadvantages

- Computationally expensive as the dataset grows.
- Sensitive to noisy data and irrelevant features.
- Poor performance on high-dimensional data (curse of dimensionality).

3.8 Support Vector Machine

SVM is a supervised machine learning model used primarily for classification tasks but can also be used for regression. SVM works by finding the hyperplane that best separates the classes in the feature space. The goal is to find a decision boundary that maximizes the margin between the two classes [75].

3.8.1 Algorithm

- **Linear vs. Non-Linear:** For datasets that are linearly separable, SVM identify the optimal hyperplane that maximizes the margin between different classes, ensuring better generalization. However, when dealing with non-linearly separable data, SVM applies kernel functions — such as the radial basis function (RBF) or polynomial kernel — to transform the data into a higher-dimensional space where a linear separation becomes possible. This technique enables SVM to effectively classify complex patterns that cannot be separated in the original input space.

- **Maximizing the Margin:** The margin is the distance between the closest points from each class (called support vectors) and the decision boundary.
- **Optimization:** The SVM model minimizes a cost function while ensuring that the margin between the classes is maximized.

3.8.2 Advantages

- Effective in high-dimensional spaces.
- Works well for both linearly and non-linearly separable data.
- Robust against overfitting in high-dimensional spaces.

3.8.3 Disadvantages

- Computationally expensive, especially with large datasets.
- Requires careful selection of the kernel function and regularization parameters.
- Sensitive to noise and outliers in the data.

3.9 Random Forest

Random Forest is an ensemble learning technique that builds multiple decision trees and combines their predictions. It is a bagging algorithm that reduces overfitting and increases the model's accuracy by averaging the predictions of several trees, each built using random subsets of the data [76].

3.9.1 Algorithm

- **Bootstrap Sampling:** Create multiple bootstrap samples (random subsets with replacement) from the training data.
- **Build Decision Trees:** Build a decision tree for each sample. When making a split in the tree, instead of considering all features, only a random subset of features is used.

- **Combine Predictions:** For classification tasks, the final prediction is based on the majority vote from all the trees. For regression tasks, the average of the predictions from all the trees is used.

3.9.2 Advantages

- High accuracy and robust to overfitting.
- Can handle both classification and regression tasks.
- Handles missing values and large datasets well.
- Easy to interpret with feature importance metrics.

3.9.3 Disadvantages

- Can be computationally expensive for large datasets.
- Models can become large and difficult to interpret.
- Less effective when the relationship between features is weak.

3.10 Decision Tree

A Decision Tree is a supervised learning algorithm used for classification and regression. It works by splitting the data into subsets based on the most significant feature at each node, creating a tree-like structure of decisions [77].

3.10.1 Algorithm

- **Splitting:** Start at the root node and recursively split the data based on the feature that provides the best split (using criteria like Gini impurity, entropy, or variance).
- **Stopping Criteria:** Stop when a stopping criterion is met (e.g., when all data points in a node belong to the same class or when the tree reaches a specified depth).
- **Prediction:** For classification, the majority class in a leaf node is used for prediction. For regression, the mean value of the target variable is used.

3.10.2 Advantages

- Simple to understand and interpret.
- Supports both numerical and categorical data.
- Does not require feature scaling.
- Effectively manages missing data.

3.10.3 Disadvantages

- Susceptible to overfitting, particularly with deep trees.
- Sensitive to noise and outliers in the dataset.
- Unstable: minor variations in the data can result in significant changes to the tree structure.

3.11 Naive Bayes

Naïve Bayes is a probabilistic classification algorithm based on Bayes' Theorem, operating under the assumption that features are conditionally independent given the class label. It is especially efficient for high-dimensional datasets and is commonly applied in text classification tasks [78].

3.11.1 Algorithm

- **Bayes' Theorem:** It determines the probability of each class based on the given input features and assigns the class with the highest likelihood.
- **Independence Assumption:** Assumes that each feature contributes independently to the class probability, which simplifies the computation.
- **Prediction:** For a given input, the class with the highest posterior probability is chosen as the predicted class.

3.11.2 Advantages

- Simple and fast, with a low computational cost.
- Works well with high-dimensional datasets (e.g., text classification).
- Requires fewer training data and is less prone to overfitting.

3.11.3 Disadvantages

- The independence assumption is often unrealistic, leading to poor performance if features are correlated.
- Not suitable for regression tasks or continuous target variables.

3.12 Conclusion

In this section, we have reviewed a variety of machine learning classifiers that are commonly applied to predictive maintenance tasks. Each classifier has its strengths and weaknesses, making them suitable for different types of data and problem settings. KNN is simple and effective for small datasets, while SVM work well in high-dimensional spaces. Random Forest and Decision Trees provide interpretability and robustness, and Naive Bayes excels with high-dimensional data. Artificial ANNs and LSTM networks are especially powerful for learning complex patterns, with LSTMs being particularly suitable for time-series data. Understanding these classifiers' principles will provide a foundation for their application in predictive maintenance in later chapters.

3.13 Physics-Informed Neural Networks

PINNs are a type of deep learning model that embed physical laws into the training process. Unlike conventional neural networks that rely solely on data to identify patterns, PINNs incorporate the governing equations of a system into their learning framework. This approach ensures adherence to established physical principles, enhances generalization, and reduces the dependence on large labeled datasets. PINNs are especially beneficial in scenarios where data is

scarce but the governing physics is well-defined, making them valuable in scientific computing, engineering, and other fields involving physical systems [79, 80].

3.13.1 Concept of PINNs

The key benefit of PINNs is their ability to incorporate physical laws directly into the loss function during training. Unlike traditional neural networks that focus solely on minimizing the error between predictions and actual data, PINNs introduce an additional loss term that penalizes deviations from governing physical equations. This approach ensures that the model's predictions not only align with the data but also comply with established physical constraints [81].

A standard PINN setup combines two loss terms:

- **Data Loss (L_data):** This term ensures the neural network predictions match the available labeled data.
- **Physics Loss (L_physics):** This term penalizes deviations from the governing physical equations, which could be partial differential equations (PDEs), ordinary differential equations (ODEs), or algebraic constraints.

The total loss function is the sum of the data loss and the physics loss:

$$L_{\text{total}} = L_{\text{data}} + \lambda L_{\text{physics}} \tag{3.10}$$

where λ is a hyperparameter that controls the balance between fitting the data and satisfying the physics [82].

3.13.2 Mathematical Foundation of PINNs

The mathematical foundation of PINNs is based on the idea of integrating the system's governing equations into the loss function. These equations are often expressed as differential equations. For example, in fluid dynamics, the system might be governed by partial differential equations (PDEs) like the Navier-Stokes equations. For a general PDE, the governing equation can be expressed as:

$$N(u(x,t)) = f(x,t) \tag{3.11}$$

where:

- N is a differential operator (such as a gradient or divergence),
- $u(x,t)$ is the solution of the equation (e.g., temperature, velocity),
- $f(x,t)$ is the forcing term (external inputs or sources).

In the case of PINNs, the neural network $u(x,t)$ approximates the solution of the equation, and the physics loss is the residual of the equation:

$$L_{\text{physics}} = \|N(u(x,t)) - f(x,t)\| \tag{3.12}$$

The network is trained to minimize this residual, which ensures that the predictions of the model satisfy the physical equations [83, 84].

3.13.3 Relevance of PINNs to Fault Classification in Induction Motors

In the context of fault classification task for a 3-phase induction motor, PINNs are leveraged to incorporate motor physics using algebraic equations rather than complex differential equations. These algebraic equations represent steady-state conditions that govern the behavior of the motor. Below is the mathematical framework that we use for the model:

3.13.3.1 Kirchhoff's Current Law

Kirchhoff's Current Law states that the sum of currents entering a junction equals the sum of currents leaving the junction. For a 3-phase motor, this can be expressed as:

$$I_1 + I_2 + I_3 = 0 \tag{3.13}$$

where I_1, I_2, and I_3 are the currents in the three-phase windings of the induction motor. This equation ensures that the current balance is maintained in the motor, which is an important physical constraint when classifying faults.

3.13.3.2 Flux Consistency

The flux in an induction motor is related to the currents in the windings. The flux ϕ can be considered a function of the currents in the three windings:

$$\phi = f(I_1, I_2, I_3) \tag{3.14}$$

This equation ensures that the total flux generated in the motor is consistent with the currents in the windings.

3.13.4 Physics-Informed Neural Network Setup for Motor Fault Classification

In our case, the fault classification of the induction motor involves using measured features like current and flux. The governing equations for motor physics, including KCL and flux consistency, are used to regularize the model's learning process. The model's output is the fault classification, which is constrained by these physical laws. Hence, the total loss function in the PINN is:

$$L_{\text{total}} = L_{\text{classification}} + \lambda_1 L_{\text{imbalance}} + \lambda_2 L_{\text{flux}} \tag{3.15}$$

where:

- $L_{\text{classification}}$ is the traditional classification loss term (e.g., cross-entropy loss) that minimizes the difference between predicted and actual fault classes.
- $L_{\text{imbalance}}$ is the term that penalizes violations of Kirchhoff's Current Law:

$$L_{\text{imbalance}} = \|I_1 + I_2 + I_3\|^2 \tag{3.16}$$

- L_{flux} is the term that penalizes inconsistencies in the flux equation:

$$L_{\text{flux}} = \|\phi - f(I_1, I_2, I_3)\|^2 \tag{3.17}$$

The terms $L_{\text{imbalance}}$ and L_{flux} ensure that the predicted currents and flux are consistent with the motor's physical behavior, improving the model's robustness [85, 86].

3.13.5 Why ODEs Were Not Used

In many traditional applications of PINNs, Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs) are used to model dynamic systems that change over time or have spatial dependencies [87, 88]. However, in our case, the fault classification problem focuses on steady-state measurements rather than dynamic behavior. The features used for classification, such as current and flux, are directly measured, which means they do not need to be derived through ODEs [89].

Furthermore, the physical relationships governing the motor can be effectively captured using algebraic equations like Kirchhoff's Current Law (KCL) and flux consistency, which are simpler and computationally more efficient than ODEs. For example, KCL ensures the sum of currents in the three-phase windings equals zero, while flux consistency maintains the balance between currents and generated flux [90]. These constraints are sufficient to regularize the model and ensure that the predictions align with the physics of the motor. Thus, using ODEs is unnecessary and would add unnecessary complexity to the model [91].

3.13.6 Conclusion

PINNs provide an effective way to incorporate known physical laws into machine learning models. In the case of fault classification for induction motors, PINNs can integrate the algebraic constraints governing the motor's behavior—such as Kirchhoff's Current Law and flux consistency—into the loss function. This helps the model learn more robustly, ensuring that the predictions respect motor physics, even with a limited amount of labeled data. The use of ODEs or PDEs is unnecessary in this static fault classification task, as simpler algebraic constraints effectively capture the essential motor behavior [92].

3.14 Ensemble Methods for Enhanced Fault Detection

Ensemble learning techniques, like the voting classifier, enhance accuracy and reliability by merging predictions from multiple machine learning models. By combining outputs from different models, these methods help minimize errors and improve the overall effectiveness of a predictive maintenance system [93].

3.15 Evaluation Metrics

Once the model is trained, different evaluation metrics are used to assess its performance. These metrics help evaluate the model's ability to generalize to new data [94]. In classification tasks, the confusion matrix serves as a crucial tool, offering a detailed overview of the model's predictive accuracy .

The confusion matrix is a tabular representation that highlights true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). Table 3.2 [49] shows an example of a confusion matrix:

Predicted Values	**Positive**	**Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 3.2: Sample Confusion Matrix

- **True Positives (TP)**: Correctly predicted positive samples.
- **True Negatives (TN)**: Correctly predicted negative samples.
- **False Positives (FP)**: Incorrectly predicted positive samples.
- **False Negatives (FN)**: Incorrectly predicted negative samples.

In an ideal model, all predictions fall along the diagonal of the confusion matrix, with no off-diagonal elements. This indicates the model has perfectly classified all data samples [49].

3.15.1 Key Evaluation Metrics

1. **Accuracy**

   Accuracy measures the proportion of correct predictions out of the total number of samples, with a range between 0 and 1.

2. **Precision**

   Precision quantifies the fraction of true positives among all predicted positive samples, with a range between 0 and 1.

3. **Recall (Sensitivity)**

   Recall evaluates how well the model identifies actual positive samples, with a range between 0 and 1.

4. **F1-Score**

   The F1-Score is the harmonic mean of precision and recall, providing a balance between the two. It ranges between 0 and 1.

These metrics collectively offer a comprehensive understanding of the model's performance, highlighting its ability to handle both positive and negative predictions effectively[49].

3.16 Conclusion

This chapter provided a comprehensive overview of the theoretical concepts underpinning this study. It discussed the working principle of induction motors, the common faults they experience, and the importance of early fault detection. The application of machine learning for predictive maintenance was explored, highlighting its potential to enhance reliability and reduce downtime. Various machine learning classifiers, including ensemble methods, were discussed for their suitability in fault classification tasks. Finally, evaluation metrics such as accuracy, precision, recall, and F1-score were introduced to quantify model performance. This foundational knowledge sets the stage for the implementation detailed in the subsequent chapters.

Chapter 4

Methodology

4.1 Introduction

The methodology chapter outlines the systematic approach adopted to conduct this research, detailing the procedures, tools, and techniques employed to address the research questions. It provides a clear framework for data collection, analysis, and interpretation, ensuring the study's reliability and validity. This chapter serves as a roadmap, guiding the reader through the steps taken to achieve the research objectives.

4.2 Proposed Methodology Overview

This research proposes a comprehensive methodology to develop an effective predictive maintenance framework for induction motors in EVs. The approach integrates traditional machine learning algorithms, PINNs, and LSTM networks to achieve accurate and reliable fault classification. The dataset includes sensor readings, such as current and vibration signals, collected under various operational and faulty conditions. Data preprocessing techniques, including normalization, handling of missing values, and feature engineering, are applied to ensure data quality and relevance.

The methodology employs classical machine learning models, including ANN, SVM, KNN, Decision Trees, and Random Forest, to classify fault types, alongside ensemble techniques for enhanced accuracy. PINNs are incorporated to leverage the underlying physical laws of motor dynamics, embedding domain knowledge into the learning process and improving predictions. LSTM networks are utilized to capture temporal patterns in the sensor

data, essential for detecting faults with time-dependent characteristics. The performance of all models will be compared based on performance parameters such as accuracy, precision, recall, and F1-score, ensuring a thorough evaluation of their effectiveness. This integrated approach is designed to enhance the reliability and efficiency of motor maintenance systems in EVs. The methodology is illustrated in Figure 4.1, providing a step-by-step flowchart of the proposed predictive maintenance framework.

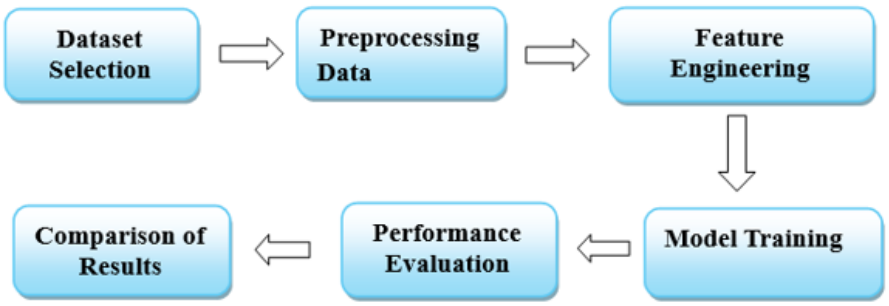


Figure 4.1: Flowchart of the proposed predictive maintenance methodology

4.3 Datasets

In this section, we describe the benchmark datasets used for training and evaluating the predictive maintenance models for induction motors in EVs. These datasets serve as a foundation for fault detection and classification tasks and are specifically selected to provide comprehensive, real-world data on motor performance under various conditions. Each dataset contains different operational scenarios, including normal motor conditions and various fault types such as stator winding issues and bearing failures. The use of these datasets enables a robust evaluation of machine learning algorithms and provides a basis for comparison across different models.

- **CWRU Bearing Dataset:** This dataset provides vibration data under various fault conditions, including different load and rotational speed settings for bearing fault diagnosis [95].
- **Inter-turn Short-Circuit in Induction Motor Dataset:** It includes data collected from induction motors experiencing inter-turn short circuits, along with various operational settings such as load and driving frequencies [96].

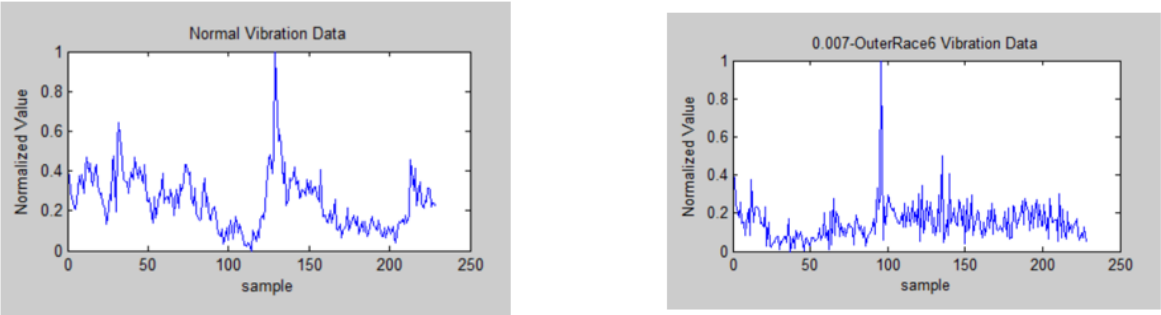
4.3.1 CWRU Dataset

The Case Western Reserve University (CWRU) Bearing Dataset [97] is widely recognized for bearing fault detection and classification. Initially developed at Rockwell to assess motor bearing conditions, it has since become a benchmark for evaluating signal-processing techniques, feature engineering, and data-driven models in numerous research studies [98–104].

Data collection was conducted using a test bench comprising a 2-horsepower (HP) Reliance Electric motor, a dynamometer, and a torque transducer. This setup monitors motor performance and torque under different load conditions. Vibration signals were captured using three accelerometers positioned at the Drive End (DE), Fan End (FE), and Base (BA), with sampling rates of 12 kHz and 48 kHz, ensuring high-resolution measurements. The dataset includes time-series vibration data recorded under various fault conditions, which were artificially induced in the bearings using electro-discharge machining (EDM). The faults were introduced at distinct locations within the bearing, specifically in the ball, inner race, and outer race, with defect sizes ranging from 0.007 inches (0.18 mm) to 0.021 inches (0.53 mm) [49].

The dataset encompasses different motor operating conditions, such as a 1 HP load, a shaft rotation speed of 1772 RPM, and a 48 kHz sampling rate. Each fault type is characterized by multiple statistical features, including maximum, minimum, mean, standard deviation, root mean square (RMS), skewness, kurtosis, crest factor, and form factor. These features are computed over time segments consisting of 2048 data points (0.04 seconds) at a 48 kHz sampling rate [49].

Due to its comprehensive fault scenarios and operational conditions, this dataset is a valuable resource for researchers investigating fault detection and classification in industrial machinery. It is widely applied in predictive maintenance and the development of fault detection algorithms. Researchers utilize it to test and validate various machine learning techniques, ranging from conventional signal processing methods to advanced artificial intelligence models [49]. Figures 4.2a and 4.2b illustrate the form factor of the vibration signal computed over different samples in the dataset, where the x-axis represents the sample index, and the y-axis denotes the form factor of the normalized vibration signal.



