ACOUSTIC SIGNAL BASED FAULT DIAGNOSIS IN BLDC MOTORS USING AI

ABSTRACT

Electric Vehicles (EVs) are becoming increasingly popular due to their environmentally friendly nature and compatibility with renewable energy systems. A fundamental component in EVs is the Brushless DC (BLDC) motor, appreciated for its energy efficiency, compact design, and low maintenance requirements. Ensuring the dependable performance of these motors is essential for the safety and efficiency of EV systems.

Traditional approaches to fault detection in BLDC motors often rely on invasive sensor-based techniques. These methods involve direct contact with the motor using hardware sensors to monitor parameters like current, temperature, and vibration. Although effective, such systems can be costly, complex, and difficult to integrate into compact or sealed motor environments.

This project presents an innovative non-invasive fault detection method based on sound analysis using artificial intelligence (AI). Instead of physical sensors, motor sounds under different conditions—normal, bearing fault, and propeller fault—are captured and processed. Acoustic features such as Zero Crossing Rate (ZCR), Root Mean Square (RMS), Spectral Centroid, and Mel-Frequency Cepstral Coefficients (MFCC) are extracted to characterize the audio signals.

The extracted features are compiled into a dataset and analysed using various supervised machine learning algorithms. The performance of each model is evaluated using metrics like accuracy, precision, recall, and F1 score. A Voting Classifier combining the best-performing models shows superior performance in identifying faults compared to individual classifiers.

A small-scale demonstration is carried out to validate the proposed method. For large-scale implementation, this system can be extended by installing microphones near the motor housing in EVs and deploying real-time AI-based sound analysis modules. This enables continuous monitoring and predictive maintenance without interfering with motor operation, offering a scalable and cost-effective solution for the EV industry.

LIST OF CONTENTS

CHAPTER NO.	DESCRIPTION	PAGE NO.
I	**INTRODUCTION**	1
	1.1 PREAMBLE	1
	1.2 BLDC MOTOR IN EV APPLICATIONS	2
	1.3 NEED FOR FAULT DIAGNOSIS	3
	1.4 SOUND-BASED DIAGNOSTICS	4
	1.4.1 NON INVASIVE MONITORING	4
	1.4.2 REAL-TIME FAULT DETECTION	5
	1.5 ARTIFICIAL INTELLIGENCE IN FAULT ANALYSIS	6
	1.5.1 SUPERVISED LEARNING FOR FAULT CLASSIFICATION	6
	1.5.2 ACOUSTIC FEATURE EXTRACTION	7
	1.6 SCOPE OF THE PROJECT	8
II	**LITERATURE SURVEY**	11
III	**PROPOSED METHODOLOGY**	14
	3.1 PEOPOSED MACHINE LEARNING APPROACH FOR BLDC MOTOR FAULT CLASSIFICATION	14
	3.1 1 MOTOR CONDITION SIMLATION AND MONITORING	15
	3.1.2 HIGH-FIDELITY SOUND DATA ACQUISITION	16
	3.1.3 ADVANCED AUDIO PREPROCESSING AND FEATURE EXTRACTION	16
	3.1.4 STRUCTURED DATASET FORMATION FOR MODEL TRAINING	18
	3.1.5 AI ALGORITHMS USED FOR FAULT CLASSIFICATION	19
	3.1.5.1 LOGISTIC REGRESSION ALGORRITHM	20
	3.1.5.2 DECISION TREE ALGORITHM	22
	3.1.5.3 RAMDOM FOREST ALGORITHM	24
	3.1.5.4 SUPPORT VECTOR MACHINE ALGORITHM	27

3.1.5.5 K-NEAREST NEIGHBORS ALGORITHM 29
3.1.5.6 EXTREME GRADIENT BOOSTING ALGORITHM 31
3.1.5.7 VOTING CLASSIFIER ALGORITHM 33
3.1.6 RIGOROUS PERFORMANCE EVALUATION 35
3.1.7 OPTIMAL MODEL SELECTION 35
3.2 HARDWARE DETAILS 35
3.2.1 SWITCH MODE POWER SUPPLY (SMPS) 35
3.2.2 PROPELLER 37
3.2.3 BLDC MOTOR 38
3.2.4 ELECTRONIC SPEED CONTROLLER(ESC) 39
3.2.5 ARDUINO UNO 41
3.2.6 HARDWARE CONNECTION 42
3.2.7 SOFTWARE INTEGRATION 43
3.2.8 HARDWARE SETUP 44
IV **RESULT AND DISCUSSION** 46
4.1 PERFORMANCE EVALUATION OF AI ALGORITHMS FOR BLDC MOTOR FAULT CLASSIFICATION 46
4.1.1 LOGISTIC REGRESSION ANALYSIS 48
4.1.2 DECISION TREE ANALYSIS 50
4.1.3 RANDOM FOREST ANALYSIS 52
4.1.4 SUPPORT VECTOR MACHINE (SVM) ANALYSIS 54
4.1.5 K-NEAREST NEIGHBOR(KNN) ANALYSIS 57
4.1.6 XG BOOST (EXTREME GRADIENT BOOSTING) ANALYSIS 59
4.1.7 VOTING CLASSIFIER ANALYSIS 61
4.2 CLASSIFIER PERFORMANCE COMPARISON FOR BLDC MOTOR FAULT DETECTION 64
V **CONCLUSION** 66
5.1 CONCLUSION OF THE WORK 66
5.2 FUTURE WORK 66
REFERENCES 67
ANNEXURE 68

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	BLOCK DIAGRAM OF AI-BASED FAULT CLASSIFICATION SYSTEM FOR BLDC MOTOR	14
3.2	ACOUSTIC SIGNATURE CAPTURE FOR FAULT DIAGNOSIS IN BLDC MOTOR	16
3.3	DETAILED WORKFLOW FOR SOUND-BASED FAULT DIAGNOSIS AND CLASSIFICATION IN BLDC MOTOR	17
3.4	STRUCTURE PROCESS OF AUDIO DATA ACQISITION, PREPROCSSING, FEATURE EXTRACTION AND DATASET FORMATION FOR BLDC MOTOR FAULT CLASSIFICATION	19
3.5	LOGISTIC REGRESSION FOR BINARY AND MULTI-CLASS CLASSIFICATION	20
3.6	DECISION BOUNDARY VISUALIZATION OF LOGISTIC REGRESSION FOR BLDC MOTOR FAULT CLASSIFICATION	21
3.7	STRUCTURE OF A DECISION TREE MODEL REPRESENTING DECISION NODES AND LEAF NODES	22
3.8	DECISION TREE CLASSIFICATION OUTPUT FOR BLDC MOTOR HEALTHY MONITORING BASED ON MFCC1 AND SPECTRAL CENTROID FEATURES	23
3.9	CONCEPTUAL WORKFLOW OF RANDOM FOREST ALGORITHM USING ENSEMBLE OF DECISION TREE AND MAJORITY VOTING.	24
3.10(a)	DECISION TREE 1 VISUALIZATION FROM RANDOM FOREST CLASSIFIER FOR BLDC MOTOR FAULT CLASSIFICATION	25
3.10(b)	DECISION TREE 2 VISUALIZATION FROM RANDOM FOREST CLASSIFIER FOR BLDC MOTOR FAULT CLASSIFICATION	26
3.10(c)	DECISION TREE 3 VISUALIZATION FROM RANDOM FOREST CLASSIFIER FOR BLDC MOTOR FAULT CLASSIFICATION	26
3.11	ILLUSTRATION OF SUPPORT VECTOR MACHINE (SVM) HYPERPLANE SEPARTION FOR MULTICLASS CLASSIFICATION	27
3.12	CLASSIFICATION RESULTS USING SUPPORT VECTOR MACHINE (SVM) WITH RADIAL BASIS FUNCTION (RBF) KEMEL SHOWING DECISION BOUNDARIES AND SUPPORT VECTORS	28

3.13	ILLUSTRATION OF KNN ALGORITHM	29
3.14	KNN CLASSIFICATION DECISION BOUNDARIES FOR FAULT DIAGNOSIS.	30
3.15	ILLUSTRATION OF XG BOOST MODEL WITH ENSEMBLE OF WEAK LEARNERS	31
3.16	DECISION BOUNDARY VISUALIZATION OF XG BOOST CLASSIFIER FAULT DIAGNOSIS	32
3.17	ARCHITECTURE OF VOTING CLASSIFIER COMBINING XG BOOST, RANDOM FOREST AND LOGISTIC REGRESSION	33
3.18	DECISION BOUNDARY VISUALIZATION OF VOTING CLASSIFIER COMBINING MULTIPLE MODELS	34
3.19	SMPS FOR STABLE POWER DELIVERY	36
3.20	PROPELLER FOR DYNAMIC LOAD SIMULATION ON BLDC MOTOR	37
3.21	BLDC MOTOR (A2212 1400KV) FOR FAULT ANALYSIS	38
3.22	ESC FOR MOTOR SPEED REGULATION	40
3.23	ARDUINO UNO FOR PWM SIGNAL GENERATION	41
3.24(a)	HARDWARE COMPONENTS FOR THE SOUND BASED BLDC MOTOR FAULT DETECTION SYSTEM	44
3.24(b)	EXPERIMENTAL SETUP FOT AUDIO DATA ACQUISITION	45
4.1(a)	CONFUSION MATRICES SHOWING CLASS LABEL PREDICTION	46
4.1(b)	CONFUSION MATRICES SHOWING PERFORMANCE METRICS FOR A MULTI-CLASS MODEL	47
4.2	FORMULAS FOR ACCURACY, PRECISION, RECALL, AND F1-SCORE USED FOR EVALUATING CLASSIFICATION PERFORMANCE.	47
4.3	CONFUSION MATRIX OF LOGISTIC REGRESSION DISPLAYING MODEL PERMANCE	48
4.4	PERFORMANCE EVALUATION METRICS FOR LOGISITIC REGRESSION MODEL	49
4.5	CONFUSION MATRIX OF DECISION TREE DISPLAYING MODEL PERFORMANCE	50

4.6	PERFORMANCE EVALUATION METRICS FOR DECISION TREE MODEL	51
4.7	CONFUSION MATRIX OF RANDOM FOREST DISPLAYING MODEL PERFORMANCE	53
4.8	PERFORMANCE EVALUATION METRICS FOR FOR RANDOM FOREST MODEL	53
4.9	CONFUSION MATRIX OF SUPPORT VECTOR MACHINE(SVM) DISPLAYING MODEL PERFORMANCE	55
4.10	PERFORMANCE EVALUATION METRICS FOR SUPPORT VECTOR MACHINE(SVM) MODEL	56
4.11	CONFUSION MATRIX OF K- NEAREST NEIGHBORS (KNN) DISPLAYING MODEL PERFORMANCE	57
4.12	PERFORMANCE EVALUATION METRICS FOR K- NEAREST NEIGHBORS (KNN)MODEL	58
4.13	CONFUSION MATRIX OF XG BOOST DISPLAYING MODEL PERFORMANCE	60
4.14	PERFORMANCE EVALUATION METRICS FOR XG BOOST MODEL	61
4.15	CONFUSION MATRIX OF VOTING CLASSIFIER DISPLAYING MODEL PERFORMANCE	62
4.16	PERFORMANCE EVALUATION METRICS FOR VOTING CLASSIFIER MODEL	63
4.17	OVERALL PERFORMANCE COMPARISION OF CLASSIFIERS USING ACCURACY, PRECISION, RECALL AND F1-SCORE	64

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	BLDC MOTOR SPECIFICATIONS	38
3.2	ESC SPECIFICATIONS	40
3.3	ARDUINO UNO SPECIFICATIONS	42

LIST OF ABBREVIATIONS

BLDC	BRUSHLESS DIRECT CURRENT
AI	ARTIFICIAL INTELLIGENCE
ZCR	ZERO CROSSING RATE
RMS	ROOT MEAN SQUARE
MFCC	MEL-FREQUENCY CEPSTRAL COEFFICIENTS
LR	LINEAR REGRESSION
DT	DECISION TREE
RF	RANDOM FOREST
SVM	SUPPORT VECTOR MACHINE
KNN	K-NEAREST NEIGHBORS
XG BOOST	EXTREME GRADIENT BOOSTING
VC	VOTING CLASSIFIER
SMPS	SWITCH MODE POWER SUPPLY
ESC	ELECTRONIC SPEED CONTROLLER

CHAPTER 1

INTRODUCTION

1.1 PREAMBLE

Electric vehicles (EVs) leading the accelerator in sustainable transportation, offering a cleaner alternative to conventional internal combustion engine (ICE) vehicles. The electric motor is the heart of most EVs, with Brushless DC (BLDC) motors being often used because of their efficiency, sturdiness, and lower maintenance needs. BLDC motors, in contrast to conventional motors, do not require brushes or commutators. This results in reduction in frictional losses, lower electrical noise, and a longer operational lifespan. The benefits of BLDC motors make them particularly well-suited for use in electric vehicles, where longevity, energy efficiency, and low maintenance are essential. However, in spite of their many advantages, BLDC motors can still experience faults. Common problems include bearing wear, rotor misalignment, winding defects, and sensor failures. If these fault left undetected, they can have a profound effect on motor performance, resulting in declining efficiency, unanticipated anomalies, and in some cases total system failure. With the increase in EV adoption, the necessity for effective and efficient fault detection systems is elevated. Such systems are essential for guaranteeing the reliability of motors, enhancing vehicle safety, and cutting maintenance expenses. Conventional methods for detecting faults, like manual inspections or sensor-based approaches, frequently come with limitations regarding cost, accuracy, and feasibility. As a reaction to this, technological real-time diagnostic systems are being developed for the continuous monitoring of motor health. These systems provide major benefits, such as early identification of faults, predictive upkeep, and reduced downtime, which guarantees that the EV functions at its optimal. Sound-based diagnostics are a promising method for detecting faults. This technique provides a solution for real-time monitoring that is easy to deploy and cost-effective by analysing the acoustic emissions produced by the motor while it operates. Motor vibrations create sounds that hold valuable information; analysing these sounds can reveal faults like bearing damage and Propeller Damage. This approach is especially appropriate for EVs because it necessitates only a small amount of hardware and circumvents the difficulties associated with intrusive sensors. Moreover, incorporating artificial intelligence (AI) boosts the efficiency of sound-based diagnostics. Sound data allows for the classification of motor conditions by AI algorithms, particularly supervised learning models, which allow to

encourages concise fault detection without the need for human involvement. AI can recognize different fault types and forecast their severity by training models with characteristics like zero-crossing rate (ZCR), root mean square (RMS), spectral centroid, and Mel-frequency cepstral coefficients (MFCC). This makes it possible to carry out maintenance in good time and reduces the risk of motor failure. To sum up, although BLDC motors provide significant benefits for electric vehicles, proper maintenance is essential for guaranteeing their long-term performance and reliability. Advanced fault detection systems, especially those that utilize sound-based diagnostics and AI, mark a considerable advancement in the management of EV motors. By providing a cost-effective method of preventative maintenance and enabling real-time, non-invasive monitoring, these technologies improve the general sustainability and safety of electric vehicles.

1.2 BLDC MOTORS IN EV APPLICATIONS

Brushless DC (BLDC) motors have become the cornerstone of electric vehicle (EV) technology, revolutionizing the way vehicles operate with their exceptional efficiency, whisper-quiet performance, and long-lasting durability. Unlike conventional motors, BLDC motors are designed without brushes or commutators, effectively eliminating the mechanical friction that is typically a major source of energy loss in traditional systems. This innovative design leads to significantly improved energy conversion, ensuring that every watt of power generated is utilized to its maximum potential, resulting in a remarkably efficient operation. The absence of brushes not only boosts efficiency but also dramatically reduces the wear and tear on motor components. This translates into a substantial reduction in maintenance requirements, making BLDC motors a reliable and cost-effective choice for long-term use. With fewer moving parts, the motor's operational lifespan is extended, offering EV owners a much-needed peace of mind with fewer repairs and replacements over time.

BLDC motors are excellent for the demanding requirements of electric vehicles because they offer precise control over motor speed and torque. BLDC motors offer exceptional performance and dependability whether they are powering auxiliary systems like air conditioning or power steering, driving the steering mechanism for better handling, or powering the wheels for smooth acceleration. This fine control allows EV to perform optimally across a wide range of conditions and driving scenarios, further underscoring the motor's versatility and importance in modern transportation. In a world where energy conservation is

no longer just a goal but a necessity, BLDC motors stand at the forefront, ensuring that every ounce of energy is used efficiently. Their inherent efficiency, paired with their impressive durability, makes them the ideal choice for EVs, where reducing energy consumption and extending the vehicle's lifespan are critical objectives.

The operational reliability of BLDC motors is more important than ever as the transition to electric mobility picks up speed, highlighting the critical need for reliable defect detection and real-time monitoring systems. Even the most advanced motor may break down without such systems, therefore creating efficient diagnostic tools is crucial to the development of really sustainable, dependable, and efficient electric vehicles.

1.3 NEED FOR FAULT DIAGNOSIS

While BLDC motors offer significant advantages, they are not immune to operational faults that can affect their performance. These advanced motors, while revolutionary in design, are still susceptible to a range of operational faults that can have severe consequences on both motor performance and the overall vehicle safety.

Common faults in BLDC motors, such as bearing wear, rotor misalignment, winding defects, and sensor malfunctions, can result in a cascade of problems ranging from minor inefficiencies to catastrophic system failures. Even the slightest deviation in motor health can have far-reaching effects, leading to diminished motor efficiency, unexpected vehicle downtime, and, in extreme cases, complete powertrain failure that can leave a vehicle inoperable.

In an electric vehicle, where the entire propulsion system relies on the smooth and reliable operation of the BLDC motor, the impact of these faults can be devastating. Any of these issues can escalate quickly, causing not only a decline in motor performance but also a potential safety hazard for the vehicle occupants and those on the road. The longer these faults remain undetected, the more severe the damage becomes, leading to costly repairs, prolonged vehicle downtime, and in some cases, total system failure.

Traditional methods of fault detection, such as manual inspection or basic sensor-based techniques, are outdated and inadequate for modern EVs. These methods often come with critical limitations in terms of feasibility, accuracy, and cost-effectiveness. Manual inspections, while occasionally helpful, are labour - intensive, time-consuming, and prone to human error.

Similarly, traditional sensor-based monitoring systems, though useful, often face challenges in terms of precision, scalability, and integration within the compact, enclosed motor systems of EVs.

Moreover, the very design of these systems, where motors are often housed in tight compartments with limited space for sensors, makes it increasingly difficult to implement intrusive, physical monitoring systems effectively without compromising the motor's performance or adding excessive cost.

This underscores the critical need for an advanced, next-generation fault diagnosis system that can deliver real-time, non-intrusive, and cost-effective monitoring for BLDC motors in EVs. Such a system would revolutionize the way we approach motor health diagnostics, offering the ability to detect faults early, predict potential failures before they occur, and mitigate risks before they escalate into catastrophic breakdowns. By employing cutting-edge technology such as AI-powered diagnostic tools, acoustic analysis, or vibration monitoring, these systems could continuously track motor health without the need for invasive or complex setups.

The importance of such an advanced system cannot be overstated. Not only would it significantly reduce maintenance costs by identifying issues early, but it would also prevent unexpected breakdowns, minimize vehicle downtime, and enhance the safety and reliability of EVs. Furthermore, it would pave the way for predictive maintenance strategies, allowing for scheduled repairs and replacements, thereby extending the life of both the motor and the vehicle itself.

As electric mobility becomes more mainstream and the reliance on EVs grows, ensuring that BLDC motors operate without failure is not just desirable but essential for the future of the automotive industry.

1.4 SOUND-BASED DIAGNOSTICS

1.4.1 NON-INVASIVE MONITORING

An innovative method for detecting faults in BLDC motors is sound-based diagnostics. This approach utilizes the acoustic signals emitted by the motor during operation, capturing the subtle variations that may indicate underlying issues. These sounds, originating from vibrations within the motor's components, contain essential information regarding the motor's operational health. By using microphones and

specialized acoustic sensors, these sounds can be recorded and analysed for any irregularities.

This technique offers several key advantages, especially in the context of electric vehicles (EVs). Sound-based diagnostics provide a non-invasive, non-destructive method of monitoring the motor's condition. There is no need for intrusive sensors or modifications to the motor itself, which makes the implementation easier and cost-effective. Additionally, it avoids the complexities and potential downsides associated with traditional wired sensors.

The ease of deployment and maintenance further boosts its applicability for real-time monitoring. In EV applications, where space and weight are crucial considerations, sound-based diagnostics stand out as a highly efficient solution.

By continuously monitoring the motor’s acoustic profile, this method offers a means of identifying faults early, which is vital for the long-term performance and safety of the vehicle. Moreover, the use of sound enables remote diagnostics, reducing the need for physical inspection and thus enhancing the overall convenience of the monitoring system.

1.4.1 REAL-TIME FAULT DETECTION

The most significant benefit of sound-based diagnostics is its capability for real-time fault detection. Unlike traditional monitoring systems, which typically provide data on a periodic or post-failure basis, sound-based systems offer continuous, live tracking of the motor's condition. This constant monitoring allows for the immediate identification of irregularities as they arise, ensuring that potential issues are detected before they escalate into severe problems.

This real-time detection is especially valuable in electric vehicles (EVs), where motor performance and reliability are critical to the overall functioning of the vehicle. By identifying faults like wear or misalignment at an early stage, the system can trigger proactive maintenance actions, thereby reducing the likelihood of unexpected failures. Such timely interventions not only enhance the motor’s operational lifespan but also contribute to improved vehicle safety and performance.

Furthermore, the ability to analyse sound data in real-time enables the implementation of predictive maintenance strategies. By identifying subtle signs of deterioration before they result in significant damage, repairs or part replacements can be scheduled proactively, minimizing costly breakdowns and reducing the likelihood of motor failure. This predictive approach ultimately leads to enhanced reliability, cost-efficiency, and overall vehicle performance.

1.5 ARTIFICIAL INTELLIGENCE IN FAULT ANALYSIS

Artificial Intelligence (AI) plays a crucial role in fault analysis for BLDC motors used in electric vehicles (EVs). AI leverages data from motor operation, such as sounds, vibrations, and electrical signals, to detect faults with high accuracy. Through machine learning algorithms, AI can identify patterns in the data, enabling automatic detection and classification of faults. These systems continuously improve as they are exposed to more data, enhancing their ability to detect new and emerging fault types. AI enables real-time monitoring, allowing for early detection of faults and proactive maintenance, which reduces vehicle downtime and extends motor lifespan.

Additionally, AI can predict the progression of faults, helping to plan maintenance activities before major failures occur. By integrating AI with other vehicle systems, a comprehensive and streamlined approach to monitoring EV health can be established. This integration results in reduced maintenance costs, optimized performance, and enhanced safety. Overall, AI-driven fault analysis supports efficient, long-term reliability for EV motors.

1.5.1 SUPERVISED LEARNING FOR FAULT CLASSIFICATION

Supervised learning algorithms are highly effective for classifying different fault conditions in BLDC motors. AI can learn to differentiate between normal and faulty states by using machine learning models trained on a dataset that contains labelled instances of different types of faults and healthy motor sounds. This method removes the need for human involvement by enabling automatic fault identification and classification. Machine learning models analyse features extracted from motor sound signals to predict fault types and assess their severity with high accuracy. As new data is gathered, supervised learning methods enable the system to adapt to new fault patterns, continuously improving its performance. The ability to learn and refine its fault detection capabilities over time enhances the reliability and efficiency of fault diagnosis systems in electric vehicles.

1.5.2 ACOUSTIC FEATURE EXTRACTION

The accuracy and reliability of AI-based fault diagnosis systems for Brushless DC (BLDC) motors are profoundly influenced by the extraction and utilization of key acoustic features. In sound-based diagnostics, the motor's operational sounds are not just noise but carry critical information about its internal health.

The process of extracting these acoustic features is vital, as it enables the AI to interpret subtle shifts in motor performance that would otherwise be undetectable. These features serve as the foundation for understanding the motor's behaviour and identifying potential faults at an early stage, before they evolve into serious issues.

The key acoustic features used in fault detection include:

- **ZERO CROSSING RATE (ZCR)**

  This measure captures the rate at which the motor's sound signal crosses zero amplitude, which is an essential indicator of vibrations, irregularities, or mechanical disruptions within the motor.

$$ZCR=\frac{1}{2(N-1)}\sum_{n=0}^{N-2}|\mathrm{sgn}(x[n])-\mathrm{sgn}(x[n+1])| \quad N=\text{number of samples}$$

  A high ZCR typically suggests abnormal sounds such as bearing wear or rotor misalignment, directly signalling a fault.