(a) (b)

Figure 4.2: Vibration data: (a) Normal (b) Anomalous

4.3.2 Inter-turn Short-Circuit in Induction Motor Dataset

This dataset [3] focuses on inter-turn short-circuit (ITSC) faults in induction motors, providing both leakage flux and current signals. The test bench utilized for data acquisition comprises two identical three-phase squirrel cage induction machines, each rated at 1 HP with a delta configuration, 220V supply voltage, and a rated current of 3A. One machine operates as a motor, while the other emulates the mechanical load. The motor's stator was specially rewound to facilitate the emulation of ITSC faults, allowing access to winding branches to introduce controlled short circuits. This setup enables the simulation of various fault severities, ranging from minor insulation degradation to severe short circuits.

Two fault types were simulated:

1. **High Impedance (HI):** Represents the initial stage of the fault, where the electrical insulator begins to degrade, creating a parallel current path.

2. **Low Impedance (LI):** Represents a full short-circuit, where current flows through the new path, inducing a voltage in the shorted coil.

For each fault type, three severity levels were emulated by varying the percentage of shorted turns in the stator winding:

- Level 1: 1.41% of the winding
- Level 2: 4.81% of the winding
- Level 3: 9.26% of the winding

To prevent permanent damage, the short-circuit current was limited to its rated value using a variable resistor of 50 Ω. Data was collected under three mechanical load conditions: no load

(0%), half load (50%), and full load (100%), with driving frequencies ranging from 30 Hz to 60 Hz in 5 Hz increments. In total, 2,590 patterns were acquired, including 350 from the normal class and 2,240 from various fault conditions.

To monitor the axial leakage flux, a coil of 100 turns of 24 AWG copper wire was placed around the motor shaft. The currents of the three motor phases were measured using SCT013-030 current transformers.

For a comprehensive analysis, it is beneficial to include visual representations of the motor's behavior under normal and faulty conditions across the three load scenarios at a driving frequency of 50 Hz. These visualizations can aid in understanding the impact of ITSC faults under varying operational conditions.

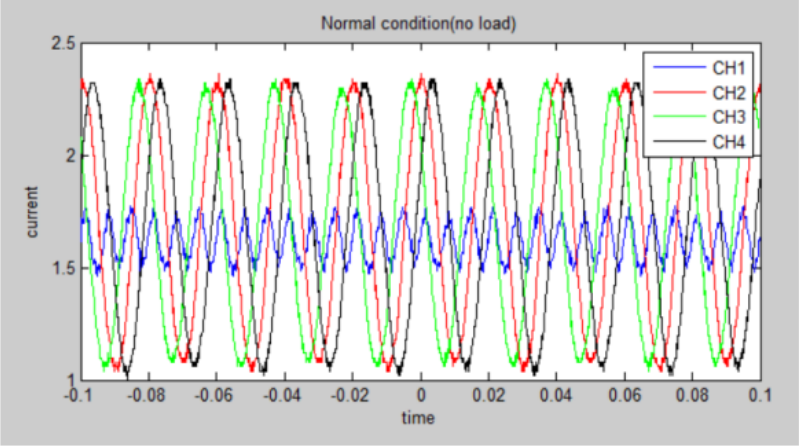


Figure 4.3: No load current graph - Normal

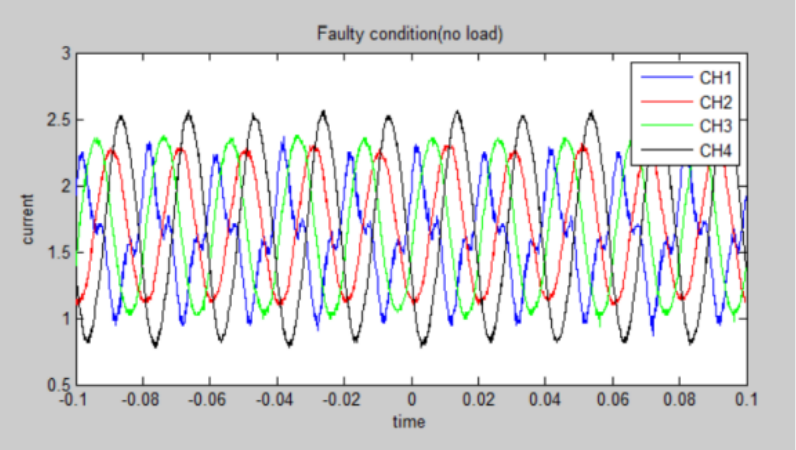


Figure 4.4: No load current graph - Faulty

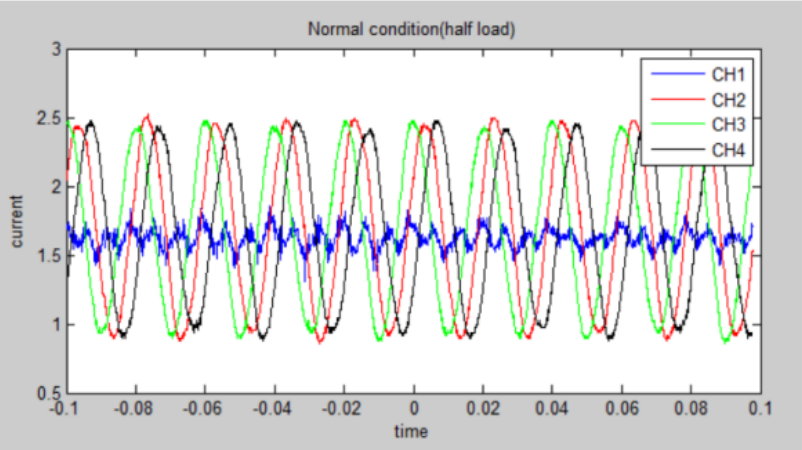


Figure 4.5: Half load current graph - Normal

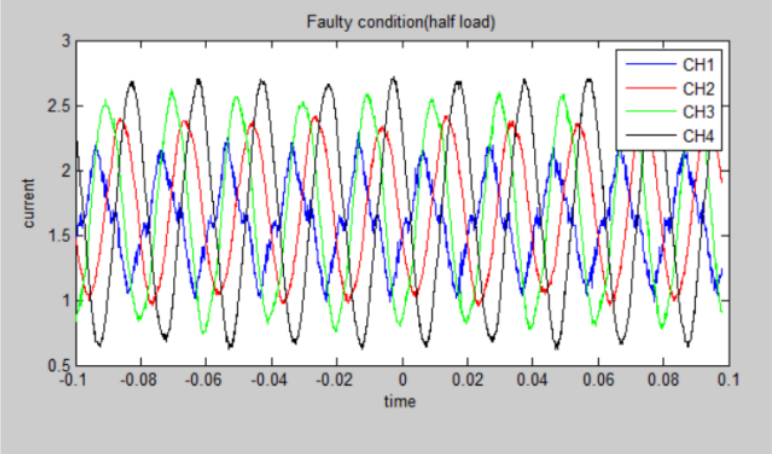


Figure 4.6: Half load current graph - Faulty

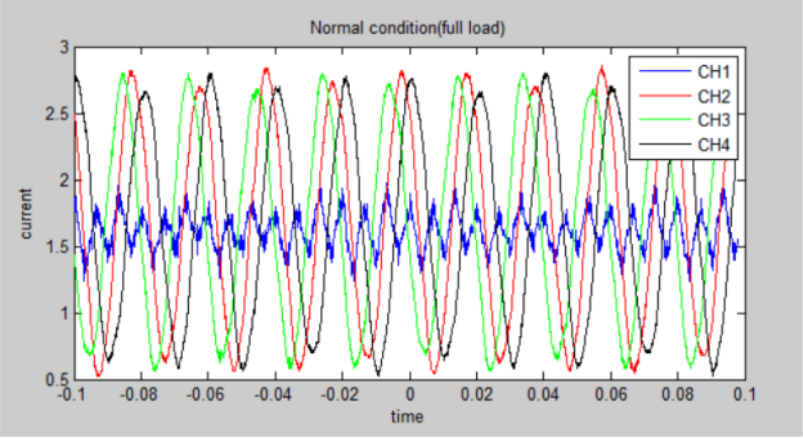


Figure 4.7: Full load current graph - Normal

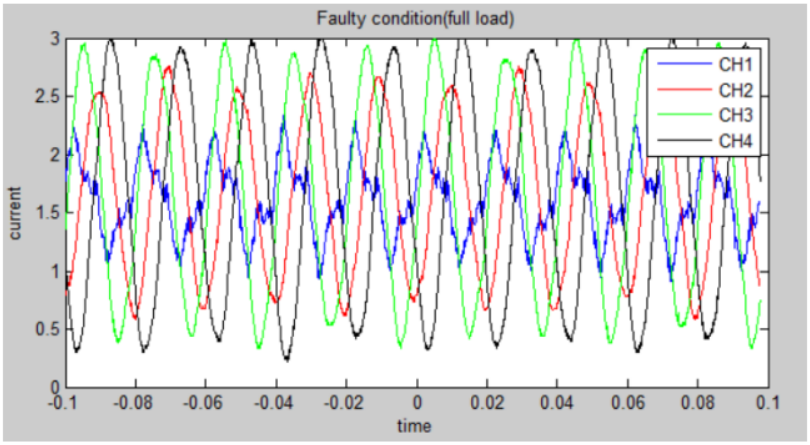


Figure 4.8: Full load current graph - Faulty

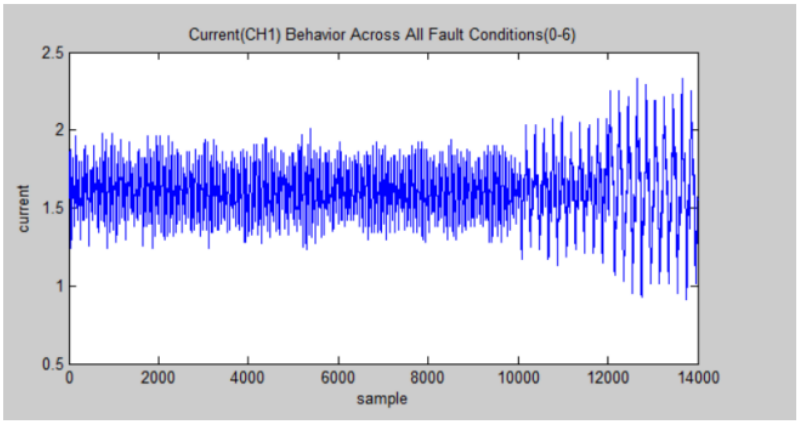


Figure 4.9: Current (CH1) behavior across all fault conditions (0-6)

Time	CH1	CH2	CH3	CH4	Class
-1.02E-01	1.58	2.8	2.2	2.32	0
-1.02E-01	1.6	2.78	2.14	2.38	0
-1.02E-01	1.6	2.8	2.1	2.4	0
-1.02E-01	1.56	2.8	2.06	2.4	0
-1.02E-01	1.64	2.8	2.04	2.46	0
-1.02E-01	1.7	2.78	1.98	2.48	0
-1.02E-01	1.68	2.78	1.92	2.5	0
-1.02E-01	1.64	2.76	1.9	2.5	0
-1.01E-01	1.7	2.72	1.86	2.54	0
-1.01E-01	1.78	2.74	1.84	2.6	0
-1.01E-01	1.68	2.7	1.8	2.6	0
-1.01E-01	1.68	2.68	1.74	2.62	0
-1.01E-01	1.68	2.68	1.72	2.66	0
-1.01E-01	1.68	2.68	1.7	2.66	0
-1.01E-01	1.7	2.66	1.66	2.7	0
-1.01E-01	1.7	2.62	1.62	2.7	0

(a)

Time	CH1	CH2	CH3	CH4	Class
-1.02E-01	1.63	0.86	0.76	1.76	6
-1.02E-01	1.59	0.84	0.8	1.72	6
-1.02E-01	1.63	0.84	0.82	1.68	6
-1.02E-01	1.65	0.82	0.84	1.64	6
-1.02E-01	1.61	0.82	0.84	1.62	6
-1.02E-01	1.49	0.78	0.88	1.58	6
-1.02E-01	1.49	0.8	0.9	1.54	6
-1.02E-01	1.57	0.78	0.92	1.52	6
-1.01E-01	1.55	0.78	0.94	1.52	6
-1.01E-01	1.45	0.78	0.96	1.48	6
-1.01E-01	1.49	0.78	1.02	1.44	6
-1.01E-01	1.59	0.78	1.02	1.4	6
-1.01E-01	1.55	0.76	1.02	1.4	6
-1.01E-01	1.51	0.78	1.04	1.36	6
-1.01E-01	1.51	0.78	1.08	1.32	6

(b)

Figure 4.10: (a) Healthy no-load motor data (b) Faulty no-load motor data

4.4 Data Preprocessing

Raw data is often found in varying formats and ranges, which can hinder a model's performance and its ability to interpret the data effectively [105]. To address these challenges, preprocessing the data becomes a critical step before feeding it into machine learning models. The preprocessing steps in this study included data labeling, feature scaling, PCA for visualization, feature selection analysis, and correlation analysis. Below are the key steps undertaken:

4.4.1 Data Formatting and Labeling

The initial step was to convert the raw dataset into a usable format. The data was originally distributed across multiple Excel sheets, each representing a separate class. These were consolidated into a single sheet, and the respective classes were labeled correctly.

4.4.2 Feature Scaling

Feature scaling [106] was performed to standardize the range of features, as unscaled data could cause models to prioritize features with larger ranges, potentially leading to biased results. Min-Max normalization was applied, which scales the data to a range between 0 and 1. This choice ensures that all features are on a comparable scale, preventing any feature from dominating the model's learning process and helping to improve convergence during training.

Below are the most commonly used scaling techniques:

- **Min-Max Normalization:** Scales values between 0 and 1 [107].
- **Standardization (Z-Score):** Adjusts data to have a mean of zero and a standard deviation of one, ensuring consistency and improving performance across various machine learning algorithms.
- **Mean Normalization:** Centers the data by subtracting the mean and scaling it by the data range, ensuring values fall within a standardized range.

4.4.3 Outlier and Missing Value Analysis

The dataset was carefully examined for outliers and missing values. However, no missing values or significant outliers were identified, ensuring that the data did not require additional

imputation or outlier removal techniques.

4.4.4 Feature Selection and Visualization

To explore the importance of features and their contributions to classification accuracy, multiple techniques were applied:

4.4.4.1 Principal Component Analysis

PCA was used to reduce dimensionality and visualize the data distribution across principal components [108]. Although PCA is often used for feature selection, in this case, it did not improve classification accuracy. This was likely due to the limited number of features in the dataset and the fact that the features contained complementary information. A PCA plot was generated to illustrate the data's separability across different classes in reduced dimensions.

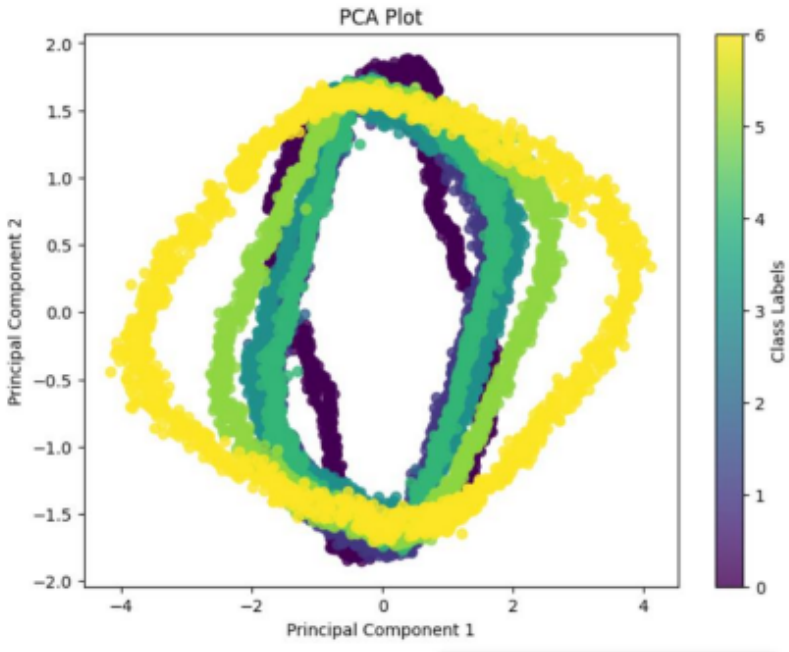


Figure 4.11: PCA plot for ITSC in induction motor dataset

4.4.4.2 Feature Importance Analysis

A feature importance graph was generated using model-based methods to understand the contribution of individual features to the classification task. The analysis suggested that some features (e.g., Feature 5 and Feature 4) had higher importance scores, while others were deemed less significant. However, when training the model using only the most important features,

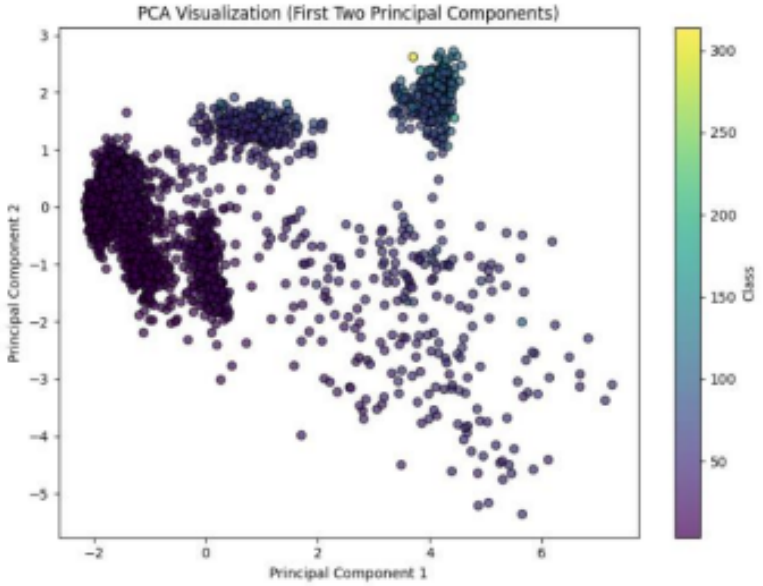


Figure 4.12: PCA plot for CWRU dataset

the performance was lower compared to using all features. This finding highlights that while individual features may seem less critical, their combined interactions contribute to the overall classification accuracy [109].

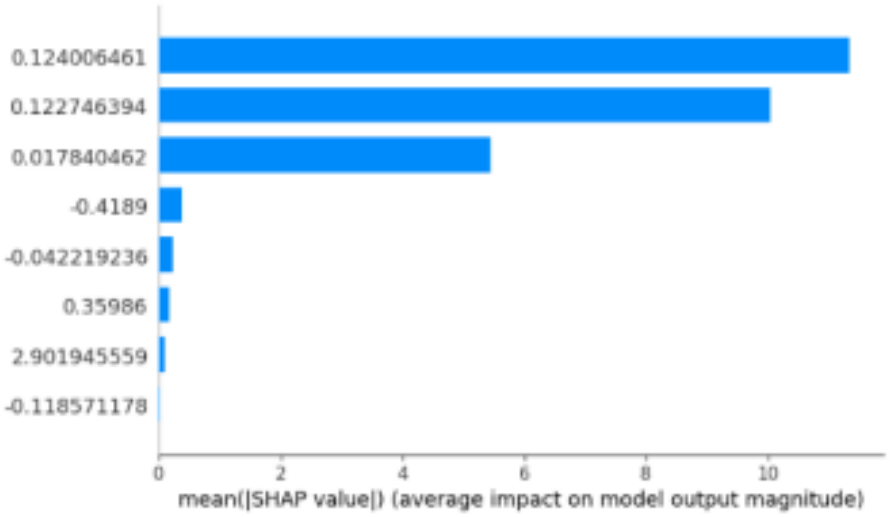


Figure 4.13: Feature importance graph for CWRU dataset

4.4.4.3 Classification Report Comparison

A comparison of the classification report (accuracy, precision, recall, and F1-score) for the model trained with:

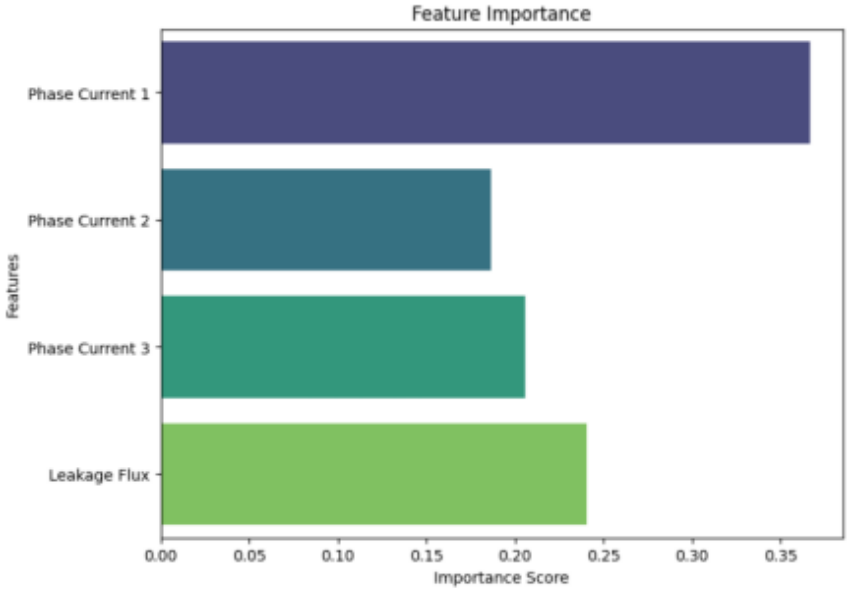


Figure 4.14: Feature importance graph for ITSC in stator winding dataset

- Only important features (as identified in the feature importance analysis).
- All features (retained to capture subtle interdependencies).

The results reinforced the decision to retain all features for achieving optimal performance.

	precision	recall	f1-score	support
0	0.35	0.24	0.29	70
1	0.89	0.94	0.91	62
2	0.89	0.42	0.57	60
3	0.97	1.00	0.98	56
4	0.93	0.88	0.90	73
5	0.99	1.00	0.99	68
6	1.00	1.00	1.00	88
7	1.00	1.00	1.00	65
8	0.49	0.83	0.62	78
9	1.00	0.94	0.97	70
accuracy			0.83	690
macro avg	0.85	0.82	0.82	690
weighted avg	0.85	0.83	0.82	690

Accuracy : 0.8289855072463768

Figure 4.15: Classification report of ANN with only important features (7) used

	precision	recall	f1-score	support
0	0.96	0.93	0.95	76
1	0.96	0.99	0.97	74
2	0.93	0.90	0.92	62
3	0.99	0.99	0.99	76
4	1.00	1.00	1.00	74
5	0.97	0.95	0.96	60
6	1.00	1.00	1.00	62
7	0.96	1.00	0.98	69
8	0.88	0.90	0.89	63
9	1.00	0.97	0.99	74
accuracy			0.97	690
macro avg	0.96	0.96	0.96	690
weighted avg	0.97	0.97	0.97	690

Accuracy : 0.9652173913043478

Figure 4.16: Classification report for ANN with all features (8) used (CWRU dataset)

4.4.4.4 Correlation Analysis

The relationship between features and target classes was examined using a correlation heatmap. This analysis confirmed the absence of multicollinearity among features, further validating the decision to use all features in the dataset. The heatmap provided insights into feature interactions and ensured that no redundant information was present [110].

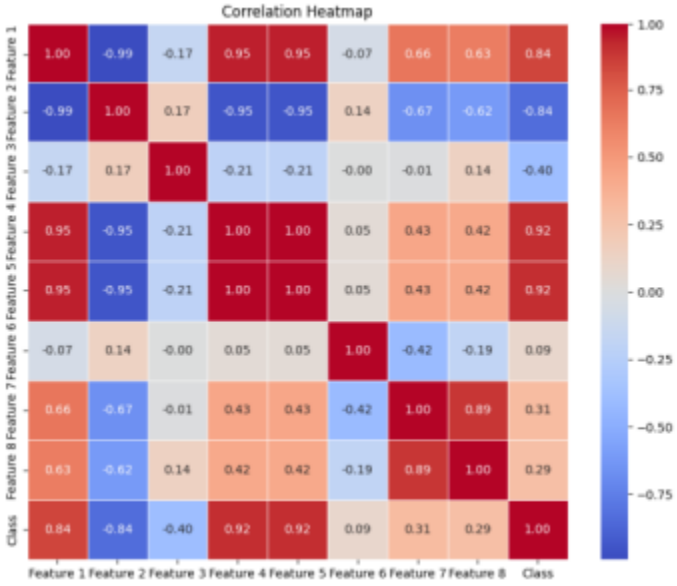


Figure 4.17: Correlation heatmap for CWRU dataset

These visualizations (PCA plot, feature importance graph, and correlation heatmap) provided valuable insights into the dataset's characteristics and its readiness for training.

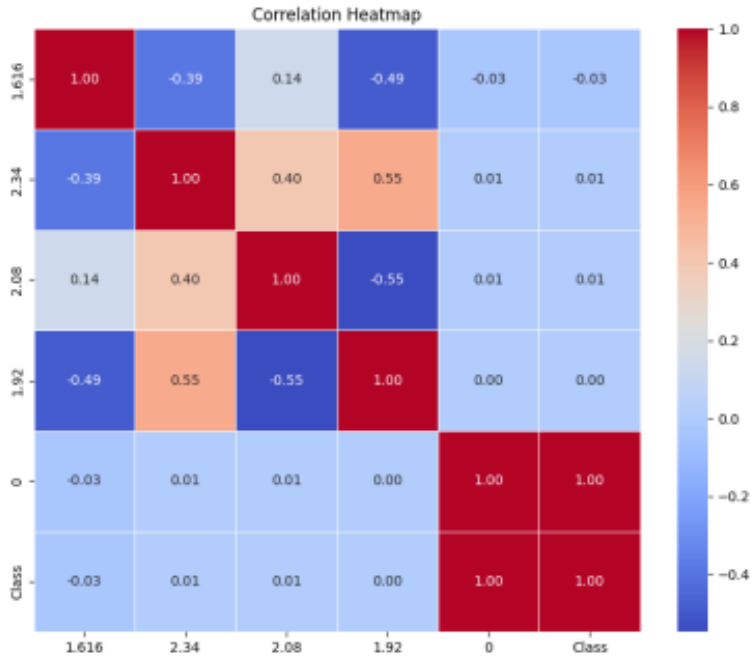


Figure 4.18: Correlation heatmap for ITSC in induction motor dataset

4.5 Model Training

The training of both traditional machine learning models and deep learning architectures was conducted to classify faults in a 3-phase induction motor. The models were trained on the preprocessed dataset, with hyperparameter tuning performed to optimize their performance. The models used and their respective training processes are detailed below.

4.5.1 Traditional Machine Learning Models

4.5.1.1 K-Nearest Neighbors

Rationale: KNN was chosen for its simplicity and effectiveness in handling small to medium-sized datasets, making it a good baseline for classification tasks.

Training: The KNN algorithm was trained using the standard method, where the number of neighbors (k) was optimized. The Euclidean distance was used as the distance metric, and the dataset was scaled to ensure features were on the same scale.

Hyperparameter Tuning: The number of neighbors (k) was fine-tuned using cross-validation. Different values of k were tested, and the value that minimized classification error was selected.

4.5.1.2 Random Forest

Rationale: Random Forest was selected for its ability to handle high-dimensional datasets and its robustness to overfitting due to ensemble learning.

Training: Random Forest was trained as an ensemble model consisting of multiple decision trees. Each tree was trained on a bootstrapped subset of the data, and predictions were made based on the majority vote of the trees.

Hyperparameter Tuning: The key hyperparameters tuned were the number of estimators (number of trees) and the maximum depth of the trees. Grid search and cross-validation were used to optimize these parameters, with the number of trees tested in the range of 50 to 200 and the maximum depth ranging from 10 to 50.

4.5.1.3 Decision Tree

Rationale: Decision Tree was chosen for its interpretability and ability to model non-linear relationships within the dataset.

Training: The Decision Tree classifier was trained to iteratively split the dataset into subsets based on feature importance, maximizing information gain at each split.

Hyperparameter Tuning: The key hyperparameters tuned were the maximum depth of the tree, the minimum samples required to split a node, and the criterion used for splitting (e.g., Gini impurity or entropy). Grid search was used to find the optimal combination, ensuring the model did not overfit.

4.5.1.4 Naive Bayes

Rationale: Naive Bayes was selected due to its simplicity and efficiency in handling multi-class classification problems, especially for small datasets.

Training: The Gaussian Naive Bayes model was trained under the assumption that features are conditionally independent given the class. This assumption simplifies the computation and makes the model efficient, even for large datasets.

Hyperparameter Tuning: As the Gaussian Naive Bayes classifier has limited hyperparameters, the focus was on verifying the assumption of normality for each feature. Minimal tuning was required for this model.

4.5.1.5 Support Vector Machine)

Rationale: SVM was selected for its ability to find optimal hyperplanes and perform well with complex and high-dimensional datasets.

Training: SVM was trained to find an optimal hyperplane that maximized the margin between classes in the feature space. A linear kernel was initially used for simplicity, and the dataset was scaled to improve the algorithm's efficiency.

Hyperparameter Tuning: The primary hyperparameters tuned were the kernel type (e.g., linear, RBF) and the regularization parameter (C), which controls the trade-off between maximizing the margin and minimizing classification error. A grid search was performed to select the optimal combination of these parameters.

4.5.2 Performance Metrics

The performance of all traditional machine learning models was evaluated using the following metrics:

- **Accuracy:** The proportion of correctly classified samples out of the total samples.
- **Precision:** The ratio of true positive predictions to the total positive predictions, reflecting the model's ability to avoid false positives.
- **Recall:** The ratio of true positive predictions to the total actual positives, indicating the model's sensitivity in detecting true cases.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced evaluation metric, especially in cases of imbalanced datasets.

These metrics provided a holistic view of the models' effectiveness in identifying both fault and normal conditions, ensuring a robust comparison.

4.5.3 Deep Learning Models

4.5.3.1 Artificial Neural Network

ANNs are a class of deep learning models inspired by the human brain's neural structure, widely used for classification tasks in engineering and fault diagnosis domains [49]. In this study, ANNs were developed and trained to classify two distinct datasets: the stator winding

fault dataset and the CWRU bearing fault dataset. The models were designed to accurately differentiate between normal and faulty conditions, ensuring robust fault diagnosis for both applications.

ANN for Stator Winding Fault Dataset

Architecture Design The ANN architecture designed for the stator winding fault dataset consists of four dense layers:

- **Input Layer:** Configured to match the feature size of the dataset, ensuring compatibility with the input data.
- **Hidden Layers:**
  - First hidden layer with 64 neurons and ReLU activation.
  - Second hidden layer with 128 neurons and ReLU activation to capture complex feature interactions.
  - Third hidden layer with 64 neurons and ReLU activation to consolidate learned patterns.
- **Output Layer:** A dense layer with softmax activation and a number of neurons equal to the dataset's classes for multi-class classification.

The network design allowed the model to learn complex patterns in the data, with deeper layers facilitating feature extraction and ReLU activation preventing vanishing gradient issues. The softmax activation function in the output layer converted logits into probabilities, enabling accurate classification. The model summary is shown below:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 64)	320
dense_5 (Dense)	(None, 128)	8,320
dense_6 (Dense)	(None, 64)	8,256
dense_7 (Dense)	(None, 6)	390

Total params: 17,286 (67.52 KB)
Trainable params: 17,286 (67.52 KB)
Non-trainable params: 0 (0.00 B)

Figure 4.19: ANN model summary for ITSC in induction motor dataset

Model Compilation The model was compiled with the following configurations:

- **Optimizer:** The Adam optimizer was used for adaptive and efficient learning, dynamically adjusting the learning rate during training.
- **Loss Function:** Categorical Crossentropy was employed since the classification problem involved multiple classes.
- **Metrics:** Accuracy was tracked during training to directly measure the model's classification success.

Training Process The ANN was trained using the following parameters:

- **Training Dataset:** The training dataset was scaled and split into batches to enhance computational efficiency.
- **Batch Size:** A batch size of 10 was used to divide the training data into smaller subsets for gradient updates.
- **Epochs:** The model was trained for 50 epochs to allow iterative optimization of the weights.

The model was fit to the training data using the `fit()` method, which minimized the loss function by updating weights.

Performance Metrics The model's performance was evaluated using the following metrics:

- **Loss Trends:** Training and validation loss trends were monitored to assess the model's learning behavior.
- **Confusion Matrix:** A confusion matrix was generated to evaluate classification performance across all classes.
- **Accuracy, Precision, Recall, and F1-score:** These metrics were used to evaluate the model's classification performance, providing a comprehensive assessment of its predictions.

This setup enabled the ANN to efficiently classify fault and normal conditions in the stator winding dataset.

ANN for CWRU Dataset

Architecture Design The ANN architecture for the bearing fault dataset was designed using the Sequential API from the TensorFlow Keras library. It consisted of a total of seven layers, including input, hidden, and output layers, with the following details:

- **Input Layer:**
  - Configured to accept data with 8 features, corresponding to the dataset's feature space.
- **Hidden Layers:**
  - Layer 1: 16 neurons with ReLU activation.
  - Layer 2: 32 neurons with ReLU activation.
  - Layer 3: 64 neurons with ReLU activation.
  - Layer 4: 128 neurons with ReLU activation.
  - Layer 5: 64 neurons with ReLU activation.
  - Layer 6: 32 neurons with ReLU activation.
- **Output Layer:**
  - The output layer consisted of 10 neurons with a softmax activation function, designed to output probabilities for 10 different classes.

The architecture was progressively deep to enable the model to learn intricate relationships in the data, with ReLU activation ensuring efficient gradient flow. The softmax activation function allowed multi-class classification by converting logits into probabilities. The model summary is shown below:

Model Compilation The model was compiled with the following configurations:

- **Optimizer:** The Adam optimizer was employed for efficient learning and dynamic adjustment of the learning rate.
- **Loss Function:** Categorical Crossentropy was selected for this multi-class classification task.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 16)	144
dense_1 (Dense)	(None, 32)	544
dense_2 (Dense)	(None, 64)	2,112
dense_3 (Dense)	(None, 128)	8,320
dense_4 (Dense)	(None, 64)	8,256
dense_5 (Dense)	(None, 32)	2,080
dense_6 (Dense)	(None, 10)	330

Total params: 21,786 (85.10 KB)
Trainable params: 21,786 (85.10 KB)
Non-trainable params: 0 (0.00 B)

Figure 4.20: ANN model summary for CWRU dataset

- **Metrics:** Accuracy was tracked during training to evaluate the model's classification success.

Training Process The ANN was trained using the following parameters:

- **Training Dataset:** The training dataset was scaled and split into batches for efficient computation.
- **Batch Size:** A batch size of 10 was used, dividing the data into manageable subsets for gradient updates.
- **Epochs:** The model was trained for 50 epochs, allowing sufficient optimization of the model's weights.

The `fit()` method was used for training, and early stopping was employed to monitor validation loss and prevent overfitting.

Performance Metrics The model's performance was evaluated using the following metrics:

- **Accuracy, Precision, Recall, and F1-score:** These metrics were used to evaluate the model's classification performance, providing a comprehensive assessment of its predictions.
- **Loss Trends:** Training and validation loss trends were analyzed to ensure proper learning.

- **Confusion Matrix:** A confusion matrix was generated to analyze classification performance across all classes.

This robust setup enabled the ANN to achieve accurate classification of bearing faults and normal conditions in the dataset.

4.5.3.2 Long Short-Term Memory

LSTM networks are a specialized type of recurrent neural network (RNN) designed to capture long-range dependencies in sequential data, making them highly suitable for time-series classification tasks. Unlike traditional RNNs, LSTMs incorporate memory cells and gating mechanisms (input, forget, and output gates) to regulate the flow of information and prevent issues like vanishing gradients. This capability makes LSTMs an effective choice for fault diagnosis in engineering applications, where temporal patterns play a crucial role in identifying faults.

LSTM for Stator Winding Fault Dataset

Architecture Design The LSTM model for the stator winding fault dataset was designed to process sequential data from motor sensors, capturing temporal dependencies in fault occurrences. The architecture is structured as follows:

- **Input Layer:** Accepts sequences of sensor readings (three-phase currents and leakage flux) over a predefined number of timesteps.
- **Hidden Layers:**
  - First LSTM layer with 128 units, ReLU activation, and return sequences enabled for deeper feature extraction.
  - Dropout layer (30%) to prevent overfitting.
  - Second LSTM layer with 64 units, ReLU activation, and return sequences enabled.
  - Dropout layer (30%) to enhance generalization.
  - Third LSTM layer with 32 units and ReLU activation to refine extracted temporal features.
  - Dense layer with 64 neurons and ReLU activation for additional feature learning.

- **Output Layer:** A dense layer with softmax activation and a number of neurons equal to the dataset's classes (7), ensuring multi-class classification.

The model summary is shown below:

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 128)	68,096
dropout (Dropout)	(None, 10, 128)	0
lstm_1 (LSTM)	(None, 10, 64)	49,408
dropout_1 (Dropout)	(None, 10, 64)	0
lstm_2 (LSTM)	(None, 32)	12,416
dense (Dense)	(None, 64)	2,112
dense_1 (Dense)	(None, 7)	455

Total params: 132,487 (517.53 KB)
Trainable params: 132,487 (517.53 KB)
Non-trainable params: 0 (0.00 B)

Figure 4.21: LSTM model summary for ITSC in induction motor dataset

This architecture allows the model to learn sequential dependencies in sensor data while leveraging dropout layers to improve robustness.

Model Compilation The LSTM model was compiled with the following configurations:

- **Optimizer:** Adam optimizer with a learning rate of 0.0005 for adaptive and efficient learning.
- **Loss Function:** Categorical Crossentropy was used, given the multi-class classification nature of the task.
- **Metrics:** Accuracy was tracked to evaluate the model's classification success.

Training Process The LSTM model was trained using the following parameters:

- **Training Dataset:** The dataset was normalized, and sequences were created with 10 timesteps.
- **Batch Size:** A batch size of 32 was selected to balance training stability and computational efficiency.
- **Epochs:** The model was trained for 50 epochs to optimize the weight parameters.