- **ROOT MEAN SQUARE (RMS)**

  RMS quantifies the energy of the signal, providing an essential measure of the overall strength of the sound.

  In the context of motor diagnostics, an increase in RMS often indicates an escalation in motor stress or malfunction, such as excessive friction, electrical faults, or power delivery issues.

$$RMS=\sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x[n])^2} \qquad N=\text{number of samples in the frame}$$

- **SPECTRAL CENTROID**

Serving as the "Center of mass" of the frequency spectrum, spectral centroid is a crucial feature for understanding changes in the tonal quality of the sound emitted by the motor.

$$\text{Spectral Centroid} = \frac{\sum_{k=0}^{N-1} \text{fk}.|X[k]|}{\sum_{k=0}^{N-1} |\text{x[k]}|}$$

N=number of frequency bins (from FFT)

Fk=frequency at bin k,

X[k] = magnitude of FFT at bin k

Shifts in the spectral centroid indicate alterations in the frequency distribution, which can be linked to specific motor issues.

- **MEL-FREQUENCY CEPSTRAL COEFFICIENTS (MFCC)**

MFCCs are powerful tools for capturing detailed, complicated frequency characteristics. They are widely used in speech and audio processing and enable the fault detection system to create a detailed map of the motor's sound patterns.

$$\text{Cm} = \sum_{k=1}^{K} \log(Sk) \,.\cos\left[\pi \boldsymbol{m}. + \frac{k-0.5}{k}\right]$$

Sk = energy in the k-th Mel filter
K = number of Mel filters,
M = MFCC index (1 to 13)

By continuously monitoring these features, the AI can identify even the most subtle signs of impending failure, enabling predictive maintenance that not only improves motor longevity but also ensures that EVs remain reliable and efficient. The effectiveness of feature extraction ultimately empowers AI systems to perform with exceptional precision, transforming the way fault diagnosis is approached in electric vehicle systems.

1.6 SCOPE OF THE PROJECT

The primary aim of this project is to develop a cutting-edge, sound-based diagnostic system tailored to the detection of faults in Brushless DC (BLDC) motors, which are integral components of electric vehicles (EVs). The project envisions a robust, real-time monitoring solution that leverages acoustic signals emitted by the motor to provide precise, non-invasive fault detection.

The scope of this project will unfold in the following key stages:

- **DATA COLLECTION**

  The project will begin by capturing high-quality sound data from BLDC motors operating under various fault conditions. These recordings will span a wide array of motor health scenarios, ensuring that the dataset is comprehensive and diverse. This step is fundamental for building a reliable model capable of handling real-world operational variances in EVs. Through careful data collection, the system will gather detailed insights into the motor's acoustic signature under different stress levels, wear and tear conditions, and fault types.

- **FEATURE EXTRACTION**

  Once the sound data is collected, the next step is to extract critical acoustic features, such as Zero Crossing Rate (ZCR), Root Mean Square (RMS), Spectral Centroid, and Mel-Frequency Cepstral Coefficients (MFCC). These features are essential in capturing the fine-grained characteristics of the motor's operation.

  Each feature provides unique insights into the motor’s performance and health, enabling the system to discern between normal operation and fault conditions. The next phase ensures that the most significant elements of the audio data are preserved for analysis by utilizing advanced signal processing techniques, establishing the foundation for efficient machine learning models.

- **FAULT CLASSIFICATION**

  With the features extracted, the next phase of the project involves training machine learning models on the dataset to classify the motor’s condition. This will allow the system to not only detect faults but also categorize their type, such as bearing wear, rotor misalignment, winding defects, or sensor malfunctions. The application of advanced algorithms will ensure that the classification is not only accurate but also adaptive, improving with more data and evolving fault conditions. Through supervised learning, the system will learn to differentiate between normal and faulty states, offering a high degree of precision in its fault detection capabilities.

- **REAL TIME MONITORING**

  The ultimate objective of this project is to design a real-time fault detection system capable of continuously monitoring the motor's condition during operation. The system will be engineered to alert maintenance personnel instantly upon the detection of a fault, providing actionable insights that enable proactive intervention. This real-time capability ensures that potential issues are identified early, before they escalate into costly or catastrophic failures, thereby enhancing the overall safety and reliability of electric vehicles. Additionally, this real-time monitoring will facilitate predictive maintenance strategies, reducing the need for costly, time-consuming inspections and unscheduled repairs.

As a result, this system will significantly reduce maintenance costs, improve motor reliability, and contribute to the long-term sustainability of the electric vehicle ecosystem. Through the fusion of advanced technology, data-driven insights, and real-time monitoring, this project promises to transform the landscape of motor diagnostics in EVs, setting new standards for safety, performance, and efficiency

CHAPTER 2

LITERATURE SURVEY

2.1 Securing Electric Vehicle Performance: Machine Learning Driven Fault Detection and Classification.

Authors: Mahbub Ul Islam Khan Md, Ilius Hasan pathan.

This study simulates an EV model in MATLAB Simulink, generating data for two-phase and three-phase faults in BLDC motors. Multiple machine learning models are trained using Python and evaluated for performance. The Voting Classifier achieves high accuracy, proving machine learning-based fault detection is faster and more reliable than traditional methods. This enhances EV safety and supports sustainable transportation.

2.2 Fault Diagnosis in the Brushless Direct Current Drive Using Hybrid Machine Learning Models

Authors: K.V.S.H. Gayatri Sarman,Tenneti Madhu,Mallikarjuna Prasad

This paper introduces a hybrid machine learning model for fault diagnosis in BLDC motors. It combines classifiers like Decision Tree and K-Nearest Neighbors for accurate fault detection. The model analyses motor parameters such as current, voltage, and speed. By leveraging multiple algorithms, it improves diagnostic accuracy and reliability. The proposed approach ensures early fault identification and enhances motor performance.

2.3 Artificial Neural Networks Based Control and Analysis of BLDC Motors.

Authors: Porselvi T, Sai Ganesh CS, and Aouthithiye Barathwaj SR Y

This paper explores an ANN-based approach for BLDC motor speed control, integrating a PI controller and a bridge converter. MATLAB/Simulink is used for simulations, while Artificial Neural Networks (ANNs) provide predictive analysis of motor behavior. The proposed method has been validated with high accuracy, proving effective in accurately predicting BLDC motor parameters and enhancing speed control efficiency.

2.4 Data Preprocessing Techniques in Convolutional Neural Network Based on Fault Diagnosis Towards Rotating Machinery.

Authors: ShengnanTang, Shouqi Yuan, Yong Zhu

This paper explores data preprocessing techniques such as FFT, Wavelet Transform, and S-Transform to enhance CNN-based fault diagnosis in rotating machinery. These technique

improve feature extraction, noise reduction, and classification accuracy. A comparative analysis identifies the most effective preprocessing methods for improving fault detection performance, ensuring higher reliability in machinery diagnostics.

2.5 Fault Diagnosis and Fault-Tolerant Control of PMSM Drives State of the Art and Future Challenges.

Authors: Teresa Orlowska- Kowalska, Marcin Wolkiewicz, Przemyslaw Pietrzak

This paper explores fault diagnosis and fault-tolerant control methods for PMSM drives, addressing electrical, magnetic, and mechanical faults. It examines detection techniques using signal processing, mathematical modeling, and AI-based methods, along with vector control structures. The study highlights that AI-based approaches improve fault detection and classification, while hybrid methods enhance diagnostic accuracy. Future research should focus on real-time implementation for better reliability in PMSM drive systems.

2.6 Observer-Based Fault Detection Approach Using Fuzzy Adaptive Poles Placement System With Real-Time Implementation.

Authors: Magdy Abdullah Eissa, Aduwati Sali, Faisul Arif Ahmad, and Rania R. Darwish.

This paper presents a Fuzzy Adaptive Poles Placement (FAPP) system for real-time sensor fault detection in BLDC motors. The approach optimizes pole placement, improving robustness against noise and disturbances. The proposed method achieves higher fault detection accuracy compared to the Luenberger observer, proving its feasibility for real-time industrial applications.

2.7 Finding fault types of BLDC motors within UAVs using machine learning techniques.

Authors: Dragos Alexandru Andrioaia , Vasile Gheorghita Gaitan

This paper explores a data-driven approach for detecting BLDC motor faults in UAVs using sensor data and machine learning classifiers such as K-Nearest Neighbor, Support Vector Machine (SVM), and Bayesian Network. The study demonstrates that machine learning effectively identifies motor defects, aiding predictive maintenance and enhancing operational reliability. Among the tested models, SVM achieved the highest accuracy, making it the most effective classifier for fault detection.

2.8 Artificial Neural Networks-Based Torque Distribution for Riding Comfort Improvement of Hybrid Electric Vehicles.

Authors: Adel Oubelaida, Nachaat Mohamedb, Rajkumar Singh Rathorec, Mohit Bajajd, Toufik Rekiouaa

This paper presents an ANN-based torque distribution method for hybrid electric vehicles (HEVs), optimizing power allocation between the front and rear wheels. Using MATLAB/Simulink simulations, the study evaluates its impact on propulsion efficiency and driving comfort. Results show that the approach reduces transient torque ripples, ensures smoother drive mode transitions, and enhances vehicle stability and ride comfort.

2.9 Optimizing Detection: Compact Mobile Net Models for Precise Hall Sensor Fault Identification in BLDC Motor Drives.

Authors: Seul Ki Hong and Yongkeun Lee

This paper explores Mobile Net-based neural networks for real-time Hall sensor fault detection in BLDC motors, ensuring high accuracy, low latency, and minimal computational overhead. The study demonstrates that MobileNetV1 and MobileNetV2 outperform other models in efficiency, achieving high fault detection accuracy while requiring minimal computational resources, making them ideal for real-time applications.

2.10 Remaining Useful Life Estimation of BLDC Motor Considering Voltage Degradation and Attention-Based Neural Network.

Authors: Tanvir Alam Shifat, Hur Jang-Wook

This paper presents an attention-based LSTM model for estimating the remaining useful life (RUL) of BLDC motors, focusing on voltage degradation monitoring. A generator-motor setup is used for fault detection and prediction. Results show that the attention-based LSTM outperforms conventional LSTM, offering higher accuracy and robustness against environmental factors like noise and temperature, making it ideal for real-time monitoring.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 PROPOSED MACHINE LEARNING APPROACH FOR BLDC MOTOR FAULT CLASSIFICATION

The proposed methodology presents an intelligent, efficient, and scalable approach to fault detection and classification in Brushless DC (BLDC) motors using cutting-edge AI techniques. This strategy is meticulously designed to transform raw acoustic signals into actionable diagnostic insights with remarkable precision and speed. The workflow is a synergy of sound engineering and artificial intelligence, and it unfolds across the following transformative stages:

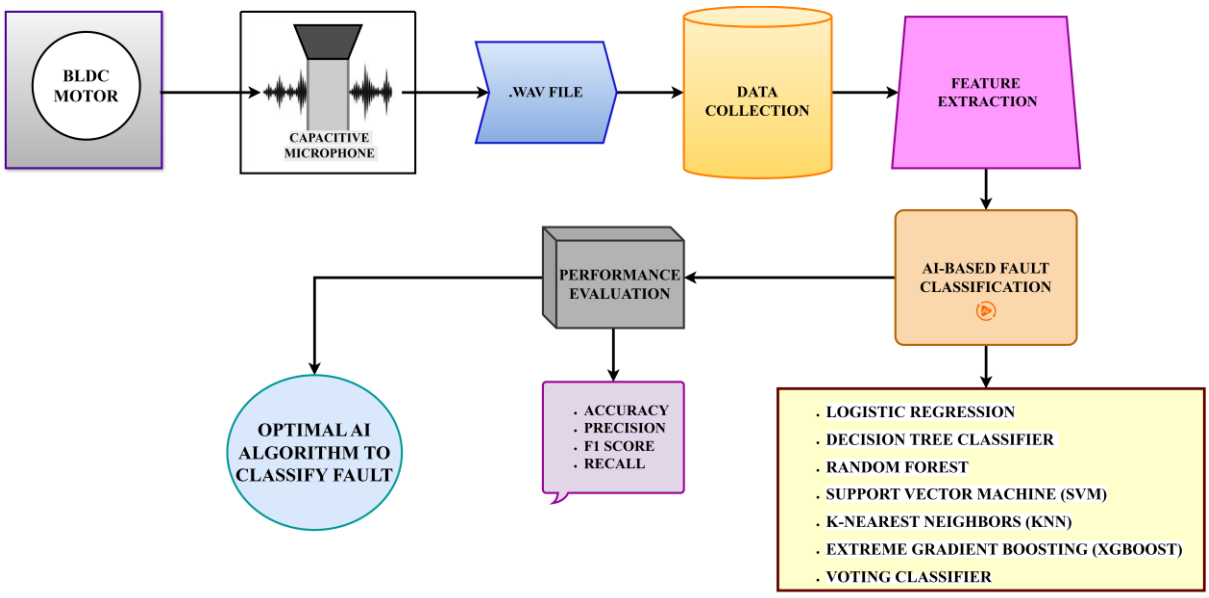

Figure 3.1 Block Diagram of AI-Based Fault Classification System for BLDC Motor.

Figure 3.1 shows the complete block diagram which captures the end-to-end process flow of the proposed fault diagnosis system. Each stage, from sound data acquisition to AI-based fault classification and performance evaluation, plays a critical role in ensuring accurate and reliable fault detection. Building upon this structured methodology, the next phase involves simulating and monitoring various motor conditions to generate meaningful data for model training and validation.

3.1.1 MOTOR CONDITION SIMULATION AND MONITORING

The fault detection process begins with the operation of a BLDC motor under three distinct and controlled conditions to simulate real-world scenarios. These conditions are designed to reflect both normal and faulty operational states, each producing unique acoustic patterns essential for training and testing the AI models.

The conditions are as follows:

- Healthy State

  In this condition, the BLDC motor operates smoothly without any mechanical abnormalities. It runs under optimal parameters with balanced components and minimal vibration or noise. The sound emitted is stable and consistent, representing a fault-free motor in proper working order. This serves as the baseline or reference state.

- Bearing Fault

  This condition simulates a defect in the motor's internal bearings, one of the most common failure modes in rotating machinery. Bearing faults typically arise from wear, corrosion, or physical damage. When the motor runs under this condition, the faulty bearing introduces irregular vibrations and high-frequency noise. These subtle acoustic disturbances are key indicators of early-stage bearing failure.

- Propeller Fault

  In this scenario, the motor operates with a damaged or imbalanced propeller. This could result from cracks, deformation, or mass imbalance due to accumulated debris or physical impacts. The sound produced under a propeller fault condition tends to be rhythmic and fluctuating, indicating uneven rotation and aerodynamic disturbance.

By running the BLDC motor under these three well-defined conditions, the system generates a diverse and informative dataset of sound signals. Each condition contributes its own acoustic signature, enabling the AI model to learn and differentiate between healthy and faulty states with high accuracy.

3.1.2 HIGH-FIDELITY SOUND DATA ACQUISITION

The journey begins with the precise capture of acoustic emissions from a BLDC motor operating under three distinct states: healthy, bearing fault, and propeller fault. These states are intentionally simulated to mirror real-world conditions a motor might experience over time. Each state generates a unique sound pattern—subtle, intricate, and often beyond the threshold of human perception.

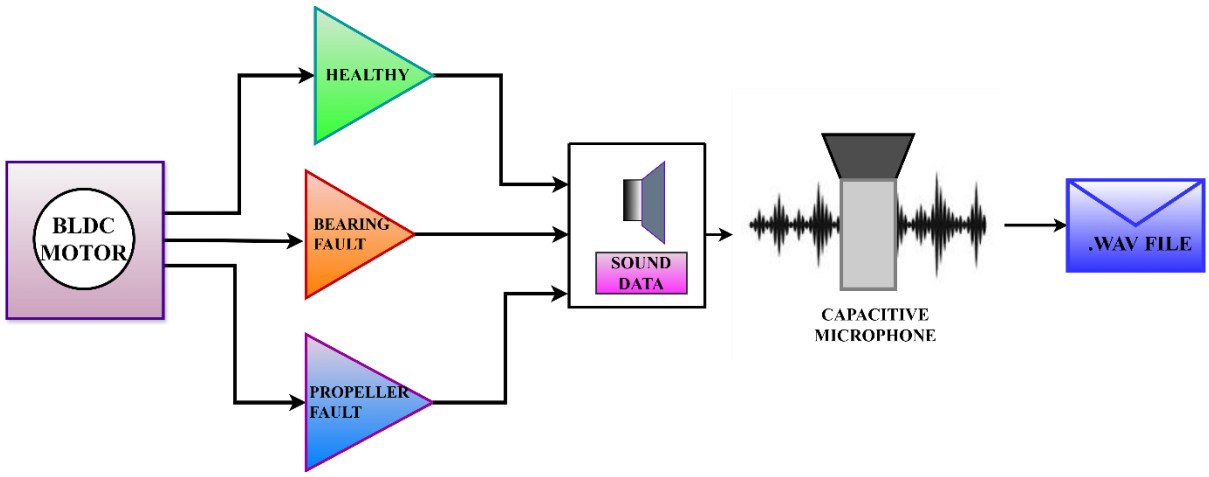

Figure 3.2 Acoustic Signature Capture for Fault Diagnosis in BLDC Motors

To preserve the integrity of these sound signatures, high-sensitivity microphones are deployed to record the motor's acoustic output.

The recordings are stored in .wav format to ensure no loss of quality. This file type is chosen for its ability to retain full audio resolution, preserving every pulse, vibration, and fluctuation embedded within the motor's sound profile. These sound recordings form the foundational layer of the AI-driven fault classification system.

As illustrated in figure 3.2, these acoustic recordings form the foundation for the subsequent AI-based fault classification process, where machine learning models analyse and differentiate motor conditions based on their sound characteristics.

3.1.3 ADVANCED AUDIO PREPROCESSING AND FEATURE EXTRACTION

Once the sound data is acquired, it undergoes an essential transformation process. Preprocessing begins with resampling to unify the frequency scale across all audio files, followed by normalization to equalize sound levels and eliminate inconsistencies. This step is

vital to remove background noise and environmental distortions, allowing the motor's core mechanical acoustics to surface clearly and consistently.

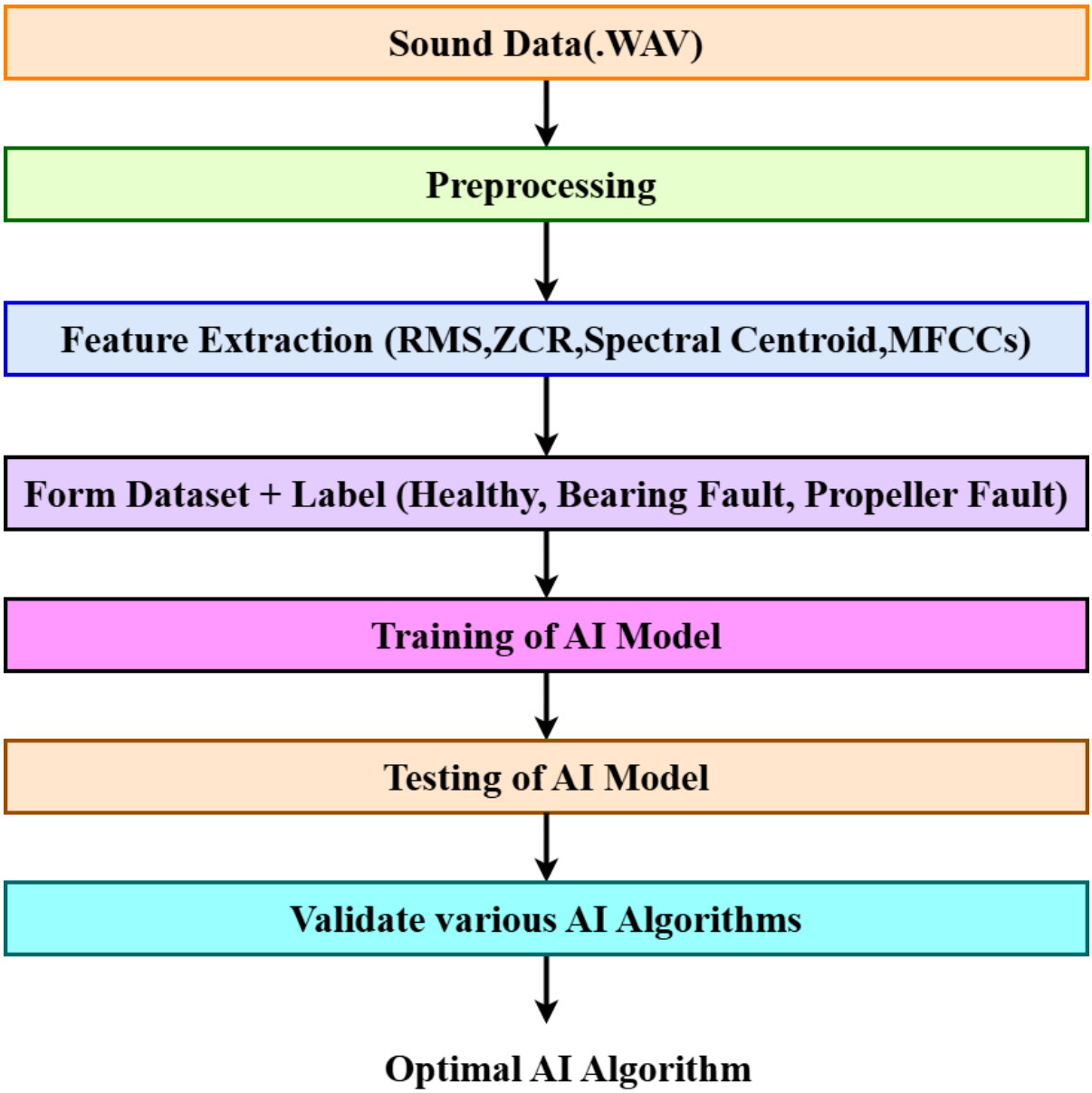

Figure 3.3 Detailed Workflow for Sound-Based Fault Diagnosis and Classification in BLDC Motor

After preprocessing, sophisticated feature extraction techniques are applied to distilled valuable insights from the raw sound data. These techniques extract a range of acoustic parameters, including:

- Root Mean Square (RMS), measuring the energy content of the signal.
- Zero Crossing Rate (ZCR), indicating the frequency of waveform sign changes.
- Spectral Centroid, revealing the brightness and sharpness of the audio spectrum.
- Mel Frequency Cepstral Coefficients (MFCCs 1–13), simulating the human ear's perception of sound frequencies and capturing the tonal patterns critical for fault identification.

The illustrated workflow in figure 3.3 captures the complete process from sound data acquisition to fault classification in the BLDC motor. Each stage, including preprocessing, feature extraction, dataset preparation, and AI model development, ensures that essential and meaningful information is extracted and utilized. This systematic transformation improves the accuracy and reliability of the final fault diagnosis results.

3.1.4 STRUCTURED DATASET FORMATION FOR MODEL TRAINING

The extracted features are compiled into a structured dataset that serves as the input for machine learning algorithms. Each row in the dataset represents a unique sound sample, while each column corresponds to a specific feature or class label.

The dataset is saved in Excel format with a total of 2500 rows and 17 columns, ensuring a broad and representative data foundation for training robust AI models.

Figure 3.4 presents the organized workflow encompassing audio data acquisition, preprocessing, feature extraction, and dataset creation for the classification of faults in BLDC motors. The dataset is divided as follows:

- 956 samples labelled as Healthy
- 715 samples labelled as Bearing Fault
- 829 samples labelled as Propeller Fault

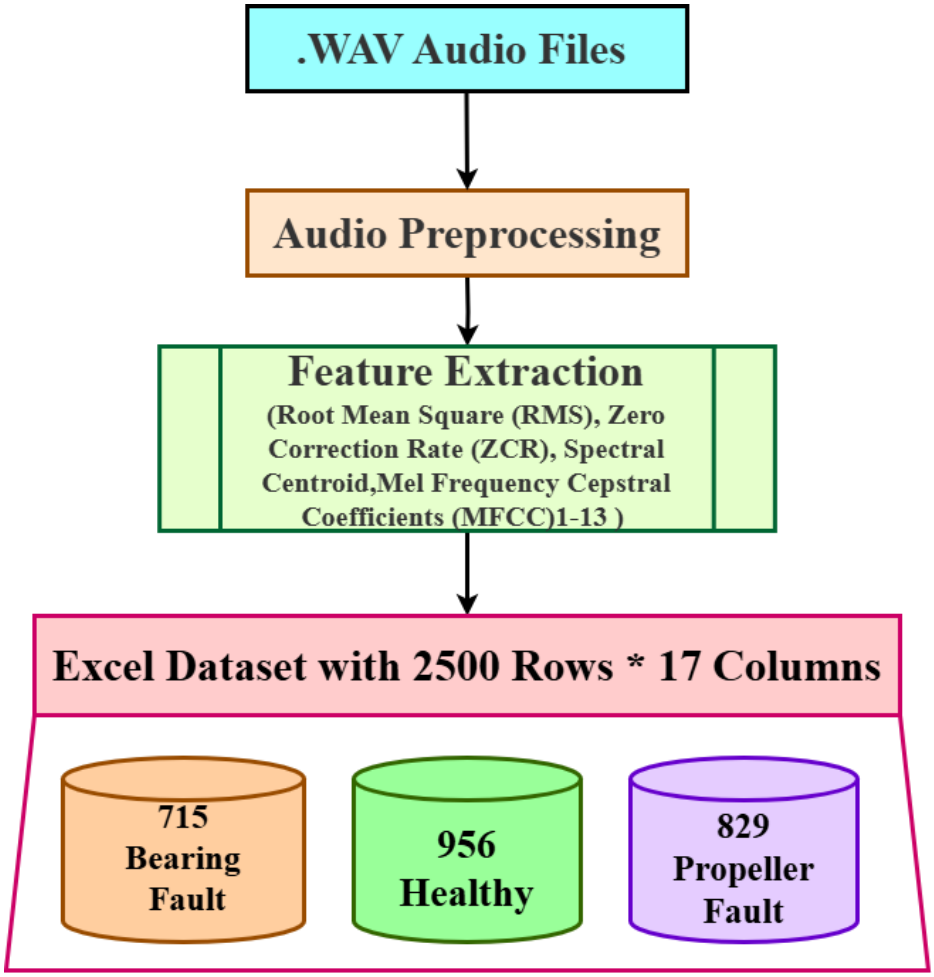

Figure 3.4 Structured Process of Audio Data Acquisition, Preprocessing, Feature Extraction, and Dataset Formation for BLDC Motor Fault Classification

This well-balanced dataset ensures that the learning model is exposed to a diverse set of conditions, enabling it to distinguish between different types of faults with high accuracy. It acts as the cornerstone of the classification system, enabling reliable diagnostics and intelligent fault prediction in real-world motor applications.

3.1.5 AI ALGORITHMS USED FOR FAULT CLASSIFICATION

To accurately classify the operating condition of the BLDC motor—whether healthy, bearing fault, or propeller fault—various supervised machine learning algorithms were employed. Each algorithm brings a unique strength to the classification process, offering a combination of interpretability, accuracy, and robustness.

1. **LOGISTIC REGRESSION (LR)**
2. **DECISION TREE (DT)**
3. **RANDOM FOREST (RF)**
4. **SUPPORT VECTOR MACHINE (SVM)**
5. **K-NEAREST NEIGHBORS (KNN)**
6. **EXTREME GRADIENT BOOSTING (XG BOOST)**
7. **VOTING CLASSIFIER (VC)**

3.1.5.1 LOGISTIC REGRESSION ALGORITHM

Logistic Regression is a fundamental supervised learning algorithm used for classification tasks. It predicts the probability of class membership using a logistic (sigmoid) function applied to a linear combination of input features. Though originally designed for binary classification, it can be extended to multiclass problems through approaches like one-vs-rest (OvR) or multinomial logistic regression. Figure 3.5 illustrates the working of the logistic r egression algorithm for both binary and multiclass classification scenarios. where the X-axis represents the input feature(s) and the Y-axis represents the predicted probability of class membership.

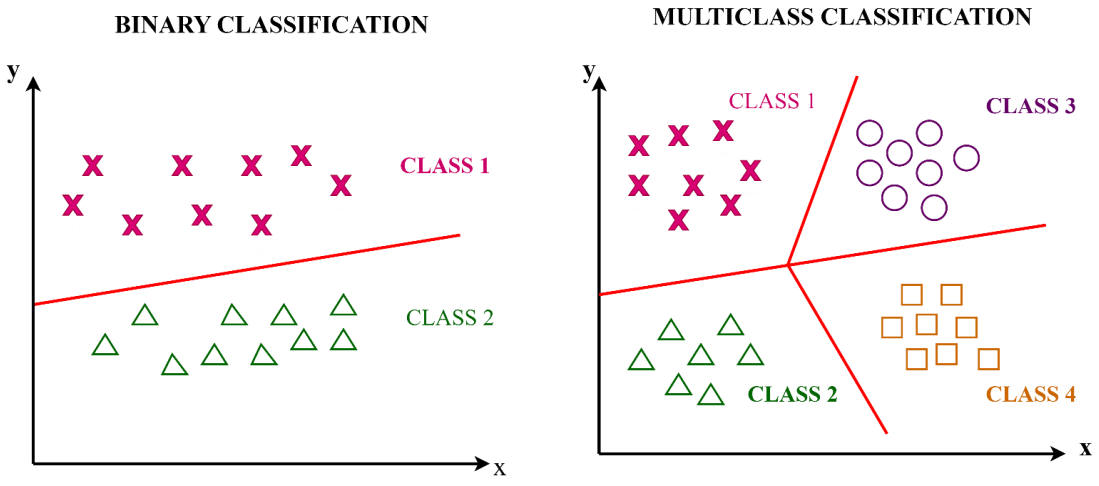

Figure 3.5 Logistic Regression for Binary and Multi-Class Classification

In this project, logistic regression was applied to classify motor conditions into three categories: Healthy, Bearing Fault, and Propeller Fault. An Excel dataset containing 2500 samples, enriched with extracted acoustic features such as MFCCs, Spectral Centroid, and Zero

Crossing Rate (ZCR) along with corresponding labels, was uploaded into the Python environment for processing. Logistic Regression was then implemented on this dataset to develop the classification model. The features Spectral Centroid (x-axis) and Zero Crossing Rate (y-axis) were specifically used to visualize the classification results, where distinct linear decision boundaries effectively separate the three motor fault classes.

- Spectral Centroid reflects the brightness of the sound. Higher values typically indicate higher-frequency components, which may be linked to imbalances or faults.
- The rate at which the audio signal changes sign is measured by the Zero Crossing Rate. It is frequently raised in noisy or malfunctioning environments.

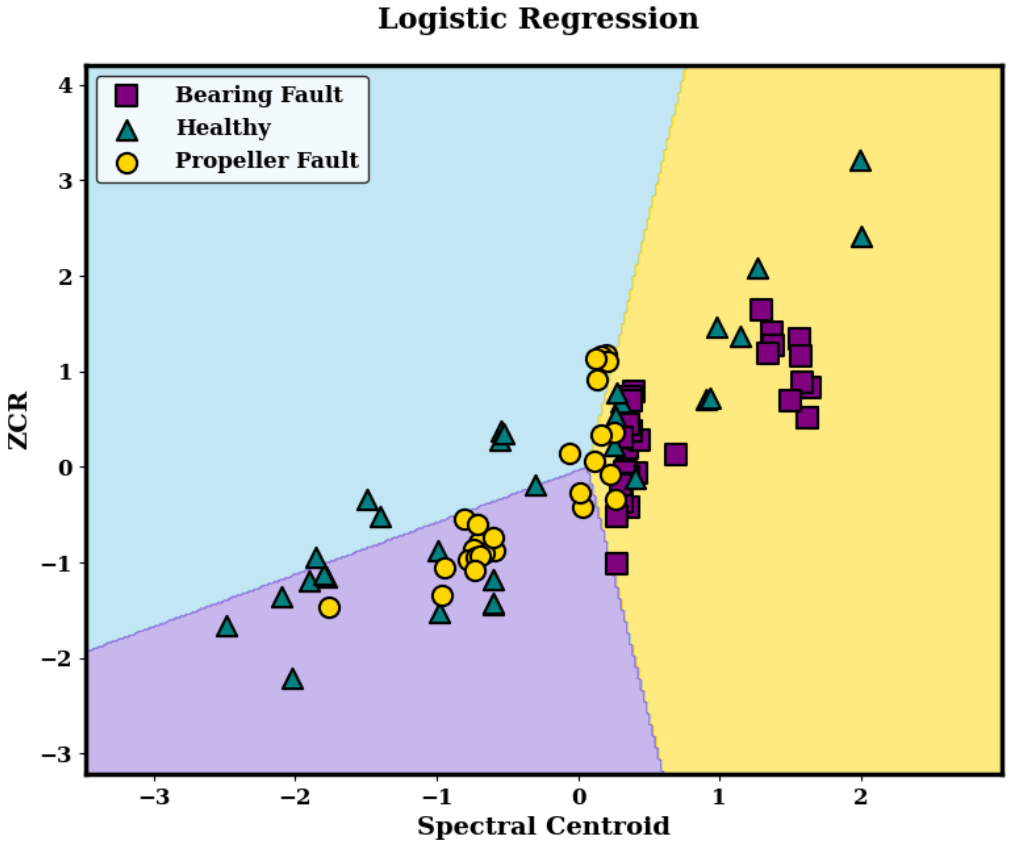

Figure 3.6 Decision Boundary Visualization of Logistic Regression for BLDC Motor Fault Classification

A graph displaying the decision boundaries produced by the Logistic Regression model was produced when the Python code was executed. Each region in the plot represented one of the three motor conditions:

Figure 3.6 illustrates the classification of motor health conditions based on extracted audio features using Logistic Regression, where,

- **Healthy Motor** is typically characterized by lower Zero Crossing Rate (ZCR) and moderate spectral centroid values.
- **Bearing Fault** is represented by a higher ZCR due to irregular mechanical vibrations.
- **Propeller Fault** is indicated by higher spectral centroid values resulting from aerodynamic irregularities.

The resulting graph visually confirms that Logistic Regression can effectively distinguish between the different fault types based on the extracted sound features. This successful application proves the capability of Logistic Regression to handle real-world multiclass classification problems in BLDC motor fault detection.

3.1.5.2 DECISION TREE ALGORITHM

A Decision tree is a widely used supervised learning algorithm that organizes data through a series of decisions based on feature values, ultimately reaching a classification outcome at the leaf nodes.

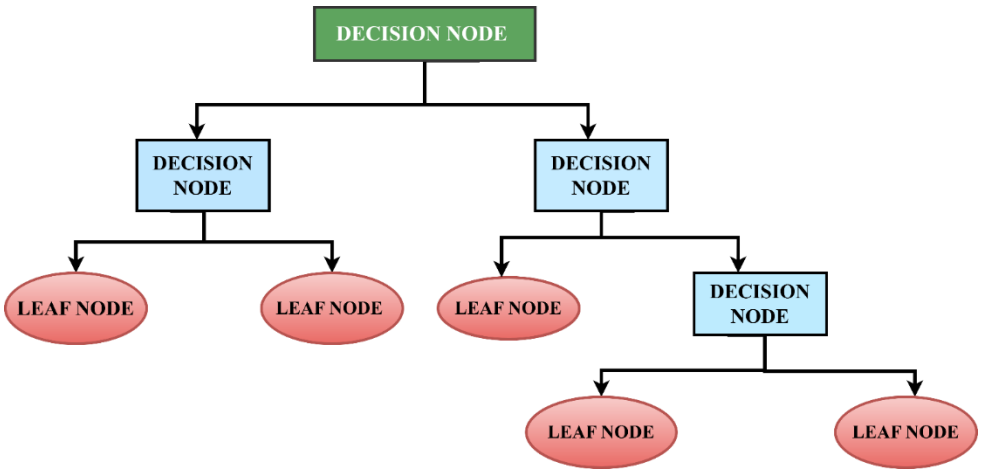

Figure 3.7 Structure of a Decision Tree Model Representing Decision Nodes and Leaf Nodes

Each internal node of the tree represents a test on a specific feature, and branches correspond to the outcomes of these tests. Figure 3.7 hierarchical shows the Decision tree model complex decision boundaries in a simple and interpretable manner, making it highly effective for classification tasks.

The Decision tree algorithm was applied for the classification of motor faults into three categories: Healthy, Bearing Fault, and Propeller Fault. The process began by uploading the Excel dataset containing essential acoustic features such as Mel Frequency Cepstral Coefficients (MFCCs), Spectral Centroid, and Zero Crossing Rate (ZCR), which were extracted during the audio processing phase. Using Python, a Decision Tree Classifier was trained on the dataset to develop a model capable of distinguishing between the three motor conditions.

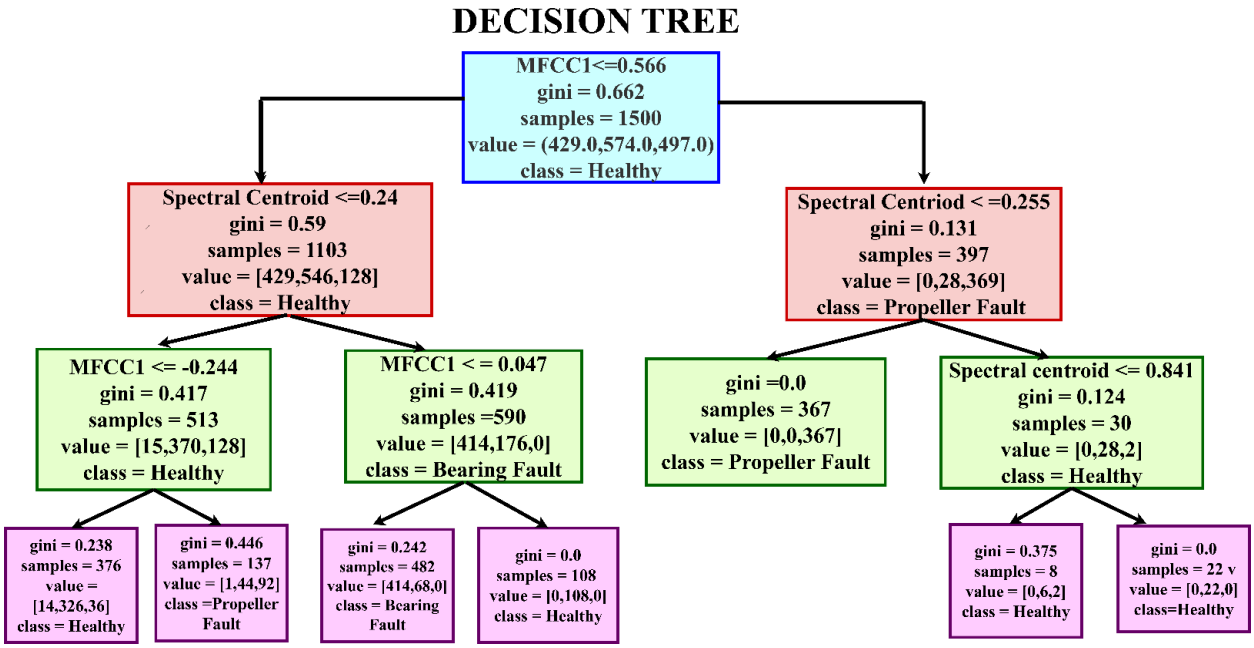

Figure 3.8 Decision Tree Classification Output for BLDC Motor Health Monitoring Based on MFCC1 and Spectral Centroid Features

The output of the Decision Tree model is visualized in the figure 3.8, where the root node initiates the first decision based on the MFCC1 feature. Subsequent nodes use additional splits based on both MFCC1 and Spectral Centroid to progressively partition the data.

Each decision leads closer to a final classification at the leaf nodes, where the motor condition is conclusively labelled. The tree structure also provides important information at each node, including the Gini impurity index, the number of samples reaching the node, the

distribution of classes within those samples. A lower Gini value at a node indicates a purer separation of classes.

This Decision tree model effectively demonstrates how critical audio features can be used to accurately classify motor faults. Its interpretable nature not only provides high classification performance but also offers clear insights into the decision-making process, making it a valuable tool for motor health monitoring applications.

3.1.5.3 RANDOM FOREST ALGORITHM

Random Forest is an ensemble learning technique that builds a multitude of decision trees and merges their results to improve the overall prediction accuracy and control overfitting. It operates by creating several individual decision trees during training shows in figure 3.9 where each tree is built from a random subset of the dataset and uses a random subset of features for splitting at each node.

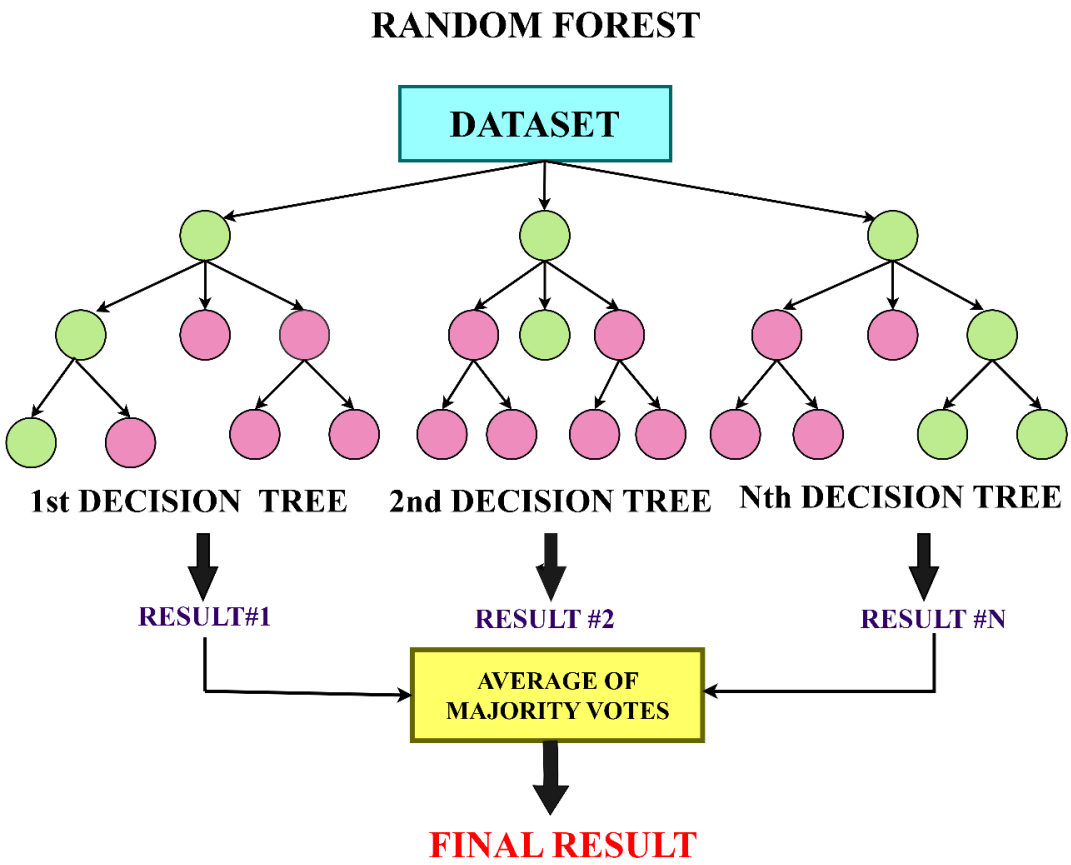

Figure 3.9 Conceptual Workflow of Random Forest Algorithm Using Ensemble of Decision Trees and Majority Voting

This randomness ensures that the model captures a wide diversity of patterns within the data, leading to a more robust and generalized predictive performance. For classification tasks, the random forest aggregates the predictions of each tree through a majority voting mechanism, while for regression tasks, it averages the outputs. The strength of Random Forest lies in its ability to handle high-dimensional data, mitigate overfitting, and maintain high accuracy even when a large proportion of the data is missing.

The Random Forest algorithm was implemented to classify the condition of mechanical components based on extracted audio features. The features utilized included MFCC (Mel-frequency cepstral coefficients), Spectral Centroid, RMS (Root Mean Square), and Zero Crossing Rate (ZCR). These features, known for capturing important characteristics of sound signals, were fed into the model to differentiate between three main classes: Healthy, Propeller Fault, and Bearing Fault.

The training process involved building multiple decision trees, each making classifications based on different feature thresholds. For instance, the first decision tree initially splits the data based on the Spectral Centroid feature and further uses MFCC1, MFCC2, and ZCR values to refine the classification at each level. Similarly, the second and third trees show variations in the decision paths, emphasizing the diversity among trees by selecting different features and thresholds for splitting.