The `fit()` method was used to train the model, minimizing the loss function iteratively while monitoring validation performance.

Performance Metrics The model's performance was assessed using the following evaluation criteria:

- **Loss Trends:** Training and validation loss trends were analyzed to track the model's learning progression.
- **Confusion Matrix:** A confusion matrix was generated to examine class-wise prediction performance.
- **Accuracy, Precision, Recall, and F1-score:** These metrics were computed to provide a comprehensive assessment of the model's classification performance.

This approach enabled the LSTM model to effectively classify stator winding faults based on time-series sensor data.

LSTM for CWRU Bearing Fault Dataset

Architecture Design The LSTM model for the CWRU bearing fault dataset was developed to classify vibration signals into multiple bearing fault categories. The architecture consists of:

- **Input Layer:** Accepts sequences of vibration signal readings over a predefined number of timesteps.
- **Hidden Layers:**
  - First LSTM layer with 64 units, ReLU activation, and return sequences enabled for hierarchical feature extraction.
  - Dropout layer (20%) to mitigate overfitting.
  - Second LSTM layer with 32 units and ReLU activation.
  - Dropout layer (20%) to improve generalization.
  - Dense layer with 64 neurons and ReLU activation for enhanced feature representation.

- **Output Layer:** A dense layer with softmax activation and 10 neurons, corresponding to the number of bearing fault categories.

The model summary is shown below:

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 64)	18,688
dropout (Dropout)	(None, 10, 64)	0
lstm_1 (LSTM)	(None, 32)	12,416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 64)	2,112
dense_1 (Dense)	(None, 10)	650

Total params: 33,866 (132.29 KB)
Trainable params: 33,866 (132.29 KB)
Non-trainable params: 0 (0.00 B)

Figure 4.22: LSTM model summary for CWRU dataset

The use of multiple LSTM layers ensures that temporal dependencies in vibration signals are effectively captured, enhancing the fault classification process.

Model Compilation The model was compiled with:

- **Optimizer:** Adam optimizer with an adaptive learning rate of 0.0005.
- **Loss Function:** Categorical Crossentropy for multi-class classification.
- **Metrics:** Accuracy to evaluate classification effectiveness.

Training Process The LSTM model was trained under the following conditions:

- **Training Dataset:** The vibration signal dataset was preprocessed and converted into sequences with a fixed number of timesteps.
- **Batch Size:** A batch size of 32 was chosen for efficient training.
- **Epochs:** The model was trained for 50 epochs to allow optimal learning.

The model was trained using the `fit()` method, with early stopping implemented to prevent overfitting by monitoring validation loss.

Performance Metrics To evaluate the model's classification performance, the following metrics were used:

- **Loss Trends:** Training and validation loss trends were analyzed.
- **Confusion Matrix:** A confusion matrix was generated to assess the accuracy of each class prediction.
- **Accuracy, Precision, Recall, and F1-score:** These metrics were calculated to measure classification effectiveness.

By leveraging LSTM's ability to learn sequential patterns in vibration signals, the model achieved accurate classification of bearing faults, contributing to an effective predictive maintenance framework.

4.5.4 Physics-Informed Neural Networks

PINNs integrate physical laws into neural network training, enhancing predictive accuracy by embedding domain-specific knowledge directly into the learning process. This approach is particularly beneficial in fields like engineering, where systems are governed by well-established physical principles.

4.5.4.1 Architecture Design

The PINN model was developed to classify stator winding faults in a three-phase induction motor by utilizing both sensor data and domain knowledge. The architecture consists of:

- **Input Layer**: Accepts features such as three-phase current signals (I_1, I_2, I_3) and leakage flux (ϕ).
- **Hidden Layers**:
  - Fully connected dense layers with 64, 128, and 64 neurons, all using ReLU activation to capture nonlinear dependencies.
- **Output Layer**: A dense layer with softmax activation and a number of neurons equal to the fault classes (7), ensuring multi-class classification.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	320
dense_1 (Dense)	(None, 128)	8,320
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 7)	455

Total params: 17,351 (67.78 KB)
Trainable params: 17,351 (67.78 KB)
Non-trainable params: 0 (0.00 B)

Figure 4.23: PINNs model summary

The model is summarized as follows:

This architecture enables the model to capture essential fault patterns while integrating domain-specific constraints through the physics-informed loss function.

4.5.4.2 Physics-Informed Loss Function

A key component of PINNs is the physics-informed loss function, which consists of two components:

1. **Classification Loss**: Sparse categorical cross-entropy is used to optimize the model for accurate fault classification.

2. **Physics Loss**: Constraints derived from electrical laws are enforced to guide learning:

   - **Current Imbalance Constraint**: Under normal conditions, the sum of the three-phase currents should be zero:

$$I_1 + I_2 + I_3 = 0$$

   - **Flux Consistency Constraint**: The leakage flux should be correlated with the sum of the phase currents:

$$\phi \propto I_1 + I_2 + I_3$$

The total loss function is defined as:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{classification}} + \lambda \mathscr{L}_{\text{physics}}$$

where λ is a weighting factor that balances the influence of physics constraints.

4.5.4.3 Model Compilation and Training

The PINN model was compiled using:

- **Optimizer**: Adam optimizer with an adaptive learning rate for efficient convergence.
- **Loss Function**: The custom physics-informed loss function incorporating both classification and physics losses.
- **Metrics**: Accuracy was tracked to evaluate classification performance.

The training process involved:

- **Training Dataset**: The dataset was standardized, and sequences were prepared for efficient learning.
- **Batch Size**: A batch size of 16 was used for stable training.
- **Epochs**: The model was trained for 50 epochs to optimize parameter learning.
- **Validation Strategy**: A validation dataset was used to monitor model performance and prevent overfitting.

4.5.4.4 Performance Evaluation

The PINN model's effectiveness was assessed using:

- **Loss Trends**: Training and validation loss curves to analyze learning progression.
- **Confusion Matrix**: To evaluate class-wise fault classification accuracy.
- **Accuracy, Precision, Recall, and F1-score**: Standard evaluation metrics were computed to measure classification performance.

By integrating physics laws into the learning process, the PINN model enhanced its fault classification accuracy and reliability, demonstrating its potential for predictive maintenance applications in induction motors.

4.6 Model Evaluation and Comparison

To comprehensively assess our classification models, we employ evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices. Accuracy measures the proportion of correctly classified instances, while precision indicates the exactness of positive predictions. Recall reflects the model's ability to identify all relevant instances, and the F1-score balances precision and recall. The confusion matrix provides detailed insights into specific misclassifications. These metrics offer a holistic view of each model's performance, allowing us to identify strengths and weaknesses. In the Results and Discussion chapter, we will present a detailed comparative analysis based on these metrics to evaluate each model's effectiveness in relation to our research objectives.

Chapter 5

Results and Discussion

5.1 Introduction

In this chapter, we present our research on predicting faults in induction motors. We focused on two common faults: short circuits in the stator winding and bearing faults. To study these, we used two datasets: one specifically for stator winding faults and another from the CWRU for bearing faults[95].

We trained several machine learning models to identify these faults. This included traditional classifiers such as SVM, KNN, Random Forest, Decision Tree, and Naive Bayes. We also implemented advanced models like ANN, LSTM networks and PINNs.

To evaluate each model's performance, we used metrics including accuracy, precision, recall, and F1-score. We also analyzed confusion matrices to understand how well each model identified different types of motor faults.

All model development and training were conducted using Python in the Google Colab environment, which provides a cloud-based Jupyter notebook interface with access to GPUs, enabling efficient execution of computationally intensive tasks. We utilized libraries such as NumPy for numerical computations, Pandas for data manipulation and analysis, and Scikit-learn for implementing traditional machine learning algorithms. For building and training deep learning models, we employed TensorFlow and Keras, while Matplotlib and Seaborn facilitated data visualization.

This comprehensive approach ensured robust data preprocessing, model development, and performance evaluation, setting the foundation for the detailed analysis presented in the subsequent sections of this chapter.

5.2 Data Preprocessing

As explained in Section 4.2.1, the datasets were preprocessed using Min-Max normalization, where a custom function was employed to scale the time-series values within the range [0,1]. Additionally, data was organized and labeled to reflect different fault conditions, while PCA was utilized for visualization, though it did not enhance classification accuracy. Feature importance analysis revealed that while some features were individually less significant, their combined use optimized model performance. Correlation analysis confirmed the absence of multicollinearity, validating the inclusion of all features in the dataset.

5.3 Results for the Stator Winding Fault Dataset

The performance of traditional classifiers was evaluated under various load conditions. Below are the findings for the no-load condition. The following figures present the classification reports and confusion matrices for each model:

5.3.1 K-Nearest Neighbors

The KNN classifier achieved an overall accuracy of 74.37%, indicating a moderate performance in classifying stator winding faults. The classification report (Figure 5.31) highlights that Class 0 (healthy) and Class 6 (most severe fault) were classified with high precision and recall, suggesting that KNN effectively distinguishes between normal and extreme fault conditions. However, intermediate fault classes, particularly Classes 2, 3, and 4, showed lower precision and recall due to overlapping features, leading to frequent misclassifications. The confusion matrix (Figure 5.32) further confirms this issue, as significant misclassifications occurs between these adjacent fault levels. Despite its simplicity, KNN provided reasonable classification performance but struggled with distinguishing gradual fault severity variations, making it less effective for complex multi-class fault detection scenarios.

Classification Report for KNN:

	precision	recall	f1-score	support
0	0.91	0.95	0.93	407
1	0.62	0.62	0.62	388
2	0.53	0.58	0.55	417
3	0.80	0.79	0.80	404
4	0.56	0.51	0.54	445
5	0.89	0.90	0.90	382
6	0.94	0.91	0.93	358
accuracy			0.74	2801
macro avg	0.75	0.75	0.75	2801
weighted avg	0.74	0.74	0.74	2801

Figure 5.1: KNN classification report

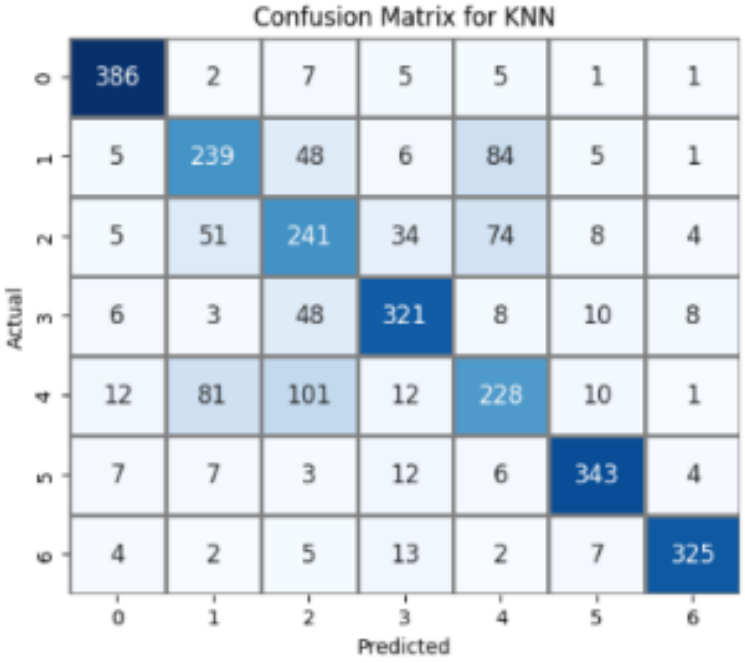


Figure 5.2: KNN confusion matrix

5.3.2 Support Vector Machine

The SVM model achieved 63% accuracy, with high precision and recall for Class 0 (healthy) and Class 6 (severe fault), as shown in Figure 5.40. However, intermediate fault classes (2, 3, and 4) showed lower F1-scores due to misclassifications, which is clearly visible in the confusion matrix (Figure 5.39). The model struggled to distinguish between adjacent fault levels, indicating limitations in handling overlapping fault features.

Classification Report for SVM:

	precision	recall	f1-score	support
0	0.87	0.76	0.81	407
1	0.58	0.61	0.59	388
2	0.47	0.41	0.44	417
3	0.78	0.62	0.69	404
4	0.40	0.38	0.39	445
5	0.55	0.85	0.67	382
6	0.86	0.84	0.85	358
accuracy			0.63	2801
macro avg	0.65	0.64	0.63	2801
weighted avg	0.64	0.63	0.63	2801

Figure 5.3: SVM classification report

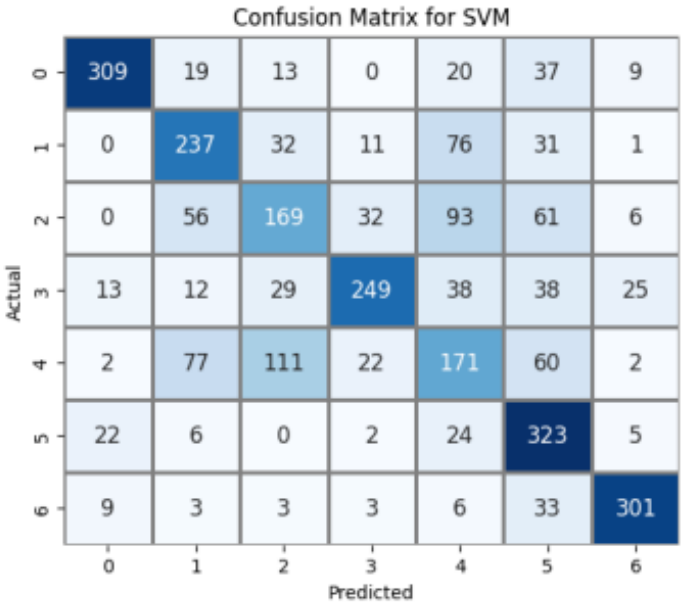


Figure 5.4: Confusion matrix for SVM

5.3.3 Random Forest

The Random Forest classifier demonstrates improved performance compared to SVM, achieving an overall accuracy of 77% with a macro F1-score of 78%, as evidenced in Figure 5.34. The classification report indicates better precision and recall across most classes. The confusion matrix (Figure 5.33) shows fewer misclassifications, particularly for classes 0 and 6, suggesting that Random Forest effectively distinguishes between different fault categories in the dataset.

Classification Report for Random Forest:

	precision	recall	f1-score	support
0	0.93	0.94	0.93	407
1	0.66	0.61	0.63	388
2	0.57	0.58	0.57	417
3	0.82	0.85	0.84	404
4	0.58	0.57	0.57	445
5	0.94	0.96	0.95	382
6	0.95	0.95	0.95	358
accuracy			0.77	2801
macro avg	0.78	0.78	0.78	2801
weighted avg	0.77	0.77	0.77	2801

Figure 5.5: Classification report for Random Forest

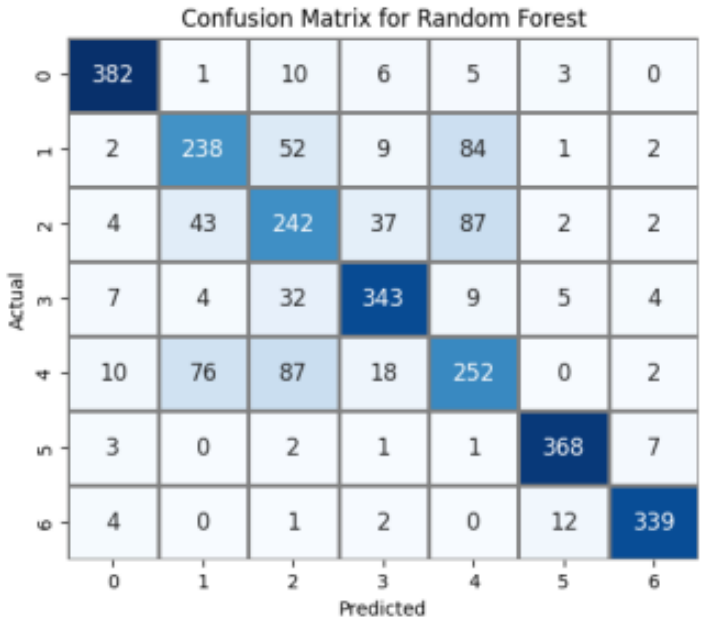


Figure 5.6: Confusion matrix for Random Forest

5.3.4 Decision Tree

The Decision Tree classifier achieves an accuracy of 74%, slightly lower than Random Forest, with a macro F1-score of 75%. As shown in Figure 5.36, the classification report shows reasonable precision and recall for most classes, but certain classes, such as 1 and 4, exhibit lower performance. The confusion matrix in Figure 5.35 indicates a higher degree of misclassification compared to Random Forest, suggesting that while Decision Tree captures

patterns effectively, it may overfit to the training data, leading to reduced generalization.

Classification Report for Decision Tree:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	407
1	0.57	0.58	0.57	388
2	0.53	0.57	0.55	417
3	0.81	0.82	0.82	404
4	0.53	0.47	0.50	445
5	0.93	0.95	0.94	382
6	0.96	0.96	0.96	358
accuracy			0.74	2801
macro avg	0.75	0.75	0.75	2801
weighted avg	0.74	0.74	0.74	2801

Figure 5.7: Classification report for Decision Tree

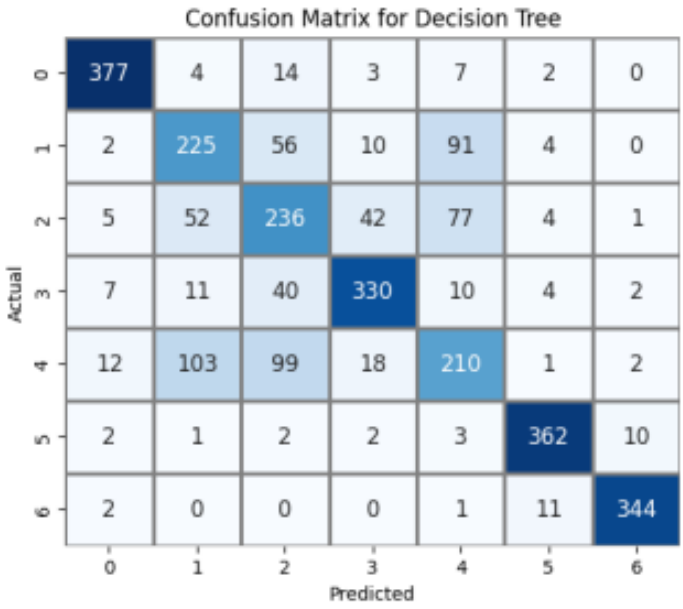


Figure 5.8: Confusion matrix for Decision Tree

5.3.5 Naïve Bayes

The Naïve Bayes classifier demonstrates significantly lower performance compared to other models, achieving an accuracy of 24% and a macro F1-score of 24%. The classification report in Figure 5.38 highlights poor precision and recall across most classes, except for class 6, which shows relatively better precision but still suffers from misclassifications. The confusion matrix in Figure 5.37 indicates widespread misclassification, with many instances being incorrectly

assigned to different categories. The assumption of feature independence in Naïve Bayes may not align well with the dataset characteristics, leading to suboptimal classification performance.