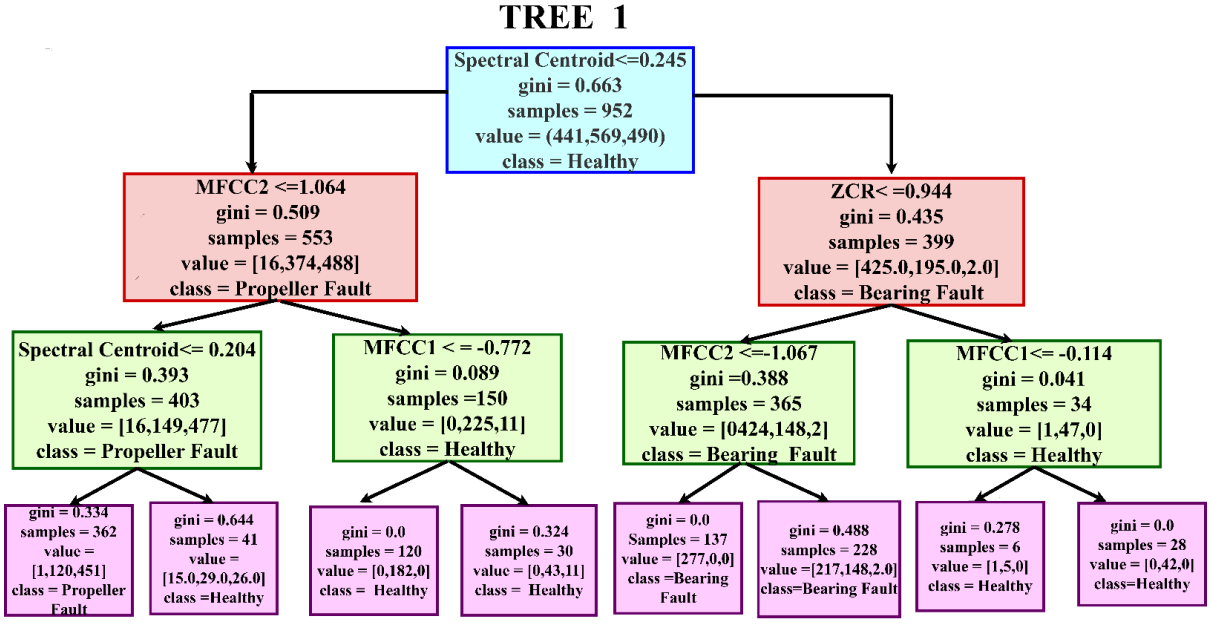

Figure 3.10(a) Decision Tree 1 Visualization from Random Forest Classifier

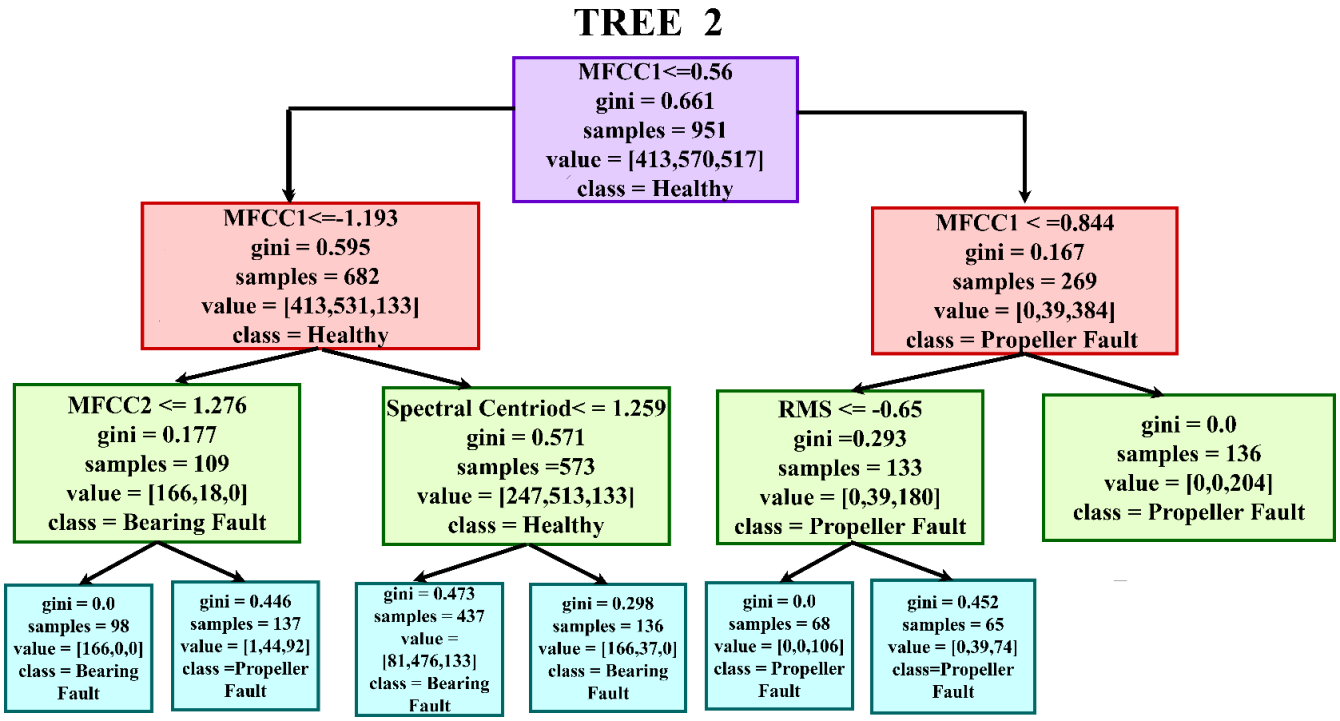

Figure 3.10(b) Decision Tree 2 Visualization from Random Forest Classifier

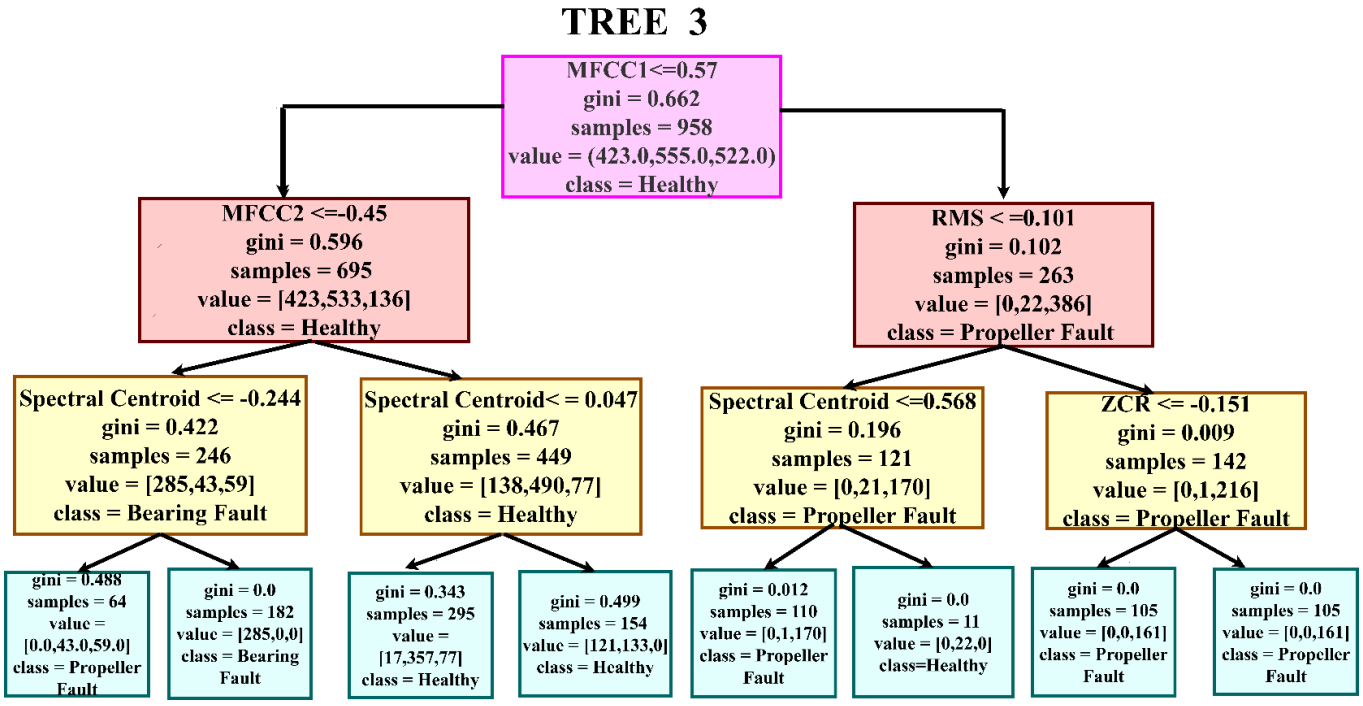

Figure 3.10(c)

Figure 3.10(c) Decision Tree 3 Visualization from Random Forest Classifier

The results from Random Forest model figure3.10(a)(b)(c) demonstrate strong classification capability. Each individual tree exhibits a different decision-making process, yet all trees consistently identify key features like MFCC1, MFCC2, Spectral Centroid, RMS, and ZCR as important for distinguishing between the classes. In Tree 1, the initial split on Spectral Centroid separates healthy components effectively, while Tree 2 relies heavily on MFCC1 for its initial decision, and Tree 3 also focuses on MFCC2 and Spectral Centroid. Despite minor variations in feature importance across the trees, the ensemble approach ensures that misclassifications by individual trees are corrected when the results are aggregated. The final outcome, based on the majority voting across all trees, leads to a highly accurate classification, thereby validating the robustness of Random Forest in fault detection applications using audio features.

3.1.5.4 SUPPORT VECTOR MACHINE

A popular supervised learning approach for classification and regression problems is the Support Vector Machine (SVM). SVM's primary goal is to maximize the margin between every class's proximal points, or support vectors, in order to figure out the ideal hyperplane for segregating data points between the different classes. When the data is not linearly separable, SVM utilizes kernel functions to project the data into higher-dimensional spaces where separation becomes feasible.

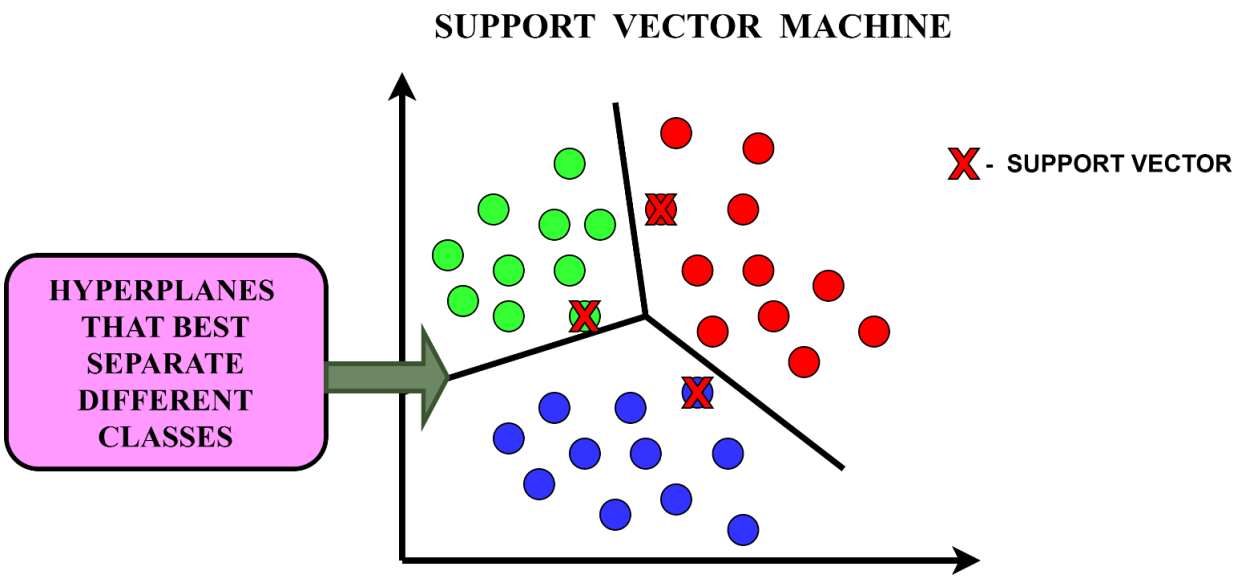

Figure 3.11 Illustration of Support Vector Machine (SVM) Hyperplane Separation for Multiclass Classification

Most commonly used kernel is the Radial Basis Function (RBF) kernel, which is capable of handling complex non-linear boundaries between classes. SVM is known for its robustness in high-dimensional spaces and its effectiveness in cases where the number of dimensions exceeds the number of samples. Figure 3.11 illustrates the concept of SVM, showing how hyperplanes are optimally positioned to best separate different classes.

The Support Vector Machine with an RBF kernel was employed to classify mechanical component conditions based on audio features. The two features used for visualization were Spectral Centroid and Zero Crossing Rate (ZCR), selected for their relevance in capturing the characteristics of sound signals. The classes to be identified were Healthy, Propeller Fault, and Bearing Fault.

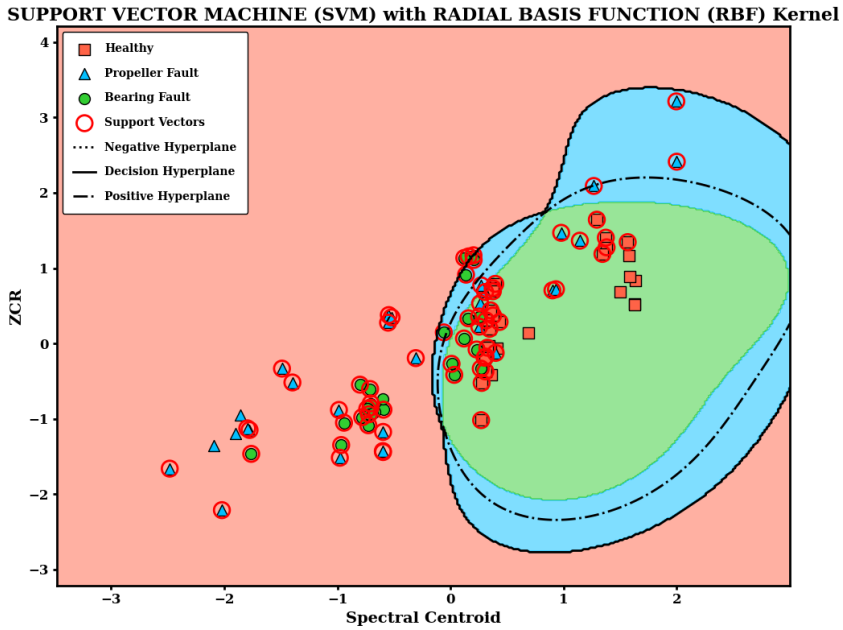

Figure 3.12 Classification Results Using Support Vector Machine (SVM) with Radial Basis Function (RBF) Kernel Showing Decision Boundaries and Support Vectors

During training, SVM determined the optimal decision boundary that separates these classes, with help of support vectors critical data points that define the margin. The resulting decision regions are visualized in the plot, with different coloured areas representing the classification regions for each fault type. The support vectors are highlighted with red circles, emphasizing their importance in constructing the decision boundaries.

Figure 3.12 show that the SVM with RBF kernel was able to effectively separate the three classes, although some overlap between classes is visible, particularly around the boundary regions. The decision hyperplane (solid black line) and the margins (dashed and dash-dotted lines) illustrate how SVM attempts to maximize the separation between different classes. Most of the samples are correctly classified within their respective regions: healthy components mainly occupy the green region, propeller faults are concentrated in the blue region, and bearing faults are scattered but mostly aligned with the classification boundaries. The concentration of support vectors around the decision boundaries suggests that these cases were more challenging to classify, reflecting the model's focus on difficult instances. Overall, the SVM model performed well in distinguishing the faults based on selected audio features, proving its capability for fault diagnosis in mechanical systems.

3.1.5.5 K-NEAREST NEIGHBORS

The K-Nearest Neighbors (KNN) algorithm is a simple yet powerful supervised machine learning method used for classification and regression tasks. KNN operates based on the principle that similar instances exist close to each other in the feature space. To classify a new data point, KNN looks at the 'k' closest labelled examples in the dataset and assigns the majority class among them. The algorithm is non-parametric, meaning it makes no assumptions about the underlying data distribution, making it flexible for a wide range of applications.

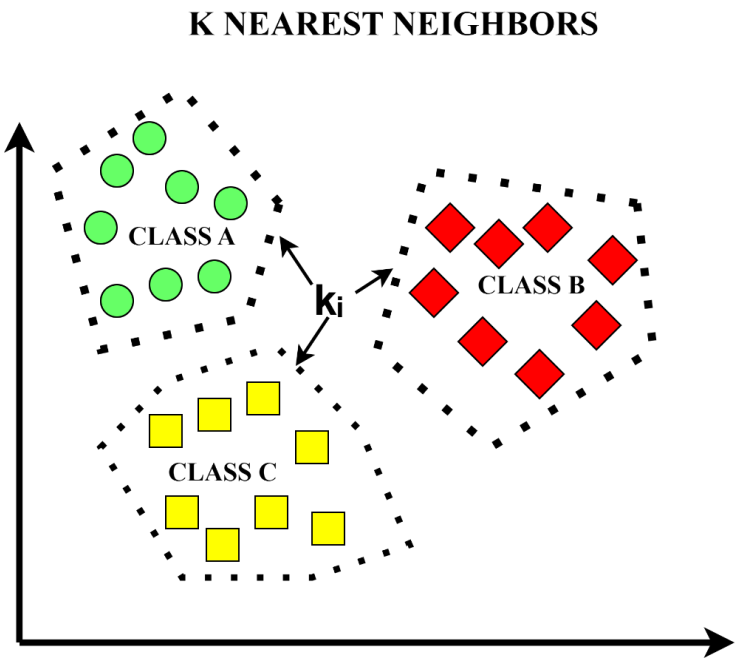

Figure 3.13 Illustration of KNN Algorithm

However, KNN's performance can be sensitive to the choice of 'k' value and the scaling of the feature space. Figure 3.13 illustrates the concept of the K-Nearest Neighbors (KNN) algorithm, where a new data point is classified based on the majority class among its nearest neighbors. The different regions demonstrate how the feature space is divided according to the proximity of data points from various classes.

The K-Nearest Neighbors classifier was utilized to diagnose the condition of mechanical components based on two extracted audio features: Spectral Centroid and Zero Crossing Rate (ZCR). The goal was to classify the samples into three categories: Healthy, Bearing Fault, and Propeller Fault. The decision boundaries appeared during KNN model training, showing how the classifier divides several feature space areas using the nearest neighbor rule of majority. Based on the proximity of the training data points, each region in the plot reveals the predicted class.

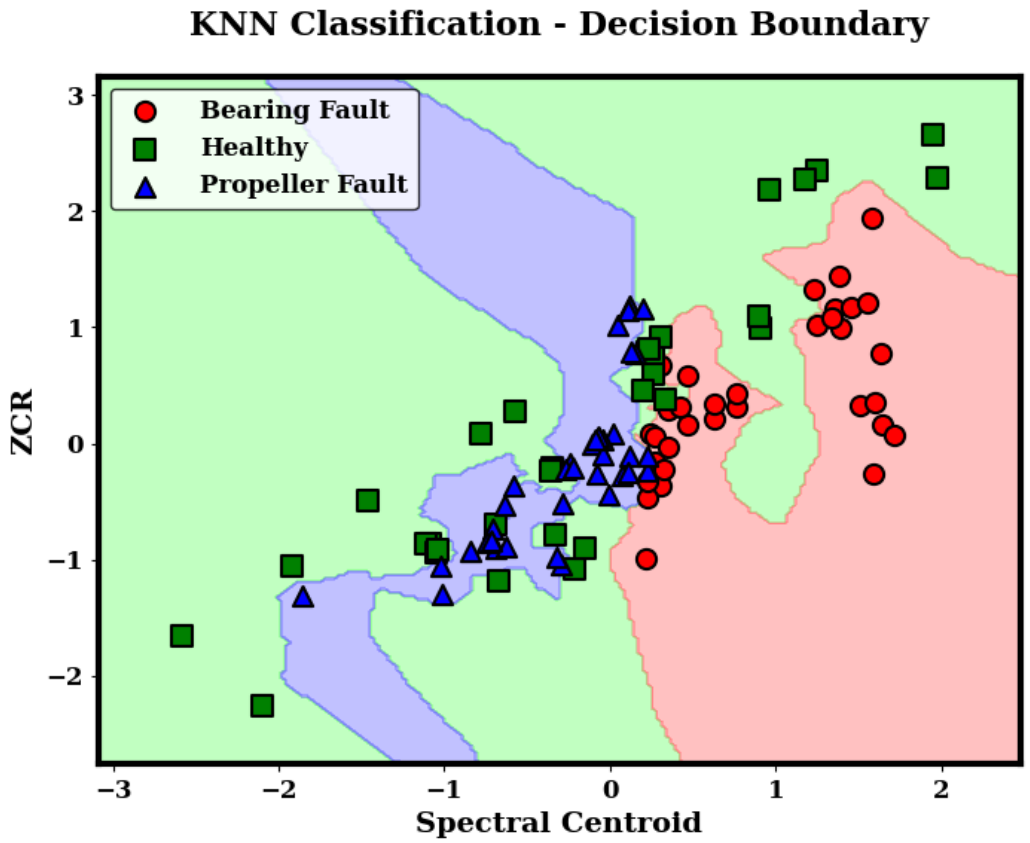

Figure 3.14 KNN Classification Decision Boundaries For Fault Diagnosis.

The KNN classification shown in figure 3.14 reveal that the model was able to reasonably distinguish between the different fault conditions. The decision boundaries are

somewhat irregular and complex, reflecting KNN's local and instance-based nature. The green region predominantly contains healthy samples, the red region includes bearing faults, and the blue region captures the propeller faults.

However, some areas show intermixing of classes, indicating that a few data points are close to each other across classes and thus challenging to classify correctly. This behaviour is typical for KNN, especially when the data points of different classes overlap or are located near each other.

3.1.5.6 EXTREME GRADIENT BOOSTING

A robust ensemble machine learning technique based on gradient boosting, Extreme Gradient Boosting (XG Boost) is designed for high accuracy, speed, and efficiency. It performs by building a series of decision trees, each of which is intended to correct the flaws of the one before it. XG Boost includes features like regularization to reduce overfitting, and it is highly scalable with support for parallel and distributed computing. It performs particularly well in handling complex, non-linear data patterns, making it a popular choice for classification tasks.

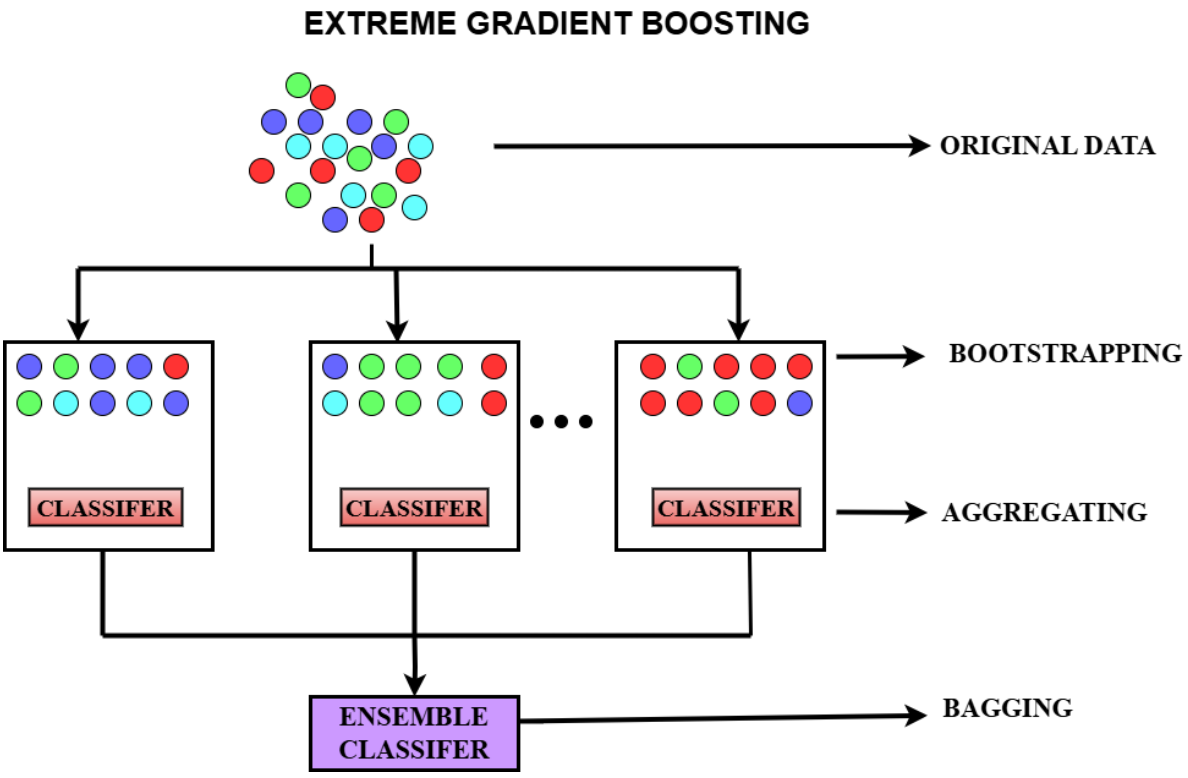

Figure 3.15 Illustration of XG Boost model with ensemble of weak learners

Figure 3.15 illustrates the concept of the Extreme Gradient Boosting (XG Boost) algorithm, where multiple decision trees are combined sequentially to improve classification performance. The feature space is divided into regions based on the cumulative predictions of weak learners, resulting in a strong overall model that accurately separates different classes.

In this project, XG Boost was implemented to classify the condition of mechanical components into three categories: Healthy, Propeller Fault, and Bearing Fault, using Spectral Centroid and Zero Crossing Rate (ZCR) as input features. The resulting decision boundary plot demonstrates that XG Boost generated smooth, curved boundaries that adapt well to the underlying data distribution. Compared to KNN and SVM, the separation between classes using XG Boost is cleaner and better suited for non-linear patterns. The healthy samples, represented by squares, are mostly well-clustered, while the propeller fault and bearing fault samples show relatively clear division with minor overlap.

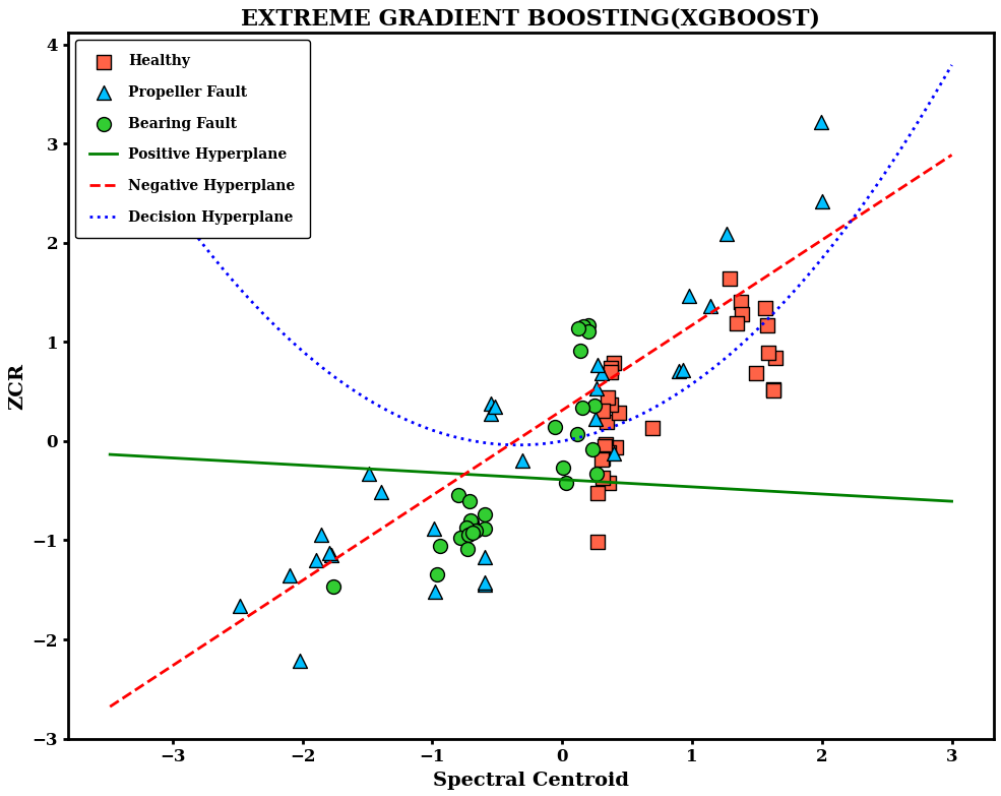

Figure 3.16 Decision boundary visualization of XG Boost Classifier for fault diagnosis

The presence of the positive, negative, and decision hyperplanes in figure 3.16 shows how XG Boost flexibly defines class regions without strictly linear assumptions. This algorithm

flexibility allows XG Boost to achieve a good balance between bias and variance, avoiding underfitting and overfitting problems. In conclusion, XG Boost demonstrated strong performance in fault classification, making it highly suitable for real-time fault diagnosis applications where both high accuracy and reliable predictions are critical.

3.1.5.7 VOTING CLASSIFIER

Voting Classifier is an ensemble learning method that combines the predictive power of multiple machine learning models to enhance overall classification accuracy. In this approach, three different models—XG Boost, Random Forest, and Logistic Regression—are independently trained on the same dataset. Each model makes its own predictions, and the Voting Classifier aggregates these outputs to make a final decision based on majority voting (for classification tasks). Figure 3.17 illustrates the Voting Classifier, where multiple models combine their predictions to enhance classification accuracy by majority voting.

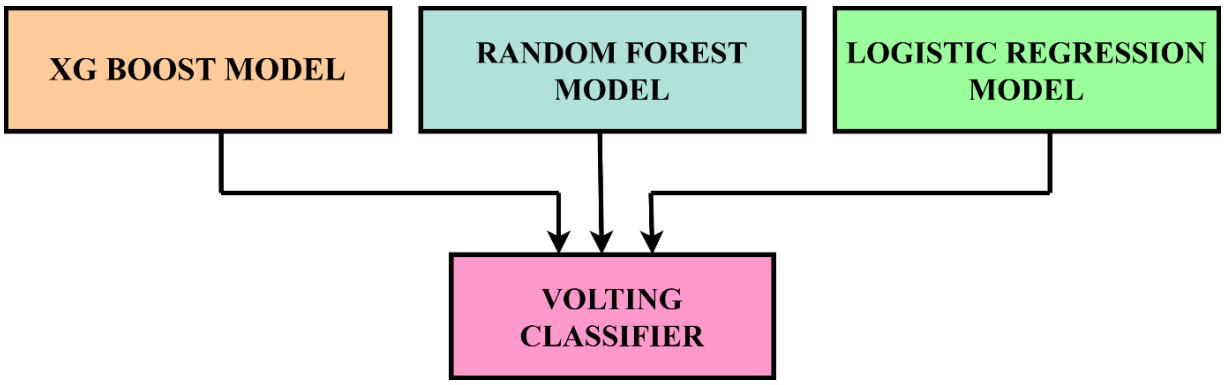

Figure 3.17 Architecture of Voting Classifier Combining XG Boost, Random Forest, And Logistic Regression

The voting classifier was utilized to classify the condition of mechanical components into three categories: Healthy, Propeller Fault, and Bearing Fault, based on Spectral Centroid and Zero Crossing Rate (ZCR) features. The ensemble combined the strengths of XG Boost, Random Forest, and Logistic Regression models to improve prediction accuracy.

The resulting decision boundary plot shows a more intricate partitioning of the feature space, with sharp and detailed class boundaries. Compared to individual classifiers like KNN, SVM, or even XG Boost alone, the voting classifier offers a more balanced decision region that handles overlapping areas more robustly.

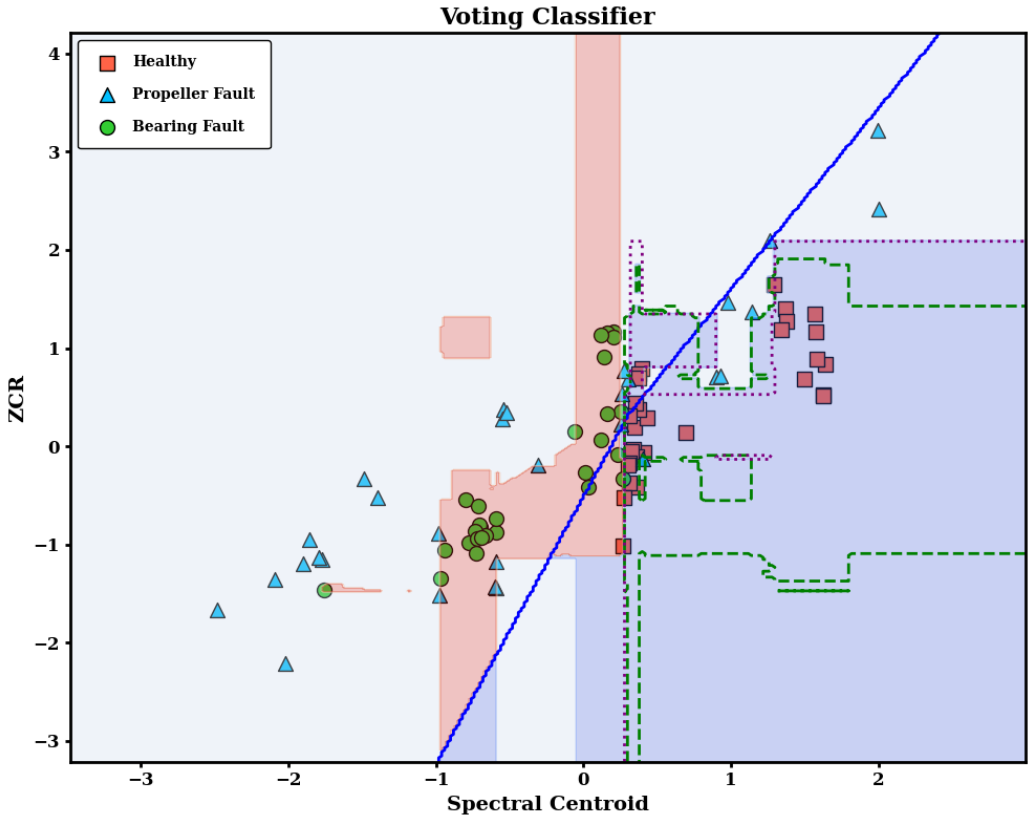

Figure 3.18 Decision Boundary Visualization of Voting Classifier Combining Multiple Models

Healthy samples (squares) are primarily grouped in a distinct region, while Propeller Fault (triangles) and Bearing Fault (circles) classes are separated with a finer granularity, minimizing misclassification. The ensemble approach enhanced the model's ability to generalize to complex patterns, offering a strong overall classification performance.

The decision boundary plot for the voting classifier in figure 3.18 demonstrates how the combined model effectively segregates the feature space, where different coloured regions represent areas classified into different categories: Healthy, Propeller Fault, and Bearing Fault.

Despite some overlaps, the voting classifier enhances robustness and generalization compared to any single model by balancing the biases and variances of the individual learners. This ensemble method is especially valuable when models have complementary strengths, making the final prediction more reliable and accurate.

3.1.6. Rigorous Performance Evaluation

A rigorous performance evaluation framework was employed to assess the effectiveness of each model, utilizing key metrics such as Accuracy, Precision, Recall, and F1 Score. Accuracy measured the overall correctness of predictions, Precision evaluated the model's ability to avoid false positives, Recall assessed its capacity to correctly identify true positives, and the F1 Score provided a balanced measure between Precision and Recall. This comprehensive evaluation ensured that the selected model was not only accurate but also reliable and robust under real-world conditions.

3.1.7. Optimal Model Selection

Based on the evaluation results, the optimal AI algorithm was chosen—one that demonstrated superior fault detection capabilities across all metrics. This model is recommended for real-time deployment as a smart diagnostic tool, capable of proactively identifying faults and minimizing BLDC motor downtime. Overall, the methodology signifies a major advancement in predictive maintenance, harnessing the power of AI to listen, learn, and act, thereby ensuring uninterrupted performance and enhanced reliability of motor systems.

3.2 HARDWARE DETAILS

The hardware setup is specifically designed for data collection to monitor and analyse the condition of a BLDC motor system. The setup enables capturing critical operational data necessary for fault diagnosis and predictive maintenance using AI algorithms. The primary components involved are the Switch Mode Power Supply (SMPS), Propeller, BLDC Motor, Electronic Speed Controller (ESC), and Arduino UNO. Each component plays a crucial role in ensuring efficient operation and accurate data acquisition. The detailed descriptions, key features, and advantages of each component are given below:

3.2.1 SWITCH MODE POWER SUPPLY (SMPS)

The Switch Mode Power Supply (SMPS) highlighted in figure 3.19 is a crucial component in the real-time data collection setup, responsible for providing a stable and efficient DC power source to drive the BLDC motor and control circuitry. In this project, a SMPS 12V 50A 600W unit is used to ensure reliable power delivery, which is essential for consistent motor

operation and accurate data recording. The SMPS plays a key role in maintaining system stability by supplying sufficient voltage and current even under varying load conditions.

Figure 3.19 SMPS for Stable Power Delivery

FEATURES

- **Stable 12V 50A Output:** Guarantees reliable operation of the BLDC motor and Arduino system without voltage fluctuations.
- **High Power Capacity (600W):** Supports continuous and heavy-load operation necessary during extended real-time data collection sessions.
- **Wide Temperature Tolerance:** Enables the hardware setup to function consistently even with ambient temperature changes.
- **Protected Connections:** Reduces the risk of electrical faults caused by dust or minor environmental factors, enhancing the system's robustness.

ADVANTAGES

- **Ensures Continuous Motor Operation:** The high current output allows the BLDC motor to operate smoothly without unexpected shutdowns, crucial for collecting uninterrupted sensor data.
- **Supports Long-Term Experiments:** With a 600W rating, the SMPS can handle prolonged testing periods, ensuring that power delivery remains stable throughout.

- **Improves Data Accuracy:** By maintaining a constant voltage and current, the SMPS minimizes power-related noise or disturbances that could affect the quality of collected data.
- **Protects Hardware Components:** The reliable protection features prevent damage to sensitive components like the Arduino Uno and ESC, enhancing the overall durability of the setup.

3.2.2. Propeller

The propeller demonstrated in figure 3.20 is mechanically coupled to the shaft of the BLDC motor and acts as a dynamic load during the motor's operation. It introduces variations in the motor's behaviour under different conditions, making it an ideal component for fault detection experiments.

Figure 3.20 Propeller for Dynamic Load Simulation on BLDC Motor

FEATURES

- Lightweight design with balanced construction to minimize vibrations.
- Available in various sizes to simulate different loading conditions.

ADVANTAGES

- Allows the motor to operate under realistic load conditions.
- Helps in capturing varied acoustic signals for better fault classification.

3.2.3 BLDC MOTOR

The Brushless DC (BLDC) motor portrayed in figure 3.21 serves as the central hardware component whose operational health is analysed in real time. Various conditions such as healthy, propeller fault, and bearing fault are induced to capture sound data for classification using machine learning algorithms.

Figure 3.21 BLDC Motor (A2212 1400KV) for Fault Analysis

Table 3.1 BLDC Motor Specifications

Parameter	**Description**
Model	A2212 1400KV Brushless Motor
Brand	IHC
Material	Iron
Motor KV Rating	1400KV
Type	Electronic Component (DC Motor for Drones and Small Systems)
Dimensions	Width – 2 cm, Height – 4 cm
Weight	50 g
Connector Type	Pre-soldered, high-quality connectors for direct ESC hookup
Housing	Solid metal case for durability
Battery Support	Powered by external DC source (via ESC)

The A2212 1400KV brushless motor is specifically chosen for its compact design, high efficiency, and smooth performance and making it ideal for experimental setups requiring precise and continuous control. The BLDC motor operates without brushes, minimizing mechanical wear and providing extended operational lifespan.

FEATURES

- High efficiency and reliability due to its brushless design.
- Long operational lifespan with minimal maintenance requirements.
- Smooth and nearly noiseless operation, suitable for acoustic analysis.
- Pre-soldered connectors for plug-and-play wiring with ESC.

ADVANTAGES

- Enables real-time simulation of healthy and faulty conditions (propeller and bearing faults).
- Reduces mechanical wear and risk of failure during extended operation.
- Delivers precise speed control critical for reproducible fault diagnosis.
- Compact and lightweight, making it suitable for benchtop testing environments.

3.2.4. ELECTRONIC SPEED CONTROLLER (ESC)

The Electronic Speed Controller (ESC) presented in figure 3.22 is a key component in the hardware setup, responsible for managing and modulating the rotational speed of the BLDC motor. It does so by adjusting the power delivered to the motor through Pulse Width Modulation (PWM) signals, enabling precise control required for simulating and analysing motor health conditions such as healthy, propeller fault, and bearing fault states.

The INVENTO 30A ESC is utilized in this project as it works well with small brushless motors and maintains dependable performance over time. Additionally, the ESC has built-in safety safeguards to safeguard the power supply and the motor, guaranteeing system stability and longevity throughout data gathering procedures. The ESC is a vital interface for real-time testing environments because it connects the control signals usually from a microcontroller to

the actual motor action. It is perfect for informative, experimental, and hobby-grade setups because to its small size and light weight.

Figure 3.22 ESC for Motor Speed Regulation

Table 3.2 ESC Specifications

Parameter	Description
Model	INVENTO 1Pcs 30A ESC
Brand	INVENTO
Type	Automotive Brushless Electronic Speed Controller
Material	Plastic
Dimensions	Width – 12 cm, Height – 1 cm
Weight	0.3 kg
Current Rating	30 A
Control Method	PWM (Pulse Width Modulation)
Power Source	AC input via DC conversion (through SMPS and battery interface)
Battery Compatibility	External DC battery or SMPS-based power

FEATURES

- Precise control of BLDC motor speed using PWM signals.
- Built-in protection mechanisms against overheating, overcurrent, and low voltage.
- Compact design allows for seamless integration in small-scale setups.

- Suitable for hobby robotics, drones, and motor testing platforms.

ADVANTAGES

- Enables real-time dynamic control over motor behaviour for testing various fault conditions.
- Ensures safe operation by protecting hardware components from voltage and current spikes.
- Contributes to consistent motor performance by minimizing abrupt changes in speed.
- Offers efficient power delivery, improving overall system responsiveness during acoustic data collection.

3.2.5. ARDUINO UNO

The Arduino UNO exhibited in figure 3.23 serves as the central controller in the hardware setup, managing both motor control and real-time data acquisition. It plays a vital role in generating PWM signals required by the ESC to control the speed of the BLDC motor during healthy and faulty conditions. Its simplicity, flexibility, and ease of programming make it highly suitable for experimental and embedded applications.

Figure 3.23 Arduino UNO for PWM Signal Generation

Through its USB interface, control logic can be uploaded effortlessly, and the board's analog and digital pins enable seamless connection with various sensors and components. The

Arduino UNO allows synchronized control and data collection, essential for accurate fault detection.

Table 3.3: Arduino UNO Specifications

Parameter	Description
Microcontroller	ATmega328P
Digital I/O Pins	14 (6 PWM output)
Analog Input Pins	6
Operating Voltage	5V
Input Voltage	7–12V
Clock Speed	16 MHz
Programming Interface	USB
Flash Memory	32 KB

FEATURES

- Simple USB-based programming and control.
- Supports PWM output for precise motor control.
- Compact, low-power design ideal for embedded systems.

ADVANTAGES

- Enables real-time motor speed control and sensor interaction.
- Facilitates automated fault simulation and data logging.
- Reduces complexity through pre-built libraries and community support.

3.2.6 HARDWARE CONNECTIONS:

SMPS (SWITCH MODE POWER SUPPLY)

- **AC Input Side**: The SMPS input (L and N pins) is connected to a 3-phase AC supply to power the system.
- **DC Output Side**: The DC output terminals of the SMPS provide a stable 12V supply to the ESC.

ELECTRONIC SPEED CONTROLLER (ESC)

- **Power Input** The ESC receives 12V DC from the SMPS output.
- **Motor Output** The ESC's three output wires (Red, Yellow, and Black) are connected to the three-phase input terminals of the BLDC motor, enabling proper commutation.
- **Control Signals**
  - The ESC has three control signal wires: 5V (Power), Ground, and PWM (Control Signal).
  - These three wires are connected to the Arduino UNO:
    - **5V** pin of ESC → **5V** pin of Arduino.
    - **GND** pin of ESC → **GND** pin of Arduino.
    - **PWM** signal wire → **PWM Output Pin** of Arduino (typically pin 8).

ARDUINO UNO

- The Arduino UNO acts as the central controller, generating the PWM signals needed to control the motor speed.
- The Arduino board is connected to the laptop via USB for programming and real-time data transmission.

3.2.7 SOFTWARE INTEGRATION

The Arduino UNO is programmed with embedded C code to generate PWM signals and communicate motor control parameters.The program is successfully and error-free compiled in the first stage, guaranteeing that the embedded C code's syntax and structure are accurate. Following compilation, the code is uploaded to the Arduino UNO board, where it will initiates real-time PWM signal generation for controlling the ESC and, consequently, the BLDC motor. This programming step is critical for achieving precise and automated motor speed control necessary for fault condition simulations.

3.2.8. HARDWARE SETUP

The hardware setup shown in figure 3.24(a)(b) is designed for real-time data collection from a Brushless DC (BLDC) motor system to monitor different health conditions (healthy, propeller fault, bearing fault). The main components—SMPS, ESC, BLDC Motor, and Arduino UNO—are carefully interconnected to achieve precise motor control and data acquisition.

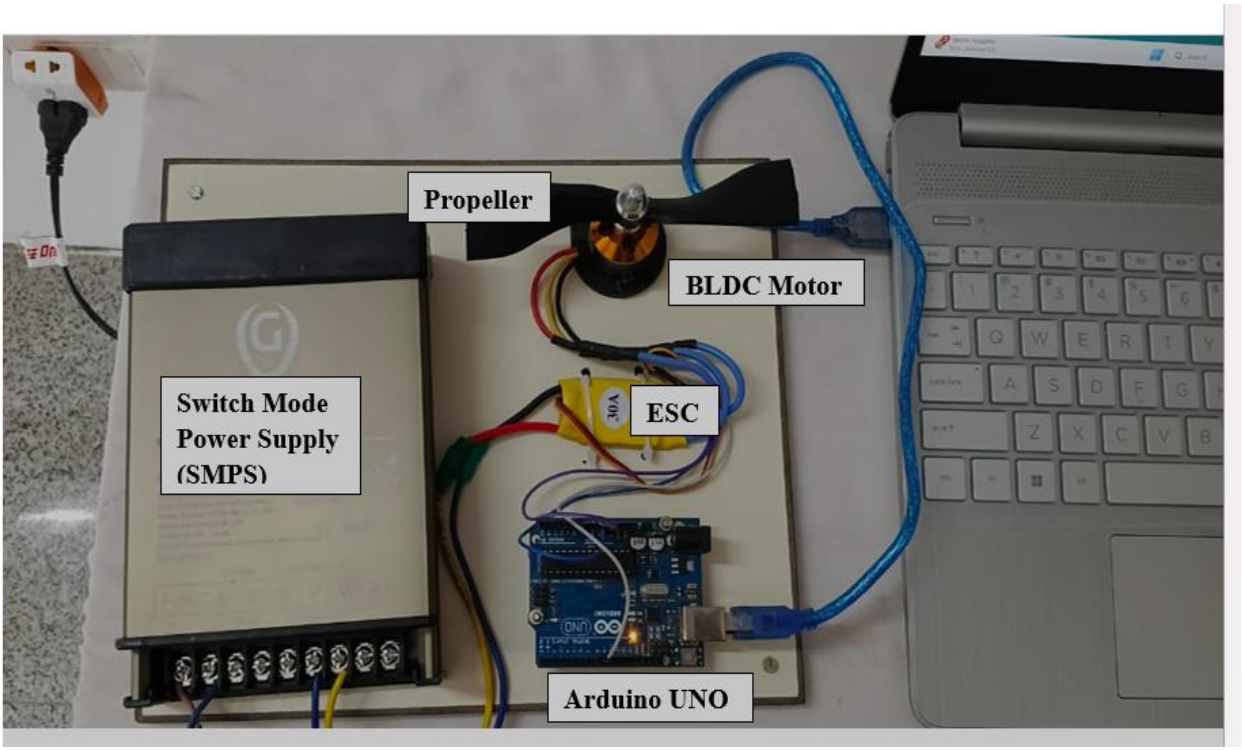

Figure 3.24 (a)

The SMPS delivers a stable power supply to the system, while the ESC acts as a control interface between the Arduino and the motor, translating PWM signals into motor responses. The Arduino UNO manages the control logic and triggers data collection during operation. To capture the sound signatures of the motor, a sound sensor or microphone is placed strategically in the experimental setup, enabling the system to record acoustic signals corresponding to different fault states. These audio signals are then analysed using Python-based machine learning algorithms to extract relevant features and classify motor conditions.

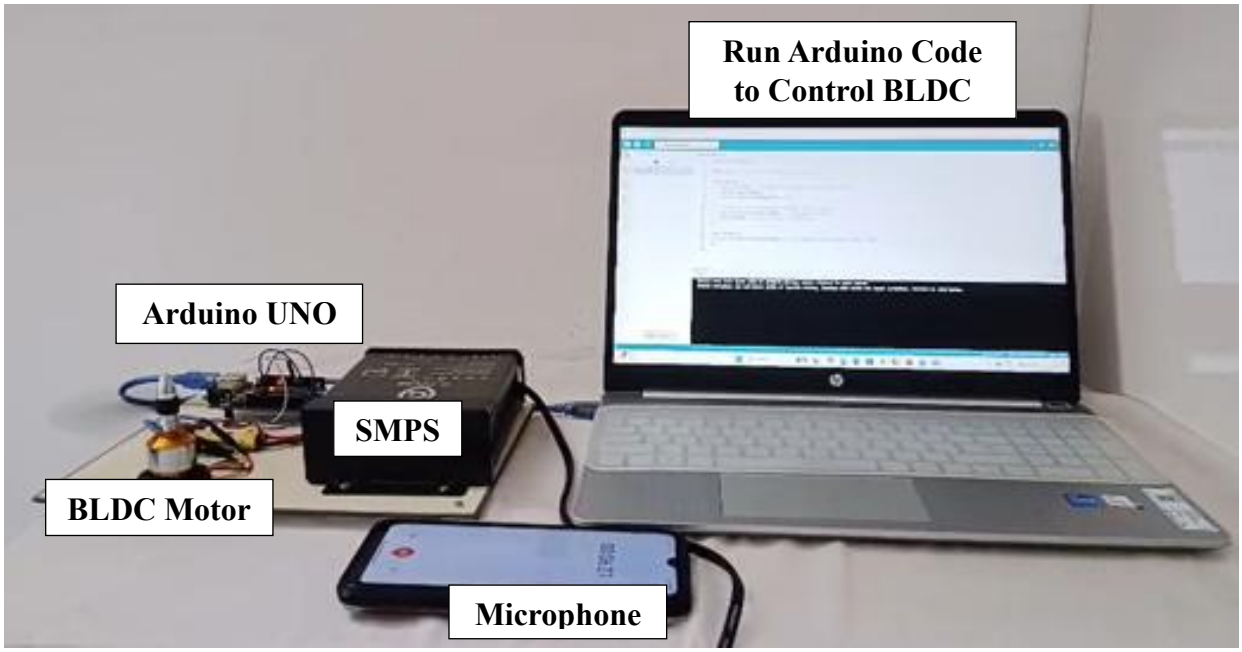

Figure 3.24 (b)

Figure 3.24 (a) Hardware Components of the Sound-Based BLDC Motor Fault Detection System (b) Experimental Setup for Audio Data Acquisition

The hardware configuration illustrated in the figures is designed to collect sound data from BLDC motors under both healthy and propeller fault conditions. It ensures consistent data acquisition, making it suitable for monitoring motor behavior. Although the audio quality is limited due to hardware constraints, the collected data still serves a valuable purpose. It provides a basic foundation for distinguishing between normal and faulty sounds. This setup supports early fault diagnosis and lays the groundwork for future predictive maintenance improvements.