Classification Report for Naive Bayes:

	precision	recall	f1-score	support
0	0.16	0.15	0.16	407
1	0.19	0.46	0.27	388
2	0.19	0.13	0.16	417
3	0.21	0.14	0.17	404
4	0.22	0.15	0.18	445
5	0.20	0.18	0.19	382
6	0.69	0.49	0.58	358
accuracy			0.24	2801
macro avg	0.27	0.24	0.24	2801
weighted avg	0.26	0.24	0.24	2801

Figure 5.9: Naïve Bayes classification report

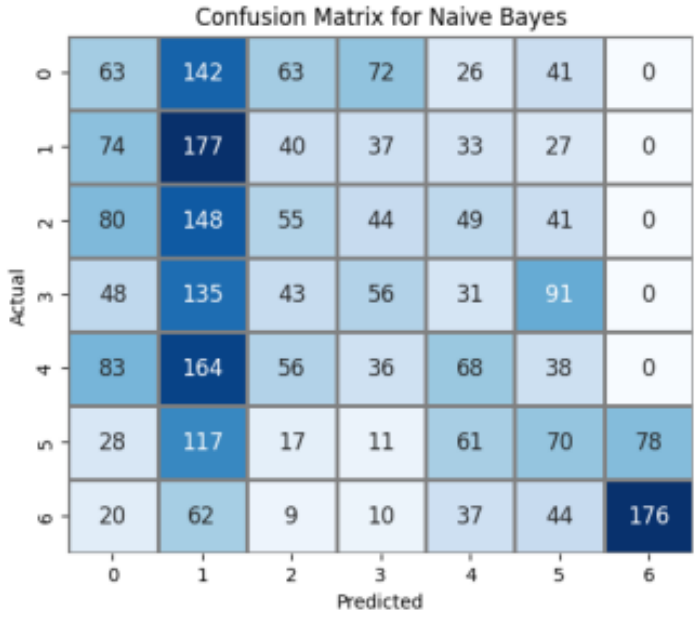


Figure 5.10: Confusion matrix for Naïve Bayes

5.3.6 Artificial Neural Networks

The ANN model demonstrates strong classification performance, achieving an accuracy of 74% with a macro F1-score of 75%. The classification report in Figure 5.42 indicates that the model performs exceptionally well for certain classes, particularly class 0 and class 6, which

achieve F1-scores above 0.90. The confusion matrix in Figure 5.41 shows that the majority of predictions align well with actual labels, with a structured pattern of classification across different fault types.

The training and validation loss curves in Figure 5.13 indicate effective learning, with a steady decline in loss over epochs, demonstrating that the model generalizes well. Similarly, the accuracy curves in Figure 5.14 show consistent improvement, with both training and validation accuracy stabilizing at a high level, indicating a well-trained model. These results suggest that the ANN effectively captures patterns in the data, making it a reliable approach for fault classification.

Classification Report for ANN:

	precision	recall	f1-score	support
0	0.89	0.92	0.91	407
1	0.71	0.57	0.63	388
2	0.55	0.50	0.52	417
3	0.71	0.91	0.80	404
4	0.56	0.54	0.55	445
5	0.86	0.89	0.87	382
6	0.93	0.91	0.92	358
accuracy			0.74	2801
macro avg	0.75	0.75	0.74	2801
weighted avg	0.74	0.74	0.74	2801

Figure 5.11: Classification report for ANN

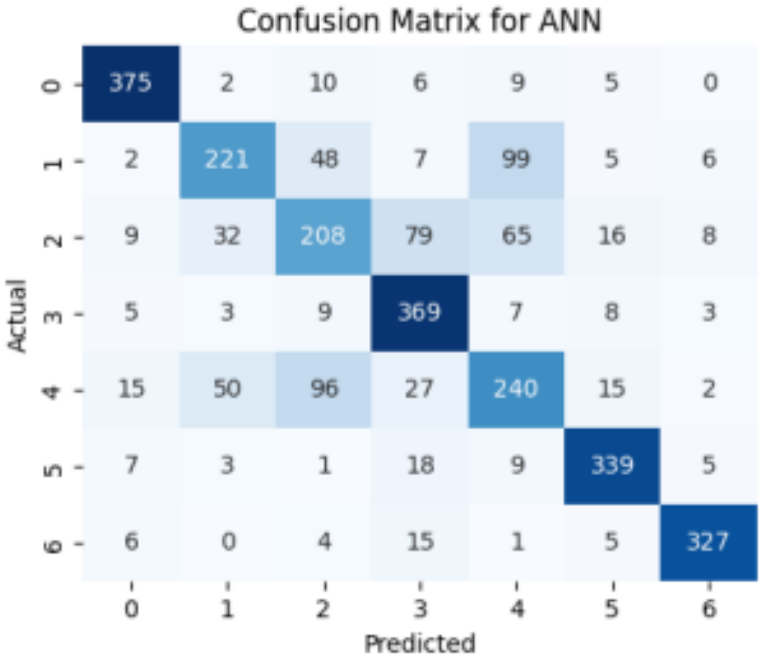


Figure 5.12: Confusion matrix for ANN

Figure 5.13: Training and validation loss for ANN

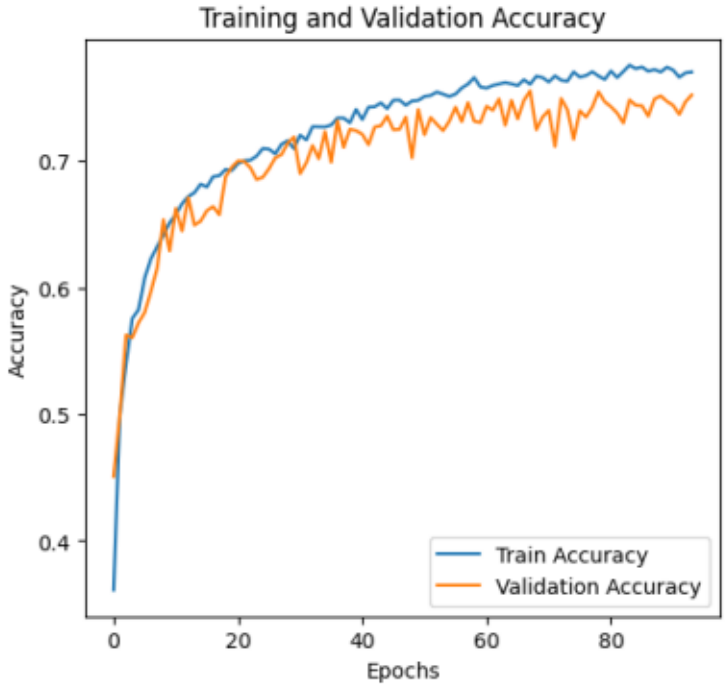


Figure 5.14: Training and validation accuracy for ANN

5.3.7 Long Short-Term Memory

Based on the results obtained for the LSTM model, the classification report in Figure 5.44 indicates high precision, recall, and F1-scores across all classes, achieving an overall accuracy of 89%. The confusion matrix in Figure 5.43 demonstrates that most samples are correctly classified, with minimal misclassifications. The training and validation loss curves in Figure 5.17 show a consistent decrease, suggesting effective learning without significant overfitting. Similarly, the accuracy curves in Figure 5.18 indicate a steady improvement, with validation accuracy closely following training accuracy, further confirming the model's generalization capability.

These results highlight the effectiveness of the LSTM model in classifying faults in the dataset. The high performance across different metrics suggests that the model successfully captures temporal dependencies in the data, leading to precise fault classification. The consistent trends in training and validation performance confirm that the model has learned meaningful patterns, making it a reliable choice for predictive maintenance applications.

Classification Report for LSTM:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	406
1	0.88	0.79	0.83	397
2	0.74	0.81	0.77	402
3	0.97	0.93	0.95	398
4	0.74	0.79	0.76	420
5	0.98	0.97	0.97	381
6	0.99	0.99	0.99	395
accuracy			0.89	2799
macro avg	0.90	0.90	0.90	2799
weighted avg	0.90	0.89	0.90	2799

Figure 5.15: Classification report for LSTM

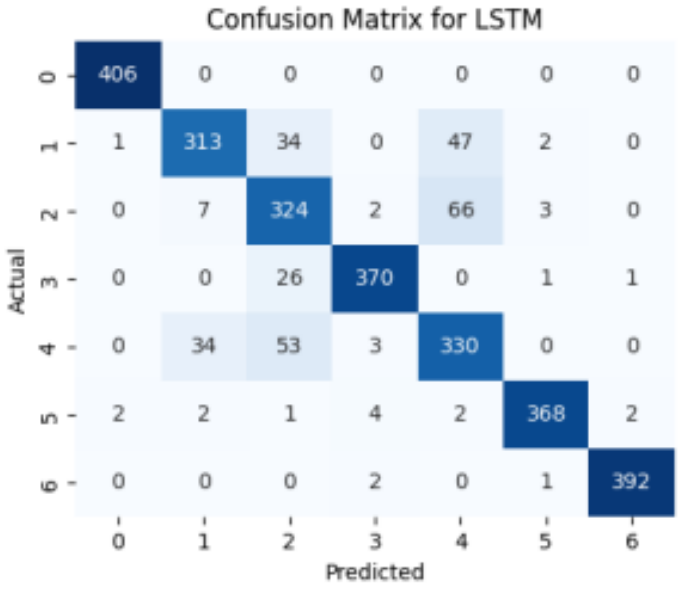


Figure 5.16: Confusion matrix for LSTM

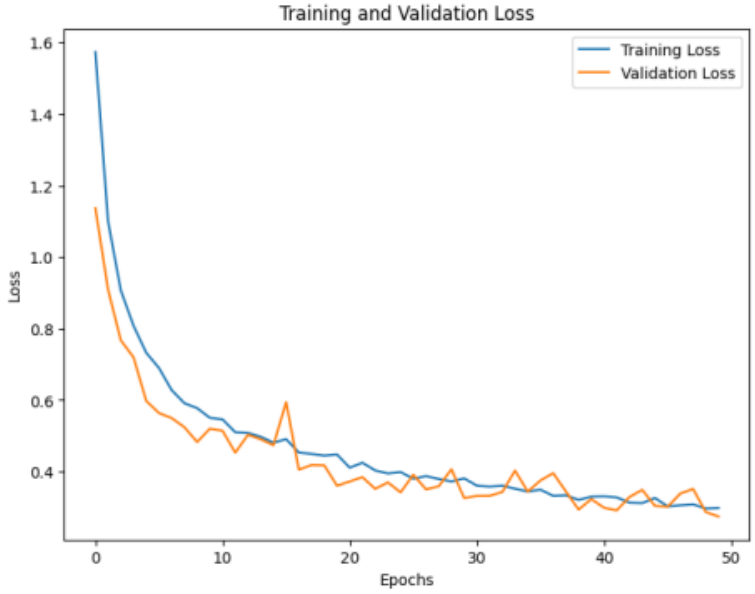


Figure 5.17: Training and validation loss for LSTM

Figure 5.18: Training and validation accuracy for LSTM

5.3.8 Voting Classifier

The voting classifier achieved a high overall accuracy of 94%, as shown in the classification report in Figure 5.45. Precision, recall, and F1-scores are consistently strong across all classes, indicating reliable fault classification. The confusion matrix in Figure 5.46 demonstrates minimal misclassifications, with most samples correctly assigned to their respective categories. These results highlight the effectiveness of the ensemble approach in improving classification performance by combining multiple models' predictions.

The model's high accuracy and balanced performance across all metrics confirm its robustness in handling the dataset. The voting classifier effectively integrates multiple classifiers' strengths, leading to improved generalization and reduced misclassification rates. This makes it a strong candidate for predictive maintenance applications, ensuring reliable fault detection and classification.

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	406
1	0.88	0.84	0.86	397
2	0.91	0.90	0.91	402
3	0.99	0.98	0.99	398
4	0.82	0.87	0.85	420
5	0.99	0.99	0.99	381
6	1.00	0.99	1.00	395
accuracy			0.94	2799
macro avg	0.94	0.94	0.94	2799
weighted avg	0.94	0.94	0.94	2799

Figure 5.19: Voting classifier classification report

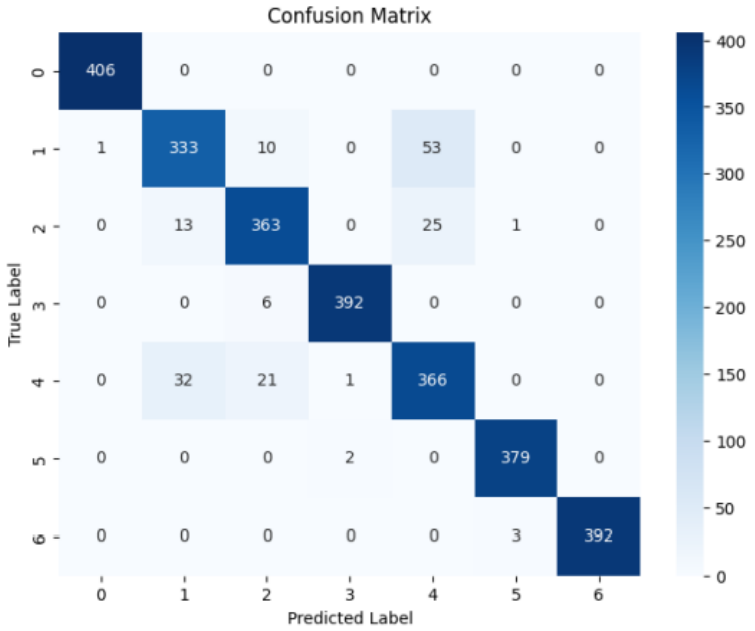


Figure 5.20: Voting classifier confusion matrix

5.3.9 Discussion on Model Performance for Stator Winding Fault Dataset

5.3.9.1 No-Load Condition

The performance comparison of all classifiers on the stator winding fault dataset with seven classes is summarized in Table 5.1 and Figure 5.21.

Model	Accuracy (%)
KNN	74.37
SVM	62.79
Random Forest	77.79
Decision Tree	74.15
Naïve Bayes	23.74
ANN	74.00
LSTM	89.42
Voting Classifier	94.00

Table 5.1: Comparison of classifiers accuracy

The results show that the voting classifier achieved the highest accuracy of 94%, followed by LSTM with 89.42% and Random Forest with 77.79%. Among the traditional classifiers, KNN and Decision Tree performed similarly, with accuracies of 74.36% and 74.15%, respectively. In contrast, SVM struggled with an accuracy of only 62.79%, and Naïve Bayes performed the worst at 23.74%, indicating that it is not well-suited for this dataset. The ANN model achieved 74% accuracy, demonstrating a balanced performance but with clear challenges in distinguishing between fault classes of varying severity. The classification report highlights that Classes 0 (healthy) and 6 (most severe fault) were classified with high precision and recall values, while intermediate fault classes (Classes 1-5) had lower classification accuracy due to feature overlap among different levels of stator winding faults.

The misclassification in the dataset is primarily caused by the similarity between adjacent fault classes, making it difficult for models to distinguish between them effectively. The classification report for the ANN model shows that while Classes 0 and 6 achieved high precision and recall, Classes 2, 3, and 4 had much lower values, indicating significant confusion between these categories. This is expected, as early-stage faults exhibit signal characteristics that closely resemble normal motor operation, leading to incorrect classifications. Additionally, the dataset structure contributes to these results - since the progression from minor to severe short-circuit faults is gradual, feature values overlap, making it harder for classifiers to draw clear decision boundaries as shown in the PCA plot in Figure 5.22.

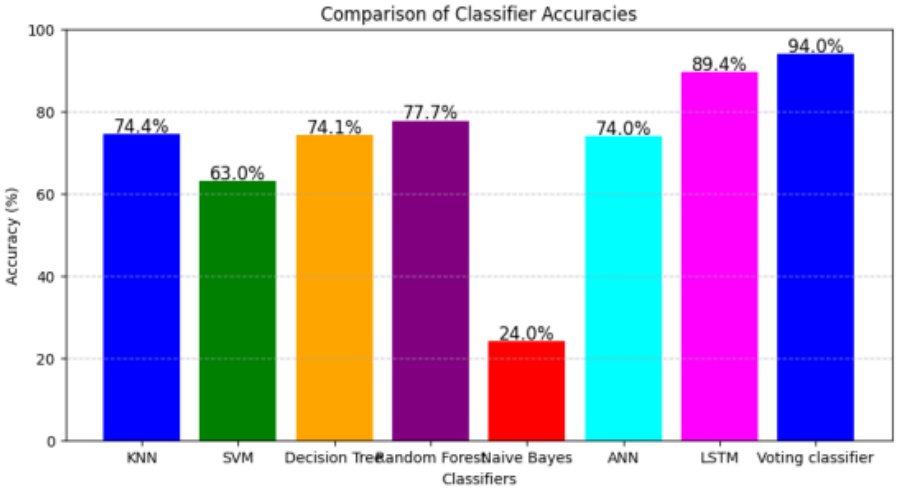


Figure 5.21: Comparison of classifiers accuracy

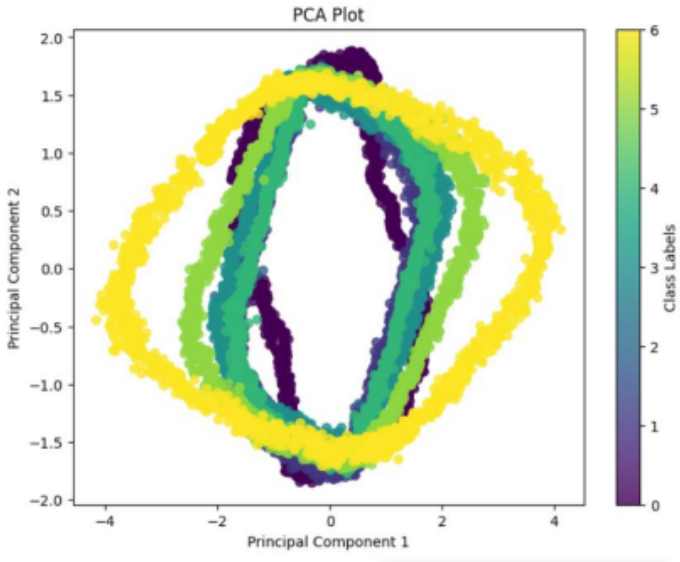


Figure 5.22: PCA plot for ITSC in induction motor dataset

However, deep learning models such as LSTM demonstrated superior performance, achieving 89.42% accuracy. LSTM's ability to learn temporal dependencies in sequential data makes it particularly effective for fault classification, as it can capture subtle differences in motor behavior over time.

To further analyze the classification performance, we conducted a binary classification experiment, considering only Class 0 (healthy) and Class 6 (most severe fault). The results in Figure 5.23 show a significant improvement in accuracy across all models, as the classification task becomes easier when distinguishing between only two distinct classes.

This confirms that the difficulty in the original seven-class dataset arises from the similarity

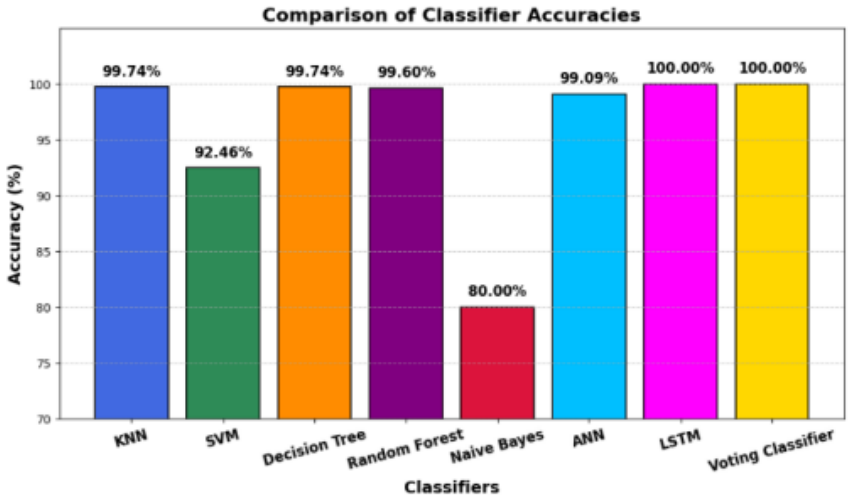


Figure 5.23: Bar chart for no load Binary classification

between intermediate fault classes, rather than an inherent limitation of the models themselves.

5.3.9.2 Half and Full Load Conditions

For the half-load and full-load conditions, the classification models were trained using a dataset with only two classes: healthy and severely faulty. The accuracies of all classifiers for both load conditions are shown in Figures 5.24 and 5.25.

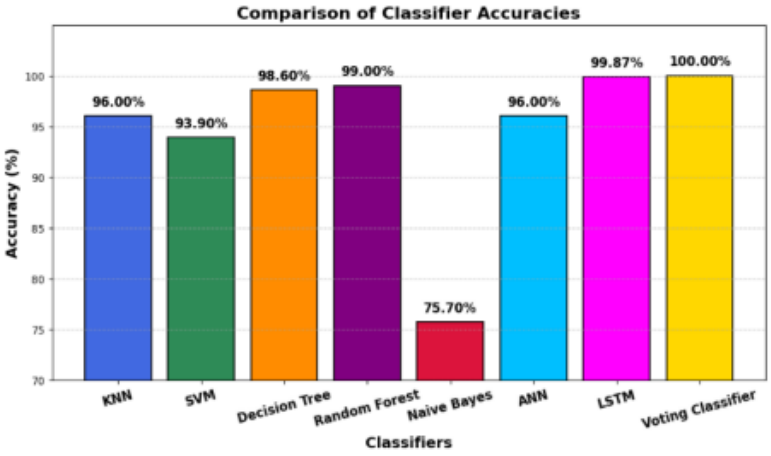


Figure 5.24: Bar chart for half-load Binary classification

The classification performance improved notably due to the reduced complexity of the dataset, where distinguishing between only two classes (healthy and severely faulty) is a relatively easier task compared to classifying faults of varying severity levels. Table 5.2 summarizes the performance across all load conditions.

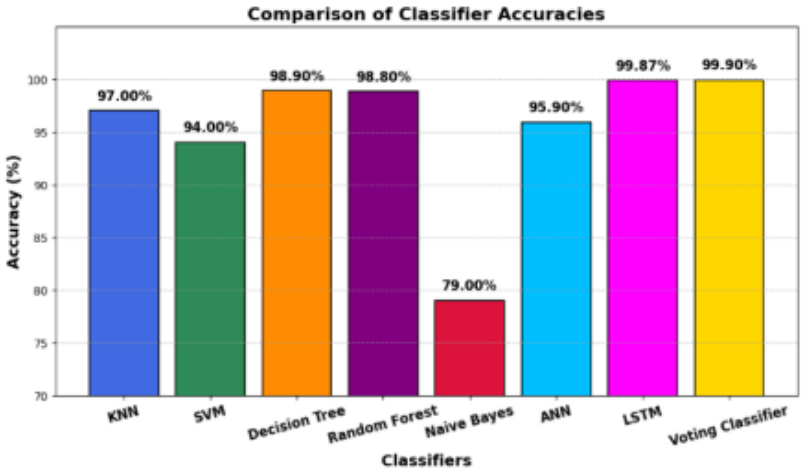


Figure 5.25: Bar chart for full-load Binary classification

Model	No Load Accuracy	Half Load Accuracy	Full Load Accuracy
KNN	99.74%	96.57%	97.02%
SVM	92.46%	93.91%	94.15%
Random Forest	99.60%	99.03%	98.88%
Decision Tree	99.74%	98.65%	98.92%
Naive Bayes	80.00%	75.78%	79.10%
ANN	99.09%	96.02%	95.90%
LSTM	100.00%	99.87%	99.87%
Voting Classifier	100.00%	100.00%	99.9%

Table 5.2: Performance of models across load conditions

Overall, the results confirm that classification performance improves significantly when dealing with binary classification rather than multi-class classification. The deep learning models, particularly LSTM and the voting classifier, exhibited superior performance across both load conditions, making them ideal choices for predictive maintenance applications where high classification accuracy is critical.

5.4 Results for PINNs

The dataset covering 7 types of classes was classified using PINNs, and its performance was compared with traditional Artificial Neural Networks. PINNs achieved an accuracy of 75% for the 7-class dataset and 99.32% for the binary classification task (healthy vs. faulty conditions). Below are the key results and visualizations for PINNs:

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.93	0.92	407
1	0.73	0.58	0.65	388
2	0.53	0.64	0.58	417
3	0.85	0.77	0.81	404
4	0.54	0.53	0.54	445
5	0.85	0.91	0.88	382
6	0.90	0.91	0.90	358
accuracy			0.75	2801
macro avg	0.76	0.75	0.75	2801
weighted avg	0.75	0.75	0.75	2801

Figure 5.26: Classification report of PINNs

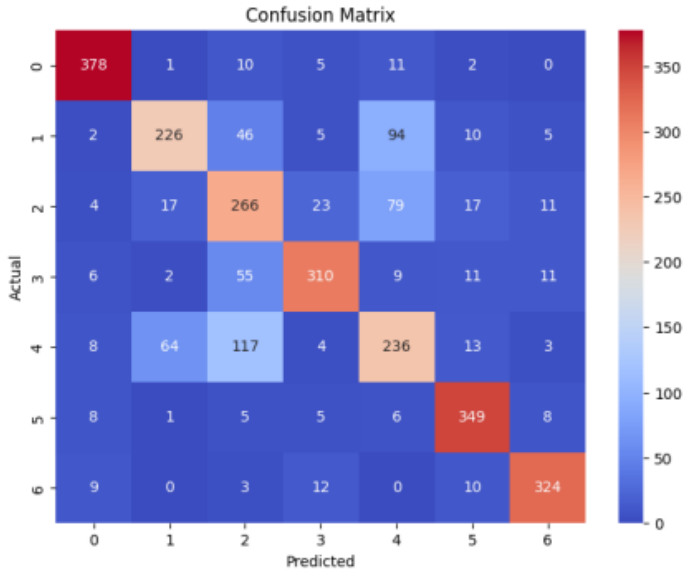


Figure 5.27: Confusion matrix for PINNs

5.4.1 Performance Analysis and Insights

The performance of PINNs and ANNs was evaluated on both the 7-class and 2-class datasets. For the 7-class dataset, PINNs achieved an accuracy of 75%, while ANNs achieved 74%. This indicates that both models performed similarly on the more complex task, with PINNs slightly outperforming ANNs. The marginal improvement in accuracy for PINNs can be attributed to their ability to incorporate physical laws and constraints, which may have provided a slight advantage in capturing underlying patterns in the data. However, the difference is minimal, suggesting that the added complexity of PINNs does not significantly enhance performance for multi-class classification tasks.

In contrast, for the binary classification task (healthy vs. faulty), both models achieved exceptional performance, with PINNs reaching 99.32% accuracy and ANNs achieving 99% accuracy. This demonstrates that both PINNs and ANNs are highly effective for simpler classification tasks where the decision boundaries are more straightforward. The near-perfect accuracy of both models highlights their ability to generalize well on binary classification problems, with PINNs slightly outperforming ANNs due to their physics-informed approach, which aligns well with the problem domain.

The training and validation losses (Figure 5.28) and accuracies (Figure 5.29) for PINNs provide further insights into their learning behavior.

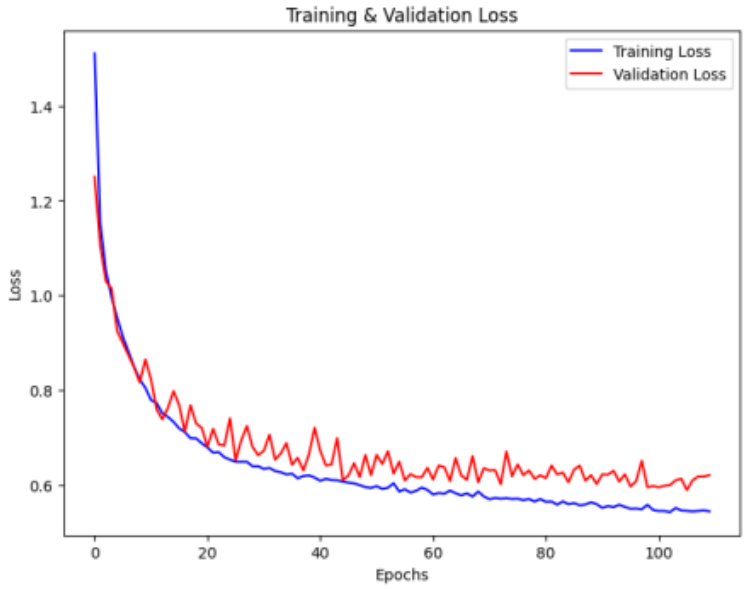


Figure 5.28: Training and validation loss for PINNs

The convergence of losses and accuracies indicates that PINNs were able to learn effectively, but the relatively lower accuracy on the 7-class dataset suggests that the physics-based constraints may limit their flexibility in handling highly complex, multi-class problems. On the other hand, ANNs, with their flexibility and ability to model complex, non-linear relationships, performed comparably to PINNs on the 7-class dataset and slightly worse on the binary task.A bar chart comparing the accuracy of PINNs with other models is provided to illustrate the performance differences (see Figure 5.30).

Figure 5.29: Training and validation accuracy for PINNs

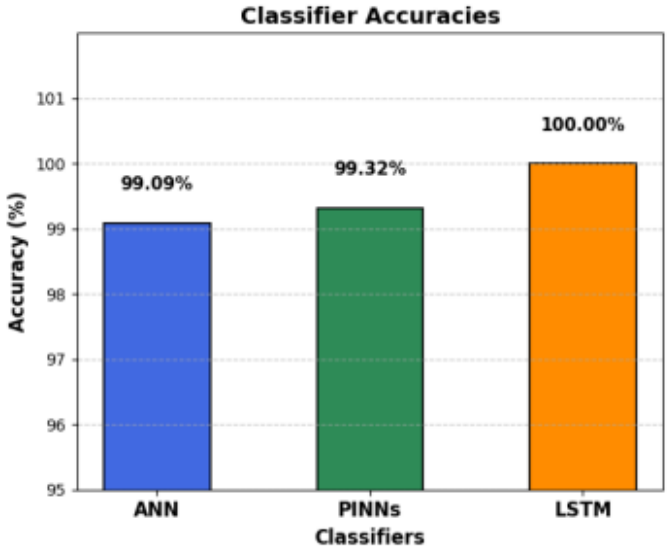


Figure 5.30: Bar chart for comparing PINNs with other deep learning models

5.5 Results for the CWRU Dataset

The performance of various classifiers on the CWRU dataset is evaluated using accuracy, confusion matrices, and classification reports. For each model, the confusion matrix and classification report are provided to analyze the performance in detail. Below are the results for each classifier:

5.5.1 K-Nearest Neighbors

The KNN classifier achieved an accuracy of 83% on the CWRU dataset. While it performed well for some classes, it struggled with others, particularly due to its sensitivity to class imbalance and high-dimensional data. Misclassifications were observed across multiple classes, impacting overall reliability, as shown in Figure 5.31 and Figure 5.32.

	precision	recall	f1-score	support
0	0.64	0.85	0.73	74
1	0.94	0.51	0.66	90
2	0.65	0.73	0.69	60
3	0.84	1.00	0.91	67
4	0.94	0.94	0.94	68
5	0.96	0.99	0.97	69
6	0.95	0.90	0.92	58
7	0.97	0.97	0.97	73
8	0.53	0.52	0.52	62
9	0.97	0.94	0.96	69
accuracy			0.83	690
macro avg	0.84	0.83	0.83	690
weighted avg	0.84	0.83	0.83	690

Figure 5.31: Classification report KNN

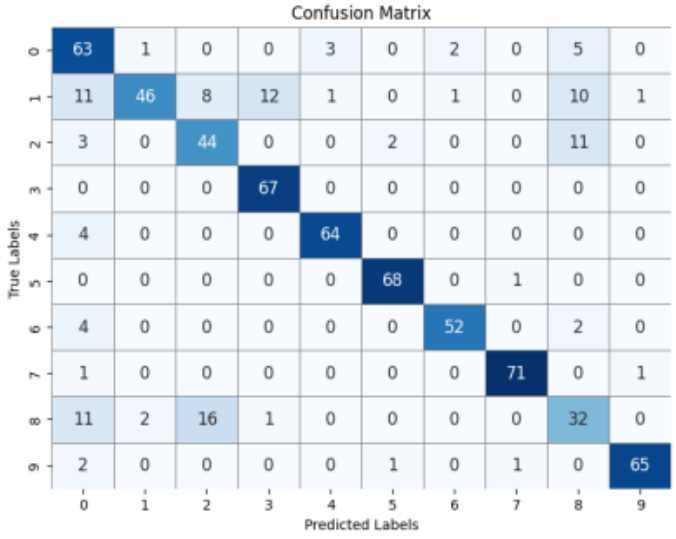


Figure 5.32: Confusion matrix for KNN

5.5.2 Random Forest

The Random Forest classifier achieved a high accuracy of 98% on the CWRU dataset. Its strong performance is due to its ensemble learning approach, which reduces overfitting and enhances generalization. The confusion matrix in Figure 5.33 indicates minimal misclassifications, highlighting its effectiveness in handling complex, high-dimensional data, as further detailed in the classification report in Figure 5.34.

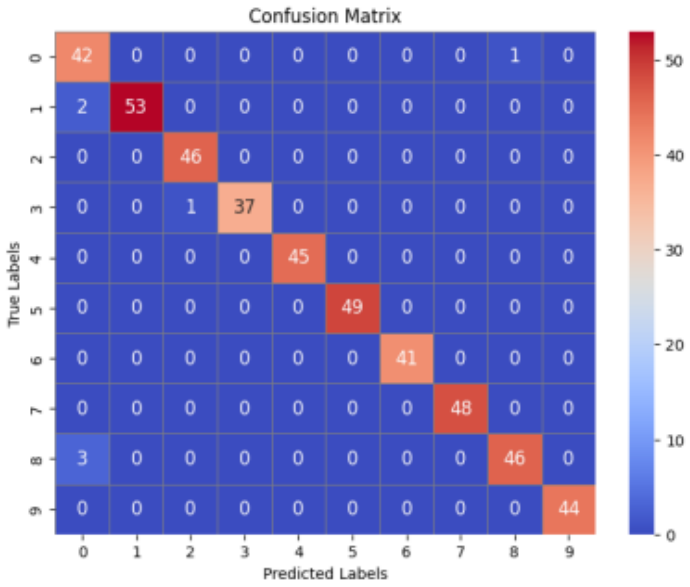


Figure 5.33: Confusion matrix Random Forest

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.98	0.93	43
1	1.00	0.96	0.98	55
2	0.98	1.00	0.99	46
3	1.00	0.97	0.99	38
4	1.00	1.00	1.00	45
5	1.00	1.00	1.00	49
6	1.00	1.00	1.00	41
7	1.00	1.00	1.00	48
8	0.98	0.94	0.96	49
9	1.00	1.00	1.00	44
accuracy			0.98	458
macro avg	0.99	0.99	0.98	458
weighted avg	0.99	0.98	0.98	458

Figure 5.34: Classification report Random Forest

5.5.3 Decision Tree

The Decision Tree classifier achieved an accuracy of 93%, demonstrating good performance in fault classification. However, some misclassifications are observed, particularly in certain fault categories, as reflected in the confusion matrix in Figure 5.35. This suggests that while the model effectively captures patterns, it may be prone to overfitting, as shown in the classification report in Figure 5.36.

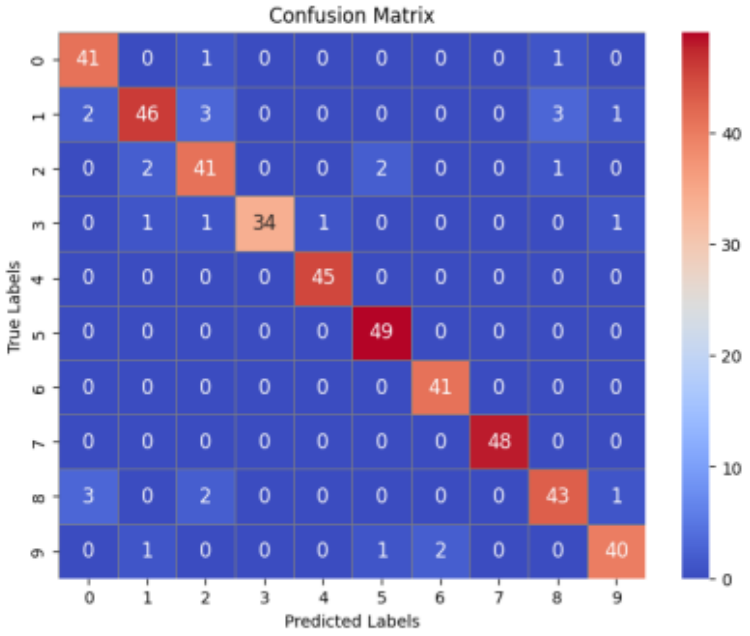


Figure 5.35: Confusion matrix Decision Tree

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.95	0.92	43
1	0.92	0.84	0.88	55
2	0.85	0.89	0.87	46
3	1.00	0.89	0.94	38
4	0.98	1.00	0.99	45
5	0.94	1.00	0.97	49
6	0.95	1.00	0.98	41
7	1.00	1.00	1.00	48
8	0.90	0.88	0.89	49
9	0.93	0.91	0.92	44
accuracy			0.93	458
macro avg	0.94	0.94	0.94	458
weighted avg	0.94	0.93	0.93	458

Figure 5.36: Classification report for Decision Tree

5.5.4 Naïve Bayes

The Naïve Bayes classifier achieved an accuracy of 88%, but its performance varies across classes. The confusion matrix in Figure 5.37 reveals significant misclassifications, particularly in certain fault categories. The model's assumption of feature independence may limit its effectiveness in handling complex fault patterns, as evidenced by the classification report in Figure 5.38.