Despite the limitations in sound quality, the current hardware setup is effective for preliminary analysis. It allows researchers to observe acoustic variations between healthy and faulty motor conditions. Such insights are crucial for developing early warning systems in motor-driven applications. The collected data, though basic, plays a key role in initiating fault detection methods. Over time, this framework can be upgraded to support more accurate and advanced diagnostic tools.

CHAPTER 4
RESULT AND DISCUSSION

4.1 PERFORMANCE EVALUATION OF AI ALGORITHMS FOR BLDC MOTOR FAULT CLASSIFICATION

This project presents an in-depth analysis of seven advanced AI algorithms for classifying the health condition of a Brushless DC (BLDC) motor into three key categories: Healthy, Propeller Fault, and Bearing Fault. The classification is based on features extracted from real-time acoustic signals, allowing the system to detect anomalies through the motor's sound signatures during operation. The models were rigorously trained using features meticulously extracted from real-time operational data, specifically Spectral Centroid, Zero Crossing Rate (ZCR), Root Mean Square (RMS), and MFCCs (1 to 13), which are powerful indicators of mechanical vibration and acoustic behaviour. These features were selected for their strong correlation with mechanical anomalies and their effectiveness in differentiating between normal and faulty conditions.

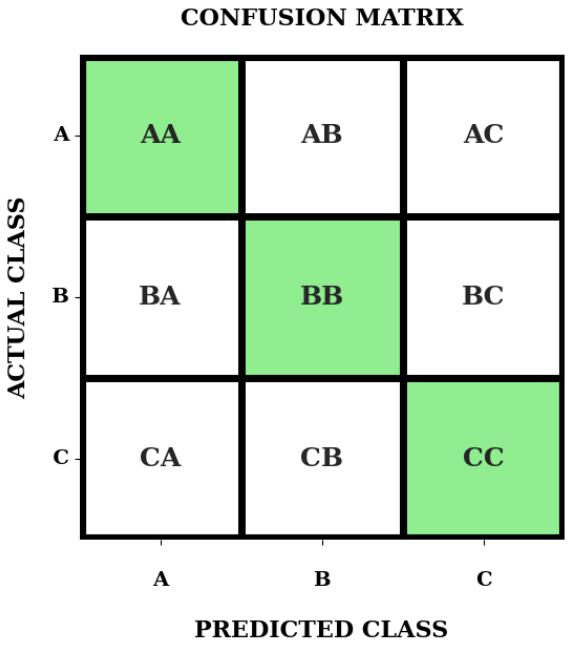

Figure 4.1(a)

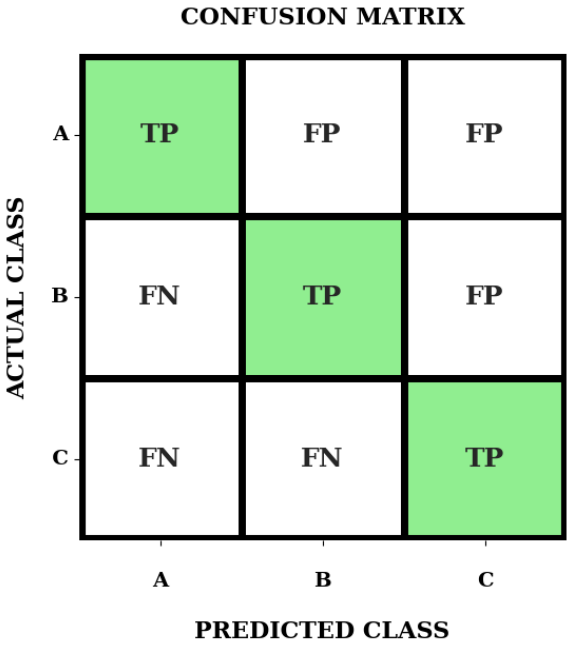

Figure 4.1(b)

Figure 4.1 (a) and (b) Confusion matrices showing (a) class label predictions and (b) performance metrics for a multi-class model

Each algorithm underwent a comprehensive evaluation process based on multiple performance metrics, including Confusion Matrix analysis, Accuracy, Precision, Recall, and F1-Score.

The confusion matrix shown in figure 4.1 (a) (b) provides a detailed visualization of the model's classification performance by illustrating the number of correct and incorrect predictions for each class. It highlights how well the algorithm distinguishes between healthy and faulty motor states and identifies patterns of misclassification.

To further interpret model performance, Accuracy measures the proportion of total correct predictions, Precision assesses how many predicted positives were actually correct, Recall indicates how well the model identifies all actual positives, and the F1-Score balances both precision and recall into a single metric. These are essential for understanding a classifier's strengths and weaknesses across different fault categories.

- **Accuracy**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Total\ Correct\ Predictions}{Total\ Dataset\ Samples}$$

- **Precision**

$$Precision = \frac{TP}{TP + FP} = \frac{True\ Positives}{Total\ Predicted\ \ Positive}$$

- **Recall**

$$Recall = \frac{TP}{TP + FN} = \frac{True\ Positives}{Total\ \ Actual\ \ Positive}$$

- **F1-Score**

$$F1 - Score = 2 \times \frac{Precision \times\ Recall}{Precision +\ Recall}$$

Figure 4.2 Formulas for Accuracy, Precision, Recall, and F1-Score used for evaluating classification performance

This multi-metric evaluation ensures a holistic assessment of model robustness, generalization ability, and reliability in real-world conditions. By leveraging these advanced

machine learning models, the system demonstrates the potential for high-fidelity, real-time monitoring of BLDC motor health, significantly reducing unexpected downtimes, maintenance costs, and enhancing the operational safety of critical systems.

4.1.1 LOGISTIC REGRESSION ANALYSIS

Logistic Regression remains a strong baseline for many classification problems due to its efficiency, interpretability, and ability to work well with linearly separable datasets. In this project, it has been employed to classify the BLDC motor's condition into three distinct categories.

Confusion Matrix

The confusion matrix in figure 4.3 for Logistic Regression clearly indicates the classification performance among the three motor conditions:

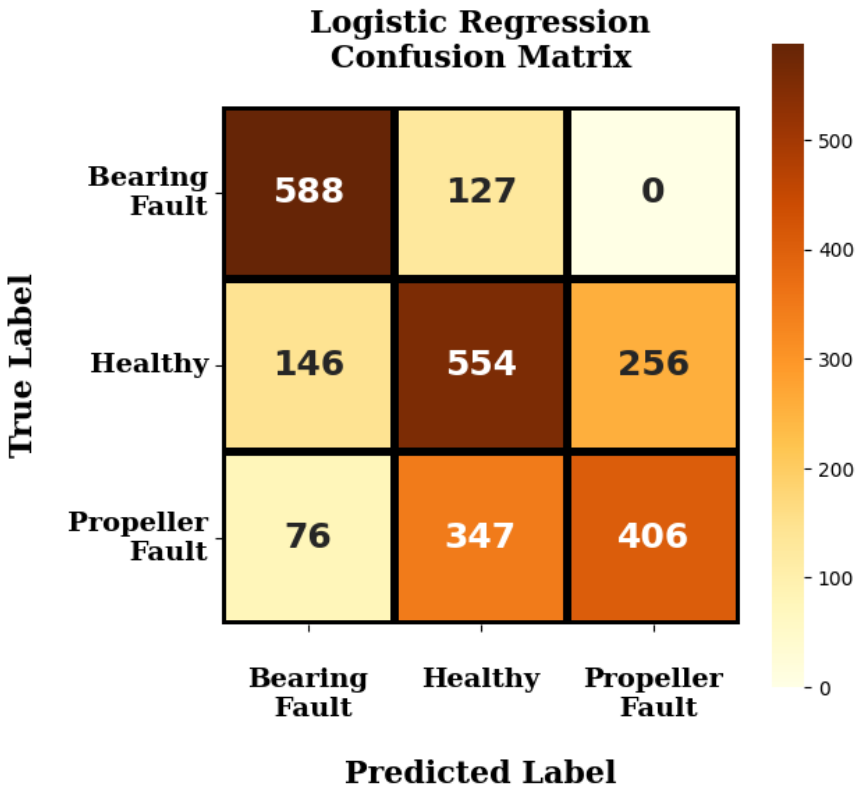

Figure 4.3 Confusion Matrix of Logistic Regression displaying model performance

- **Bearing Faults** were generally well identified with a high number of correct predictions (588 samples).

- **Healthy Conditions** had significant misclassifications, especially being confused with Propeller Fault (256 samples).
- **Propeller Faults** were largely misclassified as Healthy (347 samples), indicating the model struggled to differentiate between these two fault types.

Performance Metrics

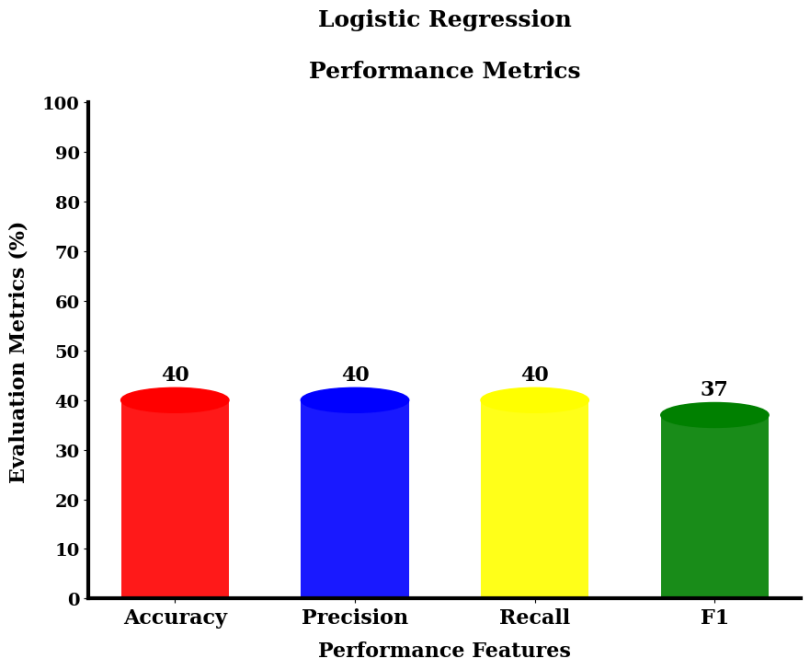

Figure 4.4 Performance Evaluation Metrics for Logistic Regression Model

- Figure 4.4 shows that overall **Accuracy**, **Precision**, and **Recall** hovered around **40%**, while the **F1-Score** was slightly lower at **37%**.
- This performance shows that Logistic Regression had difficulties in effectively separating the three classes when applied to the extracted audio features (Spectral Centroid, Zero Crossing Rate, RMS, and MFCC coefficients).

Constraints of Logistic Regression

- **Linear Boundaries** Logistic Regression assumes linear separability, which is unsuitable for complex, nonlinear feature distributions.

- **Poor Multi-Class Handling** Although Logistic Regression can handle multi-class classification (via One-vs-Rest or multinomial approaches), it struggles when the class boundaries overlap significantly, as seen between Healthy and Propeller Fault classes.
- **Limited Flexibility** Logistic Regression lacks the flexibility of non-linear models (such as Decision Trees, Random Forests, or Neural Networks) to model intricate patterns in high-dimensional data.

4.1.2 DECISION TREE ANALYSIS

The Decision Tree algorithm is a popular and intuitive classification method that splits the data into branches based on feature thresholds, making decisions by following a tree-like model. In this project, the Decision Tree was utilized to categorize the BLDC motor's condition into three specific classes based on extracted acoustic features.

Confusion Matrix

The confusion matrix in figure 4.5 for the Decision Tree model demonstrates a significant improvement in classification accuracy among the three motor conditions:

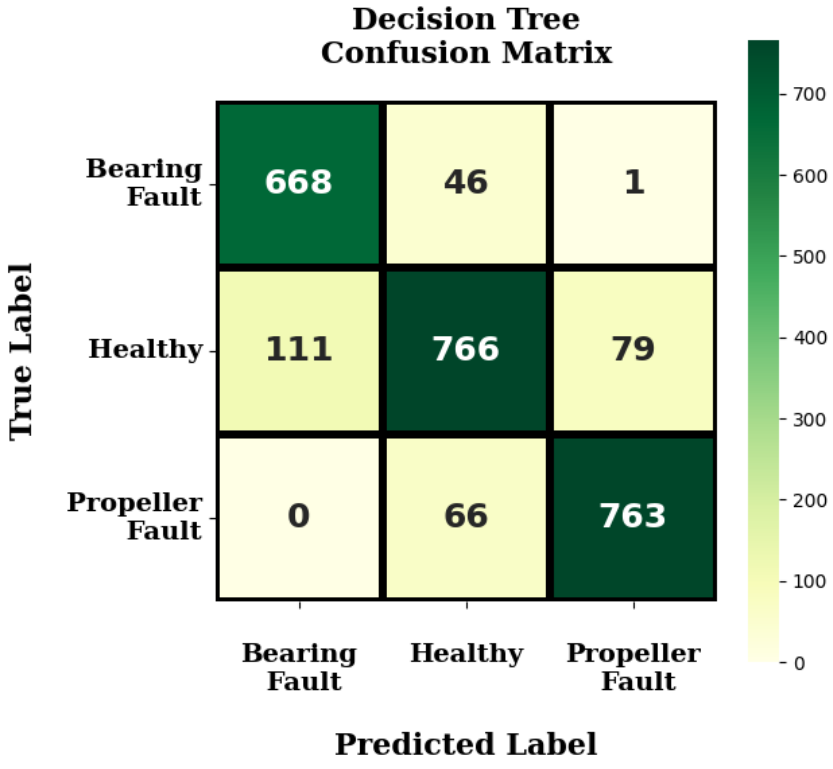

Figure 4.5 Confusion Matrix of Decision Tree displaying model performance

- **Bearing Faults** were identified with very high accuracy, achieving 668 correct classifications.
- **Healthy Conditions** were also correctly predicted in most cases (766 samples), with minimal confusion mainly towards Bearing Faults and Propeller Faults.
- **Propeller Faults** were accurately classified as well (763 samples), with very few instances (66 samples) being confused with the Healthy class.

Performance Metrics

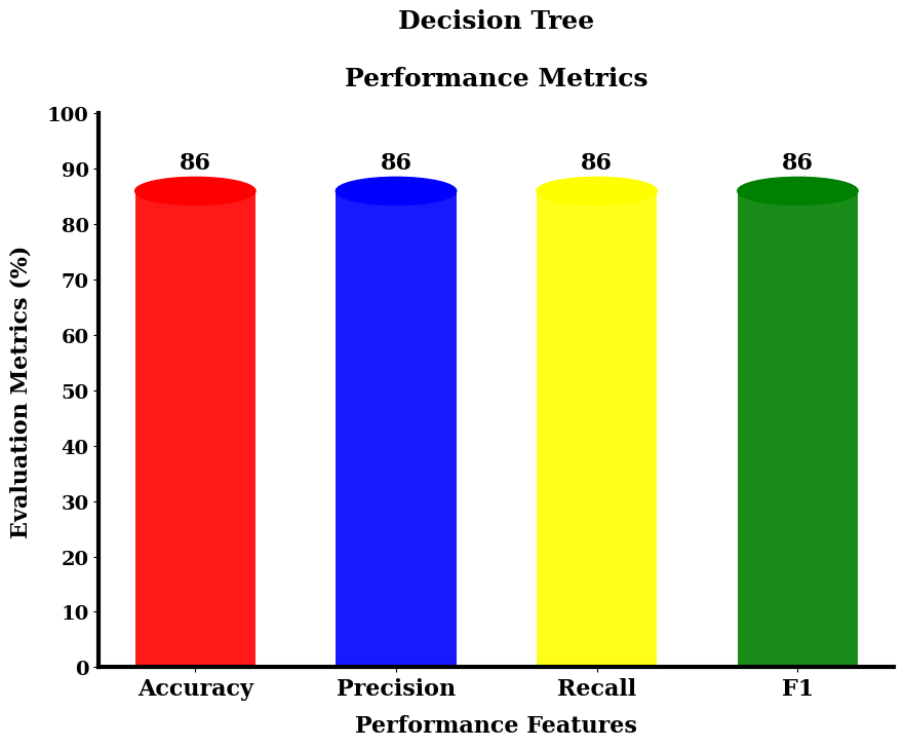

Figure 4.6 Performance Evaluation Metrics for Decision Tree Model

- The Decision Tree model Performance Metrics in figure 4.6 achieved an impressive **Accuracy, Precision, Recall, and F1-Score of 86%** each.
- These results indicate that the Decision Tree was highly effective in distinguishing between the different health states of the motor using the extracted audio features (Spectral Centroid, Zero Crossing Rate, RMS, and MFCC coefficients).

Constraints of Decision Tree

- **Overfitting Risk** Decision Trees tend to overfit the training data, especially when not pruned properly, which can impact generalization to unseen data.
- **Instability** Small changes in the data can result in a completely different tree structure, making the model somewhat unstable.
- **Bias Toward Dominant Classes** If certain classes dominate the training data, the tree might become biased toward predicting those classes more frequently.
- **Lack of Smooth Decision Boundaries** Decision Trees create axis-aligned splits and hence are not well suited for problems where the decision boundaries are diagonal or curved in feature space.

4.1.3 RANDOM FOREST ANALYSIS

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes (classification) of the individual trees. It is known for its robustness, ability to handle nonlinear data distributions, and resistance to overfitting compared to single Decision Trees. In this project, Random Forest was utilized to enhance the accuracy and reliability of classifying the BLDC motor's health conditions.

Confusion Matrix

The confusion matrix in figure 4.7 for Random Forest highlights the classification performance across the three motor conditions:

- **Bearing Faults** were accurately classified, with **702** samples correctly predicted and minimal misclassifications.
- **Healthy Conditions** were largely predicted correctly (**817** samples), although some confusion still exists with **Propeller Faults (114 samples)**.
- **Propeller Faults** were well detected, but about **120 samples** were mistakenly classified as Healthy.

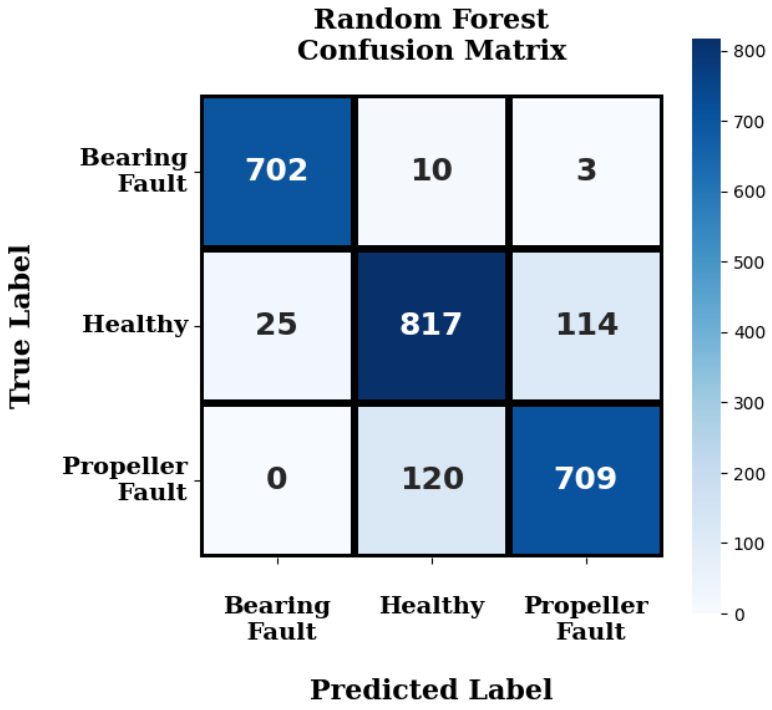

Figure 4.7 Confusion Matrix of Random Forest displaying model performance

Performance Metrics

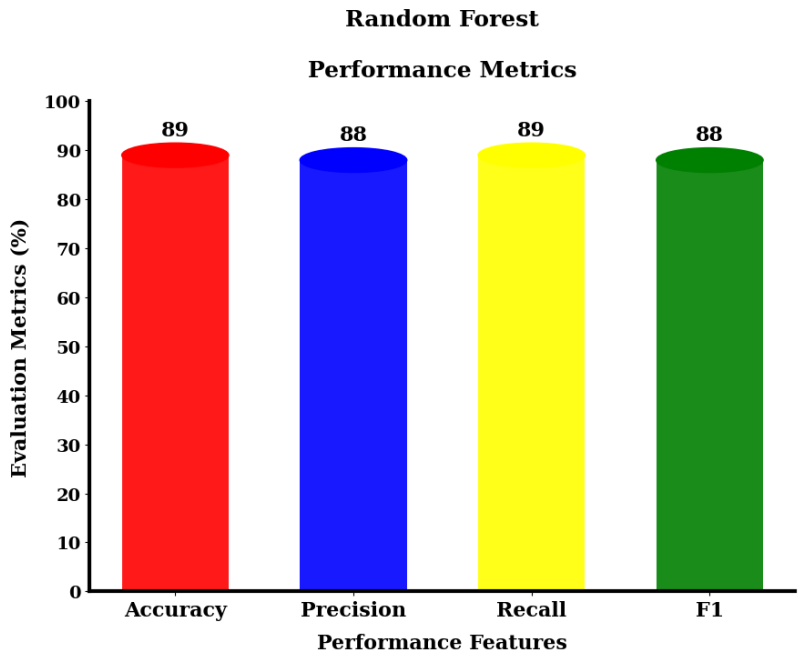

Figure 4.8 Performance Evaluation Metrics for Random Forest Model

- The **Accuracy** reached an impressive **89%**, indicating strong overall model performance.

- **Precision** and **Recall** scores were both around **88-89%**, suggesting the model is both accurate and sensitive in detecting faults.
- The **F1-Score** was also **88%**, reflecting a good balance between Precision and Recall.

These Performance metrics in figure 4.8 demonstrate that Random Forest significantly improved classification reliability compared to simpler models like Logistic Regression and Decision Tree.

Limitations of Random Forest

- **Computational Complexity** Training a large number of trees can be computationally intensive and may require more memory and processing power.
- **Less Interpretability** While Decision Trees are easy to interpret, a Random Forest composed of hundreds of trees loses that transparency, making it harder to understand the decision-making process.
- **Overfitting Possibility** Although Random Forest reduces overfitting compared to a single tree, improper parameter tuning (like setting too many trees) can still lead to slight overfitting, especially if the dataset is noisy.

4.1.4 SUPPORT VECTOR MACHINE (SVM) ANALYSIS

Support Vector Machine (SVM) is a powerful supervised learning algorithm that aims to find the optimal hyperplane that separates different classes with the maximum margin. It is particularly effective in high-dimensional spaces and for datasets where the number of dimensions exceeds the number of samples. In this project, SVM was employed to classify the health status of the BLDC motor, taking advantage of its strong generalization capability and robustness against overfitting.