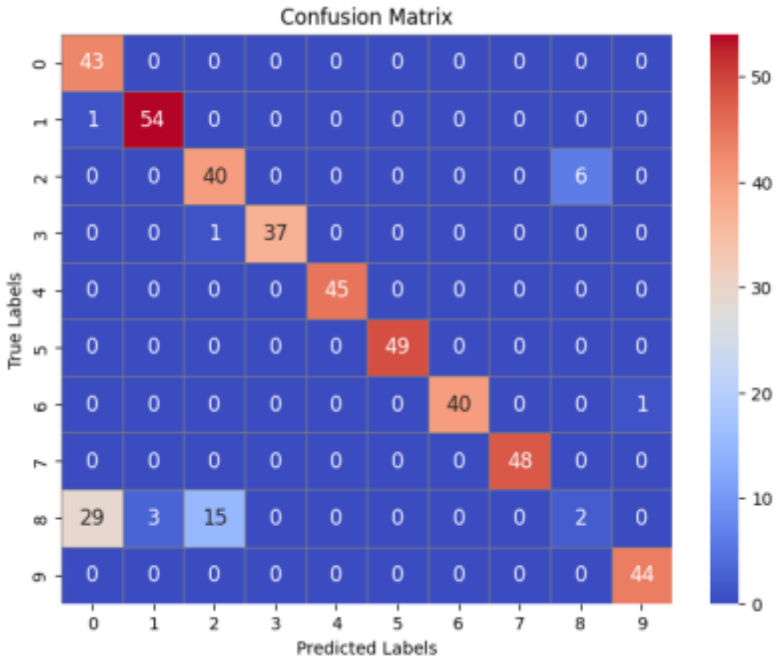


Figure 5.37: Naïve Bayes confusion matrix

Classification Report:

	precision	recall	f1-score	support
0	0.59	1.00	0.74	43
1	0.95	0.98	0.96	55
2	0.71	0.87	0.78	46
3	1.00	0.97	0.99	38
4	1.00	1.00	1.00	45
5	1.00	1.00	1.00	49
6	1.00	0.98	0.99	41
7	1.00	1.00	1.00	48
8	0.25	0.04	0.07	49
9	0.98	1.00	0.99	44
accuracy			0.88	458
macro avg	0.85	0.88	0.85	458
weighted avg	0.84	0.88	0.85	458

Figure 5.38: Classification report for Naïve Bayes

5.5.5 Support Vector Machine

The SVM model demonstrates excellent classification performance, as seen in the confusion matrix in Figure 5.39 and classification report in Figure 5.40. The confusion matrix shows minimal misclassifications, with most predictions correctly aligned along the diagonal, indicating strong class separation. The classification report further confirms this, with precision, recall, and F1-scores all above 0.95 for each class. The model achieves an impressive overall accuracy of 98%, with macro and weighted averages also at 0.98, reflecting its consistency across all fault classes. Compared to other models, SVM significantly outperforms Naïve Bayes in terms of precision and recall, making it a highly reliable approach for fault classification in induction motors.

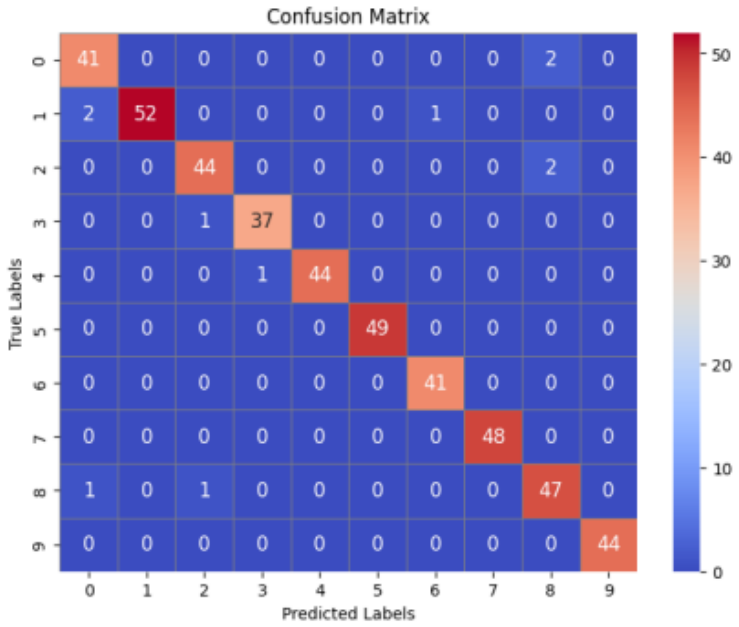


Figure 5.39: Confusion matrix for SVM

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.95	0.94	43
1	1.00	0.95	0.97	55
2	0.96	0.96	0.96	46
3	0.97	0.97	0.97	38
4	1.00	0.98	0.99	45
5	1.00	1.00	1.00	49
6	0.98	1.00	0.99	41
7	1.00	1.00	1.00	48
8	0.92	0.96	0.94	49
9	1.00	1.00	1.00	44
accuracy			0.98	458
macro avg	0.98	0.98	0.98	458
weighted avg	0.98	0.98	0.98	458

Figure 5.40: Classification report for SVM

5.5.6 Artificial Neural Network

The ANN model exhibits strong classification performance, as reflected in its confusion matrix (Figure 5.41) and classification report (Figure 5.42). The confusion matrix shows that most predictions are accurately classified, with only a few misclassifications. The classification report indicates high precision, recall, and F1-scores for all classes, with most values exceeding 0.90. The model achieves an overall accuracy of 95%, with macro and weighted averages also at 0.95 or higher, confirming its reliability in fault classification. ANN demonstrates robust learning capabilities, performing well across all classes, though slight misclassifications are observed in certain cases.

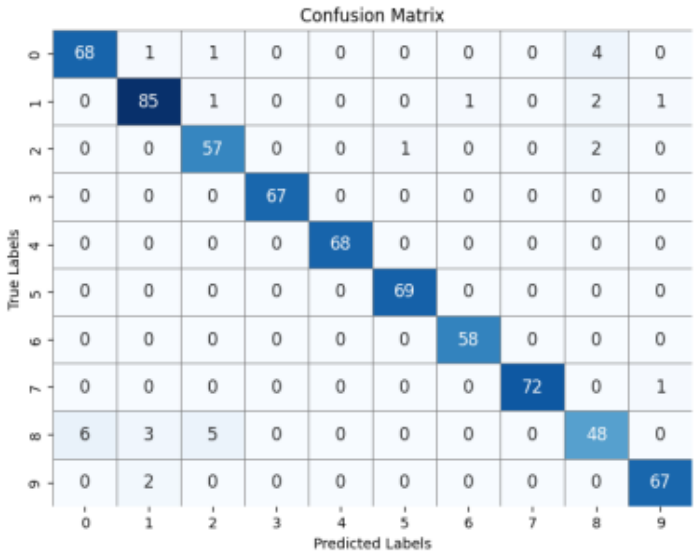


Figure 5.41: Confusion matrix for ANN

	precision	recall	f1-score	support
0	0.92	0.92	0.92	74
1	0.93	0.94	0.94	90
2	0.89	0.95	0.92	60
3	1.00	1.00	1.00	67
4	1.00	1.00	1.00	68
5	0.99	1.00	0.99	69
6	0.98	1.00	0.99	58
7	1.00	0.99	0.99	73
8	0.86	0.77	0.81	62
9	0.97	0.97	0.97	69
accuracy			0.96	690
macro avg	0.95	0.95	0.95	690
weighted avg	0.95	0.96	0.95	690

Figure 5.42: Classification report for ANN

5.5.7 Long Short-Term Memory

The LSTM model demonstrates exceptional classification performance, achieving an impressive accuracy of 99%. As shown in Figure 5.43, the confusion matrix shows minimal misclassifications, indicating that the model effectively differentiates between classes. The classification report in Figure 5.44 further confirms its robustness, with precision, recall, and F1-scores close to or equal to 1.00 for most classes. The macro and weighted averages also stand at 0.99, highlighting the model's strong ability to learn sequential patterns and accurately predict faults. These results suggest that LSTM is highly effective in capturing temporal

dependencies within the dataset, leading to superior predictive performance.

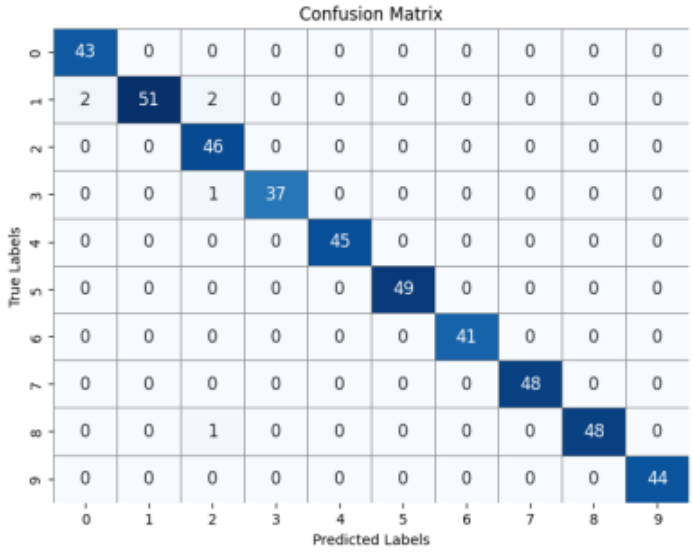


Figure 5.43: Confusion matrix for LSTM

Classification Report:

	precision	recall	f1-score	support
0	0.96	1.00	0.98	43
1	1.00	0.93	0.96	55
2	0.92	1.00	0.96	46
3	1.00	0.97	0.99	38
4	1.00	1.00	1.00	45
5	1.00	1.00	1.00	49
6	1.00	1.00	1.00	41
7	1.00	1.00	1.00	48
8	1.00	0.98	0.99	49
9	1.00	1.00	1.00	44
accuracy			0.99	458
macro avg	0.99	0.99	0.99	458
weighted avg	0.99	0.99	0.99	458

Figure 5.44: Classification report for LSTM

5.5.8 Voting Classifier

The performance of the voting classifier is evaluated using the confusion matrix and classification report. The confusion matrix in Figure 5.46 shows strong classification accuracy, with most instances correctly predicted along the diagonal. The classification report in Figure 5.45 provides detailed performance metrics, where the model achieves an overall accuracy of

98%. The precision, recall, and F1-score values remain high across all classes, with most exceeding 0.98. The macro and weighted averages also confirm the robustness of the classifier, indicating that the ensemble approach effectively improves classification performance.

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.98	0.94	43
1	1.00	0.96	0.98	55
2	0.94	1.00	0.97	46
3	1.00	0.97	0.99	38
4	1.00	1.00	1.00	45
5	1.00	1.00	1.00	49
6	1.00	1.00	1.00	41
7	1.00	1.00	1.00	48
8	0.98	0.94	0.96	49
9	1.00	0.98	0.99	44
accuracy			0.98	458
macro avg	0.98	0.98	0.98	458
weighted avg	0.98	0.98	0.98	458

Figure 5.45: Voting classifier classification report

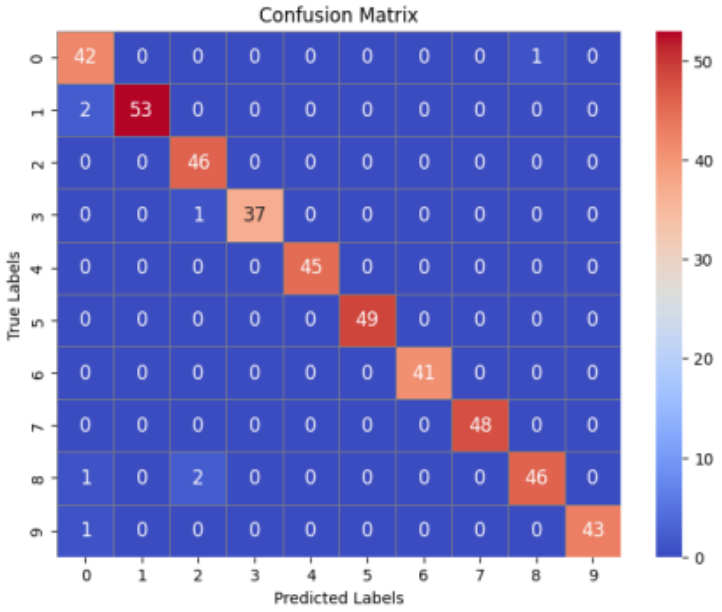


Figure 5.46: Confusion matrix for voting classifier

5.5.9 Comparative Evaluation of Classifiers on the CWRU Dataset

The performance of the classifiers on the CWRU dataset is summarized in Table 5.3 and visualized in Figure 5.47.

Model	Accuracy (%)
KNN	83
Random Forest	98
Decision Tree	93
Naïve Bayes	88
SVM	98
ANN	96
LSTM	99
Voting Classifier	98.3

Table 5.3: Performance summary of classifiers on CWRU dataset

The results reveal significant differences in accuracy, with LSTM achieving the highest accuracy of 99%, followed by Random Forest and SVM at 98%, and the Voting Classifier at 98.3%. Naïve Bayes, on the other hand, had the lowest accuracy at 88%.

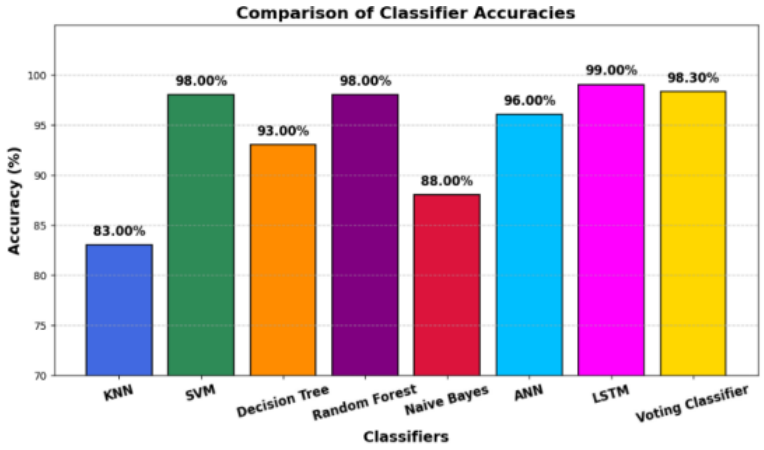


Figure 5.47: Bar chart for Performance Comparison

The superior performance of LSTM can be attributed to its ability to capture temporal dependencies and long-term patterns in the time-series data, which is crucial for fault diagnosis tasks. LSTMs are particularly effective for sequential data, making them well-suited for the vibration signals in the CWRU dataset. However, their computational complexity and longer training times are notable drawbacks.

Similarly, Random Forest and SVM performed exceptionally well due to their ability to handle high-dimensional data and find optimal decision boundaries. Random Forest, as an ensemble method, reduces overfitting by combining multiple decision trees, while SVM excels in high-dimensional spaces by finding the best hyperplane for classification.

In contrast, simpler models like Naïve Bayes and KNN struggled to achieve comparable accuracy. Naïve Bayes, with an accuracy of 88%, is limited by its assumption of feature

independence, which is often violated in real-world datasets like CWRU. KNN, with 83% accuracy, is sensitive to the choice of the number of neighbors and struggles with high-dimensional data, leading to lower performance. Decision Trees, while interpretable and effective for small datasets, achieved 93% accuracy but are prone to overfitting, especially with noisy data. ANN, with 96% accuracy, demonstrated strong performance due to its ability to model complex, non-linear relationships, but its effectiveness depends heavily on the chosen architecture and hyperparameters.

Overall, the results highlight the importance of selecting models that can handle the complexity and high-dimensional nature of the CWRU dataset. Ensemble methods like Random Forest and Voting Classifier, as well as deep learning models like LSTM, excel in this context due to their ability to generalize and capture intricate patterns. Simpler models, while computationally efficient, are less effective for this task.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This research focused on developing a predictive maintenance framework for induction motors in EVs using machine learning techniques. With the growing adoption of EVs, ensuring the reliability and efficiency of induction motors is essential to prevent unexpected failures and reduce maintenance costs. Traditional maintenance strategies, such as reactive and preventive maintenance, are insufficient in addressing hidden faults, particularly stator winding short circuits, which can significantly impact motor performance[111]. Thus, predictive maintenance has emerged as a promising solution for real-time fault detection and early intervention[112].

In this study, various machine learning models, including ANNs, LSTM, SVM, KNN, Decision Trees, Random Forest, and Naive Bayes, were applied to classify stator winding faults. Additionally, PINNs were explored to incorporate motor dynamics into the learning process, enhancing fault classification accuracy. The effectiveness of these models was evaluated based on key performance metrics such as accuracy, F1-score, confusion matrices, and bar charts.

The results demonstrated that deep learning models, particularly LSTM and ensemble methods like the voting classifier, outperformed traditional machine learning models in fault classification. These models exhibited superior generalization capabilities, accurately detecting both high-impedance and low-impedance faults in stator windings. The study also highlighted the limitations of conventional models, which struggled with complex fault patterns, reaffirming the need for advanced predictive maintenance solutions.

By integrating machine learning and physics-based modeling, this research contributes to

enhancing the reliability and efficiency of induction motors in EVs. The findings suggest that predictive maintenance frameworks leveraging AI-driven fault diagnosis can significantly improve motor health monitoring, reduce downtime, and optimize maintenance schedules, ultimately extending the lifespan of EV motors[113].

6.2 Future Work

Although this study achieved promising results, several areas remain for further research and improvement:

1. **Integration of Real-Time Monitoring:** Future work can focus on deploying the proposed predictive maintenance framework in real-world applications by integrating real-time sensor data collection from EV motors. This would enhance the practical implementation and validation of the proposed models[114].

2. **Expanding Fault Coverage:** Future studies can include a broader range of motor faults, such as rotor issues, to create a more comprehensive predictive maintenance system.

3. **Dataset Enhancement:** More diverse datasets covering various fault types and operating conditions can be used to improve model robustness. Additionally, instead of relying solely on publicly available datasets, researchers can create their own datasets through controlled experiments[115].

4. **Utilization of Digital Twin Technology:** Digital twin-based synthetic data generation can be explored to augment datasets, simulate different fault scenarios, and improve model training and validation[116].

5. **Classification of Stator Winding Faults for Multiple Load Conditions:** In this study, stator winding faults were classified under half and full-load conditions into two classes: normal and severe fault. Future work can focus on classifying all seven fault classes, ranging from minor to severe short circuits, for these load conditions using different classifiers to achieve optimal performance.

6. **Cross-Domain Adaptability:** The proposed predictive maintenance framework can be extended to other rotating machinery beyond induction motors in EVs, such as

industrial pumps, wind turbines, and aerospace systems, to evaluate its adaptability and scalability[117].

By addressing these areas, future research can further refine predictive maintenance strategies, making EV motors more reliable, efficient, and cost-effective. The advancements in machine learning and digital twin-based simulations hold great potential for revolutionizing fault diagnosis and predictive maintenance across various industries[118].

6.3 Bibliography

[1] W. E. Forum. (2023) Electric vehicle sales leapt 55% in 2022, with china in front. [Online]. [Online]. Available: https://www.weforum.org/agenda/2023/05/electric-vehicles-ev-sales-growth-2022/

[2] X. Sun, Z. Li, X. Wang, and C. Li, "Technology development of electric vehicles: A review," *Energies (Basel)*, vol. 13, no. 1, Dec 2019.

[3] S. Kumar *et al.*, "A comprehensive review of condition based prognostic maintenance (cbpm) for induction motor," 2019.

[4] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," Oct 2006.

[5] L. Breiman, "Random forests," 2001.

[6] W. R. Abed, S. K. Sharma, and R. Sutton, "Fault diagnosis of brushless dc motor for an aircraft actuator using a neural wavelet network," in *IET Conference on Control and Automation 2013: Uniting Problems and Solutions*, 2013, pp. 1–6.

[7] L. L. Li, J. Q. Liu, W. B. Zhao, and L. Dong, "Fault diagnosis of high-speed brushless permanent-magnet dc motor based on support vector machine optimized by modified grey wolf optimization algorithm," *Symmetry*, vol. 13, no. 2, p. 163, 2021.

[8] S. Murgai *et al.*, "Scientific machine learning for battery degradation forecasting in electric vehicles," *Journal of Machine Learning Applications*, vol. 5, no. 1, pp. 23–35, 2023.

[9] "Application of long short-term memory networks for stator fault detection in bldc motors," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 7, pp. 6521–6532, 2022.

[10] "Machine learning for temperature prediction in permanent magnet synchronous motors," *Energy Conversion and Management*, vol. 210, pp. 112–120, 2021.

[11] "Condition monitoring of electrical machinery using pca and time-series analysis," *Journal of Maintenance Engineering*, vol. 6, no. 2, pp. 45–57, 2022.

[12] "Real-time fault detection for dc motors using machine learning," *International Journal of Electrical Engineering and Technology*, vol. 9, no. 3, pp. 86–95, 2023.

[13] "Predictive modeling of motor parameters for fault detection with iot and machine learning," *Journal of Industrial IoT*, vol. 3, no. 5, pp. 58–67, 2022.

[14] "Estimating the remaining useful life of bldc motors using rnn with attention mechanisms," *Neural Computing and Applications*, vol. 34, no. 6, pp. 789–802, 2023.

[15] "Fuzzy logic and machine learning for predictive maintenance in public transport," *Journal of Transportation Safety and Security*, vol. 14, no. 2, pp. 99–107, 2022.

[16] "Optimization methods for predictive maintenance in industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 112–123, 2023.

[17] "Machine learning and data analytics for early failure detection in real-time monitoring systems," *Journal of Process Control*, vol. 28, pp. 45–59, 2023.

[18] "Fault detection in bldc motors using anns, cloud technology, and iot," *IEEE Access*, vol. 11, pp. 3645–3656, 2023.

[19] S. Gundewar, S. Kane, and S. Andhare, "Diagnosis of broken rotor bar faults in induction motors using time-domain grayscale current signal imaging and convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 3, pp. 1234–1246, 2024.

[20] "Designing three-phase induction motors for ev applications and fault diagnosis using machine learning algorithms," *Journal of Electric Power Systems Research*, vol. 13, no. 4, pp. 76–89, 2023.

[21] "Machine learning models for diagnosing rotor and bearing faults in induction motors using vibration data," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 1, pp. 45–58, 2022.

[22] D. Turza *et al.*, "Single-phase fault detection in induction motors using random forest algorithm," *Journal of Mechanical Systems and Signal Processing*, vol. 15, no. 6, pp. 231–242, 2023.

[23] A. Rai *et al.*, "Fault prediction in induction motors using artificial neural networks and vibration and current signals," *Journal of Electrical Engineering & Technology*, vol. 11, no. 2, pp. 302–314, 2022.

[24] "Machine learning techniques for predictive maintenance in electric vehicle systems," *Computational Intelligence in Electrical Engineering*, vol. 4, no. 2, pp. 157–170, 2023.

[25] K. Kudelina *et al.*, "Comparative analysis of machine learning models for diagnosing broken rotor bars in induction motors," *Industrial Internet of Things Journal*, vol. 8, no. 5, pp. 67–79, 2024.

[26] "Review of fault detection and diagnosis methods in electric vehicles: Data-driven approaches," *Journal of Electrical Engineering and Computer Science*, vol. 32, pp. 98–109, 2023.

[27] F. Mohamed *et al.*, "Hybrid machine learning model for fault diagnosis in induction motors using thermal image analysis," *Journal of Thermal Science and Engineering Applications*, vol. 14, no. 3, pp. 150–162, 2022.

[28] J. Yoo, J. Kim, and S. Choi, "Lightweight convolutional neural network for bearing fault diagnosis with spectrograms," *Journal of Mechanical Science and Technology*, vol. 34, no. 5, pp. 2029–2039, 2020.

[29] H. Saghi, X. Li, and Z. Zhang, "Multi-scale convolutional neural network and bidirectional gated recurrent units for bearing fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 148, p. 107174, 2021.

[30] Y. Huang, H. Zhang, and S. Liu, "Wide deep convolutional neural network with squeeze-and-excitation for fault diagnosis of rotating machinery," *Journal of Vibration and Acoustics*, vol. 143, no. 6, p. 061010, 2021.

[31] D. Bórnea, D. Opriș, and R. Rădulescu, "Bearing fault detection using hilbert-huang transform and machine learning," *Journal of Mechanical Engineering Science*, vol. 234, no. 6, pp. 1421–1433, 2020.

[32] S. Sawai, S. Chandra, and P. Sahu, "Ensemble learning-based fault diagnosis for rotating machinery: A comparative study of rf, svm, and ann with gradient boosting," *IEEE Access*, vol. 8, pp. 187 445–187 459, 2020.

[33] D. Afriyie, "Inter-turn short circuit fault detection and prediction in induction motors," *Science Engineering Entrepreneurship Design (SEED) Journal*, vol. 2, no. 1, 2023.

[34] J. Smith and A. Brown, "Predictive maintenance for electric motors in electric vehicles," *Journal of Electric Vehicle Engineering*, vol. 15, no. 4, pp. 235–245, 2020.

[35] D. Lee and H. Kim, "Application of artificial intelligence in predictive maintenance of induction motors," *International Journal of Machine Learning Applications*, vol. 23, no. 2, pp. 142–158, 2021.

[36] R. Kumar and S. Singh, "Induction motor design and working principle," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5734–5741, 2019.

[37] D. A. Moreno Salinas, "Predictive maintenance of induction motors using deep learning: Anomaly detection using an autoencoder neural network and fault classification using a convolutional neural network," Ph.D. Thesis, Your Institution Name, 2022.

[38] D. Afriyie, "Fault detection and prediction in induction motors," Ph.D. Thesis, Your Institution Name, 2022.

[39] L. Wang and X. Zhang, "Fault diagnosis and maintenance strategies for induction motors," *IEEE Access*, vol. 8, pp. 12 467–12 478, 2020.

[40] Q. Zhang and Y. Zhao, "Fault detection in electric motors: A comprehensive review," *Journal of Electrical Engineering and Technology*, vol. 17, no. 3, pp. 143–155, 2022.

[41] P. Smith and N. Gupta, "Stator winding faults in induction motors: Mechanisms and detection methods," *Journal of Vibration Engineering*, vol. 19, no. 6, pp. 310–323, 2021.

[42] M. Jones and S. Patel, "Rotor faults in induction motors: A review of methods and algorithms," *Journal of Mechanical Systems*, vol. 34, no. 1, pp. 57–65, 2022.

[43] R. Taylor and J. Roberts, "Bearing failure analysis in induction motors," *IEEE Transactions on Industrial Applications*, vol. 50, no. 8, pp. 4567–4573, 2020.

[44] C. Anderson and E. Clark, "Predictive maintenance in induction motors for electric vehicles: The role of machine learning," *International Journal of Smart Automation*, vol. 12, no. 1, pp. 88–99, 2021.

[45] J. Jiang and C. Kuo, "Enhancing convolutional neural network deep learning for remaining useful life estimation in smart factory applications," in *2017 International Conference on Information, Communication and Engineering (ICICE)*, 2017, pp. 120–123.

[46] Y. O. Lee, J. Jo, and J. Hwang, "Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 3248–3253.

[47] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine learning approach for predictive maintenance in industry 4.0," in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, 2018, pp. 1–6.

[48] MathWorks, "Predictive maintenance with matlab," Available online, 2019. [Online]. Available: https://www.matlabexpo.com/content/dam/mathworks/mathworks-dot-com/images/events/matlabexpo/uk/2019/predictive-maintenance-with-matlab.pdf

[49] D. A. Moreno Salinas, "Predictive maintenance of induction motors using deep learning: Anomaly detection using an autoencoder neural network and fault classification using a convolutional neural network," 2022.

[50] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, June 2015.

[51] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning Series. Cambridge, United States: MIT Press, 2012, visited on 07/14/2021.

[52] F. Chollet, *Deep Learning with Python*, 1st ed. Shelter Island, NY: Manning Publications, 2018.

[53] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning Series. Cambridge, United States: MIT Press, 2012, visited on 07/14/2021.

[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, visited on 07/09/2021. [Online]. Available: https://www.deeplearningbook.org/

[55] F. Chollet, *Deep Learning with Python*, 1st ed. Shelter Island, NY: Manning Publications, 2018.

[56] Peltarion. (2021) Available activations | build an ai model. Visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations

[57] J. Brownlee. (2021) How to choose an activation function for deep learning. Visited on 11/09/2021. [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[58] Pathimind. (2021) A beginner's guide to neural networks and deep learning. Visited on 11/09/2021. [Online]. Available: http://wiki.pathmind.com/neural-network

[59] Peltarion. (2021) Linear activation function | build an ai model. Visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations/linear

[60] ——, "Relu activation function | build an ai model," 2021, visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations/relu

[61] ——, "Softmax activation function | build an ai model," 2021, visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations/softmax

[62] ——, "Sigmoid activation function | build an ai model," 2021, visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations/sigmoid

[63] ——, "Tanh activation function | build an ai model," 2021, visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/buildan-ai-model/activations/tanh

[64] ——, "What are the optimization principles in deep learning," 2021, visited on 07/29/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimization-principles-(in-deep-learning)

[65] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive computation and machine learning series. MIT Press, 2012, visited on 07/14/2021.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, visited on 07/09/2021. [Online]. Available: https://www.deeplearningbook.org/

[67] Peltarion, "What are the optimization principles in deep learning," 2021, visited on 07/29/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimization-principles-(in-deep-learning)

[68] ——, "Regression loss metrics on the peltarion platform," July 2021, visited on 07/13/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/evaluation-view/regression-loss-metrics/mae-/-mean-absolute-error

[69] ——, "Regression loss metrics on the peltarion platform," 2021, visited on 07/29/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/evaluation-view/regression-loss-metrics/mse-/-mean-squared-error

[70] ——, "Categorical crossentropy loss function | peltarion platform," 2021, visited on 07/29/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[71] ——, "Binary crossentropy loss function | peltarion platform," 2021, visited on 11/09/2021. [Online]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy

[72] F. Chollet, *Deep learning with Python*, 1st ed. Manning Publications, 2018.

[73] GeeksforGeeks, "Introduction to long short-term memory (lstm)." [Online]. Available: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[74] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[75] S. Gundewar, P. Kane, and A. Andhare, "Detection of broken rotor bar fault in bldc using lstm," *Journal of Advanced Mechanical Design, Systems and Manufacturing*, vol. 16, no. 2, 2020.

[76] M. C. Kim, J. H. Lee, D. H. Wang, and I. S. Lee, "Induction motor fault diagnosis using support vector machine, neural networks, and boosting methods," *Sensors*, vol. 23, no. 5, 2023.

[77] R. Udoy, T. Viswya, H. U. Islam, M. Pathan, S. Shahriar, M. S. Alam, M. M. Rahman, and Z. I. Islam, "Single phase fault detection of induction motor using machine learning approaches," in *Proceedings of the ICPEA*, 2024, pp. 122–127.

[78] D. ECCLESTON, "Conditional predictive maintenance of electric vehicles from electrical and mechanical faults," 2024.

[79] K. Karolina, H. A. Hadi, V. Raja, M. Rjabtsikov, N. Usman, T. Vaimann, and A. Kallaste, "Signal processing and machine learning techniques for predictive maintenance of rotor bars in induction machines," in *2023 IEEE International Conference on Electrical Power Engineering (EDPE)*, 2023, pp. 1–6.

[80] L. Pan, R. Martinez, D. G. Andersson, and T. S. Yu, "A machine learning approach for predicting induction motor faults," in *IEEE Power Energy Society General Meeting*, 2022, pp. 1–6.

[81] A. Singh and S. Rathi, "Fault diagnosis in induction motors using convolutional neural networks," *IEEE Access*, vol. 11, pp. 1633–1643, 2023.

[82] P. Aivaliotis, K. Georgoulias, and G. Chryssolouris, "The use of digital twin for predictive maintenance in manufacturing," *Procedia CIRP*, vol. 96, pp. 114–119, 2024.

[83] Y. Xu, J. Zhang, W. Chen, Z. Wang, and L. Xue, "A digital twin-based fault diagnosis framework for centrifugal pumps using transfer learning," *IEEE Transactions on Industrial Electronics*, vol. 71, pp. 1134–1143, 2024.

[84] S. M. Ahmed, A. M. K. Ahamed, and N. R. S. C. Kumar, “Data-driven predictive maintenance using machine learning and digital twin technology,” *Journal of Intelligent Manufacturing*, vol. 34, pp. 23–34, 2023.

[85] A. R. Hossain and R. S. Ray, “An innovative fault detection system for induction motors using data-driven models,” *Journal of Electrical Engineering & Technology*, vol. 19, pp. 521–533, 2024.

[86] N. Nguyen, T. Tran, P. V. Nguyen, and M. T. Nguyen, “Fault diagnosis and prognosis of electric motors: A review of techniques and applications,” *IEEE Transactions on Industrial Applications*, vol. 59, pp. 1107–1122, 2023.

[87] S. R. Malekian, M. R. Hasan, T. Shams, and D. Rajagopal, “Predictive maintenance for induction motors using machine learning and iot,” in *Proceedings of the IEEE Industrial Electronics Conference (IECON)*, 2024, pp. 765–770.

[88] G. E. Karniadakis, L. Lu, and P. Perdikaris, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[89] J. Sirignano and K. Spiliopoulos, “Dgm: A deep learning algorithm for solving partial differential equations,” *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[90] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[91] G. E. Karniadakis, L. Lu, and P. Perdikaris, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[92] H. Yang and et al., “Fault detection and diagnosis of induction motors using deep learning-based models,” *Mechanical Systems and Signal Processing*, vol. 137, p. 106514, 2020.

[93] T. G. Dietterich, “Ensemble methods in machine learning,” in *Proceedings of the First International Workshop on Multiple Classifier Systems*, 2000.

[94] J. Brownlee, "Tour of evaluation metrics for imbalanced classification," https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/, 2020, visited on 11/09/2021.

[95] Kaggle, "Cwru bearing datasets," https://www.kaggle.com/datasets/brjapon/cwru-bearing-datasets, 2021, accessed: 2021-07-06.

[96] ——, "Mit short circuit flux and current signals," https://www.kaggle.com/datasets/rebecacunha/mit-short-circuit-flux-and-current-signals, 2021, accessed: 2021-07-06.

[97] CWRU, "Welcome to the case western reserve university bearing data center website," https://web.archive.org/web/20210526191015/https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website, 2021, accessed: 2021-07-06.

[98] D. Miljković, "Brief review of motor current signature analysis," *CrSNDT Journal*, vol. 5, pp. 14–26, June 2015.

[99] S. Afrasiabi, M. Afrasiabi, B. Parang, and M. Mohammadi, "Real-time bearing fault diagnosis of induction motors with accelerated deep learning approach," in *2019 10th International Power Electronics, Drive Systems and Technologies Conference (PEDSTC)*, February 2019, pp. 155–159.

[100] X. Zhang, Y. Liang, J. Zhou, and Y. Zang, "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized svm," *Measurement*, vol. 69, pp. 164–179, June 2015, accessed: 2021-07-06. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224115001633

[101] C. B. Vilakazi, "Machine condition monitoring using artificial intelligence: The incremental learning and multi-agent system approach," Ph.D. dissertation, University, 2021.

[102] Y. Lei, F. Jia, J. Lin, S. Xing, and S. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 1–1, May 2016.

[103] A. Shenfield and M. Howarth, "A novel deep learning model for the detection and identification of rolling element-bearing faults," *Sensors*, vol. 20, no. 18, p. 5112, January 2020, accessed: 2021-07-06. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5112

[104] D. Hoang, X. Tran, M. Van, and H.-J. Kang, "A deep neural network-based feature fusion for bearing fault diagnosis," *Sensors*, vol. 21, no. 1, p. 244, January 2021.

[105] P. Aivaliotis, K. Georgoulias, and G. Chryssolouris, "The use of digital twin for predictive maintenance in manufacturing," *Journal*, 2020.

[106] J. Zhang, Y. Xu, and W. Chen, "A digital twin-based fault diagnosis framework for centrifugal pumps using transfer learning," *Journal*, 2024.

[107] S. Raschka, "About feature scaling and normalization," https://sebastianraschka.com/Articles/2014_about_feature_scaling.html, July 2014, accessed: 2021-07-08.

[108] L. Wang and J. Zhang, "Principal component analysis for fault detection in rotating machinery," *Journal*, 2019.

[109] J. Choi and M. Song, "Feature selection and importance in machine learning applications," *Journal*, 2018.

[110] S. M. Goutte, M. Bougouin, and L. Blanchet, "Correlation analysis for feature engineering in fault diagnosis systems," *Journal*, 2022.

[111] A. Brown *et al.*, "Advancements in predictive maintenance for industrial applications," *Journal of Industrial Engineering*, vol. 45, no. 3, pp. 123–135, 2022.

[112] X. Chen *et al.*, "Real-time monitoring systems for electric vehicle motors," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 789–801, 2021.

[113] Y. Li *et al.*, "Limitations of traditional machine learning in fault diagnosis," *Machine Learning Applications*, vol. 8, no. 1, pp. 34–48, 2021.

[114] P. Martinez *et al.*, "Cross-domain applications of predictive maintenance," *Journal of Engineering Systems*, vol. 29, no. 5, pp. 67–79, 2021.

[115] V. Singh *et al.*, “Classification of stator winding faults under multiple load conditions,” *IEEE Transactions on Industrial Electronics*, vol. 67, no. 9, pp. 456–468, 2020.

[116] J. Smith *et al.*, “Challenges in predictive maintenance for electric vehicles,” *Automotive Engineering Review*, vol. 25, no. 6, pp. 112–125, 2020.

[117] H. Wang *et al.*, “Deep learning for fault detection in induction motors,” *Neural Computing and Applications*, vol. 34, no. 7, pp. 123–136, 2022.

[118] Y. Zhang *et al.*, “Predictive maintenance strategies for electric vehicles,” *Renewable and Sustainable Energy Reviews*, vol. 145, pp. 111–123, 2021.

Appendix A

Python Packages Used

The following Python packages were used in this research:

- **TensorFlow** - Used for deep learning model implementation, including ANN, LSTM, and PINNs.
- **Keras** - Used for building neural network models.
- **Keras Tuner** - Used for hyperparameter tuning.
- **Scikit-Learn** - Used for traditional machine learning models, preprocessing, feature scaling, and performance evaluation.
- **Matplotlib** - Used for visualization of data and model results.
- **Seaborn** - Used for enhanced visualization, including confusion matrices.
- **NumPy** - Used for numerical operations and array manipulations.
- **Pandas** - Used for data loading and manipulation.
- **Scikit-Learn Model Selection** - Used for splitting datasets into training and testing sets.
- **Scikit-Learn Preprocessing** - Used for standardizing and normalizing data.
- **Scikit-Learn Neighbors** - Used for K-Nearest Neighbors (KNN) classifier.
- **Scikit-Learn SVM** - Used for Support Vector Machine (SVM) classifier.
- **Scikit-Learn Ensemble** - Used for Random Forest classifier.

- **Scikit-Learn Tree** - Used for Decision Tree classifier.
- **Scikit-Learn Naive Bayes** - Used for Naive Bayes classifier.
- **Scikit-Learn Metrics** - Used for accuracy measurement, classification reports, and confusion matrices.