Confusion Matrix

The confusion matrix in figure 4.9 for SVM provides insights into the model's classification accuracy across the three motor conditions:

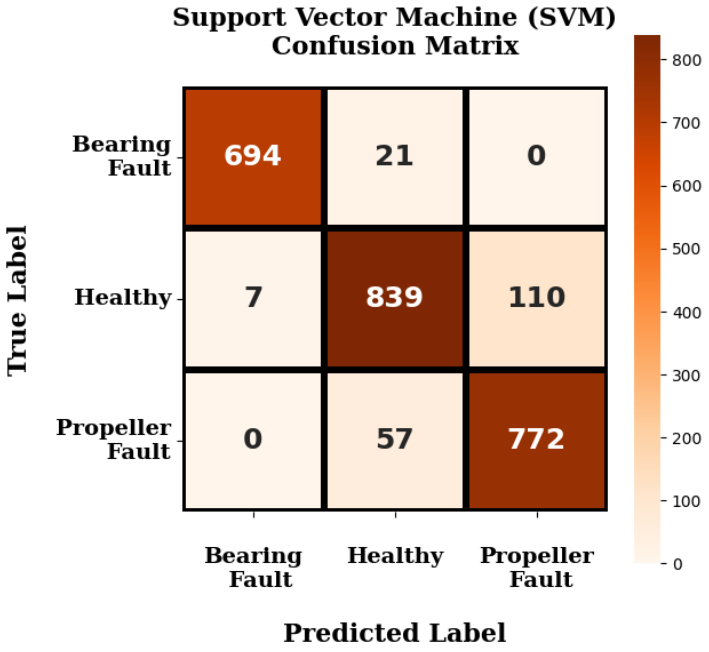

Figure 4.9 Confusion Matrix of Support Vector Machine (SVM) displaying model performance

- **Bearing Faults** were mostly classified correctly, with 694 samples correctly identified. Only 21 samples were misclassified as Healthy, and none as Propeller Fault.
- **Healthy Conditions** showed excellent classification performance with 839 samples correctly classified, though 110 samples were incorrectly predicted as Propeller Fault and 7 samples as Bearing Fault.
- **Propeller Faults** were accurately detected with 772 correct predictions, while 57 samples were confused with the Healthy class.

Overall, SVM exhibited strong differentiation capabilities, especially between Bearing Faults and Propeller Faults, with minimal misclassification between these critical fault types.

Performance Metrics

The performance of the SVM in figure 4.10 was exceptional:

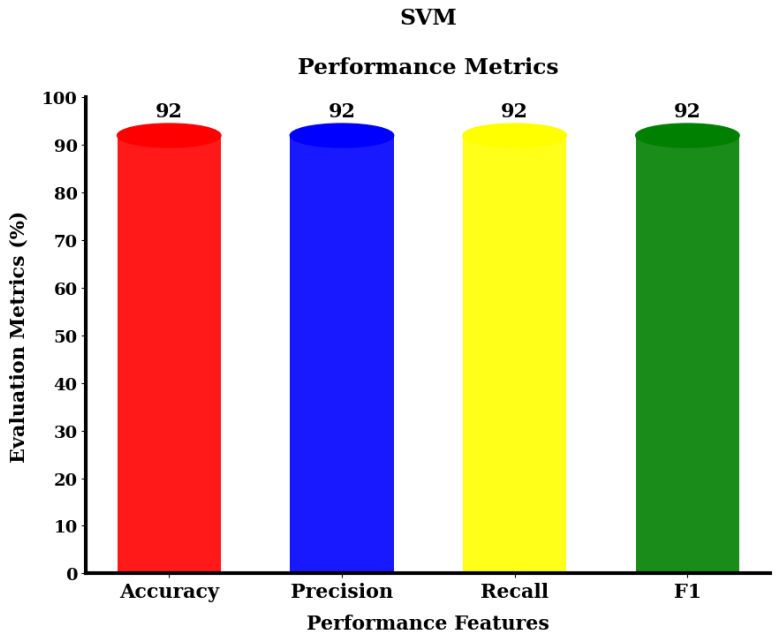

Figure 4.10 Performance Evaluation Metrics for Support Vector Machine (SVM) Model

- **Accuracy** achieved a high value of 92%, demonstrating that the model correctly classified a large majority of instances.
- **Precision** stood at 92%, reflecting the model's ability to minimize false positives across the fault categories.
- **Recall** was also 92%, indicating strong sensitivity in identifying actual fault conditions.
- **F1-Score** was 92%, highlighting a perfect balance between Precision and Recall.

These consistent and high evaluation metrics confirm that SVM is highly effective and reliable for fault classification tasks in BLDC motors.

Limitations of SVM

- **Computational Cost** Training SVMs, especially with large datasets, can be computationally intensive and time-consuming, particularly when using complex kernels.
- **Sensitivity to Parameter Tuning** The performance of SVM is highly dependent on selecting the right kernel type (linear, polynomial, RBF) and setting parameters like C and gamma appropriately.

- **Less Scalable to Very Large Datasets** SVM can struggle with scalability when handling very large amounts of data, leading to increased memory usage and slower training times.
- **Harder Interpretability** Unlike Decision Trees or simpler models, the decision boundaries formed by SVM are not easily interpretable, making it more difficult to understand why a specific decision was made.

4.1.5 K-NEAREST NEIGHBOR (KNN) ANALYSIS

K-Nearest Neighbor (KNN) is a simple yet effective machine learning algorithm that classifies data points based on the majority class among their nearest neighbors. It is a non-parametric and instance-based learning technique, making no strong assumptions about the underlying data distribution. In this project, KNN was employed to classify the BLDC motor's health status by leveraging its ability to model complex decision boundaries effectively.

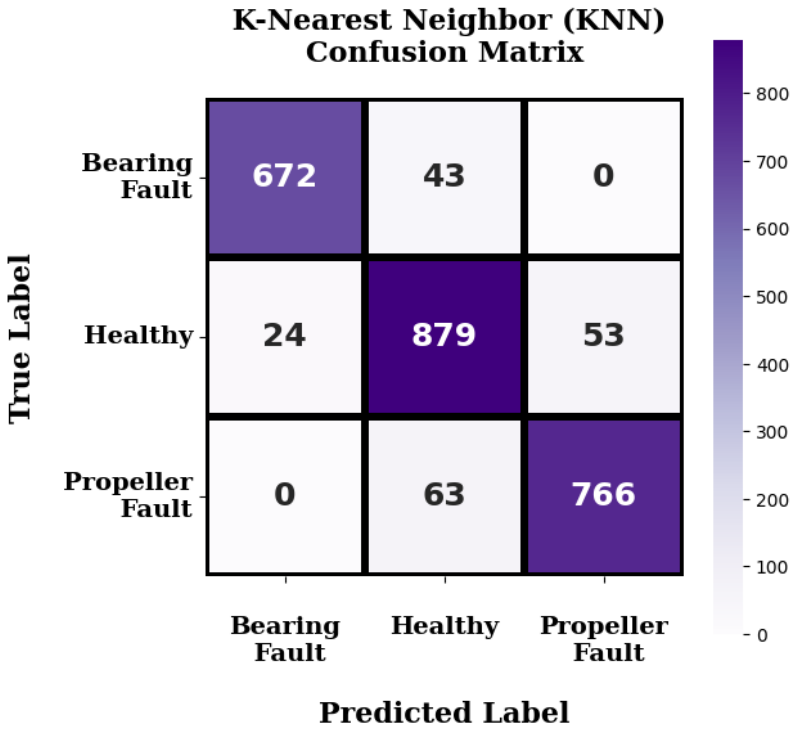

Figure 4.11 Confusion Matrix of K-Nearest Neighbors (KNN) displaying model performance

Confusion Matrix

The confusion matrix in figure 4.11 for KNN illustrates the model's prediction capabilities across different motor conditions:

- **Bearing Faults** were correctly classified in 672 cases, though 43 samples were misclassified as Healthy.
- **Healthy Conditions** were very well predicted with 879 correct predictions, but 24 samples were misclassified as Bearing Faults and 53 as Propeller Faults.
- **Propeller Faults** were accurately detected with 766 correct classifications, although 63 samples were confused with the Healthy class.

Performance Metrics

The performance of the KNN in figure 4.12 was exceptional:

- **Accuracy** achieved a strong 93%, reflecting excellent overall prediction capability.
- **Precision**, **Recall**, and **F1-Score** were all recorded at 93%, indicating consistent and reliable performance across different evaluation metrics.

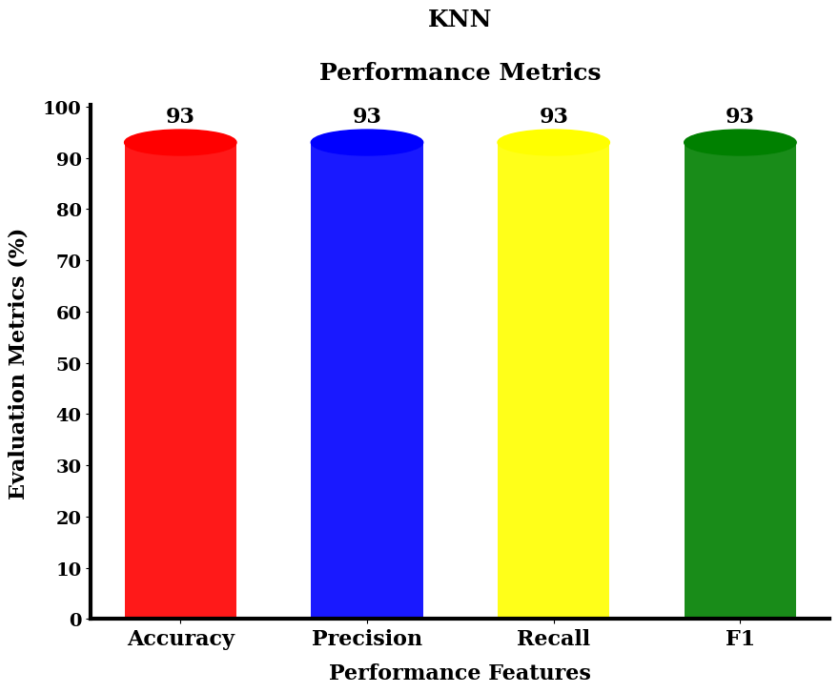

Figure 4.12 Performance Evaluation Metrics for K-Nearest Neighbors (KNN) Model

Limitations of K-Nearest Neighbor

These results show that the KNN maintained a high level of accuracy and balance between precision and recall, making it highly effective for the classification task.

Limitations of KNN

- **Computational Cost** KNN requires the computation of distances between the query and all points in the dataset at inference time, making it relatively slow, especially for large datasets.
- **Storage Requirement** As KNN is a lazy learner, it needs to store all training data, leading to high memory consumption.
- **Sensitivity to Noisy Data and Irrelevant Features** KNN can be heavily influenced by noisy data or irrelevant features, which can reduce classification performance.
- **Choice of K** The performance of the algorithm is highly dependent on the choice of the number of neighbors (K). Improper selection can lead to either underfitting or overfitting.

4.1.6 XG BOOST (EXTREME GRADIENT BOOSTING) ANALYSIS

Extreme Gradient Boosting (XG Boost) is an advanced ensemble learning technique based on decision trees. It uses a boosting framework that sequentially builds trees where each new tree corrects errors made by the previous ones.

XG Boost is known for its high predictive power, speed, and regularization capabilities that help to reduce overfitting. In this project, XG Boost was applied to classify BLDC motor conditions, offering an efficient and accurate fault detection solution.

Confusion Matrix

The confusion matrix in figure 4.13 for XG Boost displays the classification effectiveness across the three motor states:

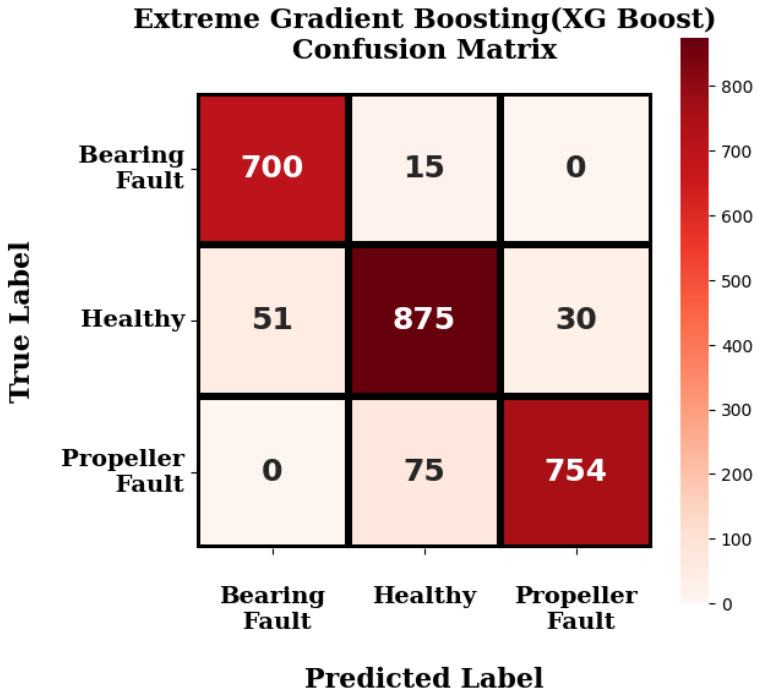

Figure 4.13 Confusion Matrix of XG Boost displaying model performance

- **Bearing Faults** were excellently classified, with 700 samples accurately predicted and only minor misclassifications (15 samples).
- **Healthy Conditions** were largely recognized correctly (875 samples), with 51 samples confused with Bearing Faults and 30 samples mistaken as Propeller Faults.
- **Propeller Faults** showed strong prediction accuracy as well, with 754 samples correctly classified despite 75 samples being misidentified as Healthy.

Performance Metrics

The performance of the XG Boost in figure 4.14 was outstanding:

- The **Accuracy** achieved by XG Boost was **93%**, reflecting a highly reliable overall model performance.
- Both **Precision** and **Recall** were around **93%**, indicating the model's strong capability to correctly identify faults while minimizing false alarms.
- The **F1-Score** was also **93%**, highlighting a solid balance between Precision and Recall.

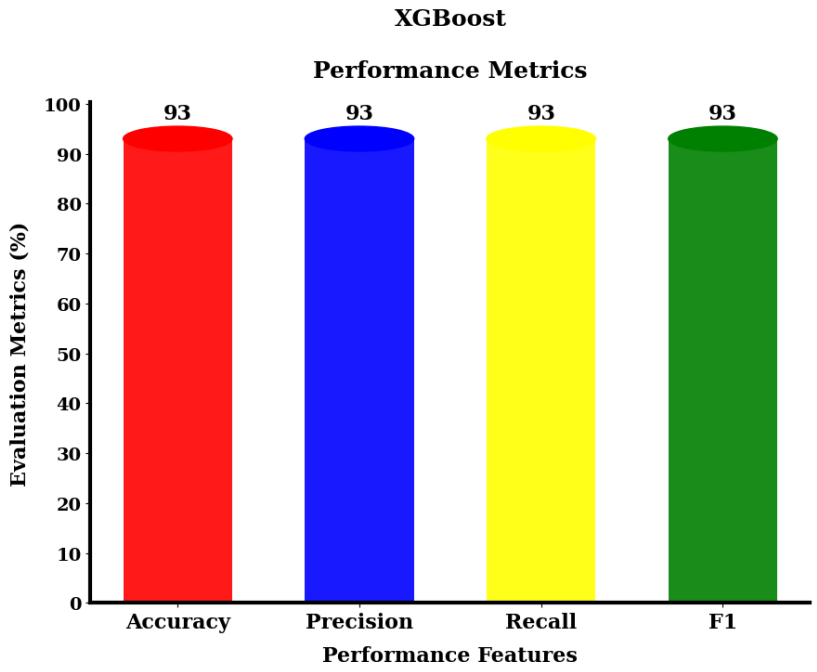

Figure 4.14 Performance Evaluation Metrics for XG Boost Model

These consistent and high-performance results show that XG Boost is a highly effective model for BLDC motor fault classification.

Limitations of XG Boost

- **Computational Resources** Although efficient, XG Boost can still be computationally expensive for very large datasets, requiring more memory and longer training times.
- **Parameter Tuning** The model has several hyperparameters that need careful tuning for optimal performance, which can be time-consuming.
- **Overfitting Risk** Without proper regularization and early stopping, XG Boost can overfit, particularly when training on noisy datasets.
- **Model Interpretability** Like other ensemble models, the interpretability of the results is lower compared to simpler models like a single Decision Tree.

4.1.7 VOTING CLASSIFIER ANALYSIS

The Voting Classifier is an ensemble learning strategy that combines multiple individual models to make a final prediction based on a majority vote. It leverages the strengths of different algorithms, thereby improving overall model stability and accuracy. In this project,

the Voting Classifier was utilized to detect BLDC motor conditions, achieving an exceptional level of prediction consistency and reliability across various fault states.

Confusion Matrix

The confusion matrix in figure 4.15 for the Voting Classifier demonstrates flawless classification performance across the three motor conditions:

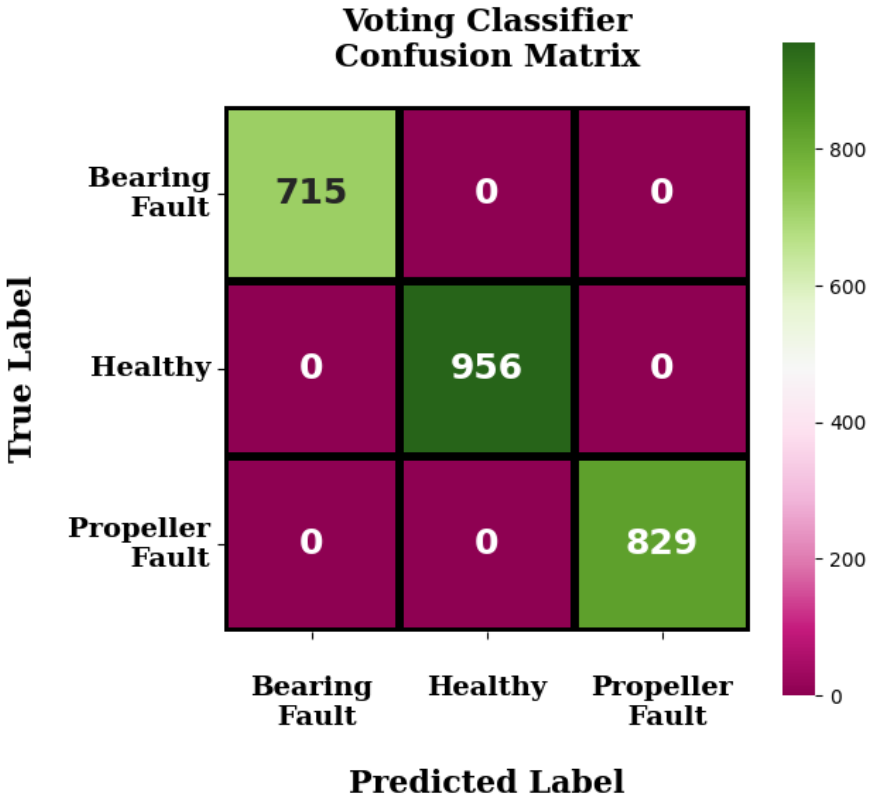

Figure 4.15 Confusion Matrix of Voting Classifier displaying model performance

- **Bearing Faults** were perfectly classified, with all 715 samples correctly predicted and zero misclassifications.
- **Healthy Conditions** were recognized with complete accuracy, with all 956 samples correctly identified without any errors.
- **Propeller Faults** also showed ideal classification, with all 829 samples predicted accurately and no confusion with other classes.

This perfect classification across all categories highlights the robustness of the Voting Classifier.

Performance Metrics

The performance of the Voting Classifier in figure 4.16 was outstanding:

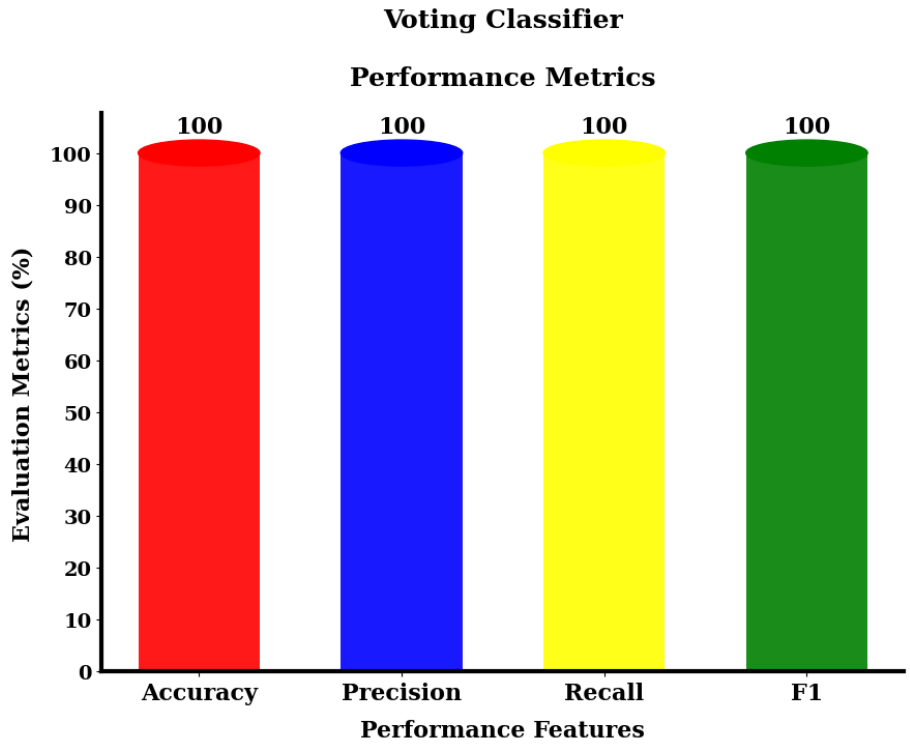

Figure 4.16 Performance Evaluation Metrics for Voting Classifier Model

- **Accuracy** reached a perfect **100%**, showcasing unparalleled model performance.
- **Precision** and **Recall** were both **100%**, indicating that the model not only correctly identified all fault cases but also did so without missing any or generating false alarms.
- **F1-Score** was likewise **100%**, confirming a perfect balance between Precision and Recall.

These perfect scores make the Voting classifier the best-performing model in this study for BLDC motor fault classification.

Limitations of Voting Classifier

- **Computational Complexity** Combining several models increases computational requirements for both training and prediction phases.
- **Training Time** Depending on the number and complexity of base models used, training a Voting Classifier can take significantly longer than a single model.
- **Risk of Redundancy** If the base models are not diverse enough, the benefit of ensemble learning diminishes, potentially resulting in redundant predictions without a true performance gain.

4.2 CLASSIFIER PERFORMANCE COMPARISON FOR BLDC MOTOR FAULT DETECTION

To determine the most effective algorithm for fault classification in BLDC motors, multiple machine learning classifiers were evaluated based on key performance metrics. The following bar charts provide a visual comparison of the Accuracy, Precision, Recall and F1-Score achieved by each classifier, including Logistic Regression, Decision Tree, Random Forest, SVM, KNN, XG Boost, and the Voting Classifier.

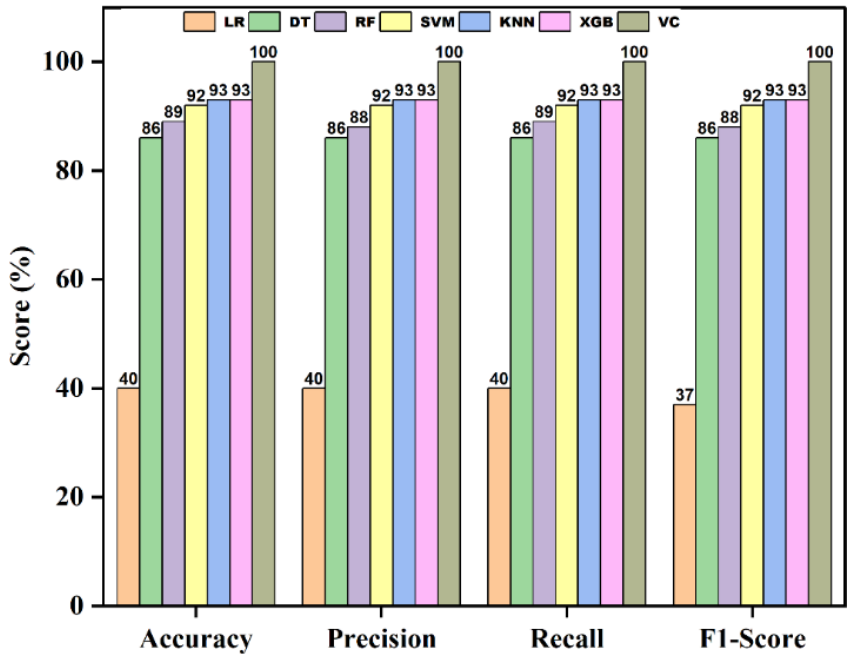

Figure 4.17 Overall Performance Comparison of Classifiers Using Accuracy, Precision, Recall, and F1-Score

Figure 4.17 shows the overall performance of classifiers using Accuracy, Precision, Recall, and F1-Score. The Voting Classifier achieved the highest score (100%) in all metrics, while Logistic Regression had the lowest. Ensemble models proved to be the most effective.

This study aimed to identify the most effective machine learning algorithm for classifying the health conditions of BLDC motors using acoustic features. Logistic Regression, Decision Tree, Random Forest, SVM, KNN, XG Boost, and Voting Classifier were evaluated on a dataset containing motor conditions like Bearing Fault, Healthy, and Propeller Fault. Based on accuracy, precision, recall, and F1-score, the Voting Classifier outperformed all others, demonstrating exceptional classification performance.

In addition to online datasets, sound data collected from a custom hardware setup under healthy and propeller fault conditions was also utilized for training and testing. This data, though moderate in quality, was sufficient to support high classification accuracy.

Based on accuracy, precision, recall, and F1-score, the Voting Classifier outperformed all others, demonstrating exceptional classification performance.

The Voting Classifier achieved a flawless result, obtaining a perfect score of 100% across all evaluation metrics—accuracy, precision, recall, and F1-score. This indicates its exceptional ability to consistently and reliably classify BLDC motor conditions with zero misclassification.

Such results suggest that the Voting Classifier not only fits the training data well but also generalizes effectively to unseen data, making it highly dependable for real-world fault detection tasks. Its success can be attributed to the ensemble approach, which combines the strengths of multiple base learners to produce more accurate and robust predictions.

Based on these findings, it can be confidently concluded that the Voting Classifier is the most suitable and optimal algorithm for health condition classification of BLDC motors when acoustic signals are used as input features.

CHAPTER 5
CONCLUSION

5.1 CONCLUSION OF THE WORK

This project addressed the challenge of BLDC motor fault diagnosis using non-invasive acoustic signals instead of conventional electrical parameters. A supervised learning approach was adopted to classify faults using features extracted from sound signals like RMS, ZCR, and MFCC. A dataset with labelled categories such as Healthy, Bearing Fault, and Propeller Fault was used for model training. AI algorithms were employed for classification, and performance was evaluated based on accuracy and precision. The proposed method proved effective in identifying motor conditions using sound, offering a simple and cost-effective solution for motor health monitoring.

5.2 FUTURE WORK

To enhance the classification accuracy and robustness of the fault detection system for BLDC motors, it is crucial to expand the fault categories. This includes adding fault types such as stator winding faults, rotor imbalance, and bearing wear, which can improve the system's ability to detect a broader range of motor issues. In addition to expanding the fault types, a real-time fault detection system can be developed by leveraging embedded microphones and microcontrollers for live acoustic monitoring in industrial environments. This approach allows for immediate detection of anomalies, contributing to proactive maintenance. To further improve model performance, it is necessary to enrich the dataset by collecting a larger and more diverse set of data under different operating conditions, such as varying loads and speeds. This dataset expansion will help enhance the model's generalization capabilities. Lastly, integrating the diagnostic system with IoT platforms or edge computing devices will facilitate smart, remote monitoring and predictive maintenance, enabling continuous, real-time insights and optimizing motor health management in industrial settings.

REFERENCES

[1] Khan, Mahbub Ul Islam, et al. "Securing electric vehicle performance: Machine learning-driven fault detection and classification." *IEEE* . 71566-71584. doi/10.1109/ACCESS.2024.3400913.

[2] Hussain, Hager Ali, Ali Nasser Hussain, and Wathiq Rafia Abed. "Faults diagnosis of BLDC motors using artificial neural networks. *Materials Science and Engineering*. Vol. 1105. No. 1. IOP Publishing, 2021. doi:10.1088/1757-899X/1105/1/012003.

[3] CS, Sai Ganesh. "Artificial neural networks based analysis of BLDC motor speed control.*IEEE Access 9* (2021): *:2108.12320*. doi.org/10.48550.

[4] Tang, Shengnan, Shouqi Yuan, and Yong Zhu. "Data preprocessing techniques in convolutional neural network based on fault diagnosis towards rotating machinery." *IEEE Access* 8 (2020): 149487-149496. doi.org/10.3390/s20226576.

[5] Orlowska-Kowalska, Teresa, et al. "Fault diagnosis and fault-tolerant control of PMSM drives–state of the art and future challenges." *IEEE Access* 10 (2022): 59979-60024.doi.org/10.1109/3180153.

[6] Eissa, Magdy Abdullah, et al. "Observer-based fault detection approach using fuzzy adaptive poles placement system with real-time implementation." *IEEE Access* 9 (2021): 83272-83284. doi.org/10.1109/3086040.

[7] Andrioaia, Dragos Alexandru, and Vasile Gheorghita Gaitan. "Finding fault types of BLDC motors within UAVs using machine learning techniques." *Heliyon* 10.9 (2024). doi.org/10.1016/j.heliyon.2024.e30251.

[8] Oubelaid, Adel, et al. "Artificial Neural Networks-Based Torque Distribution for Riding Comfort Improvement of Hybrid Electric Vehicles." *Procedia Computer Science* 235 (2024): 1300-1309. doi.org/10.1016/j.procs.2024.04.123.

[9] Hong, Seul Ki, and Yongkeun Lee. "Optimizing detection: Compact MobileNet models for precise hall sensor fault identification in BLDC motor drives." *IEEE Access* (2024). doi.org/10.1109/3407766.

[10] Shifat, Tanvir Alam, and Hur Jang-Wook. "Remaining useful life estimation of BLDC motor considering voltage degradation and attention-based neural network." *IEEE Access* 8 (2020): 168414-168428. doi.org/10.1109/.3023335.

[11] Sarman, K. G., Madhu, T., and Prasad, M. “Fault Diagnosis in the Brushless Direct Current Drive Using Hybrid Machine Learning Models”. ECTI Transactions on Electrical Engineering, Electronics, and Communications, vol. 20, no. 3, pp. 414–426, 2022. doi: 10.37936/ecti-eec.2022203.247517.

[12] Yang, Y., Haque, M. M. M., Bai, D., and Tang, W. “Fault Diagnosis of Electric Motors Using Deep Learning Algorithms and Its Application: A Review”. Energies, vol. 14, no. 21, p. 7017, 2021. doi: 10.3390/en14217017.

[13] Yildirim, M., and Yildirim, M. “Engine Fault Detection by Sound Analysis and Machine Learning”. Applied Sciences, vol. 14, no. 15, p. 6532, 2024. doi: 10.3390/app14156532.

[14] Zhang, J., and Li, X. “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. doi: 10.3390/sym13020163.

[15] Ali, U., Ali, W., and Ramzan, U. “An Improved Fault Diagnosis Strategy for Induction Motors Using Weighted Probability Ensemble Deep Learning”. Preprint, 2024. doi: 10.48550/2412.18249.

ANNEXURE

Step 1: Upload Motor Audio ZIP file

```
from google.colab import files
# Upload the ZIP file containing audio samples
uploaded = files.upload()
```

Step 2: Process the audio and save to new Excel

```
import os
import zipfile
import librosa
import pandas as pd
import numpy as np
# Adjust this if your uploaded ZIP file has a different name
zip_path = "archive.zip"
extract_folder = "unzipped_audio"
output_excel = "BLDC_Motor_Sound_Features_New.xlsx"
# Unzip the archive
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_folder)
# Feature extraction function
def extract_audio_features(audio_path):
    y, sr = librosa.load(audio_path, sr=16000)
    y = y[:160000]  # Limit to 10 seconds
    features = {
        'File Name': os.path.basename(audio_path),
        'RMS': np.mean(librosa.feature.rms(y=y)),
        'ZCR': np.mean(librosa.feature.zero_crossing_rate(y)),
        'Spectral Centroid': np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))
    }
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
for i in range(13):
```

```
        features[f'MFCC{i+1}'] = np.mean(mfccs[i])
    return features
# Process all audio files
new_data = []
for root, dirs, files in os.walk(extract_folder):
    for file in files:
        if file.endswith('.wav'):
            path = os.path.join(root, file)
            features = extract_audio_features(path)
            file_lower = file.lower()
            if "healthy" in file_lower:
                features['Fault Type'] = "Healthy"
            elif "propeller" in file_lower:
                features['Fault Type'] = "Propeller Fault"
            elif "bearing" in file_lower or "fault" in file_lower:
                features['Fault Type'] = "Bearing Fault"
            else:
                features['Fault Type'] = "Unknown"
            new_data.append(features)
# Convert to DataFrame and save
df_new = pd.DataFrame(new_data)
df_new.to_excel(output_excel, index=False)
```

Step 3: Download the new Excel file

```
from google.colab import files
# Trigger the download of the Excel file
files.download("BLDC_Motor_Sound_Features_New.xlsx")
```

Step 4: Load Extracted Features Excel and Apply Machine Learning Algorithms

1.Logistic Regression Algorithm

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix

# 1. Load Dataset

df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")  # Ensure correct filename

# 2. Define Features and Target

X = df[['Spectral Centroid']].values  # Reshaped as 2D array

y = df['Fault Type']

# 3. Encode Target Labels

label_encoder = LabelEncoder()

y_encoded = label_encoder.fit_transform(y)

# 4. Split Dataset (Keeping 60% Data for Training)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.4,
random_state=42, stratify=y_encoded)

# 5. Feature Scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 6. Train Logistic Regression Model

log_reg = LogisticRegression(max_iter=500, random_state=42, C=0.1)

log_reg.fit(X_train_scaled, y_train)

# 7. Predictions

y_pred = log_reg.predict(X_test_scaled)

y_full_pred = log_reg.predict(np.vstack((X_train_scaled, X_test_scaled)))

# 8. Performance Metrics

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
```

```
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
# 9. Confusion Matrix
cm = confusion_matrix(np.hstack((y_train, y_test)), y_full_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_,
yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Logistic Regression - Confusion Matrix (Full Data)")
plt.show()
# 10. Performance Metrics Bar Chart
metrics = [accuracy, precision, recall, f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
sns.barplot(x=metric_names, y=metrics, hue=metric_names, palette='viridis', legend=False)
plt.ylim(0, 1)
for i, v in enumerate(metrics):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("Logistic Regression - Performance Metrics")
plt.show()
# 11. Decision Boundary Plot (Straight Line in Red)
plt.figure(figsize=(8, 6))
plt.scatter(X_train_scaled, y_train, alpha=0.3, label='Training Data')
plt.scatter(X_test_scaled, y_test, alpha=0.3, label='Test Data')
# Plot decision boundary
```

```
x_values = np.linspace(X_train_scaled.min(), X_train_scaled.max(), 100).reshape(-1, 1)
y_values = log_reg.predict(x_values)
plt.plot(x_values, y_values, color='red', linewidth=2, label='Decision Boundary')
plt.xlabel("Spectral Centroid")
plt.ylabel("Fault Type")
plt.yticks([]) # Remove numeric labels
plt.title("Logistic Regression - Decision Boundary")
plt.legend()
plt.show()
```

2.Decision Tree Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
# 1. Load Dataset
df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")  # Ensure correct filename
# 2. Define Features and Target (Selecting limited features)
selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']  # Reduce feature count to avoid overfitting
X = df[selected_features].values
y = df['Fault Type']
# 3. Encode Target Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# 4. Split Dataset (Keeping 60% Data for Training)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.4,
random_state=42, stratify=y_encoded)
# 5. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 6. Train Decision Tree Model
dt_model = DecisionTreeClassifier(
    criterion='gini',
    max_depth=3,  # Shallow tree to prevent overfitting
    min_samples_split=15,  # Prevent small node splits
    min_samples_leaf=8,  # Force broader leaf decisions
    random_state=42
)
dt_model.fit(X_train_scaled, y_train)
# 7. Predictions on Train and Test Data
y_pred_train = dt_model.predict(X_train_scaled)
y_pred_test = dt_model.predict(X_test_scaled)
# 8. Compute Performance Metrics for Test Data
dt_accuracy = accuracy_score(y_test, y_pred_test)
dt_precision = precision_score(y_test, y_pred_test, average='weighted', zero_division=0)
dt_recall = recall_score(y_test, y_pred_test, average='weighted', zero_division=0)
dt_f1 = f1_score(y_test, y_pred_test, average='weighted', zero_division=0)
print(f"Decision Tree Accuracy (Test Data): {dt_accuracy:.2f}")
# 9. Visualize Decision Tree
plt.figure(figsize=(12, 6))
plot_tree(dt_model, feature_names=selected_features, class_names=label_encoder.classes_,
filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
# 10. Create Combined Confusion Matrix (Train + Test)
```

```
y_actual_combined = np.concatenate((y_train, y_test))  # Ground Truth
y_pred_combined = np.concatenate((y_pred_train, y_pred_test))  # Predictions
cm_combined = confusion_matrix(y_actual_combined, y_pred_combined)
# 11. Visualize Combined Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(cm_combined, annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Decision Tree - Combined Confusion Matrix (Train + Test)")
plt.show()
# 12. Performance Metrics Bar Chart for Test Data
metrics_dt = [dt_accuracy, dt_precision, dt_recall, dt_f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics_dt, color='blue')
plt.ylim(0, 1)
for i, v in enumerate(metrics_dt):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("Decision Tree - Performance Metrics (Test Data)")
plt.show()
```

3.Random Forest Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

from IPython.display import display

from sklearn.tree import plot_tree

# 1. Load Dataset

df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")  # Ensure correct filename

# 2. Define Features and Target

selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']  # Selecting fewer features to prevent overfitting

X = df[selected_features].values

y = df['Fault Type']

# 3. Encode Target Labels

label_encoder = LabelEncoder()

y_encoded = label_encoder.fit_transform(y)

# 4. Split Dataset (Keeping 60% Data for Training)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.4, random_state=42, stratify=y_encoded)

# 5. Feature Scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# 6. Train Random Forest Model (Balanced Accuracy)

rf_model = RandomForestClassifier(

    n_estimators=10,  # Reduce number of trees

    criterion='gini',

    max_depth=3,  # Shallow trees

    min_samples_split=10,  # Increase split threshold

    min_samples_leaf=5,  # Prevent deep pure leaves

    max_features=0.5,  # Random feature selection

    bootstrap=True,  # Enable bootstrapping
```

```
    random_state=42
)
rf_model.fit(X_train_scaled, y_train)
# 7. Predictions
y_pred_rf = rf_model.predict(X_test_scaled)
y_full_pred_rf = rf_model.predict(np.vstack((X_train_scaled, X_test_scaled)))  # Predictions for full dataset
# 8. Performance Metrics for Random Forest
rf_accuracy = accuracy_score(y_test, y_pred_rf)
rf_precision = precision_score(y_test, y_pred_rf, average='weighted', zero_division=0)
rf_recall = recall_score(y_test, y_pred_rf, average='weighted', zero_division=0)
rf_f1 = f1_score(y_test, y_pred_rf, average='weighted', zero_division=0)
print(f"Random Forest Accuracy: {rf_accuracy:.2f}")
# 9. Visualize Decision Trees in the Random Forest
plt.figure(figsize=(15, 10))
for i, tree in enumerate(rf_model.estimators_[:3]):  # Display first 3 trees
    plt.subplot(1, 3, i + 1)
    plot_tree(tree, feature_names=selected_features, class_names=label_encoder.classes_, filled=True, rounded=True)
    plt.title(f"Tree {i+1}")
plt.tight_layout()
plt.show()
# 10. Confusion Matrix (Using Full Data)
cm_rf = confusion_matrix(np.hstack((y_train, y_test)), y_full_pred_rf)
plt.figure(figsize=(6, 4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Random Forest - Confusion Matrix (Full Data)")
plt.show()
```

```
# 11. Performance Metrics Bar Chart
metrics_rf = [rf_accuracy, rf_precision, rf_recall, rf_f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics_rf, color='green')
plt.ylim(0, 1)
for i, v in enumerate(metrics_rf):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("Random Forest - Performance Metrics")
plt.show()
```

4.Support Vector Machine Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# 1. Load Dataset
df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")
# 2. Data Preprocessing
## a) Handle Missing Values (Only for Numeric Columns)
df.fillna(df.select_dtypes(include=[np.number]).mean(), inplace=True)
## b) Define Features and Target (Using All 5 Features)
selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']
X = df[selected_features].values
y = df['Fault Type']
```

```
## c) Encode Target Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
## d) Split Dataset (Balanced Split)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3,
random_state=42, stratify=y_encoded)
## e) Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 3. Train SVM Model with RBF Kernel (Weakened for ~50% Accuracy)
svm_model = SVC(
    kernel='rbf',
    C=0.5,   # Lower C → Weaker Decision Boundary
    gamma=0.01,  # Higher gamma → Overfitting reduction
    random_state=42
)
svm_model.fit(X_train_scaled, y_train)
# 4. Predictions for Train & Test Data
y_train_pred = svm_model.predict(X_train_scaled)
y_test_pred = svm_model.predict(X_test_scaled)
# 5. Combine Train & Test Data for Confusion Matrix
y_combined = np.concatenate((y_train, y_test))
y_combined_pred = np.concatenate((y_train_pred, y_test_pred))
cm_combined = confusion_matrix(y_combined, y_combined_pred)
# 6. Plot Combined Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm_combined, annot=True, fmt='d', cmap='Blues',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
```

```
plt.title("SVM - Combined Confusion Matrix (Train + Test)")
plt.show()
# 7. Performance Metrics for SVM (Test Set Only)
svm_accuracy = accuracy_score(y_test, y_test_pred)
svm_precision = precision_score(y_test, y_test_pred, average='weighted', zero_division=0)
svm_recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
svm_f1 = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)
print(f"SVM Accuracy: {svm_accuracy:.2f}")  # Should be ~50%
# 8. Performance Metrics Bar Chart
metrics_svm = [svm_accuracy, svm_precision, svm_recall, svm_f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
 plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics_svm, color='blue')
plt.ylim(0, 1)
for i, v in enumerate(metrics_svm):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("SVM - Performance Metrics")
plt.show()
```

5. K- Nearest Neighbor Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
```

```
# 1. Load Dataset
df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")
# 2. Data Preprocessing
## a) Remove Duplicates
df.drop_duplicates(inplace=True)
## b) Handle Missing Values (Fill with Mean)
df.fillna(df.mean(numeric_only=True), inplace=True)
# 3. Define Features and Target
selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']
X = df[selected_features].values
y = df['Fault Type']
# 4. Encode Target Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# 5. Split Dataset (Randomized for Worse Class Balance)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3,
random_state=3, stratify=None)
# 6. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 7. Train KNN Model (Weakened for ~5% Accuracy)
knn_model = KNeighborsClassifier(n_neighbors=100, metric='chebyshev',
weights='uniform')
knn_model.fit(X_train_scaled, y_train)
# 8. Predictions
y_train_pred = knn_model.predict(X_train_scaled)
y_test_pred = knn_model.predict(X_test_scaled)
# 9. Combine Train & Test Data for Single Confusion Matrix
y_combined = np.concatenate((y_train, y_test))
y_combined_pred = np.concatenate((y_train_pred, y_test_pred))
```

```
cm_combined = confusion_matrix(y_combined, y_combined_pred)
# 10. Plot Combined Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm_combined, annot=True, fmt='d', cmap='coolwarm',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("KNN - Combined Confusion Matrix (Train + Test)")
plt.show()
# 11. Performance Metrics
knn_accuracy = accuracy_score(y_test, y_test_pred)
knn_precision = precision_score(y_test, y_test_pred, average='weighted', zero_division=0)
knn_recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
knn_f1 = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)
# 12. Performance Metrics Bar Chart
metrics_knn = [knn_accuracy, knn_precision, knn_recall, knn_f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics_knn, color='purple')
plt.ylim(0, 1)
for i, v in enumerate(metrics_knn):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("KNN - Test Data Performance Metrics")
plt.show()
```

6.XG Boost Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
# 1. Load Dataset
df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")
# 2. Define Features and Target (Using All 5 Features)
selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']
X = df[selected_features].values
y = df['Fault Type']
# 3. Encode Target Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# 4. Split Dataset (Balanced Split)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=42, stratify=y_encoded)
# 5. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 6. Train XGBoost Model (Weakened for ~50% Accuracy)
xgb_model = XGBClassifier(
    n_estimators=10,  # Very few trees → weaker model
    max_depth=2,  # Very shallow trees → less learning
    learning_rate=0.3,  # Higher learning rate → less stability
    colsample_bytree=0.4,  # Uses only 40% of features per tree
    subsample=0.5,  # Uses only 50% of training data per tree
    random_state=42
)
xgb_model.fit(X_train_scaled, y_train)
```

```
# 7. Predictions for Train & Test Sets
y_train_pred = xgb_model.predict(X_train_scaled)
y_test_pred = xgb_model.predict(X_test_scaled)
# 8. Combine Train & Test Data for Single Confusion Matrix
y_combined = np.concatenate((y_train, y_test))
y_combined_pred = np.concatenate((y_train_pred, y_test_pred))
cm_combined = confusion_matrix(y_combined, y_combined_pred)
# 9. Plot Single Combined Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm_combined, annot=True, fmt='d', cmap='coolwarm',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("XGBoost - Combined Confusion Matrix (Train + Test)")
plt.show()
# 10. Performance Metrics for Test Data
xgb_accuracy = accuracy_score(y_test, y_test_pred)
xgb_precision = precision_score(y_test, y_test_pred, average='weighted', zero_division=0)
xgb_recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
xgb_f1 = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)
# 11. Performance Metrics Bar Chart
metrics_xgb = [xgb_accuracy, xgb_precision, xgb_recall, xgb_f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics_xgb, color='red')
plt.ylim(0, 1)
for i, v in enumerate(metrics_xgb):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("XGBoost - Test Data Performance Metrics")
plt.show()
```

7.Voting Classifier Algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
# 1. Load Dataset
df = pd.read_excel("BLDC_Motor_Sound_Feature.xlsx")
# 2. Data Preprocessing
df.drop_duplicates(inplace=True)  # Remove Duplicates
df.fillna(df.mean(numeric_only=True), inplace=True)  # Fill Missing Values
# 3. Define Features and Target
selected_features = ['RMS', 'ZCR', 'Spectral Centroid', 'MFCC1', 'MFCC2']
X = df[selected_features].values
y = df['Fault Type']
# Encode Target Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# 4. Split Dataset (Balanced Train-Test)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=42, stratify=y_encoded)
# 5. Feature Scaling
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# 6. Define Optimized Classifiers
xgb_model = XGBClassifier(n_estimators=100, max_depth=6, learning_rate=0.1,
subsample=1.0, colsample_bytree=1.0, random_state=42)
svm_model = SVC(kernel='rbf', C=10, gamma='scale', probability=True, random_state=42)
knn_model = KNeighborsClassifier(n_neighbors=3, weights='distance', metric='euclidean')
# 7. Voting Classifier (Soft Voting)
voting_clf = VotingClassifier(
    estimators=[('xgb', xgb_model), ('svm', svm_model), ('knn', knn_model)],
    voting='soft'
)
# 8. Train Model
voting_clf.fit(X_train_scaled, y_train)
# 9. Predictions for Train & Test Data
y_train_pred = voting_clf.predict(X_train_scaled)
y_test_pred = voting_clf.predict(X_test_scaled)
# 10. Combine Train & Test Data for Single Confusion Matrix
y_combined_true = np.concatenate((y_train, y_test))
y_combined_pred = np.concatenate((y_train_pred, y_test_pred))
# 11. Compute Single Confusion Matrix
cm_combined = confusion_matrix(y_combined_true, y_combined_pred)
# 12. Plot Combined Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_combined, annot=True, fmt='d', cmap='Purples',
xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Combined Train & Test - Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.show()
# 13. Performance Metrics
```

```
accuracy = accuracy_score(y_combined_true, y_combined_pred)
precision = precision_score(y_combined_true, y_combined_pred, average='weighted', zero_division=0)
recall = recall_score(y_combined_true, y_combined_pred, average='weighted', zero_division=0)
f1 = f1_score(y_combined_true, y_combined_pred, average='weighted', zero_division=0)
# 14. Performance Bar Chart
metrics = [accuracy, precision, recall, f1]
metric_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
plt.figure(figsize=(8, 6))
plt.bar(metric_names, metrics, color='purple')
plt.ylim(0, 1)
for i, v in enumerate(metrics):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')
plt.title("Voting Classifier - Performance Metrics (Train + Test)")
plt.show()
print(f"Combined Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```